Adv. Java means DURGA SIR..

ADV.JAVA

With

SCWCD / OCWCD

JSP Material

4. Building JSP Pages using Tag Libraries(JSTL)

DURGA M.Tech

(Sun certified & Realtime Expert)

Ex. IBM Employee

Trained Lakhs of Students
for last 14 years across INDIA

Building JSP Pages Using Java Standard Tag Library (JSTL)

Agenda :

- Core Library (14 tags)
  1) General purpose tags :
     - <c:out>
     - <c:set>
     - <c:remove>
     - <c:catch>
     - Summary of General Purpose tags
  2) Conditional tags :
     - <c:if>
     - <c:when>
     - <c:choose>
     - <c:otherwise>
     - Summary of Conditional tags
  3) Iteration tags :
     - <c:forEach>
     - <c:forTokens>
     - Summary of Iteration Tags
  4) Re-directional tags : (URL related tags)
     - <c:url>
     - <c:redirect>
     - <c:import>
     - <c:param>
     - Summary of url tags
- Sql Tag Library
- XML Tag Library
- Formatting Tag Library
- Functional Tag Library

Introduction :

1. Sun people encapsulated the core functionality which is common to many web application in the form of JSTL, programmer can use this predefined library without writing on his own.
2. The main objective of EL is to eliminate java code from the jsp but it fails to replace java code complete elimination.
3. Which process some functionality , we can resolve this problem by using JSTL hence the main objective of JSTL is to remove java code from the Jsp.

Entire JSTL divided into 5 parts :

1. Core Library (14 tags)
   - It defines some standard actions to perform general programming like implementing condition and iterational statements.
   - It can also perform jsp fundamental tasks such as setting attributes, writing output to the jsp page and redirect in the request to other pages and etc.,
2. Sql Tag Library : It defines several standard actions it can be used for DataBase operations.
3. XML Tag Library : It defines several standard actions useful for writing and formatting XML data.
4. Formatting Tag Library : It defines several standard actions which are useful for I18Npurpose.
5. Functional Tag Library : It defines several standard actions which are used for manipulating for Collection and Objects.

Core Library :

JSTL core library divided into the following 4 parts

1) General purpose tags :
   - <c:out>
   - <c:set>
   - <c:remove>
   - <c:catch>

2) Conditional tags :
   - <c:if>
   - <c:when>
   - <c:choose>
   - <c:otherwise>

3) Iterational tags :
   - <c:forEach>
   - <c:forTokens>

4) Re-directional tags : (URL related tags)
   - <c:url>
   - <c:redirect>

- <c:import>
- <c:param>

Installing JSTL :

By default JSTL functionality is not available to the jsp, we can provide JSTL functionality by placing the following jar files in web-application lib folder.

1. jstl.jar : defines several classes and interfaces provided by Sun.
2. standard.jar : provide library implementation classes by vendor.

Note : The above 2 jar files we have to download from net and place it either in web-application lib folder or at server level lib folder(i.e., ex : tomcat/lib)

To make core library available to the jsp , we have to declare taglib directive as follows

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

OR

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

General Purpose tags :

<c:out>

We can use this tag for writing template text and expressions to the jsp

```
<c:out>

c --- prefix
out ---- tagname
```

form 1 :

```
<c:out value="ashok" />
```

It prints template text ashok to the Jsp

```
<c:out value="${param.uname}" />
        // runtime expressions
```

form 2 :

```
<c:out value="${param.uname}" default="ashok" />
```

- If the main value is not possible or if it is not then we can provide default value by using default attribute.

- If the specified request parameter 'uname' is not available then returns a null , in that case only default will get the cahance.

form 3 :

```
<c:out value="${result}" escapeXml="false" />
```

If the result contains xml data , if we want to xml data then take *escapeXml="false"* , if we don't want to xml data then take *escapeXml="true"* , in that case the result value considered template text.

Ex 1 :

```
<%@page isELIgnored="false" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:out value="ashok" />
The Entered name :
<c:out value="${param.uname}" default="charan" />
```

Ex 2 :

```
<%@page isELIgnored="false" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <head>
        <title>The attribute escapeXml in  c:out </title>
    </head>
    <body>
        <c:set  var="test" scope="session">
        We are going to use the attribute escapeXml of c:out to show the difference
        <table border="5">
            <tr><td bgcolor="green">Arun </td>
            <td bgcolor="yellow">Akshay </td>
            <td bgcolor="red">SaiChran </td></tr>
        </table>
</c:set>

<h1>out with escapeXml="false"</h1>
<c:out value="${test}" escapeXml="false" /><br>

<h1>out with escapeXml="true"</h1>
<c:out value="${test}" escapeXml="true" /><br>

</body>
</html>
```

<c:out> defines following 3 attributes :

1. value : It is a mandatory attribute to provide required value , it can be String literal or runtime expression.
2. default : It is an optional attribute and it is for providing default value, Jsp engine consider it's value if and only if the value attribute evaluates null.

3. escapeXml : It is an optional attribute the default value is true.
   "true" if the tag should escape special xml characters,
   "false" it process that xml data.

<c:set>

The <jsp:setProperty> tag can do only one thing set property of bean but

1) if we want to set a value in map (or)
2) if we want to make one entry in map (or)
3) if we simply want to create new request scope attribute.

We can get all these things by using <c:set>

<c:set> comes in 2 flavors var and target

1. var : The var version is for setting attribute variable.
2. target : The target version is for setting bean properties or map values.

Each of the 2 flavors comes in 2 variations with or without body.

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

<c:set  var="x" value="13"/>
<c:set  var="y" value="23"/>
<c:set var="result" scope="application" >
 ${x*y}
</c:set>

The result of the 2 numbers is :
<c:out value="${result}"/>
```

Ex :

```
<c:set var="fedo" value="${person.dog}" />
  // value -- need not be a String
```

Ex :

```
<c:set var="x" scope="session">
  ashok, arun
</c:set>
```

setting a target property or value with <c:set> :

Without body :

```
<c:set target="${map}" property="raja" value="SCWCD"/>
```

bean :

```
<c:set target="${person}" property="name">
 ${person.name}
</c:set>
```

Example 1 :

In Servlet Code :

```
package com.jstl;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ServletDemo extends HttpServlet {
        public void service(HttpServletRequest request,HttpServletResponse response)
                                throws ServletException,IOException{

                foo.Person person = new foo.Person();
                foo.Dog dog = new foo.Dog();

                dog.setName("spike");
                person.setDog(dog);

               System.out.println(person);

                request.setAttribute("person", person);

                RequestDispatcher rd = request.getRequestDispatcher("pages/demo.jsp");
                rd.forward(request, response);
        }

}
```

demo.jsp

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

<c:set target="${person}" property="name" value="Ashok"/>
<c:out value="${person.name}"/>
```

Example 2 :

```
package com.jstl;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ServletDemo extends HttpServlet {

    public void service(HttpServletRequest request,HttpServletResponse response)
                        throws ServletException,IOException {
            Map map = new HashMap();
            request.setAttribute("map", map);
            RequestDispatcher rd = request.getRequestDispatcher("pages/demo.jsp");
            rd.forward(request, response);
    }

}
```

demo.jsp

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

<c:set target="${map}" property="ashok" value="SCWCD"/>
<c:out value="${map}"/> <br/>
<c:out value="${map.ashok}"/>
```

web.xml

```
<web-app>
<servlet>
 <servlet-name>ServletPerson</servlet-name>
 <servlet-class>com.jstl.ServletDemo</servlet-class>
</servlet>

<servlet-mapping>
 <servlet-name>ServletPerson</servlet-name>
 <url-pattern>/fs</url-pattern>
</servlet-mapping>

</web-app>
```

output :

```
{ashok=SCWCD}
```

SCWCD

http://localhost:8080/jsp/fs

<c:set> conclusions :

- We can never have both var, target attributes <c:set> it will gives unpreductable results.
- Scope is an optional attribute, default scope is page.
- If the "value" is null, the attribute named by var will be removed.

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

<c:set var="x" value="${param.user}"/>
<c:out value="${x}"/>
 //Assume there is no request parameter user
 output : blankspace
```

- If the attribute var doesn't exist <c:set> will be created but if value is not null.
- If the target expression is null then container throws an Exception.
- If the target expression is not a map/bean then container throws an Exception.
- If the target expression is a bean but the bean doesn't have a property matches with property attribute then container throws an exception saying Invalid property in <set>.

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

<c:set target="${person}" property="name" value="Ask" scope="session"/>
<c:out value="${person.name}"/>
```

output :

```
org.apache.jasper.JasperException:
   Illegal scope attribute without var in "c:set" tag.
```

note : In <c:set> tag all attributes are optional , when ever we are taking scope attribute compulsory we can take var attribute otherwise we will get exception.

<c:remove>

We can use this tag to remove attribute in the specified scope this tag contains the following 2 attributes.

1. var : represents name of the attributes.
2. scope : represents the scope in which attribute present.

Ex :

```
<c:remove var="x" />
```

If the scope is not specified then JSP engine will search page scope for the specified attribute, if it is available then JSP engine will removes the attribute, If the specified attribute is not available it will search in request scope followed by session, application.

Note : The <c:remove> compulsary should be a var attribute but not expression.

```
<c:remove var="x"/>    //valid
<c:remove var="${x}"/>  //invalid
```

Note 2 : In <c:remove> tag var attribute is the mandatory attribute and scope attribute is a optional attribute.

Ex : remove.jsp

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

<c:set var="x" value="10"/>
<c:set var="y" value="20"/>
<c:set var="result" value="${x*y}" scope="application"/>
Before removal of result : <c:out value="${result}"/> //200
<br/>

<c:remove var="x"/>
<c:remove var="y"/>
removal of x & y : <c:out value="${result}"/> //200
<br/>

<c:remove var="result"/>
After removal of result is :
   <c:out value="${result}" default="8888"/> //8888
```

Output :

```
Before removal of result : 200 //200
removal of x & y : 200 //200
After removal of result is : 8888 //8888
```

<c:catch>

This can be used to catch and suppress that exception, so that the result of the code will be executed normally
We have to place risky code as a body of <c:catch> tag.

Syntax :

```
<c:catch>
 Risky Code
```

```
</c:catch>
```

- If an exception raised an risky code when this tag suppress that expression and rest of the code will be executed normally.
- We can hold the raised exception object by using var attribute, which is page scoped attribute.

Note : in <c:catch> var attribute is optional.

Ex 1 :

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

<c:catch var="e">  //page scope we can't access outside
 <% int age=Integer.parseInt("ten"); %>
 The Raised Exception : ${e}
</c:catch>
<c:if test="${e!=null}">
  <h1>OOPS! -- Exception Raised </h1> ${e}
</c:if>
```

output :

```
 //page scope we can't access outside
OOPS! -- Exception Raised

java.lang.NumberFormatException: For input string: "ten"
```

Ex 2 :

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

The UserName is : ${param.uname}
<c:catch var="e">
 <%int age=Integer.parseInt(request.getParameter("age"));%>
 The  Age is : ${param.age}
</c:catch>

<c:if test="${e!=null}">
 <h1>OOPS!--Exception Raised</h1> <h3>${e}</h3>
</c:if>

The Height is : ${param.height}
```

output :

```
http://localhost:8081/jstl/WebRoot/pages/jstl.jsp?uname=Ashok&age=ten&height=5.5

The UserName is : Ashok
OOPS!--Exception Raised
```

```
java.lang.NumberFormatException: For input string: "ten"

The Height is : 5.5
```

Ex 3 :

```
<c:catch var="e">
 ${Person.age}
</c:catch>

Raised Exception is : ${e}
```

age is not a property of Person hence it raises PropertyNotFoundException but <c:catch> handles that exception and continue rest of the JSP normally.

Summary of General Purpose tags :

Tag	Description	Attributes
<c:out>	For writing template text and expression to the JSP page.	value, default, escapeXml value is mandatory attribute
<c:set>	To set some attribute in some scope and to set bean property and add to entries in the Map.	var, target, value, scope, property
<c:remove>	To remove an attribute in the specified scope, if we are not mension any scope page followed request, session, application.	var, scope var is mandatory attribute
<c:catch>	For suppress an Exception and continue rest of the JSP normally.	var

Conditional Tags :

<c:if>

If we can use this tag to implement core java if statement , there are 2 forms are <c:if> available.

Without body :

```
<c:if test="testcondition" var="x" scope="session"/>
```

In this case test condition will be evaluated and result store into var x , If the rest of the Jsp page where ever the same test condition is required , we can use its directly without re-evaluated once again.

- In this case test attribute is mandatory.
  var, scope attributes are optional.
- But when ever the scope attribute is specified compulsory we should take var attribute.

With body :

```
<c:if test="testcondition" var="x" scope="session">
  ----------
  ----------
</c:if>
```

The test condition is true then the body will be executed otherwise without executing the body , the rest of the JSP will be executed.

- In this case also we can store test results into var variable.
- scope, var attributes are optional.

Ex :

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

<c:if test="${uname eq 'Ashok'}">
 <jsp:forward page='demo.jsp'/>
</c:if>
```

if.jsp

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

<h1>If Conditional Tag with Body</h1>
<c:set var="x" value="10" scope="request"/>
<c:if test="${x eq 10}" var="y" scope="session">
 x  value is : ${x}   <br/>
 The result is : ${y}
</c:if>
```

output :

```
If Conditional Tag with Body

x value is : 10
The result is : true
```

<c:choose>, <c:when>, <c:otherwise> we can use these tags for implements if-else , switchstatements.

implementing if-else :

JSTL doesn't contain any tag for else , we can implement if-else statement by using the above tags

```
<c:choose>
<c:when test="testcondition">
   //Action 1 (if)
</c:when>

<c:otherwise>
  //Action 2  (else)
</c:otherwise>
</c:choose>
```

If test condition is true Action 1 will be executed else Action 2 will be executed.

implementing switch statement :

```
<c:choose>
<c:when test="testCondition1">
   //Action 1
</c:when>
<c:when test="testCondition2">
   //Action 2
</c:when>

<c:otherwise>
  //default Action
</c:otherwise>
</c:choose>
```

- <c:choose> should compulsory contains atleast one <c:when> , but <c:otherwise> is optional.
- Every <c:when> implicitly contains break stastement hence there is no chance fall-through inside switch.
- We have to take <c:otherwise> as a last statement only.
- <c:choose> and <c:otherwise> won't take any attribute but <c:when> tag can contains only one mandatory attribute i.e., test.

Ex :

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

<html>
<body>
<form>
<b>Select the Number</b>
<select name="day">
 <option>1</option>
```

```
 <option>2</option>
 <option>3</option>
 <option>4</option>
 <option>5</option>
 <option>6</option>
 <option>7</option>
</select>
<br>
<input type="submit"/>
</form>

<c:set var="day" value="${param.day}"/>
Today is :
<c:choose>
 <c:when test="${day eq 1}">
  <c:out value="Sunday"/>
 </c:when>
 <c:when test="${day eq 2}">
  <c:out value="Monday"/>
 </c:when>
 <c:when test="${day eq 3}">
  <c:out value="Tuesday"/>
 </c:when>
 <c:when test="${day eq 4}">
  <c:out value="Wednesday"/>
 </c:when>
 <c:when test="${day eq 5}">
  <c:out value="Thursday"/>
 </c:when>
 <c:when test="${day eq 6}">
  <c:out value="Friday"/>
 </c:when>
 <c:when test="${day eq 7}">
  <c:out value="Saturday"/>
 </c:when>

 <c:otherwise>
  <c:out value="Select the values between 1 to 7"/>
 </c:otherwise>
</c:choose>


</body>
</html>
```

Summary of Conditional tags :

Tag	Description	Attributes
<c:if>	To implement core java if statement	test, var, scope test attribute is mandatory
<c:choose> <c:when> <c:otherwise>	To implement if-else and switch statements	<c:choose> and <c:otherwise> won't take any attributes, <c:when> take one attribute i.e., test.

Iteration Tags :

<c:forEach> tag :

<c:forEach> to implement general purpose for loop.

form 1 :

```
<c:forEach begin="0" end="11" step="1">
 Learning Jstl (11 times)
</c:forEach>
Here
```

- This loop internally maintain one couter variable , which is incremented by step attribute value.
- The default value for the step attribute is "1" , and it is optional attribute.

form 2 : <c:forEach> with var attribute

```
<c:forEach begin="0" end="11" var="count" step="2">
 ${count}
</c:forEach>

output :
0 2 4 6 8 10
```

form 3 : </c:forEach> with items attribute

```
<c:forEach items="${Array/Collections}" var="obj">
.....
</c:forEach>
```

- items attribute should contains either Collection Obj (OR) Arrays.
- This action will integrate over each item in the collection untill all the elements.
- We can represents current collection obj by using var attribute.

Types of items attribute	Types of var attribute
primitive array Ex : int[]	Corresponding wrapper class Integer

Collection	java.lang.Collection
Map	Map.entry
Object Array Student[]	Corresponding object class type Student
List of String seperated by ","	String

Ex :

```
<%@page isELIgnored="false" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%
      String[]  s = {"A","B","C","D"};
      session.setAttribute("s", s);
%>

<c:forEach items="${s}" var="obj">
 The Current object is : ${obj}  <br>
</c:forEach>

 output :
The Current object is : A
The Current object is : B
The Current object is : C
The Current object is : D
```

header.jsp

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

<table>
<tr><th>Header name</th><th>Header value</th></tr>
<tr><td><c:forEach items="${header}" var="x"></td></tr>
 <tr><td>${x.key}</td><td>${x.value}</td></tr>
<tr><td></c:forEach></td></tr>
</table>
```

cookie.jsp

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

<%
      Cookie c1 = new Cookie("uname", "Agastya");
      Cookie c2 = new Cookie("mail", "info@jobs4times.com");
      Cookie c3 = new Cookie("mobile", "9822334455");
```

```
        Cookie c4 = new Cookie("address", "IND");
        response.addCookie(c1);
        response.addCookie(c2);
        response.addCookie(c3);
        response.addCookie(c4);
%>
<c:forEach items = "${cookie}" var="x">
      <h2>${x.value.name} ---- ${x.value.value}</h2>
</c:forEach>

output :
address ---- IND
uname ---- Agastya
mail ---- info@jobs4times.com
mobile ---- 9822334455
JSESSIONID ---- 62a82c98e954f45a4f5967a745ff
```

Write a program to print all the session scoped attributes (attribute names and attribute values)

in Servlet code :

```
package info;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class ServletDemo extends HttpServlet {
        private static final long serialVersionUID = 1L;

        protected void processRequest(HttpServletRequest request, HttpServletResponse response)
                                                        throws ServletException, IOException {
                HttpSession session = request.getSession();
                session.setAttribute("Ashok", "SCJP");
                session.setAttribute("Arun", "SCWCD");

                RequestDispatcher rd = request.getRequestDispatcher("myJsp.jsp");
                rd.forward(request, response);
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
                                                        throws ServletException, IOException {
              processRequest(request, response);
}
```

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
                                                    throws ServletException, IOException {
            processRequest(request, response);
  }
}
```

myJsp.jsp

```
<%@page isELIgnored="false" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:forEach items="${sessionScope}" var="obj">
 <h2>${obj.key} ---- ${obj.value}</h2>
</c:forEach>
```

output :

```
Arun ---- SCWCD
Ashok ---- SCJP
```

Example :

In Servlet Code :

```
String[] movies1={"A","B","C"};
String[] movies2={"MovieA","MovieB","MovieC"};

java.util.ArrayList list=new java.util.ArrayList();
list.add(movies1);
list.add(movies2);

request.setAttribute("moviesList", list);

 RequestDispatcher rd=request.getRequestDispatcher("myJsp.jsp");
 rd.forward(request, response);
```

myJsp.jsp

```
<c:forEach items="${moviesList}" var="listElements">
 <c:forEach items="${listElements}" var="movie">
  ${movie}
 </c:forEach>
</c:forEach>
```

output :

```
A B C MovieA MovieB MovieC
```

form 4 : <c:forEach> with varStatus attribute

- This attribute dicuss status of the iteration like current iteration number is 1st iteration or not.
- This attribute is the type of javax.servlet.jsp.jstl.core.LoopTagStatus
- This class contains several methods, which are useful during iterations.

public Object getCurrent() :	it returns the current item in the iteration.
public int getIndex() :	returns current index
public int getCount() :	returns the no. of iterations that have already perform including current iteration.
public boolean isFirst() :	returns information about whether the current iteration is first , then it returns "true" else returns "false"
public boolean isLast() :	returns information about whether the current iteration is last , then it returns "true"
public Integer getBegin() :	returns the value of begin attribute for the associate tag, (OR) null if no begin attribute is specified.
public Integer getEnd() :	returns the value of end attribute for the associate tag, (OR) null if no end attribute is specified.
public Integer getStep() :	returns the value of step attribute for the associate tag, (OR) null if no step attribute is specified. (i.e., there is no default value)

myJsp.jsp

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

<c:forEach items="Sai,Shiva,Vishnu" varStatus="status">
 The current Item : ${status.current} <br>
 The index : ${status.index} <br>
 The count : ${status.count} <br>
 Is it first Item ? ${status.first} <br>
 Is it last Item ? ${status.last} <br>
 The begin Item : ${status.begin} <br>
```

```
 The end Item : ${status.end} <br>
 The step Item : ${status.step} <br>
</c:forEach>
```

output :

```
The current Item : Sai
The index : 0
The count : 1
Is it first Item ? true
Is it last Item ? false
The begin Item :
The end Item :
The step Item :
The current Item : Shiva
The index : 1
The count : 2
Is it first Item ? false
Is it last Item ? false
The begin Item :
The end Item :
The step Item :
The current Item : Vishnu
The index : 2
The count : 3
Is it first Item ? false
Is it last Item ? true
The begin Item :
The end Item :
The step Item :
```

Example 2 :

myJsp.jsp

```
<c:forEach begin="0" end="10" step="1" varStatus="status">
 The begin Item : ${status.begin} <br>
 The end Item : ${status.end} <br>
 The step Item : ${status.step} <br>
</c:forEach>
```

output :

```
The begin Item : 0
The end Item : 10  //11 times
The step Item : 1
```

<c:forTokens> :

- It is specialized version of forEach to perform String tokenization based on some delimeter(seperator).

- It behaves exactly same as StringTokenizer in core Java.

```
<c:forTokens items="ask,sai,raja,raki" delims=";" var="obj">
.............
</c:forTokens>
```

For each token according to the seperator, the body will be executed we can store the current Token by using var attribute.

Ex :

myJsp.jsp

```
<c:forTokens items="ask,sai" delims="," var="obj">
 The current object : ${obj}  <br>
</c:forTokens>

output :
The current object : ask
The current object : sai
```

<c:forTokens> can take the following attributes :

begin :	specifies the index where iteration should start the index of first token is zero.
end :	specifies the index where iteration should terminates.
step :	counter increments value between iterations.
varStatus :	It specifies status of the iteration like it is a first iteratin or not.

Ex :

```
<c:forTokens items="ask,sai,raja,raki" delims=";" >
No one is Good !!!
</c:forTokens>
```

Ex :

```
<c:forTokens items="ask,sai,raki" delims=";" varStatus="status">
 Is it first element ? ${status.first} <br>
Is it last element ? ${status.last} <br>
The begin index : ${status.begin} <br>
The end index : ${status.end} <br>
The step index : ${status.step} <br>
No one is Good !!!
</c:forTokens>
output :
Is it first element ? true
Is it last element ? true
```

```
The begin index :
The end index :
The step index :
No one is Good !!!
```

- In case of <c:forTokens> items attribute should be String only but in the case of<c:forEach> items can be "Collection/Array, Map or String".
- Hence <c:forTokens> is considered as a specialized version of forEach loop.

Ex :

The combination of varStatus, var is allowed.

```
<c:forTokens items="ask,sai,raki" delims="," varStatus="status" var="x">
The Current Element : ${status.current} -- ${x} <br>
</c:forTokens>

output :
The Current Element : ask -- ask
The Current Element : sai -- sai
The Current Element : raki -- raki
```

Summary of Iteration Tags :

Tags	description	attribute
<c:forEach>	general puepose for loop and enhanced for loop	begin, end, items, step, var, varStatus

begin, end ---> These are mandatory in the case of normal for loops.
items ---> This is mandatory in the case of enhanced for loops.
(According to jsp specification all attributes are optional)

<c:forTokens>	Specialized version of StringTokenization	begin, end, step, var, varStatus, items, delims

items, delims ---> These are mandatory attributes.

URL related tags :

<c:import>

- By using <c:import> to include the response of some other JSP into Current JSP at request processing time,
  Hence this inclusion is called Dynamic Include.
  It is exactly equal to <jsp:include> standard action.
- <jsp:include> and include directive applicable with in the same server/container
  but <c:import> can be applicable either with in the same server or outside of the server.
  It is always recommended to use outside of the web server.

```
<jsp:include page="http://localhost:8080/jstl/myJsp.jsp"/>//invalid
<%@include file="http://localhost:8080/jstl/myJsp.jsp"%>//invalid
<c:import url="http://localhost:8080/jstl/myJsp.jsp"/>  //valid
```

form 1 :

demo.jsp

```
Hello Demo Jsp World!
```

myJsp.jsp

```
Hello, this is from myJsp.jsp on GlassFish server.
<c:import url="http://localhost:8080/jstl/demo.jsp"/>

output :
Hello, this is from myJsp.jsp on GlassFish server.
Hello Demo Jsp World!
```

form 2 :

myJsp.jsp

```
Hello, this is from myJsp.jsp on GlassFish server.
<c:import url="/demo.jsp" context="/jstl" />
  // absolute paths
```

We can import the resources from outside of current application also
(i.e., cross context communication also possible)

form 3 :

We can store the result of imported page into a variable specified by var attribute,
Where ever the rest of the JSP, we can use directly that variable without import once again.

myJsp.jsp

```
Hello,
<c:import url="/demo.jsp" var="result" scope="session"/>
The result is :${result}

output :
Hello, The result is :
Hello Demo Jsp World!
```

Whenever we are using var attribute the result of target JSP store into var attribute , if we want that result we have to retrieve from that var attribute.

form 4 :

The more convinient way to store the result of <c:import> is to use Reader object, it is alternative to var attribute.
Hence var and varReader should not come symultaneously.

```
Hello,
```

```
<c:import url="demo.jsp" varReader="myReader"/>

<%
java.io.Reader myReader=(java.io.Reader)pageContext.getAttribute("myReader");
int i=myReader.readLine();
write(i!=null){
 //you can perform your own operations.
 //once checked again
}
%>
```

form 5 :

While performing import we can send parameters to the target jsp for this we should use <c:param> tag these parameters are available in the target Jsp in the form of request parameters (or) form parameters. demo.jsp

```
Hello Demo Jsp World! <br>

The form parameter is : ${param.c1} <br>
The form parameter is : ${param.c2}
```

myJsp.jsp

```
Hello, <br>
<c:import url="http://localhost:8080/jstl/demo.jsp">
 <c:param name="c1" value="SCJP"/>
 <c:param name="c2" value="SCWCD"/>
</c:import>

output :
Hello,
Hello Demo Jsp World!
The form parameter is : SCJP
The form parameter is : SCWCD
```

<c:redirect> :

We can use this tag to redirect the request to another page, it is exactly equal to sendRedirect of ServletResponse.

form 1 :

```
Hello, <br>
<c:redirect url="/demo.jsp" />
      // here absolute path is optional

output :
Hello Demo Jsp World!
```

form 2 :

```
<c:redirect url="/demo.jsp" context="/jstl" />
```

We can redirect request to some other web application resources also.

form 3 :

While performing redirection we can pass parameters of target resources for this we have to use <c:param> tag.

demo.jsp

```
Hello Demo Jsp World! <br>
The form parameter is : ${param.c1} <br>
The form parameter is : ${param.c2}
```

myJSP.jsp

```
Hello, <br>
<c:redirect url="/demo.jsp" context="/jstl">
 <param name="c1" value="SCJP"/>
 <param name="c2" value="SCWCD"/>
</c:redirect>
```

output :

```
Hello Demo Jsp World!
The form parameter is :
The form parameter is :
```

<c:url> :

We can use this tag to rewrite the url by appending the session information and form parameters to the URL.

In servlet code :

```
PrintWriter out=response.getWriter();
HttpSession session=request.getSession();
out.println(" <a href=\" "+
      response.encodeURL("test.do")+
       " \">  click </a> ");
```

In Jsp :

```
<a href="<c:url value="demo.do" />">click me</a>
```

form 1 :

```
<c:url value="demo.jsp" var="x"/>
<a href="${x}">Click Me</a>
```

form 2 :

```
<c:url value="/demo.jsp" var="x" context="/jstl" scope="session" />
```

form 3 :

```
<c:url value="demo.jsp" var="x" scope="request">
 <c:param name="c1" value="SCJP"/>
 <c:param name="c2" value="SCWCD"/>
</c:url>
```

- <c:url> tag rewrite the value of var attribute by appending session id, iff cookies are disabled and store into the var attribute.
- Suppose if we are not disable in the cookie <c:url> won't append the session id to the url.

```
<c:url value="demo.jsp" var="x">
 <c:param name="c1" value="SCJP"/><br>
 <c:param name="c2" value="SCWCD"/> <br>
</c:url>

 The value of x : ${x}  <br>
<a href="${x}">Click Me</a>


output :
The value of x : demo.jsp?c1=SCJP&c2=SCWCD
Click Me

Hello Demo Jsp World!
The form parameter is : SCJP
The form parameter is : SCWCD
```

- URL-encoding means replacing the unsafe (or) reserved characters with other characters and the whole thing is decoded again on the server side.
- Ex : spaces are not allowed in URL but we can substitute "+" sign for the spaces.
- The problem in <c:url> is doesn't automatically encode your URLs.
- We can encode the URLs by using <c:param> tag
- <c:url> tag can do only rewrite the URL but not encode the URL.

Using <c:url> with a query string :

myJsp.jsp

```
<c:set var="first" value="Ashok"/>
<c:set var="last" value="Agg"/>

<a href="<c:url value='demo.jsp?first=${first}&last=${last}' var='x'/>">
                        Click Me</a> <br>
The URL using param is :${x} <br>

Using Param tag in the body url-rewriting and Url-encoding<br>
<c:url value="demo.jsp" var="y">
```

```
 <c:param name="first" value="${first}"/><br>
 <c:param name="last" value="${last}"/> <br>
</c:url>

 The value of y : ${y}  <br>
<a href="${y}">Click Me</a>
```

demo.jsp

```
Hello Demo Jsp World! <br>
The form parameter is : ${param.first} <br>
The form parameter is : ${param.last}
```

output :

```
Click Me
The URL using param is : demo.jsp?first=Ashok&last=Agg
Using Param tag in the body url-rewriting and Url-encoding
The value of y : demo.jsp?first=Ashok&last=Agg
Click Me

Hello Demo Jsp World!
The form parameter is : Ashok
The form parameter is : Agg
```

Summary of url tags :

Tag	Description	Attributes
<c:import>	For importing the response of other page into current page into request processing time.	url, var, scope, varReader, context, charEncoding
<c:redirect>	To redirect the request to other web componants.	url, context
<c:url>	To rewrite url by appending session information and parameters.	value, var, scope, context
<c:param>	To send the parameters while implementing and Redirecting.	name, value