Librería Estándar de Etiquetas JSP (JSTL – *JSP Standard Tag Library*)

La Librería Estándar de Etiquetas JSP (JSTL) es una especificación complementaria que amplía el conjunto básico de etiquetas JSP. La especificación 1.0 de la JSTL incluye:

- Lógica de iteración y condicionales.
- Lenguaje de Expresiones (EL).
- Tratamiento de URL.
- Internacionalización.
- Tratamiento XML.
- Acceso a Bases de Datos.

Instalar la JSTL

Se pueden descargar los ficheros de instalación desde http://jakarta.apache.org pero si estamos trabajando con Tomcat, es suficiente con copiar los ficheros `standard.jar` y `jstl.jar` que podemos encontrar en el directorio `/jsp-examples/WEB-INF/lib` al directorio `/WEB-INF/lib` de nuestra aplicación Web.

Versiones de la librería JSTL

Existen dos versiones principales de la librería JSTL la 1.0 y la 1.1. Es importante hacer esta distinción porque la forma de declararlas para su utilización difiere bastante.
En la versión 1.1 sólo existe una versión de la librería y se declara con la uri: `http://java.sun.com/jsp/jstl/XXXX` dónde XXXX es el conjunto de etiquetas que queremos utilizar.
En la versión 1.0 existen dos versiones de la librería: una para trabajar con EL y otra para trabajar con expresiones en tiempo de ejecución y se declaran con la uri: `http://java.sun.com/jstl/XXXX` dónde XXXX es el conjunto de etiquetas que queremos utilizar para trabajar con EL y uri: `http://java.sun.com/jstl/XXXX_rt` dónde XXXX es el conjunto de etiquetas que queremos utilizar para trabajar con expresiones en tiempo de ejecución.

Etiquetas del núcleo (*Core Tags*)

Etiquetas de propósito general

<c:out>

La etiqueta **out** toma una expresión JSTL la evalúa y la envía a la salida.

JSTLout.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>&lt;c:out&gt; Ejemplo de Accion</title>
</head>
<body>
<c:out value="${'<tag> , &'}"/>
</body>
</html>
```

<c:set>

La etiqueta **set** es una versión de la acción **setProperty**. Permite evaluar una expresión y utilizar el resultado para establecer el valor de una propiedad de un **JavaBean** o de un objeto **java.util.Map**.
Tiene los siguientes atributos:

- **value**: Es la expresión que se va a evaluar.
- **var**: es el nombre de la variable donde se almacena el resultado de la evaluación de la expresión.
- **scope**: define el ámbito de la variable anterior. Puede ser `page`, `request`, `session` y `application`, siendo `page` el valor por defecto.
- **target**: es el nombre del objeto cuya propiedad se va a establecer.
- **property**: es el nombre de la propiedad que se va a establecer en el objeto definido en **target**.

```
<c:set value="<expresión>" target="<objeto destino>"
property="<nombre propiedad>" />
<c:set value="<expresión> var="<nombre de variable>" />
```

<c:remove>
La etiqueta **remove** se utiliza para eliminar variables, dentro de un ámbito.
Tiene los siguientes atributos:

- **scope**: Es el ámbito en que se va a buscar la variable a eliminar.
- **var**: es el nombre de la variable a eliminar.

<c:catch>
La etiqueta **catch** es una forma de tratar los errores que se puedan producir en determinadas partes de la página JSP.

- **var**: es el nombre de la variable donde se almacena la excepción capturada.

Iteraciones

Estas etiquetas son una alternativa a los bucles **for**, **while** o **do-while** de Java.

IteracionScriptlet.jsp

```
<html>
<head>
<title>Ejemplo de iteraci&uacute;on</title>
</head>
<body>
<%
java.util.Iterator set =
(java.util.Iterator)request.getAttribute("set");
while (set.hasNext()) { %>
El valor es <%= set.next() %>
<% } %>
</body>
</html>
```

milteracion.html

```
<html>
<head>
<title>Selecci&oacute;n m&uacute;ltiple</title>
</head>
<body>
<table width="550" border="0" bgcolor="#006633" align=center>
<tr>
<td><img src="machado.gif"></td>
<td>
<font color="#FFFFFF" face="Arial "size="5">
<b>  I.E.S. Antonio Machado - Sevilla  </b>
</font>
</td>
</tr>
```

```
</table>
    <FORM action="http://localhost:8080/dfsi/IteracionJSTL.jsp"
method="post">
         <P>
         <SELECT NAME="piezas" SIZE=4 MULTIPLE>
            <OPTION VALUE="Ordenador" SELECTED>Ordenador</OPTION>
            <OPTION VALUE="Monitor">Monitor</OPTION>
            <OPTION VALUE="Impresora" SELECTED>Impresora</OPTION>
            <OPTION VALUE="Escaner" SELECTED>Escaner</OPTION>
         </SELECT>
         <INPUT TYPE="submit" VALUE="Items Seleccionados">
      </FORM>
</body>
</html>
```

IteracionJSTL.jsp

```
<%@ taglib uri=" http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Ejemplo de iteraci&oacute;n JSTL</title>
</head>
<body>
<c:forEach var="item" begin="0" items="${paramValues.piezas}">
El valor es <c:out value="${item}"/>
<br>
</c:forEach>
</body>
</html>
```

La diferencia es mínima, pero la sintaxis del ejemplo con JSTL resulta más limpia.

<forEach>

La etiqueta **forEach** permite la iteración sobre una colección de objetos (un array, **Collection**, **Iterator**, **Enumeration** y **Map**). Tiene los siguientes atributos:

- **var**: Es el nombre del objeto actual durante la iteración.
- **items**: Define la colección de objetos sobre la que se va a iterar.
- **varStatus**: Define el nombre de la variable que almacena el estado de la iteración.
- **begin**: Es un valor de tipo `int` que establece el inicio de la iteración.
- **end**: Es un valor de tipo `int` que establece el final de la iteración. Si no se especifica, la iteración termina con el último elemento de la colección.
- **step**: Es un valor de tipo `int` que establece el incremento de la iteración.

Iteracion2JSTL.jsp

```
<%@ taglib uri=" http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Ejemplo de iteraci&uacute;on JSTL</title>
</head>
<body>
<c:forEach var="item" begin="0" items="${set}"
      begin=”2” step=”3”>
El valor es <%= item %>
</c:forEach>
</body>
</html>
```

<forTokens>
La etiqueta **forTokens** permite descomponer un **String** en varios, según un patrón determinado. Después se recorren de forma similar a **forEach**. Todos los atributos de **forEach** son válidos y se introduce uno nuevo:

- **delim**: Establece el valor del patrón que delimita las subcadenas.

forTokensJSTL.jsp
```
<%@ taglib uri=" http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Ejemplo de iteracion JSTL</title>
</head>
<body>
<c:forTokens var="item" delims=”~” items="token1~token2~token3">
El valor es <%= item %>
</c:forTokens>
</body>
</html>
```

Condicionales
Estas etiquetas son una alternativa a las sentencias Java **if** y **switch**.

<c:if>
La etiqueta **if** permite la ejecución de un bloque dependiendo del valor de la condición. Tiene los siguientes atributos:

- **test**: Es la condición a evaluar.
- **var**: Es un atributo opcional que permite almacenar el resultado del test anterior.

ifJSTL.jsp
```
<%@ taglib uri=" http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Ejemplo de condicional JSTL</title>
</head>
<body>
<c:if test=”${user==null}”>
<form>
Nombre: <input name=”nombre”>
Password: <input name=”password”>
</form>
</c:if>
<c:if test=”${user!=null}”>
Bienvenido, ${user.nombre}
</c:if>
<!—
  Resto de la página principal…
-->
</body>
</html>
```

<choose>, <when> y <otherwise>
Permiten la ejecución de un bloque **switch**.

chooseJSTL.jsp
```
<%@ taglib uri=" http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Ejemplo de condicional JSTL</title>
```

```
</head>
<body>
<c:choose>
<c:when test=”${user==null}”>
<form>
Nombre: <input name=”nombre”>
Password: <input name=”password”>
</form>
</c:when>
<c:otherwise>
Bienvenido, ${user.nombre}
</c:otherwise>
</c:choose>
<!—
  Resto de la página principal…
-->
</body>
</html>
```

Manipulación de URL

JSP tiene un soporte limitado de la manipulación de URL. **JSTL** aporta algunas etiquetas que facilitan estas tareas.

<c:import>

La etiqueta **import** aporta toda la funcionalidad de la acción **include** pero además permite la inclusión de URL absolutas. Por defecto, la información se dirige hacia el objeto **JSPWriter** pero se pueden utilizar algunos atributos para especificar un objeto **Reader** dónde se pueda acceder al contenido. La etiqueta **import** tiene los siguientes atributos:

- **url**: Es el valor de la URL que hay que importar. Puede ser relativa o absoluta.
- **context**: Permite especificar un contexto de otra aplicación Web que se esté ejecutando en el contenedor Web.
- **var**: Permite especificar una variable que referencie el contenido del recurso.
- **scope**: Establece el ámbito de la variable definida anteriormente. Los valores válidos son: `page`, `request`, `session` y `application`.
- **charEncoding**: Especifica la codificación de caracteres del recurso de entrada. Por defecto se asume el `ISO-8859-1`.
- **varReader**: Permite crear un objeto **Reader** que nos dé acceso al contenido importado.

Hay dos formas de utilizar la etiqueta import. La primera es importar contenido local o remoto y enviarlo directamente al cliente.

```
<c:import url=”<url>” />
```

La segunda es importar el contenido a una variable o a un objeto **Reader**.

importJSTL.jsp

```
<%@ taglib uri=" http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Ejemplo de import JSTL</title>
</head>
<body>
Aquí está el código fuente de la página principal.
<pre>
<c:import url=”index.html” var=”valor” />
<c:out value=${valor}” />
</pre>
```

```
</body>
</html>
```

<c:param>

La etiqueta **param** complementa la etiqueta **import** permitiendo especificar parámetros en la petición de la URL.
Tiene dos atributos:

- **name**: Es el nombre del atributo que se va a especificar en al URL.
- **value**: Es el valor que tomará dicho atributo.

<c:url>

La etiqueta **url** permite codificar la URL con información de la sesión y los parámetros que se hayan definido.
Tiene los siguientes atributos:

- **value**: Es el valor de la URL que se va a procesar.
- **context**: Permite especificar el contexto cuando se hace referencia a una URL de un contexto diferente.
- **var**: Permite guardar el valor de la URL generada en una variable.
- **scope**: Establece el ámbito de la variable definida anteriormente. Los valores válidos son: `page`, `request`, `session` y `application`.