JSTL

Víctor Custodio

¿Qué es JSTL?

- JSTL no es más que un conjunto de librerías de etiquetas simples y estándares que encapsulan la funcionalidad principal que es usada comúnmente para escribir páginas JSP. Las etiquetas JSTL están organizadas en 5 librerías:
  - **core**: Comprende las funciones script básicas como loops, condicionales, y entrada/salida.
  - **xml**: Comprende el procesamiento de xml
  - **fmt**: Comprende la internacionalización y formato de valores como de moneda y fechas.
  - **sql**: Comprende el acceso a base de datos.
  - **Fn**: Para trabajar con la longitud de conjuntos así como manejo de Strings

¿Cuál es el problema con los scriptlets JSP?

- La especificación JSP ahora se ha convertido en una tecnología estándar para la creación de sitios Web dinámicos en Java pero….
  - El código Java embebido en scriptlets es desordenado.
  - Un programador que no conoce Java no puede modificar el código Java embebido, anulando uno de los mayores beneficios de los JSP: permitir a los diseñadores y personas que escriben la lógica de presentación que actualicen el contenido de la página.
  - El código de Java dentro de scriptlets JSP no pueden ser reutilizados por otros JSP, por lo tanto la lógica común termina siendo re-implementado en múltiples páginas.
  - La recuperación de objetos en los ambientes de HTTP Request y Session es complicada. Es necesario hacer el Casting de objetos y esto ocasiona que tengamos que importar más Clases en los JSP.

¿Como mejoran esta situación la librería JSTL?

- Las etiquetas JSTL **son XML**, estas etiquetas se integran limpia y uniformemente a las etiquetas HTML.
- Las 5 librerías de etiquetas **JSTL incluyen la mayoría de funcionalidad que será necesaria en una página JSP**. Las etiquetas JSTL son muy sencillas de usar para personas que solo conocen HTML
- Las etiquetas JSTL **encapsulan la lógica como el formato de fechas y números**. Usando los scriptlets JSP, esta misma lógica necesitaría ser repetida en todos los sitios donde es usada
- Las etiquetas JSTL **pueden** referenciar objetos que se encuentren en los ambientes Request y Session sin conocer el tipo del objeto y **sin necesidad de hacer el Casting**.

¿Desventajas de JSTL?

- Mayor sobrecarga en el servidor. Con Scriplets el servidor solo copiaba código java, ahora necesita hacer una traducción de etiquetas a código java.
- A pesar que las etiquetas JSTL proporciona un potente conjunto de librerías reutilizables, no puede hacer todo lo que el código Java puede hacer

Ejemplo

- Ambas páginas implementan la misma lógica.
  - Graban una lista de objetos AddressVO del Request
  - Luego iteran a través de la lista, imprimiendo el atributo apellido de cada objeto (si el apellido no es null y de longitud diferente a 0). En cualquier otro caso, imprimirá "N/A".
  - Finalmente, la página imprime la fecha actual.

Ejemplo: con scriplets JSP

```
<%@ page import="com.ktaylor.model.AddressVO, java.util.*"%>

  <p><h1>Customer Names</h1></p>
  <%
  List addresses = (List)request.getAttribute("addresses");
  Iterator addressIter = addresses.iterator();
   while(addressIter.hasNext()) {
     AddressVO address = (AddressVO)addressIter.next();
     if((null != address) &&
    (null != address.getLastName()) &&
    (address.getLastName().length() > 0)) {
%>
 <%=address.getLastName()%><br/>
<%
   }
  else {
%>
   N/A<br/>
<%
   }
 }
%>
 <p><h5>Last Updated on: <%=new Date()%></h5></p>
```

Ejemplo: sin scriplets JSP

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
 <p><h1>Customer Names</h1></p>
 <c:forEach items="${addresses}" var="address">
  <c:choose>
     <c:when test="${not empty address.lastName}" >
       <c:out value="${address.lastName}"/><br/>
     </c:when>
    <c:otherwise>
      N/A<br/>
    </c:otherwise>
  </c:choose><br/>
</c:forEach><br/>

<jsp:useBean id="now" class="java.util.Date" />

 <p><h5>Last Updated on: <c:out value="${now}"/></h5></p>
```

EL: ¿Qué es eso que hay entre ${} ?

- **EL** (Expression Language) es un lenguaje utilizado en las paginas **jsp** para interactuar con los datos servidos por parte del servidor, combinado con la librería **JSTL Core** nos permite construir toda la lógica de las **jsp** de una forma mucho mas amena y eficaz.
- **Accediendo a variables.**
  Para obtener el valor de cualquier variable, sea del tipo que sea, lo único que tenemos que hacer es escribir su nombre entre **${}**, de tal modo, si nuestra variable "**miVariable**" es un String, Integer, Date()… bastara con escribir "**${miVariable}**" para acceder a ella.

EL

- Notacion por puntos. Imaginaros que tenemos un objeto Persona con nombre “persona” almacenado en la sesión. En el servlet:

```
Persona persona = new Persona();

HttpSession session = request.getSession(true);

session.setAttribute("persona", persona);
```

Todo lo que tenemos que hacer para acceder a esta variable y sus propiedades es escribir el nombre del objeto, mas el punto y la propiedad como si se tratase de un javabean

EL

- Podemos mostrar el nombre de la persona con esto: <c:out value=”**${sessionScope.persona.edad}** o esto directamente entre etiquetas HTML: ”**${sessionScope.persona.edad}.**
- Además del ámbito de session tenemos otros elementos predefinidos en EL:
  - **pageScope**
  - **requestScope**
  - **sessionScope**
  - **applicationScope**
  - **param y paramValues**
  - **header y headerValues**
  - **initParam**
  - **cookie**
  - **pageContext**

EL

- Si ningún otro objeto existe en otro ambito con el mismo nombre, podemos acceder al objeto directamente por su nombre, en el ejemplo anterior podríamos usar: ${**persona.edad**}
- Al ejecutarse, se buscará en todos los ámbitos (Scopes), empezando por el más pequeño y terminando por el más grande.

EL: Operaciones

- Además del acceso a los objetos y sus propiedades también podemos realizar algunas operaciones con EL. Estás son:

.	Access a bean property or Map entry
[]	Access an array or List element
()	Group a subexpression to change the evaluation order
+	Addition
-	Subtraction or negation of a value
*	Multiplication
/ or div	Division
% or mod	Modulo (remainder)
== or eq	Test for equality
!= or ne	Test for inequality
< or lt	Test for less than
> or gt	Test for greater than
<= or le	Test for less than or equal
>= or ge	Test for greater than or equal
&& or and	Test for logical AND
|| or or	Test for logical OR
! or not	Unary Boolean complement
empty	Test for empty variable values

Ejemplo:
<c:if test=”**${empty persona.nombre}**“>
“No se conoce el nombre
</c:if>

Instalación y configuración del JSTL

- La librería JSTL es distribuida como un conjunto de archivos JAR que simplemente tenemos que agregarlo en el classpath del contenedor de servlets.
- Descargar la implementación JSTL de la página de proyecto Jakarta TagLibs
- Copia la librería al directorio /WEB-INF/lib de tu aplicación Web
- Si utilizamos un servidor de aplicaciones JEE no será necesario añadir las librerías ya que el servidor tendrá su propia implementación.

Instalación y configuración del JSTL

- Importa en las páginas JSP cada librería JSTL que se use:

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

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

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

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

HelloWorld

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
  Setting the value: "Hello World!"
  <c:set var="hello" value="Hello World!"/>
  <p/>
  <c:out value="${hello}"/>
```

Core

- Las etiquetas JSTL pertenecientes a este grupo nos proveen la siguiente funcionalidad:
- **a.** Manipular variables, dichas etiquetas son :
  <c:out>, <c:set>, <c:remove> y <c:catch>
- **b.** Crear condicionales.
  <c:if>, <c:choose>, <c:when> y <c:otherwise>
- **c.** También permiten crear ciclos repetitivos (bucles):
  <c:forEach> y <c:forTokens>
- **d.** Por último, nos permiten Manipular URL's con las etiquetas:
  <c:import>, <c:url>, <c:redirect> y <c:param>
- Pruebas

FMT

- Internacionalización : http://chuwiki.chuidiang.org/index.php?title=Internacionalizaci%C3%B3n_de_JSP_con_JSTL
- Formateo de fechas, ejemplo:

fmt

```
<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>
```

FN

- Nos permite trabajar con Strings y Arrays en el lenguaje EL de JSP resolviendo la mayor parte de las necesidades típicas en programación. Aquí muestro los ejemplos de las más utilizadas.
- Pruebas