élogos

CARACTERÍSTICAS DE LA TECNOLOGÍA JSP Y COMPONENTES

élogos

ÍNDICE

CARACTERÍSTICAS DE LA TECNOLOGÍA JSP Y COMPONENTES

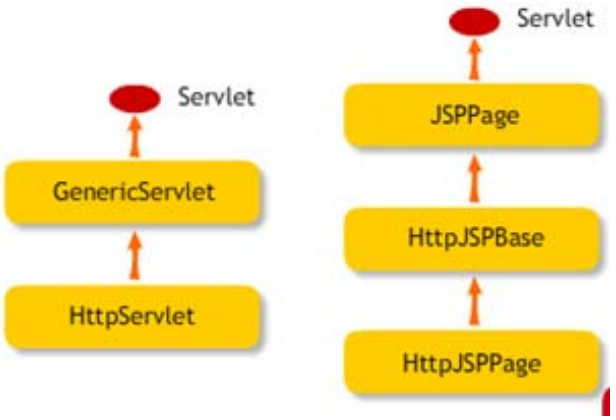
1. Visión general de todos sus elementos

Introducción

La tecnología JSP simplifican la creación dinámica de páginas al poder combinar en un archivo de texto bloques HTML con código Java.

Objetivo del JSP

Un objetivo que persigue JSP es separar la presentación de la lógica de negocio. Con esto se descompone el trabajo entre el diseñador de la página web y el programador que realiza la lógica del negocio.



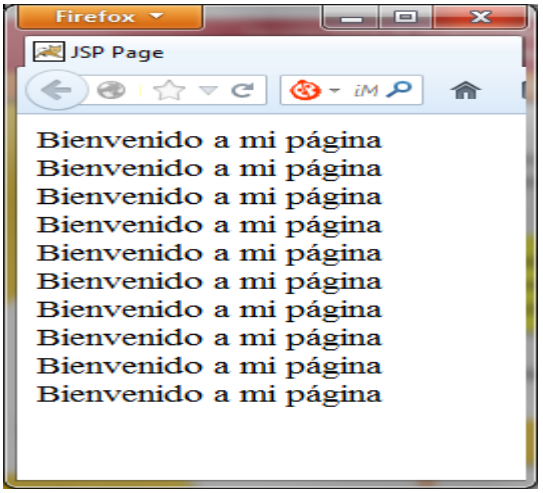
Estructura de una página JSP

Una página JSP es un archivo de texto en el que se combinan bloque de texto HTML estático, con bloque de código Java. El código Java tiene como misión generar de forma dinámica parte de la página .

Los bloques de texto HTML se introducen tal cual en el documento, evitando los incómodos out.println() utilizados en los servlets para generar HTML. Por su parte, el código Java insertado en la página se delimita por los símbolos <% y %>. Los bloque de código HTML se pueden introducir en cualquier parte de la página y tantos como se necesiten.

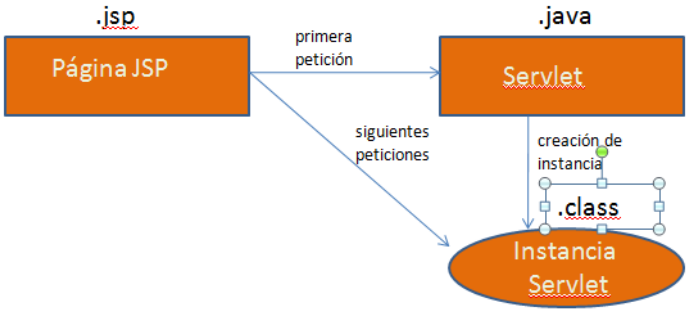
El siguiente ejemplo de página JSP genera una página Web que nos muestra 10 veces un mensaje de saludo:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <title>Ejemplo JSP</title>
   </head>
   <body>
      <%for(int i=1;i<=10;i++){%>
         Bienvenido a mi página<br/>
      <%}%>
   </body>
</html>
```



Ciclo de vida de una página JSP

Las páginas JSP representan un importante elemento que facilita la construcción de aplicaciones, sin embargo, de cara al servidor de aplicaciones no existe diferencia entre páginas JSP y Servlets, puesto que toda página JSP es trasformada en un Servet.



La transformación de la página JSP en un Servlet tiene lugar cuando se produce la primera petición de la página. A partir de ahí, las siguientes peticiones son dirigidas directamente al Servlet.

Todo este proceso es transparente tanto para el usuario como para el programador, pues es realizado de forma implícita por un Servlet interno que incorporan los servidores de aplicaciones Java EE.

El Servlet creado implementa la interfaz HttpJspPage, que es una subinterfaz de Servlet.

Características de la Tecnología JSP y Componentes

Elementos JSP

Dentro de una página JSP lo primero que se encuentra es texto HTML pero además se van a encontrar otros tres elementos.

Clase JSPWriter

La característica de JSPWriter es que guarda el documento en un buffer y no lo envía hasta que no este generado completo el documento. El comportamiento de JSPWriter mediante directivas puede ser modificado.

Para esta regla hay dos opciones:

1. Cuando se quiera poner en la salida la cadena <% habrá que utilizar el siguiente símbolo <\%
2. Respecto a los comentarios

    Comentario HTML:<!--Contenido-->

    Comentario JSP:<%--comentario--%>

Los comentarios de JSP no son mandados al browser

Las expresiones

El valor de la expresión se va a sustituir por una llamada al método:

void <JSPWriter>.print(Object b)

void <JSPWriter>.print(int i)

El método print está sobrecargado para objetos o para tipos fundamentales de datos mientras que al método write sólo se le puede pasar texto.

Ejemplo

```
Hora actual: <%= new java.util.Date()%>
out.print (new java.util.Date());
```

Las expresiones también se pueden escribir en XML:

```
<JSP:expresion>
código
</JSP:expresion>
```

Variables implícitas

Para simplificar el desarrollo de página JSP, existen una serie de variables implícitas creadas por el servidor de aplicaciones, que podemos utilizar directamente en los bloques de código Java de la página. Son las siguientes:

Requets	Es un objeto de tipo HttpServletRequest que representa la petición al Servlet.
response	Es un objeto de tipo HttpServletResponse que representa la respuesta del Servlet.
session	Es de tipo HttpSession y representa la sesión de un cliente. Cada cliente va a tener asociada una sesión.
out	Es un JSPWriter y sirve para escribir la salida.
application	Es un objeto de tipo ServletContext. En cada aplicación sólo existe un objeto application
config	Es un objeto de tipo ServletConfig, es un objeto que se corresponde con la inicialización.
pageContext	Es un objeto que introduce JSP para acceder atributos de la página y también va a permitir colocar parámetros compartidos. Es una instancia de la clase PageContext.
page	Es un alias de this.

Scriptlets

Los Scriptlets representan los bloques de código de la página. Como sabemos, se representan así:

<% código %>

Se pueden intercalar scriptlets con texto HTML para así generar dinámicamente la página:

```
<%
int dia = new java.util.Date().getDay();
if (dia == 1)
%>
Hoy toca cobrar. Enhorabuena!!
<% }
else {
%>
Hace <%=dia - 1 %> días que cobraste
<% } %>
```

Los Scriptles también se pueden poner en XML:

```
<JSP:scriptlet>
 Código
</JSP:scriptlet>
```

Declaraciones

Las declaraciones son códigos Java que se quieren poner en la clase del Servlet pero fuera del método _JSPService(). En concreto estas declaraciones se van a poner a nivel de métodos o atributos de la clase. El marcador es el siguiente:

<%! declaracion %>

Ejemplo

```
<HTML>
  <HEAD>
        <TITLE> Contador de Visitas </TITLE>
  </HEAD>
  <BODY>
        <%! private int contador = 0; %>
        Accesos desde la última vez que se cerró el servidor:
        <%=++contador%>
  </BODY>
</HTML>
```

Las declaraciones también se pueden poner en XML:

```
<JSP:declaration>
declaración
</JSP:declaration>
```

Errores de compilación

Cuando se realizan páginas JSP se pueden producir tres tipos de errores:

- Errores de compilación: son errores de sintaxis.
- Errores lógicos: la compilación de los módulos es correcta pera la ejecución se produce de forma no deseada.
- Excepciones: son errores en tiempo de ejecución que impiden que el programa siga adelante.

En los marcadores JSP (se ha cerrado mal un marcador): son fáciles de detectar, lo indica el generador de Servlet. El error lo envía el Servlet Engine a un fichero o bien de la consola. Si se produce un error de compilación hay que enviar un código error 500 Syntax Error.

En el código de los scriptles: en este caso el mensaje de error es más confuso, porque indica la línea de JSP donde está el error.

Gestión de excepciones

Las excepciones se pueden capturar en JSP de una forma más sencilla, utilizando una directiva.

Mediante erroPage, indicamos al contenedor la página a la que debemos redirigir al usuario si se produce la excepción:

```
<%@ page errorPage "mensajeerror.JSP" %>
```

mensajerror.JSP

```
<%@ page isErrorPage = "true" %>
<%out.print((exception) %>
<% if (exception instanceof SocketException) %>
```

La directiva include

<%@ include file="URL" %>

La directiva include permite incluir el contenido de otro fichero dentro del fichero JSP. La ventaja de esta directiva es que bastará con modificar el fichero especificado en include para recoger en toda la aplicación el cambio realizado.

Un problema que encontramos es que si se modifica un fichero incluido, los ficheros que lo incluyen no quedan modificados:

1. touch *.JSP (unix), que permite actualizar la fecha de los ficheros que se le indican. Como ha cambiado lo vuelve a recompilar y recoge las modificaciones incluidas dentro del fichero de la directiva include.

2. En Windows se tendrán que grabar de nuevo todos los ficheros, así se actualiza la fecha y se recompilan los módulos JSP.

Acciones

Las acciones permiten reutilizar componentes de uso común para evitar tener que hacer la misma operación múltiples veces. También se les denomina tags ya que la operación resolver, se indica utilizando un marcador XML. Las acciones sólo están vigentes en Servlet Engine con versión de JSP 1.1.

Los tags se subdividen en dos:

- Tags de sistema, son tags que vienen junto a la especificación de JSP y todos los Servlets Engine los tienen, son estándar. Utilizan el namespace JSP.

```
<JSP: nombre-accion>
. . .
</JSP:nombre-accion>
```

- Tags personalizados, es la forma que tienen los programadores de añadir pequeños programas que luego reutilizan los diseñadores de páginas webs.

```
<nomina:lista>
    .
    .
    .
</nomina:lista>
```

La acción <JSP:include>

La acción <JSP:include>, tiene el siguiente formato:

```
<JSP:include page="URL" flush="true">
```

Incluye un fichero (URL) en el punto donde aparezca esta orden, cuando el browser realice la petición al servlet engine, a diferencia de la directiva que lo realiza antes de la generación del servlet.

El atributo flush obligatoriamente tiene que estar a true.

Con esta acción no es necesario recompilar el JSP cuando se modifique el fichero, debido a que la inclusión en ejecución el fichero incluido no puede tener JSP ya que no se recompila.

Ventajas de la tecnología JSP

La tecnología JSP tiene diferentes ventajas. Conozcamos cuáles son éstas:

ASP es una tecnología similar de Microsoft. Las ventajas de JSP están duplicadas:

1. La parte dinámica está escrita en Java, no en Visual Basic, otro lenguaje específico de MS, por eso es mucho más poderosa y fácil de usar.
2. Es portable a otros sistemas operativos y servidores Web.

Características de la Tecnología JSP y Componentes

Con relación a Active Server Pages (ASP).

ASP es una tecnología similar de Microsoft. Las ventajas de JSP están duplicadas:

1. La parte dinámica está escrita en Java, no en Visual Basic, otro lenguaje específico de MS, por eso es mucho más poderosa y fácil de usar.
2. Es portable a otros sistemas operativos y servidores Web.

Con relación a los Servlets.

Es mucho más conveniente escribir (y modificar) HTML normal que tener que hacer un billón de sentencias println que generen HTML. Los expertos en diseño de páginas Web pueden construir el HTML, dejando espacio para que los programadores de servlets inserten el contenido dinámico.

Con relación a Server-Side Includes (SSI).

SSI es una tecnología ampliamente soportada que incluye piezas definidas externamente dentro de una página Web estática. JSP es mejor porque nos permite usar servlets en vez de un programa separado para generar las partes dinámicas.

Con relación JavaScript.

JavaScript puede general HTML dinámicamente en el cliente. Ésta una capacidad útil, pero sólo maneja situaciones donde la información dinámica está basada en el entorno del cliente.
JavaScript no puede acceder a los recursos en el lado del servidor, como bases de datos, catálogos, información de precios, etc.