élogos

GENERACIÓN DINÁMICA DE PÁGINAS. OBJETOS IMPLÍCITOS

élogos

ÍNDICE

GENERACIÓN DINÁMICA DE PÁGINAS. OBJETOS IMPLÍCITOS

1. Resumen de sintaxis JSP

A continuación mostramos una tabla con un resumen de sintaxis JSP indicando:
- La sintaxis
- La interpretación
- Una serie de notas que ayuden a su comprensión

Elemento JSP	Síntaxis	Interpretación	Notas
Expresión JSP	<%= expression %>	La expresión es evaluada y situada en la salida.	El equivalente XML es <jsp:expression> expression </jsp:expression>. Las variables predefinidas son request, response, out, session, application, config, y pageContext.
Scriptlet JSP	<% code %>	El código se inserta en el método service.	El equivalente XML es: <jsp:scriptlet> code </jsp:scriptlet>.
Declaración JSP	<%! code %>	El código se inserta en el cuerpo de la clase del servlet, fuera del método service.	El equivalente XML es: <jsp:declaration> code </jsp:declaration>.
Directiva page JSP	<%@ page att="val" %>	Dirige al motor servlet sobre la configuración general.	El equivalente XML es: <jsp:directive.page att="val"\>. Los atributos legales son (con los valores por defecto en negrita): import="package.class" contentType="*MIME-Type*" isThreadSafe="true|false" session="true|false" buffer="*size*kb|none" autoflush="true|false" extends="package.class" info="*message*" errorPage="*url*" isErrorPage="true|false" language="java"
Directiva include JSP	<%@ include file="url" %>	Un fichero del sistema local se incluirá cuando la página se traduzca a un Servlet.	El equivalente XML es: <jsp:directive.include file="url"\>. La URL debe ser relativa. Usamos la acción jsp:include para incluir un fichero en el momento de la petición en vez de en el momento de la traducción.
Comentario JSP	<%-- comment --%>	Comentario ignorado cuando se traduce la página JSP en un servlet.	Si queremos un comentario en el HTML resultante, usamos la sintaxis de comentario normal del HTML <-- comment -->.
Acción jsp:include	<jsp:include page="*relative URL*" flush="true" />	Incluye un fichero en el momento en que la página es solicitada.	Aviso: en algunos servidores, el fichero incluido debe ser un fichero HTML o JSP, según determine el servidor (normalmente basado en la extensión del fichero).
Acción jsp:useBean	<jsp:useBean att=val*/> o <jsp:useBean att=val*> ... </jsp:useBean>	Encuentra o construye un Java Bean.	Los posibles atributos son: id="*name*" scope="page|request|session|application" class="package.class" type="package.class" beanName="package.class"
Acción jsp:setProperty	<jsp:setProperty att=val*/>	Selecciona las propiedades del bean, bien directamente o designando el valor que viene desde un parámetro de la petición.	Los atributos legales son: name="*beanName*" property="*propertyName*|*" param="parameterName" value="*val*"
Acción jsp:getProperty	<jsp:getProperty name="*propertyName*" value="*val*" />	Recupera y saca las propiedades del bean.	
Acción jsp:forward	<jsp:forward page="*relative URL*"/>	Reenvía la petición a otra página.	
Acción jsp:plugin	<jsp:plugin attribute="*value*"> ...</jsp:plugin>	Genera etiquetas OBJECT o EMBED, apropiadas al tipo de navegador, pidiendo que se ejecute un applet usando el Java Plugin.	

2. Elementos de Script JSP

Los elementos de script que nos permiten insertar código Java, dentro del servlet que se generará desde la página JSP actual, son de 3 formas:

1. Expresiones de la forma: `<%= expresión %>`

Que son evaluadas e insertadas en la salida.

2. Scriptlets de la forma : `<% código %>`

Que se insertan dentro del método service del servlet.

3. Declaraciones de la forma: `<%! código %>`

Que se insertan en el cuerpo de la clase del servlet, fuera de cualquier método existente.

Expresiones de la forma <%= expresión %>

Una expresión JSP se usa para insertar valores Java directamente en la salida. Tiene la siguiente forma:

<%= expresión Java %>

La expresión Java es evaluada, convertida a un string e insertada en la página. Esta evaluación se ejecuta durante la ejecución (cuando se solicita la página) y así tiene total acceso a la información sobre la solicitud.

Por ejemplo, puede acceder a los parámetros de la petición a través del objeto request, o a los atributos de sesión mediante sesión.

<%= expresión %>

- Para simplificar estas expresiones, hay un gran número de variables predefinidas que podemos usar.
- Los autores de XML pueden usar una sintaxis alternativa para las expresiones JSP, son sensibles a las mayúsculas, por eso asegúrate de usar minúsculas.

Generación dinámica de páginas. Objetos implícitos

Scriptlets de la forma <% código %>

Si queremos hacer algo más complejo que insertar una simple expresión, los scriptlets JSP nos permiten insertar código arbitrario dentro del método servlet que será construido al generar la página.

Los Scriptlets tienen la siguiente forma: <% Código Java %>

Los Scriptlets tienen acceso a las mismas variables predefinidas que las expresiones. Por eso, por ejemplo, si queremos que la salida aparezca en la página resultante, tenemos que usar la variable out

```
<%
String queryData = request.getQueryString();
out.println("Attached GET data: " + queryData);
%>
```

Los scriptlets no necesitan completar las sentencias Java, y los bloques abiertos pueden afectar al HTML estático fuera de los scriplets. Por ejemplo, el siguiente fragmento JSP, contiene una mezcla de texto y scritplets:

```
<% if (Math.random() < 0.5) { %>
Have a <B>nice</B> day!
<% } else { %>
Have a <B>lousy</B> day!
<% } %>
Que se convertirá en algo como esto:
if (Math.random() < 0.5) {
  out.println("Have a <B>nice</B> day!");
} else {
  out.println("Have a <B>lousy</B> day!");
}
```

sentencias print

Si queremos usar los caracteres "%>" dentro de un scriptlet, debemos poner "%\>". Finalmente, observa que el equivalente XML de <% Código %> es

```
<jsp:scriptlet>
Código
</jsp:scriptlet>
```

Declaraciones de la forma <%! código %>

Una declaración JSP nos permite definir métodos o campos que serán insertados dentro del cuerpo principal de la clase servlet (fuera del método service que procesa la petición).

Tienen la siguiente forma:
<%! Código Java%>

<%! Código Java%>

Como las declaraciones no generan ninguna salida, normalmente se usan en conjunción con expresiones JSP o scriptlets.

Por ejemplo, aquí tenemos un fragmento de JSP que imprime el número de veces que se ha solicitado la página actual desde que el servidor se arrancó (o la clase del servlet se modificó o se recargó):

```
<%! private int accessCount = 0; %>
Accesses to page since server reboot:
<%= ++accessCount %>
```

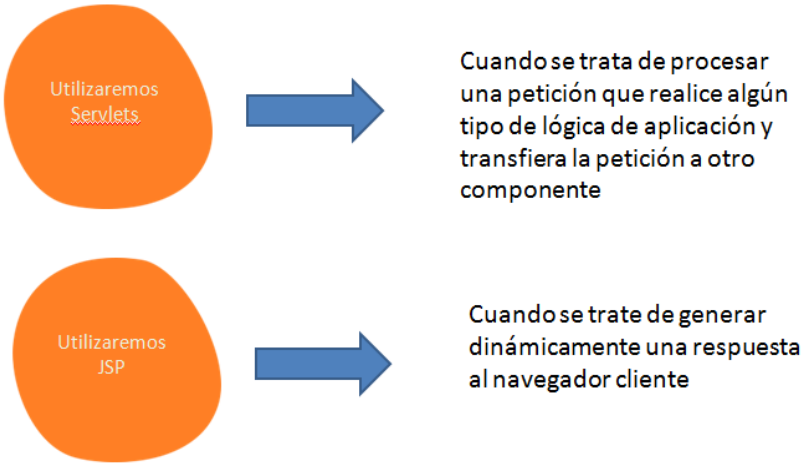
Como con los scriptlet, si queremos usar los caracteres "%>", ponemos "%\>". Finalmente, observa que el equivalente XML de <%! Código %> es...

```
<jsp:declaration>
Código
</jsp:declaration>
```

Generación dinámica de páginas. Objetos implícitos

Combinar servlets y JSP

Las páginas JSP no tienen como objetivo sustituir a los servlets a la hora de crear una aplicación Web, sino complementarlos. La mayoría de las aplicaciones utilizan tanto servlets como páginas JSP.



Ejemplo

```
public class Servlet1 extends HttpServlet {
   @Override
   public void init(ServletConfig config) throws ServletException {
      //inicializa el contador de visitas globales a 0
      //esto sucederá con la primera petición
      //realizada al servlet
      ServletContext sc=config.getServletContext();
      sc.setAttribute("global",0);
      super.init(config);
   }
   protected void service(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      //incrementa el contador cada vez que un usuario
      //solicita el servlet
      ServletContext sc=this.getServletContext();
      int contador=(Integer)sc.getAttribute("global");
      contador++;
      sc.setAttribute("global", contador);
   }
```

```
}
```

muestra.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Bienvenido</h1>
        El número de visitas realizadas al sitio es de: <%=application.getAttribute("global")%>
    </body>
</html>
```

3. Objetos Implícitos

Para simplificar el código en expresiones y scriplets JSP, tenemos ocho variables definidas automáticamente, algunas veces llamadas objetos implícitos.

Las variables disponibles son:

request

Éste es el HttpServletRequest asociado con la petición, y nos permite mirar los parámetros de la petición (mediante getParameter), el tipo de petición (GET, POST, HEAD, etc.), y las cabeceras HTTP entrantes (cookies, Referer, etc.).

response

Éste es el HttpServletResponse asociado con la respuesta al cliente. Observa que, como el stream de salida (ver out más abajo) tiene un buffer, es legal seleccionar los códigos de estado y cabeceras de respuesta, aunque no está permitido en los servlets normales una vez que la salida ha sido enviada al cliente.

out

Éste es el PrintWriter usado para envíar la salida al cliente. Esta es una versión con buffer de PrintWriter llamada JspWriter, podemos ajustar el tamaño del buffer, o incluso desactivar el buffer, usando el atributo buffer de la directiva page. Se usa casi exclusivamente en scriptlets.

session

Éste es el objeto HttpSession asociado con la petición. Recuerda que las sesiones se crean automáticamente, por esto esta variable se une incluso si no hubiera una sesión de referencia entrante. La única excepción es usar el atributo session de la directiva page para desactivar las sesiones, en cuyo caso los intentos de referenciar la variable session causarán un error en el momento de traducir la página JSP a un servlet.

aplication

Éste es el ServletContext obtenido mediante getServletConfig().getContext().

config

Éste es el objeto ServletConfig para esta página.

pageContext

JSP presenta una nueva clase llamada PageContext para encapsular características de uso específicas del servidor como JspWriters de alto rendimiento.

page

Esto es sólo un sinónimo de this, y no es muy útil en Java. Fue creado como situación para el día en que los lenguajes de script puedan incluir otros lenguajes distintos de Java.