UT04_01 Introducción a Java Server Pages (JSP)

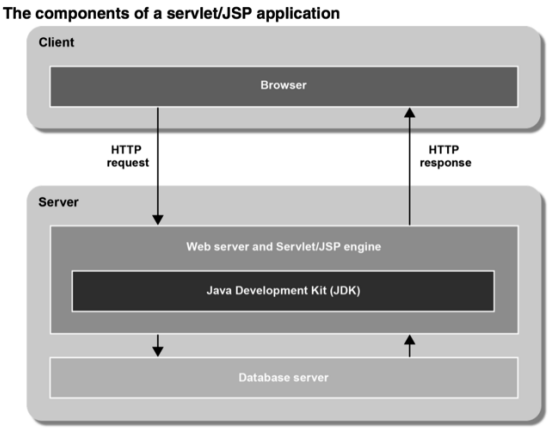
1. Repaso.

Hoy en día, puede encontrar la tecnología Java en redes y dispositivos que comprenden desde Internet y superordenadores científicos hasta portátiles y teléfonos móviles, pasando por simuladores de mercado, juegos y tarjetas de crédito. Proporciona la posibilidad de crear soluciones empresariales seguras, independientes de plataforma, robustas y escalables. Está tecnología está soportada por una comunidad de usuarios que la enriquecen y mejoran abriendo posibilidades en: Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma.

- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, formularios HTML…
- Desarrollar eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier dispositivo digital.

Definiendo algunos conceptos:

JDK o SDK: Es el ambiente en el cual es posible desarrollar cualquier aplicación Java que incluye; el API de Java, el compilador de Java, así como el JVM (Java Virtual Machine) de la plataforma correspondiente.



JDBC: La tecnología JDBC (Java DataBase Connectivity) es una interface de programación de aplicaciones (API) que permite acceder, desde el lenguaje de programación Java, a virtualmente cualquier fuente de datos tabulados.

JSP (Java Server Pages): Este es un tipo de programa Java que contiene HTML, para ejecutar un JSP se requiere de un servlet engine como Tomcat. JSP separa la interface de usuario de la generación de contenidos, permitiendo cambiar el formato de la página sin alterar el contenido dinámico subyacente.

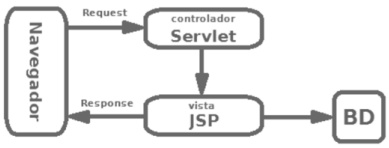
Servlet: Un servlet es una clase (como un applet) Java que se ejecuta en un servidor de aplicaciones. Antes de los servlets, la creación de contenidos dinámicos se hacía a través de CGI. Un servlet generalmente es utilizado para procesar solicitudes de usuarios, para procesos de verificación de usuarios, generación de contenido dinámico. Similar a un JSP, un JSP se convierte eventualmente en un Servlet, la diferencia es que un Servlet solo contiene lenguaje Java desde el inicio, mientras que un JSP contiene Java y HTML.

En resumen:

✓JSP: facilita la presentación de los documentos. Su objetivo es estructurar la interfaz de usuario, de una forma clara y ordenada.

✓Servlet: son programas que se ejecutan en el lado del servidor para generar páginas Web. Provee de una mayor flexibilidad en la programación de la lógica de negocio.

Modelo vista controlador JSP y Servlet:



2. Introducción y ejemplos.

La siguiente tabla enumera las otras ventajas de usar JSP sobre otras tecnologías:

frente a páginas Active Server (ASP) : Las ventajas de JSP son dobles. En primer lugar, la parte dinámica está escrita en Java, no en Visual Basic u otro lenguaje específico de MS, por lo que es más potente y fácil de usar. En segundo lugar, es portátil para otros sistemas operativos y servidores web que no sean de Microsoft.

contra Servlets puros: Es más conveniente escribir (y modificar) HTML normal que tener muchas instrucciones println que generen el HTML.

f**rente al servidor incluye (SSI):** SSI solo está pensado para inclusiones simples, no para programas "reales" que usan datos de formularios, conexiones de bases de datos y similares.

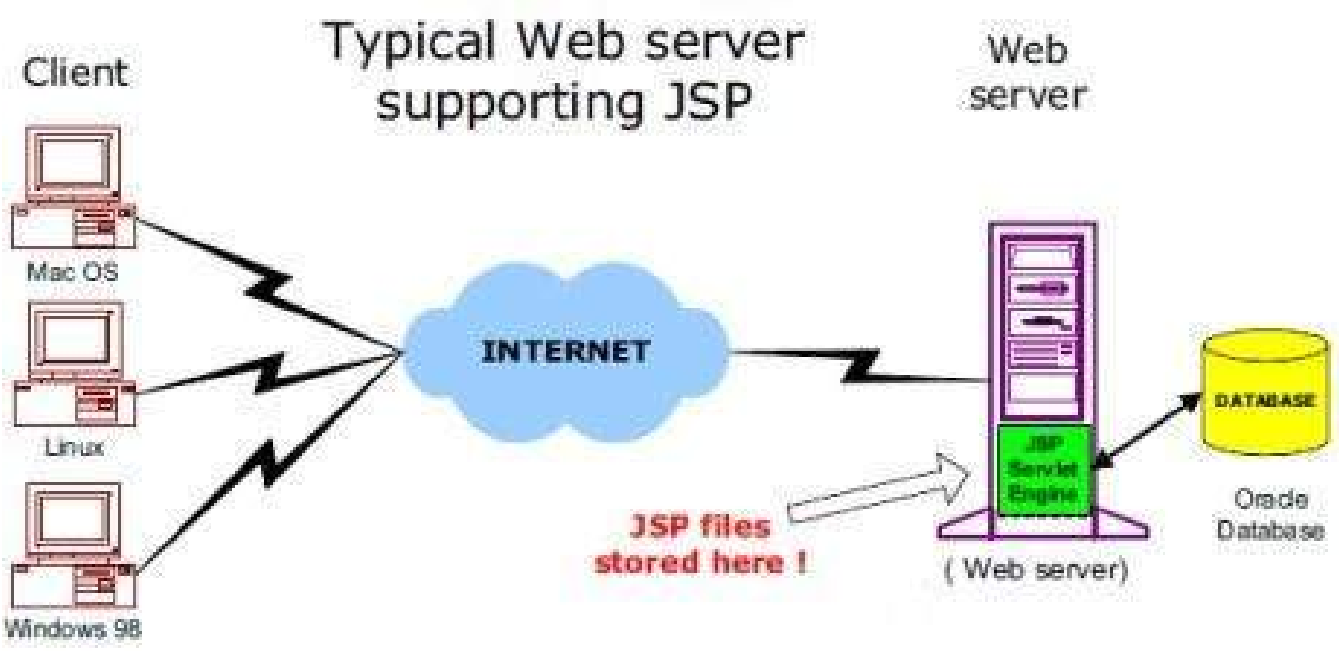
contra JavaScript: JavaScript puede generar HTML dinámicamente en el cliente, pero apenas puede interactuar con el servidor web para realizar tareas complejas como el acceso a la base de datos y el procesamiento de imágenes, etc.

frente a HTML estático: HTML normal, por supuesto, no puede contener información dinámica.

El servidor web necesita un motor JSP, es decir, un contenedor para procesar páginas JSP. El contenedor JSP es responsable de interceptar las solicitudes de páginas JSP. Este tutorial utiliza Apache que tiene un contenedor JSP incorporado para soportar el desarrollo de páginas JSP.

Un contenedor JSP funciona con el servidor web para proporcionar el entorno de tiempo de ejecución y otros servicios que necesita un JSP. Sabe cómo comprender los elementos especiales que forman parte de JSP.

El siguiente diagrama muestra la posición del contenedor JSP y los archivos JSP en una aplicación web.

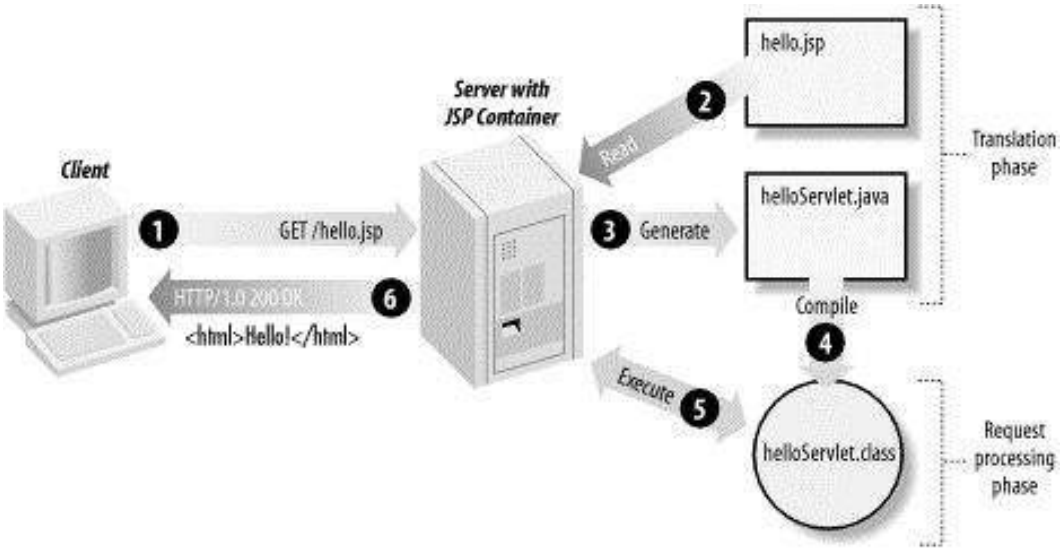


2.1 Procesamiento de JSP

Los siguientes pasos explican cómo el servidor web crea la página web usando JSP:

- Al igual que con una página normal, su navegador envía una solicitud HTTP al servidor web.
- El servidor web reconoce que la solicitud HTTP es para una página JSP y la reenvía a un motor JSP. Esto se hace utilizando la página URL o JSP que termina en **.jsp en** lugar de **.html** .
- **El motor JSP carga la página JSP desde el disco y la convierte en un contenido de servlet.** Esta conversión es muy simple, en la que todo el texto de la plantilla se convierte en instrucciones println () y todos los elementos JSP se convierten en código Java. Este código implementa el comportamiento dinámico correspondiente de la página.
- El motor JSP compila el servlet en una clase ejecutable y reenvía la solicitud original a un motor de servlet.
- Una parte del servidor web llamada servlet engine carga la clase Servlet y la ejecuta. Durante la ejecución, el servlet produce una salida en formato HTML. El motor de servlet transfiere la salida al servidor web dentro de una respuesta HTTP.
- El servidor web reenvía la respuesta HTTP a su navegador en términos de contenido HTML estático.
- Finalmente, el navegador web maneja la página HTML generada dinámicamente dentro de la respuesta HTTP exactamente como si fuera una página estática.

Todos los pasos mencionados anteriormente se pueden ver en el siguiente diagrama:



Normalmente, el motor JSP comprueba si ya existe un servlet para un archivo JSP y si la fecha de modificación en el JSP es anterior al servlet. Si el JSP es anterior a su servlet generado, el contenedor JSP supone que el JSP no ha cambiado y que el servlet generado aún coincide con el contenido del JSP. Esto hace que el proceso sea más eficiente que con los otros lenguajes de scripts (como PHP) y, por lo tanto, más rápido.

Entonces, de alguna manera, una página JSP es realmente solo otra manera de escribir un servlet sin tener que ser un programador de Java. Excepto por la fase de traducción, una página JSP se maneja exactamente como un servlet regular.

2.2 Declaraciones JSP

Una declaración declara una o más variables o métodos que puede usar en el código Java más adelante en el archivo JSP. Debe declarar la variable o el método antes de usarlo en el archivo JSP.

A continuación está la sintaxis de las declaraciones JSP:

```
<%! declaration; [ declaration; ]+ ... %>
```

Puede escribir el equivalente XML de la sintaxis anterior de la siguiente manera:

```
<jsp:declaration>
   code fragment
</jsp:declaration>
```

A continuación se muestra un ejemplo de declaraciones JSP:

```
<%! int i = 0; %>
<%! int a, b, c; %>
<%! Circle a = new Circle(2.0); %>
```

2.3 Expresión JSP

Un elemento de expresión JSP contiene una expresión de lenguaje de scripting que se evalúa, se convierte a String y se inserta donde aparece la expresión en el archivo JSP.

Como el valor de una expresión se convierte en una cadena, puede usar una expresión dentro de una línea de texto, esté o no etiquetada con HTML, en un archivo JSP.

El elemento de expresión puede contener cualquier expresión que sea válida de acuerdo con la Especificación del lenguaje Java, pero no puede usar un punto y coma para finalizar una expresión.

A continuación está la sintaxis de JSP Expression -

```
<%= expression %>
```

Puede escribir el equivalente XML de la sintaxis anterior de la siguiente manera:

```
<jsp:expression>
   expression
</jsp:expression>
```

El siguiente ejemplo muestra una expresión JSP -

```
<html>
   <head><title>A Comment Test</title></head>
   <body>
      <p>Today's date: <%= (new java.util.Date()).toLocaleString()%></p>
   </body>
```

```
</html>
```

El código anterior generará el siguiente resultado:

Today's date: 11-Sep-2010 21:24:25

2.4 Comentarios JSP

El comentario JSP marca texto o declaraciones que el contenedor JSP debe ignorar. Un comentario JSP es útil cuando desea ocultar o "comentar", una parte de su página JSP.

A continuación está la sintaxis de los comentarios de JSP:

```
<%-- This is JSP comment --%>
```

El siguiente ejemplo muestra los comentarios JSP -

```
<html>
   <head><title>A Comment Test</title></head>
   <body>
      <h2>A Test of Comments</h2>
      <%-- This comment will not be visible in the page source --%>
   </body>
</html>
```

El código anterior generará el siguiente resultado:

A Test of Comments

Hay una pequeña cantidad de construcciones especiales que puede usar en varios casos para insertar comentarios o caracteres que de otro modo serían tratados de manera especial. Aquí hay un resumen:

S.No.	Sintaxis y propósito
1	**<% - comentario -%>** Un comentario JSP. Ignorado por el motor JSP.
2	**<! - comentario ->** Un comentario en HTML Ignorado por el navegador.
3	**<\%** Representa static <% literal.
4	**% \>** Representa static%> literal.
5	**\ '** Una comilla simple en un atributo que usa comillas simples.
6	**\ "**

Una comilla doble en un atributo que usa comillas dobles.

2.5 Directivas JSP

Una directiva JSP afecta la estructura general de la clase servlet. Por lo general tiene la siguiente forma:

```
<%@ directive attribute="value" %>
```

Hay tres tipos de etiqueta directiva:

S.No.	Directiva y descripción
1	**<% @ página ...%>** Define atributos dependientes de la página, como el lenguaje de scripts, la página de error y los requisitos de almacenamiento en búfer.
2	**<% @ incluir ...%>** Incluye un archivo durante la fase de traducción.
3	**<% @ taglib ...%>** Declara una biblioteca de etiquetas, que contiene acciones personalizadas, utilizadas en la página

Explicaremos la directiva JSP en un capítulo separado JSP - Directivas

2.6 Acciones JSP

Las acciones JSP usan **construcciones** en sintaxis XML para controlar el comportamiento del motor de servlet. Puede insertar dinámicamente un archivo, reutilizar componentes de JavaBeans, reenviar el usuario a otra página o generar HTML para el complemento de Java.

Solo hay una sintaxis para el elemento Acción, ya que se ajusta al estándar XML:

```
<jsp:action_name attribute="value" />
```

Los elementos de acción son básicamente funciones predefinidas. La siguiente tabla enumera las acciones JSP disponibles:

S.No.	Sintaxis y propósito
1	**jsp: incluir** Incluye un archivo en el momento en que se solicita la página.
2	**jsp: useBean** Encuentra o crea una instancia de JavaBean.
3	**jsp: setProperty** Establece la propiedad de un JavaBean.
4	**jsp: getProperty** Inserta la propiedad de un JavaBean en la salida.

5	**jsp: adelante** Reenvía al solicitante a una nueva página.
6	**jsp: plugin** Genera código específico del navegador que crea una etiqueta OBJECT o EMBED para el complemento de Java.
7	**jsp: elemento** Define elementos XML dinámicamente.
8	**jsp: atributo** Define el atributo del elemento XML definido dinámicamente.
9	**jsp: cuerpo** Define el cuerpo del elemento XML definido dinámicamente.
10	**jsp: texto** Se usa para escribir texto de plantilla en páginas y documentos JSP.

Para más explicaciones sobre acciones de JSP ver TutorialsPoint.

2.7 Objetos implícitos JSP

JSP admite nueve variables definidas automáticamente, que también se llaman objetos implícitos. Estas variables son -

S.No.	Descripción del objeto
1	**solicitud** Este es el objeto **HttpServletRequest** asociado con la solicitud.
2	**respuesta** Este es el objeto **HttpServletResponse** asociado con la respuesta al cliente.
3	**fuera** Este es el objeto **PrintWriter** utilizado para enviar resultados al cliente.
4	**sesión** Este es el objeto **HttpSession** asociado con la solicitud.
5	**solicitud** Este es el objeto **ServletContext** asociado con el contexto de la aplicación.
6	**config** Este es el objeto **ServletConfig** asociado a la página.
7	**pageContext**

Esto encapsula el uso de funciones específicas del servidor como **JspWriters de** mayor rendimiento.

página

8 Esto es simplemente un sinónimo de **esto** , y se usa para llamar a los métodos definidos por la clase de servlet traducida.

Excepción

9 El objeto **Excepción** permite que los datos de excepción sean accedidos por JSP designado.

2.8 Ejemplos: Netbeans, Java, Application Web, Java EE6 para que sea index.jsp

EJEMPLO 1. Hello World

```
<html>
   <head><title>Hello World</title></head>
   <body>
      Hello World!<br/>
      <%
         out.println("Your IP address is " + request.getRemoteAddr());
      %>
   </body>
</html>
```

EJEMPLO 2. Probando el entorno del servidor un script JSP y un script Servlet. Procesos de compilación.

```
——————————Ejemplo código JSP——————————-
<%@ page language='java' contentType="text/html" %>
<%! int count=1; %>
<html>
<head><title>Ejemplo JSP</title></head> <body>
Ha entrado
<%= count++ %>
<% if (count == 1) { %>
vez<% } else { %>
veces <% } %>
</body>
</html>
```

EJEMPLO 3:

```
<%! int day = 3; %>
<html>
```

```
   <head><title>IF...ELSE Example</title></head>
   <body>
      <% if (day == 1 | day == 7) { %>
         <p> Today is weekend</p>
      <% } else { %>
         <p> Today is not weekend</p>
      <% } %>
   </body>
</html>
```

EJEMPLO 4:

```
<%! int day2 = 3; %>
<html>
   <head><title>SWITCH...CASE Example</title></head>
   <body>
      <%
         switch(day2) {
            case 0:
               out.println("It\'s Sunday.");
               break;
            case 1:
               out.println("It\'s Monday.");
               break;
            case 2:
               out.println("It\'s Tuesday.");
               break;
            case 3:
               out.println("It\'s Wednesday.");
               break;
            case 4:
               out.println("It\'s Thursday.");
               break;
            case 5:
               out.println("It\'s Friday.");
               break;
            default:
               out.println("It's Saturday.");
         }
      %>
   </body>
</html>
```

Ejemplo 5:

```
<%@page import="java.util.Enumeration"%>
<html>
<head><title>Un Scriptlet que extrae informaci&oacute;n</title></head>
<body>
<b>Http headers:</b><br/>
<%-- primero --%>
<%
for (Enumeration<String> e = request.getHeaderNames();
e.hasMoreElements(); ) {
String header = e.nextElement();
out.println(header + ": " + request.getHeader(header) + "<br/>");
}
String message = "Eso eso todo";
%>
<hr/>
<%-- segundo --%>
<%out.println(message);%>
</body>
</html>
```