¿ QUÉ ES JSP?

Java Server Pages: Páginas de Servidor Java

¿Qué es una tecnología del lado del servidor?

Es una tecnología que nos va a permitir crear Aplicaciones Web.

| ¿Qué es una Aplicación Web? | ¿Qué es una página Dinámica? |

JSP es una tecnología que nos va a permitir crear aplicaciones y sitios web, cuyas páginas se generan dinámicamente dependiendo de la acción del usuario, se generarán unas u otras.

Un ejemplo sería realizar una búsqueda en internet y que el resultado de esa búsqueda nos lance una opción de disponibilidad variable debido a que mi solicitud no ha podido ser cubierta, es decir, si le decimos al navegador que queremos ir a "China" en una fecha en concreto, el buscador de la web que hayamos elegido, nos redirigirá a una ***página Dinámica*** en la que nos ofrecerá una "alternativa" a dicha solicitud.

Esto es lo que pueden realizar tecnologías como ***PHP***, ***ASP*** y ***JSP*** del lado del servidor en una ***Aplicación Web*** con una Web que crea ***páginas Dinámicas*** en tiempo real.

Ahora vamos a ver qué ocurre entre bambalinas....

Cuando nosotros estámos programando en ***JSP***, y creamos un formulario, cuadros de texto, menús desplegables, etc.

Al escoger y enviar, lo que realmente ocurre es que estamos enviando información al servidor, y este consulta a la ***BBDD*** para ver si hay disponibilidad, a su vez, la ***BBDD*** envía la información de vuelta al Servidor; y éste genera la página dinámica, donde ubicará toda la información obtenida.

Resumiendo:

- Es un ***código Java*** que se ejecuta en un servidor.
- Lee las acciones del usuario casi siempre desde formularios ***HTML***.
- Devuelve una ***página HTML*** que se genera de forma dinámica dependiendo de las acciones del usuario.

El ***código JSP*** está alojado en un servidor.

Nunca estará alojado en nuestro ordenador local.

Introducción JSP

JSP (JavaServer Pages) El denominado contenedor, un componente del ***servidor web*** que se encarga de tomar la página web, sustituye el ***código Java*** por el resultado devuelto tras la petición del servidor a la base de datos.

"Mezcla ***HTML*** estático con ***HTML*** generado dinámicamente. Se escribe el ***HTML*** regular de la forma normal, usando cualquier herramienta de construcción web y se encierra el código de las partes dinámicas en unas etiquetas especiales, la mayoría de las cuales empiezan con <% y terminan con "%>"" .

<%@page contentType="text/html"%>

<%@page pageEncoding="UTF-8"%>

Veamos un ejemplo sencillo de página JSP:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Mi primera página JSP</title>
</head>
<body>
<h1> Hoy es:
<%= new java.util.Date() %>
</h1>
</body>
</html>
```

Esto es una página sencilla que se puede utilizar en cualquier aplicación web, al contrario de lo que sería con los ***Servlets***, que deben ir en un ***directorio específico***.

La última versión de la especificación ***JSP es la 2.0***, aunque es de reciente aparición (***Tomcat 4.x*** implementa la versión anterior, ***la 1.2***). Es una especificación paralela al ***API*** de ***servlets***, concretamente a la ***versión 2.3***. Puede encontrar más información sobre JSP en:

http://java.sun.com/products/jsp

Aunque ***JSP*** y ***servlets*** parecen tecnologías distintas, internamente el servidor web traduce el ***JSP*** a un ***servlet***, lo compila y finalmente lo ejecuta cada vez que el cliente solicita la página ***JSP***.

Sus características los hacen apropiados para distintas tareas

Los ***JSP*** son mejores para generar páginas con gran cantidad de contenido estático.

Los ***SERVLETS*** que realizan la misma función deben incluir gran cantidad de sentencias del tipo `out.printLn()` para producir el ***HTML***.

Los ***Servlets*** son mejores en tareas de poca salida, datos binarios, o páginas de contenido variable.

En proyectos complejos lo recomendable es combinar ambas tecnologías: Los ***servlets*** para procesar información y los ***JSP*** para presentar los datos a los clientes.

Traducción de los JSP a Servlets

Como ya se ha comentado, la primera vez que se solicita una página JSP, el servidor genera un servlet equivalente, lo compila y lo ejecuta. Para siguientes solicitudes, solo será necesario ejecutar el código compilado.

El servlet generado de manera automática tiene un método `_jspService` que es el equivalente al service de los ***servlets*** "generados manualmente". En este método es donde se genera el ***código HTML***, mediante instrucciones `printLn` y donde se ejecuta el ***código Java*** insertado en la página.

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse  response)
                        throws java.io.IOException, ServletException {
   JspWriter out = null;
   response.setContentType("text/html;ISO-8859-1");
   out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0
               Transitional//EN\">);
   out.println("<html>");
   out.println("<head>");
   out.println("<title>Mi primera pagina JSP</title>");
   out.println("</head>");
   out.println("<body>");
   out.print("Hoy es ");
   out.println(new java.util.Date());
   out.println("</body>");
   out.println("</html>");
}
```

El directorio donde se coloca el servlet generado, así como su nombre, depende del servidor web, por ejemplo, ***Tomcat*** utiliza su directorio

`work/Catalina/localhost/aplicacion_web.`

En caso de que la página esté en ***webapps/ROOT***, el nombre de la aplicación se sustituye por un guion bajo (**_**).

Software Necesario

Java JEE IDE (Eclipse, NetBeans, etc.).

Necesitamos un entorno de desarrollo llamado ***IDE***, que puede ser lo que conocemos ***Eclipse***, ***NetBeans***, etc. Pero, muy importante, ese ***IDE*** debe ser la ***versión JEE***. que como recordamos son las:

Java Enterprise Edition : Java Ediciones Empresariales

Estas aplicaciones están compuestas de tres partes:

CLIENTE --> SERVIDOR --> BBDD

Una vez que ya estamos trabajando con un entorno de desarrollo ***JEE*** vamos a instalar un ***servidor web***, y en este caso vamos a elegir ***Tomcat***.

¿Porqué Tomcat? Porque tiene que ser un servidor que entienda ***JSP***.

Antiguamente, ***Tomcat*** era una ***librería, plugin o motor*** que permitía a un servidor previamente instalado entender ***JSP.***

Desde hace un tiempo dejó de serlo y ahora se puede simular que nuestro ordenador es remoto y poder probar nuestras aplicaciones antes de subirlas a un servidor real.

Elementos de JSP

Existen tres tipos de elementos que podemos insertar:

- ***Código*:** podemos "incrustar" código Java de distintos tipos (declaraciones de variables y/o métodos, expresiones,sentencias) para que lo ejecute el ***contenedor JSP.***
- ***Directivas*:** Permiten controlar distintos parámetros del servlet resultante de la traducción automática del JSP
- ***Acciones***: normalmente sirven para alterar el flujo normal de ejecución de la página (p.eje.redirecciones), auqnue tienen usos variados.

Se pueden poner comentarios en una página JSP entre los símbolos <%--y--%>. El contenedor JSP ignorará todo lo contenido entre ambos. Dentro de los fragmentos de código Java también se pueden colocar comentarios siguiendo la sintaxis habitual del lenguaje Java.

Inserción de código en páginas JSP

Hay tres formas de insertar código Java en una página JSP

- ***Expresiones:*** de la forma `<%=expresión%>` : en este caso, la expresión se evalúa, su resultado se convierte a `string` y se inserta en la salida.
- ***Scriptlests:*** de la forma `<% código %>` el código se ejecuta dentro del método _jspService del servlet generado.

- ***Declaraciones:*** de la forma `<%! código %>` se insertan en el cuerpo del servlet generado, fuera de sus métodos.

1. *Expresiones*. tras convertir en un `string` (El objeto predefinido `out`). La forma de traducir una expresión a código de servlet es imprimiendo en `out`(mediante sentencia `out.write(expresion)`) o similar.

2. *Scriptlets*. Permiten ejecutar código arbitrario, cuyo resultado no es necesario enviar a la salida. Si desde un ***scriptlet*** se desea escribir algo en ésta, bastará con utilizar el objeto predefinido en `out`. Un uso común de los ***scriptlets*** es hacer que ciertas partes de código ***HTML*** aparezcan o no en función de una condición. Por ejemplo:

```
<%
  java.util.Calendar ahora = java.util.Calendar.getInstance();
  int hora = ahora.get(java.util.Calendar.HOUR_OF_DAY);
%>
<b> Hola mundo, <i>
<% if ((hora>20)||(hora<6)) { %>
     buenas noches
<% }
   else if ((hora>=6)&&(hora<=12)) { %>
          buenos días
<%      }
        else { %>
           buenas tardes
<%      } %>
</i> </b>
```

3.Declaraciones. Permite definir variables o métodos que se insertarán dentro del cuerpo del servlet generado. Esto da la posibilidad de sobreescribir los métodos `jsInit` y `jsDestroy` que son el equivalente en ***JSP*** del `init` y `destroy` de los servlets. Las variables declaradas conservarán su valor entre sucesivas llamadas a la página, ya que son variables miembros del ***servlet*** y no locales al método `jspService`. Esto nos permite, por ejemplo, creare un contador de acceso a la página:

```
<%! private int accesos = 0; %>
<h1> Visitas: <%= ++accesos %> </h1>
```

JavaBeans: Es un componente software reutilizable escrito en ***Java***. un ***Bean*** no es más que una clase ***Java*** escrita siguiendo unas ciertas convenciones. Convenciones que hacen posible que herramientas automáticas puedan acceder a sus propiedades y manipularlas sin necesidad de modificar el código.

esto puede servir en caso de un ***IDE***, por ejemplo, para realizar "programación visual". En ***JSP*** el uso principal de los ***Beans*** es manipular componentes ***Java***sin necesidad de incluir código en la página, accediendo a sus propiedades mendiante etiquetas.

El uso de ***Beans*** en páginas ***JSP*** ofrece diversas ventajas con respecto al uso directo de ***código Java***:

A. Se evita el uso de ***sintaxis Java***, en su lugar se emplean etiquetas con ***sintaxis XML***. Esto permite separar más fácilmente el trabajo de ***programadores*** y ***diseñadores web.***

B. Se simplifica la creación y uso de objetos compartidos entre varias páginas.

C. Se simplifica la creación de objetos a partir de los parámetros de la petición.

Ventajas de JSP

- **Contra Active Server Pages (ASP)**

ASP es una tecnología similar de Microsoft. Las ventajas de ***JSP*** están duplicadas. Primero, 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, segundo, es portable a otros sistemas operativos y servidores Web.

JSP --> escribe una vez, ejecuta donde quieras-.

ASP--> limitada para arquitecturas basadas en tecnología Microsoft-.

- ***Contra los Servlets***

JSP no nos da nada que no pudieramos en un principio hacer con un servlet. Pero es mucho más conveniente escribir (y modificar!) ***HTML***normal que tener que hacer un billón de sentencias println que generen ***HTML***. Además, separando el formato del contenido podemo poner diferentes personas en diferentes tareas: nuestros expertos en diseño de páginas ***Web*** pueden construir el ***HTML***, dejando espacio que nuestros programadores de servlets inserten el contenido dinámico

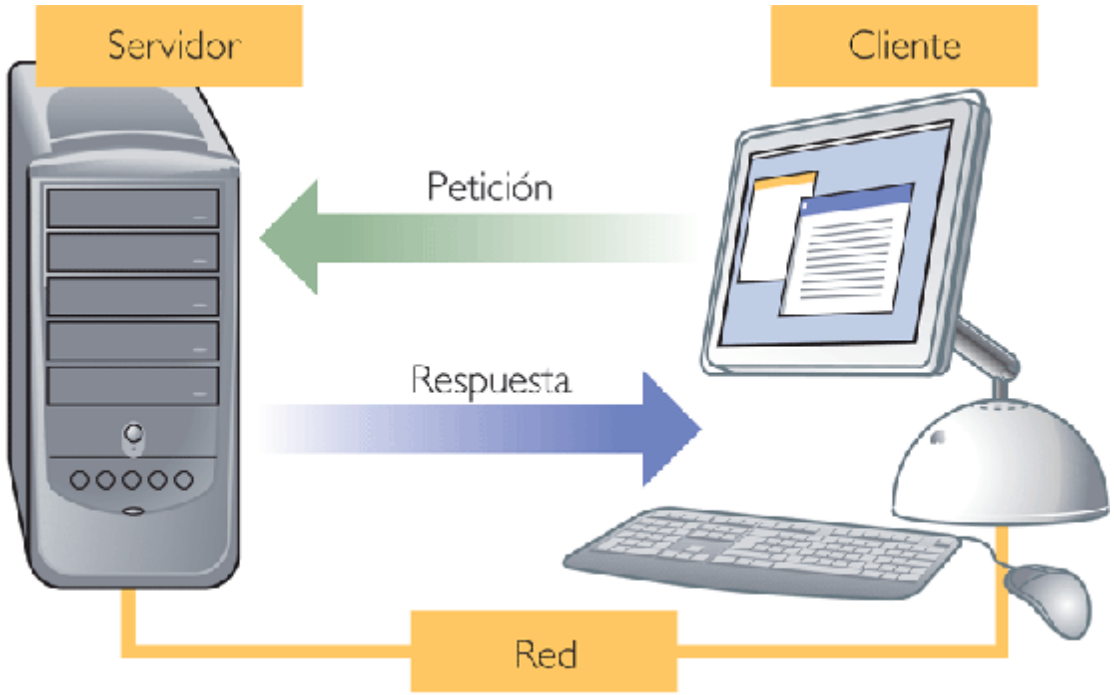
- ***Contra JavaScript***

JavaScript es ejecutado en la máquina cliente cuando ésta se descarga en la página web, mientras que ***JSP*** se ejecuta en el servidor, devolviendo al cliente el resultado final de la página.

El ***JavaScript*** no puede usar clases java que la máquina cliente tenga instaladas, mientras que con ***JSP*** se pueden usar todas las clases Java que el servidor tenga.

JavaScript no está diseñado para acceder a ***BBDD***, con ***JSP***, usando ***JDBC***, clases específicas o los llamados ***JavaBeans***, sí que buedes acceder fácilmente a ***BBDD***.

A nivel programación son muy similares, basándose los dos en el lenguaje ***Java***. La única diferencia es que el ***JavaScript*** solo puedes usar los objetos definidos dentro de los navegadores, mientras que ***JSP*** puede acceder a todos los objetos y clases que existan en el servidor.


Servidor
Cliente
Petición
Respuesta
Red