élogos

LENGUAJE EL Y LA LIBRERÍA DE ACCIONES ESTÁNDAR JSLT

élogos

El lenguaje EL y la librería de acciones estándar JSLT

JSTL (JSP Standard Tag Library)

La librería JSTL es un componente dentro de la especificación del Java 2 Enterprise Edition (J2EE) y es controlada por Sun MicroSystems. 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 4 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.

Problema con los scriptlets JSP

Un scriptlet JSP: <% int contador = 100; %>

La especificación JSP ahora se ha convertido en una tecnología estándar para la creación de sitios Web dinámicos en Java, y el problema es que han aparecido algunas debilidades:

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 puede ser reutilizado por otros JSP, por lo tanto la lógica común es que termina siendo reimplementado en múltiples páginas.

La recuperación de objetos fuera del 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.

Debido a que las etiquetas JSTL son XML, estas etiquetas se integran limpia y uniformemente en las etiquetas HTML.

LA LIBRERÍA DE ACCIONES ESTÁNDAR JSLT

Las 4 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 no saben de programación, ya que como mucho necesitarán conocimientos de etiquetas del estilo 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, o necesitaría ser movida a Clases de ayuda.

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.

Los JSP EL (Expression Language) facilitan las llamadas a los métodos Get y Set en los objetos Java. Esto no es posible en la versión JSP 1.2, pero ahora está disponible en JSP 2.0. Expression Language es usado extensamente en la librería JSTL.

Desventajas de JSTL

Los JSTL pueden agregar mayor sobrecarga en el servidor. Los scriptlets y las librerías de etiquetas son compilados a servlets, los cuales luego son ejecutados por el contenedor. El código Java embebido en los scriptlets es básicamente copiado en el servlet resultante. En cambio, las etiquetas JSTL, causan un poco más de código en el servlet. En la mayoría de casos esta cantidad no es importante pero debe ser considerada.

Los scriptlets son más potentes que las etiquetas JSTL. Si se desea hacer todo en un script JSP es muy probable que se inserte todo el código en Java en él. A pesar de que las etiquetas JSTL proporcionan un potente conjunto de librerías reutilizables, no puede hacer todo lo que el código Java puede hacer. La librería JSTL está diseñada para facilitar la codificación en el lado de presentación que es típicamente encontrado en la capa de Vista si hablamos de la arquitectura Modelo-Vista-Controlador.

Ejemplo. JSTL sin scriptlets

Vamos a ver 2 versiones de una página simple. La primera versión usa etiquetas scriptlets, y la segunda usa etiquetas JSTL. Ambas páginas implementan la misma lógica. Graban una lista de objetos AddressVO del Request y 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.

LA LIBRERÍA DE ACCIONES ESTÁNDAR JSLT

Con scriptlets 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>
```

LA LIBRERÍA DE ACCIONES ESTÁNDAR JSLT

Con JSTL:

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

Instalación y configuración del JSTL

La librería JSTL es distribuida como un conjunto de archivos JAR que simplemente tenemos que agregar en el classpath del contenedor de servlets.

1. Debemos usar un contenedor de servlets compatible con la versión JSP 2.0 para usar el JSTL El Apache Tomcat 5.0 [http://jakarta.apache.org/tomcat/] por ejemplo. En caso contrario, es posible usar el JSTL 1.0 en un contenedor que soporte por lo menos el JSP 1.2 y servlets 2.3.

2. Descargar la implementación JSTL de la página de proyecto Jakarta TagLibs [http://jakarta.apache.org/taglibs/binarydist.html]. La distribución binaria viene empaquetada como .zip o tar.gz. Desempaquete estos archivos en un directorio temporal.

3. Del directorio temporal, copie todos los archivos JAR que se encuentran en jakarta-taglibs/standard-1.0/lib al directorio /WEB-INF/lib de su aplicación Web. Esto incluirá los JAR específicos del JSTL así como también los otros JAR que depende de él.

4. Finalmente, importamos en las páginas JSP cada librería JSTL que la página necesitará. Eso lo hacemos agregando las directivas taglib apropiadas al inicio de la página JSP. Las directivas son:

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

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

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

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

Creamos una página JSP y probamos algunas etiquetas simples para asegurarnos de que la configuración del JSTL está bien hecha.

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

Cuando accedemos a esta página desde el navegador Web, si la configuración es correcta, deberíamos ver:

Setting the value: "Hello World!"

En la línea 1 la página helloword.jsp usa la directiva taglib para importar la librería núcleo de la etiqueta y asigna la letra "c" como prefijo. La letra "c" es usada para referenciar a las etiquetas núcleo en la página JSP. Las únicas 2 etiquetas que usamos en el ejemplo son c:set y c:out.

En la línea 4 c:set lo usamos para asignar el valor "Hello World" a la variable hello.

En la línea 6 c:out es usado para evaluar la variable hello con la expresión EL ${$hello}.

Usando las etiquetas JSTL, podemos reusar más nuestro código en las páginas JSP, podemos tener JSP más ordenados y legibles, y, lo mejor de todo, podemos tener a los diseñadores y a los proveedores de los contenidos trabajando en el mantenimiento de las páginas JSP.

Expresiones EL

El lenguaje de expresiones de JSP tiene como objetivo reemplazar a las clásicas expresiones JSP basadas en la utilización de código Java, contribuyendo así a la reducción e incluso eliminación en algunos casos de la utilización de scriptlets Java dentro de una página JSP.

Las expresiones EL devuelven un valor al lugar de la página donde esté situada la expresión. Su sintaxis es:

${expresion}

donde expresion es cualquier expresión sintáctica válida EL que devuelva un resultado, como una llamada a un objeto EL o a alguna variable JSTL.

Las expresiones EL se combinan con las acciones JSTL para generar dinámicamente las páginas. Los valores de muchos atributos de acciones JSTL se establecen mediante expresiones EL.

Objetos implícitos EL

El lenguaje EL incluye una serie de objetos implícitos que permiten acceder de una forma sencilla a toda la información que los distintos objetos del API servlet proporcionan a la aplicación.

Algunos de los objetos implícitos más interesante que forman parte del lenguaje EL son:

- **pageScope**. Proporciona acceso a las variables de ámbito de página.
- **requestScope**. Proporciona acceso a las variables de ámbito de petición. Por ejemplo, dado el siguiente bloque de sentencias incluidas en una página JSP:

```
<%int codigo=Math.ceil(Math.random()*500);
      request.setAttribute("codigo", codigo);%>
<jsp:forward page="prueba.jsp"/>
```

  Si quisiéramos mostrar en la página prueba.jsp el valor de la variable de petición “codigo”, utilizaríamos:

```
El código generado es: ${requestScope.codigo}
```

- **sessionScope**. Proporciona acceso a las variables de ámbito de sesión. Por ejemplo, supongamos que en una página JSP tenemos el siguiente bloque de instrucciones:

```
<jsp:useBean id="ob" class="javabeans.Datos"
       scope="session"/>
<jsp:setProperty name="ob"
      property="numero" value="100"/>
<%response.sendRedirect("prueba.jsp");%>
```

  El acceso a la propiedad numero del objeto desde prueba.jsp sería:

```
${sessionScope.ob.numero}
```

- **applicationScope**. Proporciona acceso a las variables de ámbito de aplicación.
- **param**. Mediante este objeto tenemos acceso a los parámetros enviados en la petición. Por ejemplo, el siguiente bloque de sentencias inicializaría la propiedad "nombre" del JavaBean "usuario" con el valor del parámetro "user" recibido en la petición:

```
<jsp:useBean id="usuario" class="javabeans.Usuario"/>
<jsp:setProperty name="usuario" property="nombre"
value="${param.user}"/>
```

- **cookie**. Proporciona acceso a las cookies enviadas en la petición. Cada elemento de la colección Map asociada representa un objeto cookie. La siguiente expresión de ejemplo mostraría en la página el contenido de la cookie "user":

  Usuario: ${cookie.user.value}

Principales acciones JSTL

Entre las principales acciones del core JSTL tenemos las siguientes:

<c:set>

Establece el valor de una variable:

```
<c:set var="dato" value="${6+10}"/>
```

También puede utilizarse para establecer el valor de una propiedad de un JavaBean:

```
<c:set target="persona"
      property="usuario" value="${param.user}"/>
```

<c:if>

Evalúa el cuerpo de la acción si el resultado de la condición indicada en su atributo test es true. El código del siguiente ejemplo mostraría en la página de respuesta el valor de la variable "n" en caso de que éste sea un número par:

```
<c:if test="${n%2 == 0}">
       El número es <c:out value="${n}"/>
</c:if>
```

<c:choose>

Su formato es similar al de la acción <c:if>, sólo que en este caso se comprueban varias condiciones, evaluándose el cuerpo de la primera que resulte verdadera.

El siguiente ejemplo muestra distintos mensajes en la página de respuesta en función del valor de la variable dat:

```
<c:choose>
       <c:when test="${dat>7 && dat<13}">
              Buenos días
       </c:when>
       <c:when test="${dat>13 && dat<20}">
              Buenas tardes
```

```
        </c:when>
        <c:otherwise>
              Buenas noches
        </c:otherwise>
</c:choose>
```

<c:forEach>

Se emplea para iterar una colección de objetos. Muy util cuando se quiere mostrar en una página el resultado de una búsqueda en una base de datos.

El siguiente ejemplo muestra el valor de la propiedad nombre de todos los javabeans almacenados en la colección personas de ámbito de sesión:

```
<c:forEach var="persona" items="${sessionScope.personas}">
        <c:out value="${persona.nombre}"/>
</c:forEach>
```