JSTL

JSP Standard Tag Library (JSTL) encapsula en etiquetas simples funcionalidades básicas de muchas aplicaciones Web. Por ejemplo, en lugar de iterar sobre listas usando un scriptlet, JSTL cuenta con etiquetas que permiten iterar de la misma forma en todas partes. Esta estandarización posibilita aprender una sola etiqueta y utilizarla en múltiples contenedores JSP. Además, cuando las etiquetas son estándar, los contenedores pueden reconocer y optimizar sus implementaciones.

JSTL es una colección de cuatro librerías de tags:

- *Core*: Acciones de propósito generales, tales como iteraciones y condicionales.
- *XML*: Manipulación de documentos XML.
- *FMT*: Internacionalización y formato de configuración regional como moneda y fechas.
- *SQL*: Acceso a base de datos relacionales.

Debido a que JSTL 1.0 fue liberado antes que JSP 2.0, este soportaba completamente los contenedores JSP 1.2 y no EL. Por ello, para soportar ambos mundos, scripting y EL, se crearon un conjunto de librerías gemelas. Los URI de las dos librerías son similares, pero a las librerías de runtime se le agregaba "_rt" tanto al URI como al PREFIX. Por ello, fue posible mezclar el uso de acciones runtime y EL.
A partir, JSTL 1.1 se incorporó extensamente el uso de EL.
En la actualidad, la última versión estable de JSP es la 2.1 y de JSTL es la 1.2.

(1) Tag Libraries:

Librería	URI	PREFIX
Core	http://java.sun.com/jsp/jstl/core	c
XML	http://java.sun.com/jsp/jstl/xml	x
FMT	http://java.sun.com/jsp/jstl/fmt	fmt
SQL	http://java.sun.com/jsp/jstl/sql	sql

(2) Runtime Tag Libraries (Solo si se trabaja con JSTL 1.0):

Librería	URI	PREFIX
Core	http://java.sun.com/jstl/core_rt	c_rt
XML	http://java.sun.com/jstl/xml_rt	x_rt
FMT	http://java.sun.com/jstl/fmt_rt	fmt_rt
SQL	http://java.sun.com/jstl/sql_rt	sql_rt

Core Tag Library

La librería JSTL Core cubre las siguientes áreas funcionales:

- Manipulación de variables
- Condicionales
- Looping e iteraciones
- Manipulación de URL

Manipulación de variables

Sintaxis 1
<c:out value="expression" Out-Specification />

Sintaxis 2
<c:out value="expression" Out-Specification>
Out-Body
</c:out>

value: Expresión que se mostrará. Es equivalente a <%= expresión %>

Out-Specification: Atributos adicionales, entre los que encontramos:

Atributo	Descripción
default="expression" o Out-Body	Expresión que se mostrará por defecto en el caso de que el atributo "value" sea una cadena vacía o nula.
escapeXml="boolean"	Indica si la expresión que se visualiza debe ser tratada o no como HTML o XML. En el caso de que el valor sea "false", se considera que la expresión es HTML o XML por lo que será interpretada por parte del navegador, de lo contrario, el valor expresado se muestra en

	pantalla.

Cuando el nombre de una variable aparece en una expresión, la variable se busca en el siguiente orden:

(1) page
(2) request
(3) session
(4) application

Si la variable no es encontrada, no se retorna nada.

Para acceder a una variable dentro de un alcance en particular, procedemos de la siguiente manera:

```
${pageScope.variable}
${requestScope.variable}
${sessionScope.variable}
${applicationScope.variable}
```

Veamos algunos ejemplos sencillos,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Ejemplo</title>
   </head>
   <body>
    <c:out value="¡Hola Mundo!! :)"/>
    <br>
    <c:out value="${producto.id}" default="Sin definir"/>
    <br>
    <c:out value="${param.nombre}">
     No informado
    </c:out>
    <br>
    <c:out value="<b>Texto en negrita</b>" default="Sin valor" escapeXml="false"/>
    <br>
    <c:out value="<b>Texto en negrita</b>" default="Sin valor" escapeXml="true"/>
   </body>
</html>
```

Sintaxis 1
<c:set var="name" value="value" scope="Var-Scope" />

Sintaxis 2
<c:set var="name" scope="Var-Scope">
value
</c:set>

Establece el valor de una variable en un ámbito en particular.

var: Nombre de la variable a definir.

value: Valor asignado a la variable.

scope: Ámbito de visibilidad de la variable, entre ellos:

Ámbito	Duración
page	La variable solo será válida dentro de la página JSP y será regenerada en cada nueva petición.
request	La variable será válida a lo largo de la petición y estará disponible para incluirse o reenviarse

	a otras páginas JSP.
session	La variable se asocia a una sesión en particular, la cual es responsable de su creación. La variable será válida mientras la sesión exista.
application	La variable es común a todas las sesiones y será válida hasta que la aplicación Web finalice.

Veamos un ejemplo sencillo,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Ejemplo</title>
   </head>
   <body>
    <c:set var="browser" value="${header['User-Agent']}" scope="session" />
    <p>Navegador usado: <c:out value="${browser}"/></p>
   </body>
</html>
```

Veamos como declarar una variable con un valor por defecto,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Ejemplo</title>
   </head>
   <body>
        <c:set var="sessionid" scope="session">
          <c:out value="${cookie['JSESSIONID'].value}" default="---"/>
        </c:set>
        <p>Identificador de Sesi&oacute;n: ${sessionid}</p>
   </body>
</html>
```

<c:remove var="name" scope="Var-Scope">

Remueve una variable de un determinado alcance.

var: Nombre de la variable a eliminar.

scope: Ámbito en que se elimina la variable. Este puede ser page, request, session o application.

Tomando como base un de los ejemplos antes planteados,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Ejemplo</title>
   </head>
```

```
  <body>
    <c:set var="browser" value="${header['User-Agent']}" scope="session" />
    <p>Navegador usado: <c:out value="${browser}"/></p>
    <c:remove var="browser" scope="session"/>
    <p>Variable removida</p>
    <p>Navegador usado: <c:out value="${browser}"/></p>
  </body>
</html>
```

```
<c:catch var="name">
   catch-body
</c:catch>
```

Provee un mecanismo para capturar cualquier excepción java.lang.Throwable que puede ocurrir durante la ejecución de cualquier acción anidada.

var: Nombre de la variable que se define al ejecutar la etiqueta.

Condicionales

```
<c:if test="expression" var="name" scope="scope">
  If-Body
</c:if>
```

var: Nombre de la variable que contiene el retorno de la evaluación de la condición.

scope: Ámbito en el que es válida la variable. Para más detalles ver la etiqueta <c:set>

test: Expresión a evaluar. Si el resultado de evaluar la expresión es "true", el contenido del cuerpo será procesado en forma estándar por el contenedor JSP y cualquier salida será retornada al corriente JspWriter.

Veamos un ejemplo,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core " prefix="c" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Ejemplo</title>
   </head>
   <body>
      <h1>Validador</h1>
      <form action="index.jsp" method="post">
         <p>Ingresa un valor: <input type="text" name="nro" maxlength="10" size="7"/></p>
         <input type="submit" value="Procesar"/>
      </form>

      <c:set var="i" value="${param.nro}" scope="request" />
      <c:catch var="exception">
         <c:if test="${!empty i}">
            <c:if test="${i mod 2 eq 0}">
               <p>El valor ingresado es par</p>
            </c:if>
            <c:if test="${i mod 2 ne 0}">
               <p>El valor ingresado es impar</p>
            </c:if>
         </c:if>
      </c:catch>
      <c:if test="${!empty exception}">
         <p>El valor ingresado debe ser un número</p>
      </c:if>
   </body>
```

```
</html>
```

```
<c:choose>
  <c:when test="expression">
   When-Body 1
  </c:when>
  <c:when test=" expression">
   When-Body 2
  </c:when>
  ...
  <c:when test=" expression">
   When-Body n
  </c:when>
  <c:otherwise>
   Otherwise-Body
  </c:otherwise>
</c:choose>
```

Un número ilimitado de <c:when> puede existir en una acción <c:choose> pero solo puede haber una única etiqueta <c:otherwise>

Cuando se remplazan varios <c:if> por <c:choose> podemos notar:

(1) El código es más legible y autodescriptivo para una persona que no se dedica a programar.
(2) El código es más eficiente ya que una vez que una expresión es evaluada y resulta verdadera, no continúa evaluando todas las otras expresiones que siguen a continuación.
(3) Con el empleo de <c:choose> nos aseguramos que las acciones son mutuamente excluyentes, mientras que un error en la lógica de los if, puede permitir ejecutar dos o más condiciones en forma simultánea.

Veamos el ejemplo anterior,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core " prefix="c" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Ejemplo</title>
   </head>
   <body>
      <h1>Validador</h1>
      <form action="index.jsp" method="post">
         <p>Ingresa un valor: <input type="text" name="nro" maxlength="10" size="7"/></p>
         <input type="submit" value="Procesar"/>
      </form>

      <c:set var="i" value="${param.nro}" scope="request" />
      <c:catch var="exception">
         <c:if test="${!empty i}">
            <c:choose>
               <c:when test="${i mod 2 eq 0}">
                  <p>El valor ingresado es par</p>
               </c:when>
               <c:otherwise>
                  <p>El valor ingresado es impar</p>
               </c:otherwise>
            </c:choose>
         </c:if>
      </c:catch>
      <c:if test="${!empty exception}">
         <p>El valor ingresado debe ser un número</p>
      </c:if>
   </body>
</html>
```

Looping e iteraciones

Sintaxis 1: Iteración sobre una colección de objetos

```
<c:forEach [var="varName"] [varStatus="varStatusName"] items="collection" [begin="begin"]
[end="end"] [step="step"]>
  forEach-body
</c:forEach>
```

Sintaxis 2: Iteración de un número fijo de veces

```
<c:forEach [var="varName"] [varStatus="varStatusName"] begin="begin" end="end" [step="step"]>
  forEach-body
</c:forEach>
```

var: Nombre de la variable que contiene el valor actual de la iteración.

varStatus: Nombre de la variable que representa el estado de la iteración. Contiene información acerca de la iteración en curso. Sus propiedades son:

- index: Índice del actual ítem en la iteración
- count: Posición del ciclo de iteración
- first: Determina si el ciclo actual es el primero
- last: Determina si el ciclo actual es el último

items: Puede ser un objeto de uno de los siguiente tipos:

- Un vector
- Una implementación de java.util.Collection
- Una implementación de java.util.Iterator
- Una implementación de java.util.Enumeration
- Una implementación de java.util.Map
- Una cadena de valores separados por coma

Si es nulo ninguna iteración es llevada a cabo

begin: Si es especificado, debe ser un valor mayor o igual a cero. Si este se informa, la iteración comienza en el ítem que coincide con el valor especificado, de lo contrario, comienza en cero.

end: Si es especificado, debe ser mayor o igual al valor del atributo "begin". Si este se informa, la iteración termina cuando la posición del ítem tratado coincide.

step: Si es especificado, debe ser mayor o igual a uno. El valor por defecto es uno. Representa el incremento aplicado en cada iteración.

Veamos algunos ejemplos,

Ejemplo 1

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Ejemplo</title>
   </head>
   <body>
      <h1>Libros</h1>
      <form action="index.jsp" method="post">
         <input type="checkbox" id="bookA" name="books" value="A"/><label for="bookA">Libro A</label>
         <input type="checkbox" id="bookB" name="books" value="B"/><label for="bookB">Libro B</label>
         <input type="checkbox" id="bookC" name="books" value="C"/><label for="bookC">Libro C</label>
         <input type="checkbox" id="bookD" name="books" value="D"/><label for="bookD">Libro D</label>
         <br>
         <input type="submit" value="Mostrar"/>
```

```
    </form>

    <c:set var="bo" value="${paramValues.books}"/>
    <c:if test="${!empty bo}">
      <ol>
       <c:forEach items="${bo}" var="b">
         <c:if test="${!empty b}">
           <li>${b}</li>
         </c:if>
       </c:forEach>
      </ol>
    </c:if>
  </body>
</html>
```

Ejemplo 2

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ejemplo</title>
  </head>
  <body>
    <h1>Múltiplos</h1>
    <table border="1" cellspacing="0" cellpadding="1">
      <c:forEach var="row" begin="1" end="10">
        <tr>
          <c:forEach var="col" begin="1" end="10">
            <td>${row * col}</td>
          </c:forEach>
        </tr>
      </c:forEach>
    </table>
  </body>
</html>
```

<c:forTokens [var="varName"] [varStatus="varStatusName"] items="stringOfTokens" delims="delimiters" [begin="begin"] [end="end"] [step="step"]>
** forTokens-body**
</c:forTokens>

Interactúa sobre una cadena separada por un conjunto de delimitadores.

var: Nombre de la variable que contiene el valor actual de la iteración.

varStatus: Nombre de la variable que representa el estado de la iteración. Contiene información acerca de la iteración en curso. Sus propiedades son:

- index: Índice del actual ítem en la iteración
- count: Posición del ciclo de iteración
- first: Determina si el ciclo actual es el primero
- last: Determina si el ciclo actual es el último

items: Cadena a tratar. Atributo obligatorio.

delims: Separador utilizado en la cadena. Atributo obligatorio.

begin: Si es especificado, debe ser un valor mayor o igual a cero. Si este se informa, la iteración comienza en el ítem que coincide con el valor especificado, de lo contrario, comienza en cero.

end: Si es especificado, debe ser mayor o igual al valor del atributo "begin". Si este se informa, la iteración termina cuando la posición del ítem tratado coincide.

step: Si es especificado, debe ser mayor o igual a uno. El valor por defecto es uno. Representa el incremento aplicado en cada iteración.

Veamos un ejemplo sencillo,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Ejemplo</title>
   </head>
   <body>
      <ul>
         <c:forTokens items="a;b;c;d;e;f" delims=";" var="current" begin="2" end="4">
          <li>${current}</li>
         </c:forTokens>
      </ul>
   </body>
</html>
```

Manipulación de URL

<c:import var="name" url="url" />

Importa el contenido de un recurso basado en un URL. Reemplaza a <jsp:include>.
La etiqueta <jsp:include> permite incluir archivos solo desde la aplicación Web actual, mientras que esta etiqueta permite incluir archivos desde otros sitios Web.

<c:url value="url" />

Provee una forma sencilla de construir correctamente una URL aplicando las reglas de reescritura.

<c:redirect url="url" />

Envía una redirección al cliente.

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

Permite definir parámetros en <c:import>, <c:url> o <c:redirect>, usando la siguiente sintaxis:

```
<c:url value="url" >
  <c:param name="param1">value1</c:param>
  <c:param name="param2">value2</c:param>
</c:url>
```