JSP Standard Tag Library

Note: In order to use JSTL, we have to include a jar file called jstl-1.2.jar in lib folder along with servlet-api.jar and jsp-api.jar

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

* **Core Tags**
* **Formatting tags**
* **SQL tags**
* **JSTL Functions**

**JSTL Core tags**

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

There are following Core JSTL Tags:

|  |  |
| --- | --- |
| Tag | **Description** |
| <c:out > | Like <%= ... >, but for expressions. |
| <c:set > | Sets the result of an expression evaluation in a 'scope' |
| <c:remove > | Removes a scoped variable (from a particular scope, if specified). |
| <c:catch> | Catches any Throwable that occurs in its body and optionally exposes it. |
| <c:if> | Simple conditional tag which evalutes its body if the supplied condition is true. |
| <c:choose> | Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise> |
| <c:when> | Subtag of <choose> that includes its body if its condition evalutes to 'true'. |
| <c:otherwise > | Subtag of <choose> that follows <when> tags and runs only if all of the prior conditions evaluated to 'false'. |
| <c:import> | Retrieves an absolute or relative URL and exposes its contents to either the page, a String in 'var', or a Reader in 'varReader'. |
| <c:forEach > | The basic iteration tag, accepting many different collection types and supporting subsetting and other functionality . |
| <c:forTokens> | Iterates over tokens, separated by the supplied delimeters. |
| <c:param> | Adds a parameter to a containing 'import' tag's URL. |
| <c:redirect > | Redirects to a new URL. |
| <c:url> | Creates a URL with optional query parameters |

c:out

It is just like JSP expression tag but it is used for exression. It renders data to the page.

*Example of c:out*

Let's see the simple example of c:out.

**index.jsp**

* <form action="process.jsp" method="post">
* FirstName:<input type="text" name="fname"/><br/>
* LastName:<input type="text" name="lname"/><br/>
* <input type="submit" value="submit"/>
* </form>

**process.jsp**

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

First Name:<c:out value="${param.fname}"></c:out><br/>

Last Name:<c:out value="${param.lname}"></c:out>

c:import

It is just like jsp include but it can include the content of any resource either within server or outside the server.

*Example of c:import*

Let's see the simple example of c:import to display the content of other site.

**index.jsp**

* <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
* <h1>ABC.com</h1>
* <c:import url="http://www.google.com"></c:import>

*Example of c:import to display the source code*

Let's see the simple example of c:import to display the source code of other site.

**index.jsp**

* <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
* <h1>ABC.com</h1>
* <hr/>
* <c:import var="data" url="http://www.google.com"></c:import>
* <h2>Data is:</h2>
* <c:out value="${data}"></c:out>

c:forEach

It repeats the nested body content for fixed number of times or over collection.

*Example of c:forEach*

Let's see the simple example of c:forEach.

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

<c:forEach var="number" begin="5" end="10">

<c:out value="${number}"></c:out>

</c:forEach>

c:if

It tests the condition.

*Example of c:if*

Let's see the simple example of c:if.

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

<c:set var="number" value="${200}">

<c:if test="${number<500}">

<c:out value="number is less than 500"></c:out>

</c:if>

c:redirect

It redirects the request to the given url.

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

**SQL tags:**

The JSTL SQL tag library provides tags for interacting with relational databases (RDBMSs) such as Oracle, mySQL, or Microsoft SQL Server.

Following is the syntax to include JSTL SQL library in your JSP:

<%@ taglib prefix="sql"

uri="http://java.sun.com/jsp/jstl/sql" %>

Following is the list of SQL JSTL Tags:

|  |  |
| --- | --- |
| Tag | **Description** |
| <sql:setDataSource> | Creates a simple DataSource suitable only for prototyping |
| <sql:query> | Executes the SQL query defined in its body or through the sql attribute. |
| <sql:update> | Executes the SQL update defined in its body or through the sql attribute. |
| <sql:param> | Sets a parameter in an SQL statement to the specified value. |
| <sql:dateParam> | Sets a parameter in an SQL statement to the specified java.util.Date value. |
| <sql:transaction > | Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction. |

**SELECT Operation:**

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

<%@ page import="java.io.\*,java.util.\*,java.sql.\*"%>

<%@ page import="javax.servlet.http.\*,javax.servlet.\*" %>

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

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

<html>

<head>

<title>SELECT Operation</title>

</head>

<body>

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

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

<sql:query dataSource="${snapshot}" var="result">

SELECT \* from Employees;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Emp ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Age</th>

</tr>

<c:forEach var="row" items="${result.rows}">

<tr>

<td><c:out value="${row.id}"/></td>

<td><c:out value="${row.first}"/></td>

<td><c:out value="${row.last}"/></td>

<td><c:out value="${row.age}"/></td>

</tr>

</c:forEach>

</table>

</body>

</html>

**INSERT Operation:**

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

<%@ page import="java.io.\*,java.util.\*,java.sql.\*"%>

<%@ page import="javax.servlet.http.\*,javax.servlet.\*" %>

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

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

<html>

<head>

<title>JINSERT Operation</title>

</head>

<body>

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

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

<sql:update dataSource="${snapshot}" var="result">

INSERT INTO Employees VALUES (104, 2, 'Nuha', 'Ali');

</sql:update>

<sql:query dataSource="${snapshot}" var="result">

SELECT \* from Employees;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Emp ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Age</th>

</tr>

<c:forEach var="row" items="${result.rows}">

<tr>

<td><c:out value="${row.id}"/></td>

<td><c:out value="${row.first}"/></td>

<td><c:out value="${row.last}"/></td>

<td><c:out value="${row.age}"/></td>

</tr>

</c:forEach>

</table>

</body>

</html>

**DELETE Operation:**

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

<%@ page import="java.io.\*,java.util.\*,java.sql.\*"%>

<%@ page import="javax.servlet.http.\*,javax.servlet.\*" %>

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

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

<html>

<head>

<title>DELETE Operation</title>

</head>

<body>

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

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

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

<sql:update dataSource="${snapshot}" var="count">

DELETE FROM Employees WHERE Id = ?

<sql:param value="${empId}" />

</sql:update>

<sql:query dataSource="${snapshot}" var="result">

SELECT \* from Employees;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Emp ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Age</th>

</tr>

<c:forEach var="row" items="${result.rows}">

<tr>

<td><c:out value="${row.id}"/></td>

<td><c:out value="${row.first}"/></td>

<td><c:out value="${row.last}"/></td>

<td><c:out value="${row.age}"/></td>

</tr>

</c:forEach>

</table>

</body>

</html>

**UPDATE Operation:**

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

<%@ page import="java.io.\*,java.util.\*,java.sql.\*"%>

<%@ page import="javax.servlet.http.\*,javax.servlet.\*" %>

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

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

<html>

<head>

<title>DELETE Operation</title>

</head>

<body>

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

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

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

<sql:update dataSource="${snapshot}" var="count">

UPDATE Employees SET last = 'Ali' where empid=?

<sql:param value="${empId}" />

</sql:update>

<sql:query dataSource="${snapshot}" var="result">

SELECT \* from Employees;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Emp ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Age</th>

</tr>

<c:forEach var="row" items="${result.rows}">

<tr>

<td><c:out value="${row.id}"/></td>

<td><c:out value="${row.first}"/></td>

<td><c:out value="${row.last}"/></td>

<td><c:out value="${row.age}"/></td>

</tr>

</c:forEach>

</table>

</body>

</html>

**JSTL Functions:**

JSTL includes a number of standard functions, most of which are common string manipulation functions. Following is the syntax to include JSTL Functions library in your JSP:

<%@ taglib prefix="fn"

uri="http://java.sun.com/jsp/jstl/functions" %>

Following is the list of JSTL Functions:

|  |  |
| --- | --- |
| Function | **Description** |
| fn:contains() | Tests if an input string contains the specified substring. |
| fn:containsIgnoreCase() | Tests if an input string contains the specified substring in a case insensitive way.  boolean containsIgnoreCase(java.lang.String, java.lang.String) |
| fn:endsWith() | Tests if an input string ends with the specified suffix.  boolean endsWith(java.lang.String, java.lang.String) |
| fn:indexOf() | Returns the index withing a string of the first occurrence of a specified substring.  int indexOf(java.lang.String, java.lang.String) |
| fn:join() | Joins all elements of an array into a string.  String join (java.lang.String[], java.lang.String)  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  <html>  <head>  <title>Using JSTL Functions</title>  </head>  <body>  <c:set var="string1" value="This is first String."/>  <c:set var="string2" value="${fn:split(string1, ' ')}" />  <c:set var="string3" value="${fn:join(string2, '-')}" />  <p>Final String : ${string3}</p>  </body>  </html> |
| fn:length() | Returns the number of items in a collection, or the number of characters in a string.  int length(java.lang.Object)  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  <html>  <head>  <title>Using JSTL Functions</title>  </head>  <body>  <c:set var="string1" value="This is first String."/>  <c:set var="string2" value="This is second String." />  <p>Length of String (1) : ${fn:length(string1)}</p>  <p>Length of String (2) : ${fn:length(string2)}</p>  </body>  </html> |
| fn:replace() | Returns a string resulting from replacing in an input string all occurrences with a given string.  boolean replace(java.lang.String, java.lang.String, java.lang.String)  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  <html>  <head>  <title>Using JSTL Functions</title>  </head>  <body>  <c:set var="string1" value="This is first String."/>  <c:set var="string2" value="${fn:replace(string1,  'first', 'second')}" />  <p>Final String : ${string2}</p>  </body>  </html> |
| fn:split() | Splits a string into an array of substrings.  java.lang.String[] split(java.lang.String, java.lang.String) |
| fn:startsWith() | Tests if an input string starts with the specified prefix.  boolean startsWith(java.lang.String, java.lang.String)  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  <html>  <head>  <title>Using JSTL Functions</title>  </head>  <body>  <c:set var="string" value="Second: This is first String."/>  <c:if test="${fn:startsWith(string, 'First')}">  <p>String starts with First</p>  </c:if>  <br />  <c:if test="${fn:startsWith(string, 'Second')}">  <p>String starts with Second</p>  </c:if>  </body>  </html> |
| fn:substring() | Returns a subset of a string.  java.lang.String substring(java.lang.String, int, int)  **Example:**  Following is the example to explain the functionality of this function:  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  <html>  <head>  <title>Using JSTL Functions</title>  </head>  <body>  <c:set var="string1" value="This is first String."/>  <c:set var="string2" value="${fn:substring(string1, 5, 15)}" />  <p>Final sub string : ${string2}</p>  </body>  </html>  Output: is first S |
| fn:substringAfter() | Returns a subset of a string following a specific substring.  java.lang.String substringAfter(java.lang.String, java.lang.String)  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  <html>  <head>  <title>Using JSTL Functions</title>  </head>  <body>  <c:set var="string1" value="This is first String."/>  <c:set var="string2" value="${fn:substringAfter(string1, 'is')}" />  <p>Final sub string : ${string2}</p>  </body>  </html>  OUTPUT: is first String. |
| fn:substringBefore() | Returns a subset of a string before a specific substring.  java.lang.String substringBefore(java.lang.String,  java.lang.String)  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  <html>  <head>  <title>Using JSTL Functions</title>  </head>  <body>  <c:set var="string1" value="This is first String."/>  <c:set var="string2" value="${fn:substringBefore(string1,  'first')}" />  <p>Final sub string : ${string2}</p>  </body>  </html>  Final sub string : This is |
| fn:toLowerCase() | Converts all of the characters of a string to lower case.  java.lang.String toLowerCase(java.lang.String)  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  <html>  <head>  <title>Using JSTL Functions</title>  </head>  <body>  <c:set var="string1" value="This is first String."/>  <c:set var="string2" value="${fn:toLowerCase(string1)}" />  <p>Final string : ${string2}</p>  </body>  </html>  Final string : this is first string. |
| fn:toUpperCase() | Converts all of the characters of a string to upper case.  java.lang.String tolowercase(java.lang.String)  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  <html>  <head>  <title>Using JSTL Functions</title>  </head>  <body>  <c:set var="string1" value="This is first String."/>  <c:set var="string2" value="${fn:toUpperCase(string1)}" />  <p>Final string : ${string2}</p>  </body>  </html>  Final string : THIS IS FIRST STRING. |
| fn:trim() | Removes white spaces from both ends of a string.  java.lang.String trim(java.lang.String)  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  <html>  <head>  <title>Using JSTL Functions</title>  </head>  <body>  <c:set var="string1" value="This is first String "/>  <p>String (1) Length : ${fn:length(string1)}</p>  <c:set var="string2" value="${fn:trim(string1)}" />  <p>String (2) Length : ${fn:length(string2)}</p>  <p>Final string : ${string2}</p>  </body>  </html>  String (1) Length : 29  String (2) Length : 20  Final string : This is first String |

The fn:contains() function determines whether an input string contains a specified substring.

**Syntax:**

The fn:contains() function has following syntax:

boolean contains(java.lang.String, java.lang.String)

**Example:**

Following is the example to explain the functionality of this function:

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

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

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

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

<c:if test="${fn:contains(theString, 'test')}">

<p>Found test string<p>

</c:if>

<c:if test="${fn:contains(theString, 'TEST')}">

<p>Found TEST string<p>

</c:if>

</body>

</html>

This would produce following result:

Found test string

Example

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

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

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="This <abc> is second String.</abc>"/>

<p>Index (1) : ${fn:indexOf(string1, "first")}</p>

<p>Index (2) : ${fn:indexOf(string2, "second")}</p>

</body>

</html>

Formatting Tags

<fmt:formatNumber> To render numerical value with specific precision or format.

<fmt:parseNumber> Parses the string representation of a number, currency, or percentage.

<fmt:formatDate> Formats a date and/or time using the supplied styles and pattern.

<fmt:setLocale> Stores the given locale in the locale configuration variable.

<fmt:formatNumber> demo

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

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

<html>

<head>

<title>JSTL fmt:formatNumber Tag</title>

</head>

<body>

<h3>Number Format:</h3>

<c:set var = "balance" value = "120000.2309" />

<p>Formatted Number (1): <fmt:formatNumber value = "${balance}"

type = "currency"/></p>

<p>Formatted Number (2): <fmt:formatNumber type = "number"

maxIntegerDigits = "3" value = "${balance}" /></p>

<p>Formatted Number (3): <fmt:formatNumber type = "number"

maxFractionDigits = "3" value = "${balance}" /></p>

<p>Formatted Number (4): <fmt:formatNumber type = "number"

groupingUsed = "false" value = "${balance}" /></p>

<p>Formatted Number (5): <fmt:formatNumber type = "percent"

maxIntegerDigits="3" value = "${balance}" /></p>

<p>Formatted Number (6): <fmt:formatNumber type = "percent"

minFractionDigits = "10" value = "${balance}" /></p>

<p>Formatted Number (7): <fmt:formatNumber type = "percent"

maxIntegerDigits = "3" value = "${balance}" /></p>

<p>Formatted Number (8): <fmt:formatNumber type = "number"

pattern = "###.###E0" value = "${balance}" /></p>

<p>Currency in USA :

<fmt:setLocale value = "en\_US"/>

<fmt:formatNumber value = "${balance}" type = "currency"/>

</p>

</body>

</html>

Example – 2

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

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

<html>

<head>

<title>JSTL fmt:parseNumber Tag</title>

</head>

<body>

<h3>Number Parsing:</h3>

<c:set var = "balance" value = "1250003.350" />

<fmt:parseNumber var = "i" type = "number" value = "${balance}" />

<p>Parsed Number (1) : <c:out value = "${i}" /></p>

<fmt:parseNumber var = "i" integerOnly = "true"

type = "number" value = "${balance}" />

<p>Parsed Number (2) : <c:out value = "${i}" /></p>

</body>

</html>

fmt:dateNumber demo

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

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

<html>

<head>

<title>JSTL fmt:dateNumber Tag</title>

</head>

<body>

<h3>Number Format:</h3>

<c:set var = "now" value = "<% = new java.util.Date()%>" />

<p>Formatted Date (1): <fmt:formatDate type = "time"

value = "${now}" /></p>

<p>Formatted Date (2): <fmt:formatDate type = "date"

value = "${now}" /></p>

<p>Formatted Date (3): <fmt:formatDate type = "both"

value = "${now}" /></p>

<p>Formatted Date (4): <fmt:formatDate type = "both"

dateStyle = "short" timeStyle = "short" value = "${now}" /></p>

<p>Formatted Date (5): <fmt:formatDate type = "both"

dateStyle = "medium" timeStyle = "medium" value = "${now}" /></p>

<p>Formatted Date (6): <fmt:formatDate type = "both"

dateStyle = "long" timeStyle = "long" value = "${now}" /></p>

<p>Formatted Date (7): <fmt:formatDate pattern = "yyyy-MM-dd"

value = "${now}" /></p>

</body>

</html>

setLocale Demo

package com.sssit;

import java.util.ListResourceBundle;

public class Example\_es\_ES extends ListResourceBundle {

public Object[][] getContents() {

return contents;

}

static final Object[][] contents = {

{"count.one", " ఒకటి "},

{"count.two", " రెండు "},

{"count.three", " మూడు "},

};

}

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

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

<html>

<head>

<title>JSTL fmt:setLocale Tag</title>

</head>

<body>

<fmt:bundle basename = "com.sssit.Example">

<fmt:message key = "count.one"/><br/>

<fmt:message key = "count.two"/><br/>

<fmt:message key = "count.three"/><br/>

</fmt:bundle>

<!-- Change the Locale -->

<fmt:setLocale value = "es\_ES"/>

<fmt:bundle basename = "com.sssit.Example">

<fmt:message key = "count.one"/><br/>

<fmt:message key = "count.two"/><br/>

<fmt:message key = "count.three"/><br/>

</fmt:bundle>

</body>

</html>

Custom Tags:

Example 1

Step 1 ---- HelloTag.java

import javax.servlet.jsp.tagext.\*;

import javax.servlet.jsp.\*;

import java.io.\*;

public class HelloTag extends SimpleTagSupport {

public void doTag() throws JspException, IOException {

JspWriter out = getJspContext().getOut();

out.println("Hello Custom Tag!");

}

}

Step 2 ---- custom.tld(must be placed in web-inf folder)

<taglib>

<tlib-version>1.0</tlib-version>

<jsp-version>2.0</jsp-version>

<short-name>Example TLD</short-name>

<tag>

<name>Hello</name>

<tag-class>com.sssit.HelloTag</tag-class>

<body-content>empty</body-content>

</tag>

</taglib>

Step 3 ---- use in jsp page

<%@ taglib prefix="ex" uri="WEB-INF/custom.tld"%>

<html>

<head>

<title>A sample custom tag</title>

</head>

<body>

**<ex:Hello/>**

</body>

</html>

Output: Hello Custom Tag!