JSP
(JAVA SERVER PAGES)

Eduard Lara

INDICE

1. Introducción a JSP
2. Ciclo de vida de las páginas JSP
3. Incluir código Java en un JSP
4. Directivas
5. Acciones
6. Objetos implícitos de JSP

1. INTRODUCCIÓN A JSP

- La tecnología JSP está basada en el lenguaje de programación Java y encaminada a facilitar el desarrollo de sitios web.
- Mediante el uso de páginas JSP podemos incorporar contenido dinámico en sitios web mediante código Java embebido a través de etiquetas especiales < % % >.
- Las páginas JSP son archivos de texto con extensión .jsp que contienen etiquetas HTML, junto con código Java embebido, que permite el acceso de la página a datos desde ese código Java ejecutado en el servidor.

1. INTRODUCCIÓN A JSP

- Cuando se solicita una página JSP, la parte HTML se procesa en el cliente, sin embargo, el código Java se ejecuta en el momento de recibir la petición y el contenido dinámico generado por ese código se inserta en la página antes de devolverla al usuario.
- Esto proporciona una separación entre la parte de presentación HTML de la página y la parte de lógica de programación incluida en el código Java.

2. CICLO DE VIDA DE LAS PÁGINAS JSP

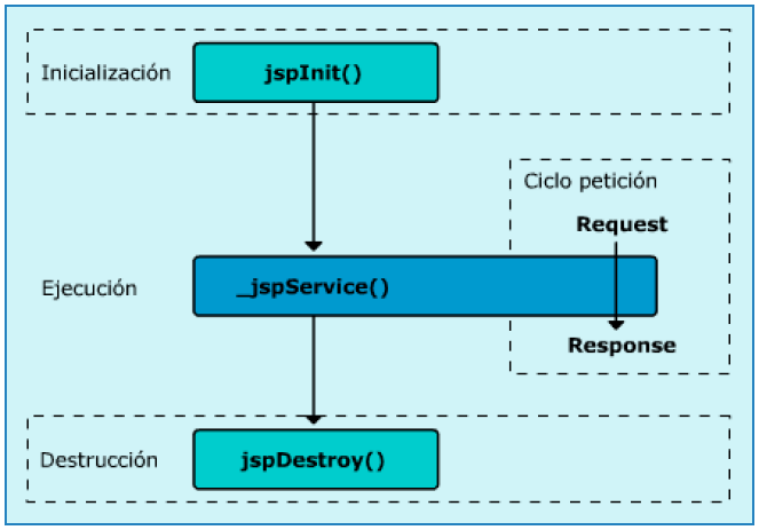
El tratamiento de la petición HTTP que se hace en el servidor web hasta que se devuelve la respuesta al cliente se resume en 4 pasos:

- El motor JSP analiza la página solicitada y crea un fichero .java correspondiente al servlet.
- El servlet generado se compila para obtener el archivo .class, que pasa al control del motor servlet, que lo ejecuta del mismo modo que si se tratase de cualquier otro servlet.
- El motor servlet carga la clase del servlet generado para ejecutarlo.
- El servlet se ejecuta y devuelve su respuesta al solicitante.

2. CICLO DE VIDA DE LAS PÁGINAS JSP

El servlet se genera con los siguientes métodos:

- jspInit(); Inicializa el servlet generado y sólo se llama en la primera petición
- jspService(request, response); Se invoca en cada petición, incluso en la primera. Es quien se encarga de manejar las peticiones.
- jspDestroy(); Invocada por el motor para eliminar el servlet.



3. INCLUIR CÓDIGO JAVA EN UN JSP

Podemos "incrustar" código Java de distintos tipos (declaraciones de variables y/o métodos, expresiones, sentencias) para que lo ejecute el contenedor JSP.

Hay varias formas de insertar código Java en una página JSP:

- 3.1. Declaraciones
- 3.2. Expresiones
- 3.3. Scriptlets
- 3.4. Comentarios

3.1. DECLARACIONES

- Las declaraciones son elementos que se utilizan para declarar una variable o un método que se insertarán dentro del cuerpo del servlet generado.
- No generarán ninguna salida, por lo que pueden utilizarse conjuntamente con otros elementos de las páginas JSP.
- Su sintaxis es la siguiente:

```
<%! DeclaraciónClaseJava %>
```

3.1. DECLARACIONES

```
<%! public static final String DEFAULT_NAME = "World"; %>

<%! public String getName(HttpServletRequest request) {
      return request.getParameter("name");
    }
%>

<%! int counter = 0; %>
```

- Una declaración de una variable o método solamente es válida para la página JSP en la que se ha declarado.
- Las variables conservarán su valor entre sucesivas llamadas a la página, ya que son variables miembro del servlet.

3.2. EXPRESIONES

- Las expresiones son un mecanismo que evita tener que escribir el código completo de la sentencia out.println().
- Su sintaxis:

```
<%= ExpresiónJava %>
```

```
La fecha y hora actual es: <%= new java.util.Date() %>
```

- Las expresiones que se incluyen en estos elementos son evaluados, convertidos a un objeto de tipo String e incluidos como parte del código HTML en el lugar donde aparece en la página JSP.
- Las expresiones están orientadas a la generación de datos.

3.3. SCRIPTLETS

- Un scriptlet es un bloque de código Java insertado en la página y ejecutado durante el procesamiento de la respuesta (en el motor JSP).
- El resultado del código Java no es necesario enviarlo a la salida.
- Su sintaxis es la siguiente:

```
<% CódigoJava %>
```

3.3. SCRIPLETS

- Los scriptlets no están limitados a una sola línea, pueden ocupar varias.

```
<% if ( i > 10 ) { %>
        Soy un número alto.
<% } else { %>
        Soy un número bajo.
<% } %>
```

- Un uso común de los scriptlets es hacer que ciertas partes de código HTML aparezcan o no en función de una condición.

3.4. COMENTARIOS

La documentación es importante para cualquier aplicación. Las páginas de JSP admiten tres tipos de comentarios:

1) Comentarios HTML

Los comentarios HTML se consideran texto de plantilla de HTML. Estos comentarios se envían en el flujo de respuesta HTTP. Por ejemplo:

```
<!-- Esto es un comentario HTML. Aparecerá en la respuesta. -->
```

3.4. COMENTARIOS

2) Comentarios de página de JSP

Los comentarios de página de JSP sólo se ven en el archivo de la página de JSP. Estos comentarios no se incluyen en el código fuente del servlet durante la fase de conversión ni aparecen en la respuesta HTTP. Por ejemplo:

```
<%-- Esto es un comentario de JSP. Sólo se verá en el código JSP.
     No aparecerá en el código del servlet ni en la respuesta.
--%>
```

3.4. COMENTARIOS

3) Comentarios Java

Se pueden incrustar comentarios Java con etiquetas de scriptlet y de declaración. Estos comentarios se incluyen en el código fuente del servlet durante la fase de conversión, pero no aparecen en la respuesta HTTP. Por ejemplo: fuente del servlet durante la fase de conversión ni aparecen en la respuesta HTTP. Por ejemplo:

```
<%
  /* Esto es un comentario Java. Aparecerá en el código de servlet.
     No aparecerá en la respuesta. */
%>
```

4. DIRECTIVAS

Las directivas proporcionan información global de la página utilizada en la fase de traducción. Se utilizan para definir y manipular una serie de atributos dependientes de la página que afectan a todo el JSP y, por lo tanto, influyen en la estructura que tendrá el servlet generado. Para indicar la presencia de una directiva se utiliza el signo de la arroba.

```
<%@ NombreDirectiva [attr="valor"]* %>
```

Hay tres tipos de directivas:

- 4.1. page
- 4.2. include
- 4.3. taglib

4.1. DIRECTIVA PAGE

- La directiva page se emplea para especificar atributos para toda la página JSP en su conjunto.
- Su sintaxis es:

  <%@ page [atributo="valor" atributo=valor ...]%>
- Aunque puede haber varias directivas page, sólo se puede declarar un determinado atributo una vez por página.
- Esto se aplica a todos los atributos excepto import.
- Una directiva page puede situarse en cualquier lugar del archivo JSP.
- Es conveniente que la directiva page sea la primera sentencia del archivo JSP.

4.1. DIRECTIVA PAGE

La directiva page define algunas propiedades dependientes de la página y las comunica al contenedor web durante la conversión.

- El atributo **language** especifica el lenguaje de secuencia de comandos que se debe emplear en la página. El único valor actualmente definido es java, el predeterminado.
- El atributo **extends** determina el nombre de clase (completo) de la superclase de la clase de servlet que se genera con la página de JSP.
- El atributo **buffer** define el tamaño del búfer empleado en el flujo de salida (un objeto JspWriter). Su valor es none o Nkb. El tamaño de búfer predeterminado es kilobytes (Kbytes) o más. Por ejemplo: buffer="8kb" o buffer="none".

4.1. DIRECTIVA PAGE

- El atributo **autoFlush** determina si la salida del búfer debe borrarse automáticamente cuando se llena el búfer o si se lanza una excepción. Su valor es true (borrado automático) o false (lanzar una excepción). El valor predeterminado es true.
- El atributo **session** determina si la página de JSP interviene en una sesión HTTP. Su valor es true (predeterminado) o false.
- El atributo **import** define del conjunto de clases y paquetes que deben importarse en la definición de clase de servlet. El valor de este atributo es una lista delimitada por comas de nombres de clase o paquetes completos. Por ejemplo: import="java.sql.Date,java.util.*,java.text.*"

4.1. DIRECTIVA PAGE

- El atributo **isThreadSafe** permite declarar si la página de JSP estáprotegida ante subprocesos. Si el valor se define en false, este atributo instruye al analizador JSP para que escriba el código de servlet de manera que sólo se procese simultáneamente una solicitud HTTP. El valor predeterminado es true.
- El atributo **info** define una cadena informativa sobre la pág JSP.
- El atributo **contentType** define un tipo MIME del flujo de salida. El valor predeterminado es text/html.
- El atributo **pageEncoding** determina la codificación de caracteres del flujo de salida. El valor predeterminado es ISO-8859-1. Otras codificaciones de caracteres permiten incluir juegos de caracteres no latinos, como kanji o cirílico.

4.1. DIRECTIVA PAGE

- El atributo **isELIgnored** especifica si se omiten los elementos de lenguaje de expresiones de la página. Su valor es true o false (predeterminado). Si se define en true, no se evalúa el lenguaje de expresiones de la página.
- El atributo **isErrorPage** establece que la página de JSP se ha diseñado para ser el destino del atributo errorPage de otra página de JSP. Su valor es true o false (predeterminado). Todas las páginas de JSP erróneas tienen acceso automático a la variable implícita exception.
- El atributo **errorPage** indica otra pág. JSP que manejará todas las excepciones de tiempo de ejecución lanzadas por esta pág. JSP. El valor es una URL relativa a la jerarquía web actual o a la raíz de contexto. Por ejemplo, errorPage="error.jsp" (es relativa a la jerarquía actual) o errorPage="/error/formErrors.jsp" (es relativa a la raíz de contexto de la aplicación web).

4.2. DIRECTIVA INCLUDE

- La directiva include permite incluir código JSP en una página en tiempo de compilación. El código puede ser una página HTML, un archivo java, un fichero de texto u otra página JSP.
- La página final que va a procesar el motor JSP es la formada por la página base más el contenido del fichero que se haya incluido.
- Sintaxis:
    - <%@ include file="nombre del fichero" %>
- Una vez incluido el fichero, si se modifica el fichero no se verá reflejado en el servlet.
- Se suele utilizar para incluir cabeceras y pies de página estándar o cualquier otro texto en formato común en las páginas JSP.

4.3. DIRECTIVA TAGLIB

La directiva taglib permite extender los marcadores de JSP con etiquetas o marcas generadas por el propio usuario (etiquetas personalizadas). Se hace referencia a una biblioteca de etiquetas que contiene código Java compilado definiendo las etiquetas que van a ser usadas y que han sido usadas y que han sido definidas por el usuario.

- Sintaxis:

  <%@ taglib uri="taglibraryURI" prefix="tagPrefix"%>

- La variable uri hace referencia a la dirección que identifica a la biblioteca de etiquetas.
- prefix define el prefijo que se coloca a cada una de las etiquetas de la biblioteca, que se utiliza para distinguir las etiquetas personalizadas.

5. ACCIONES

- Normalmente sirven para alterar el flujo normal de ejecución de la página (p.ej. redirecciones), aunque tienen usos variados.
- Las acciones proporcionan al motor JSP información sobre lo que debe hacer a la hora de procesar la página.
- Las acciones son marcas estándar, con formato XML, que afectan al comportamiento en tiempo de ejecución del JSP y la respuesta se devuelve al cliente.
- Hay acciones predefinidas y también se pueden incorporar nuevas acciones personalizadas (incluidas a través de la directiva taglib).

5. ACCIONES

- Sintaxis:

  <nombre_etiqueta [atr="valor" atr="valor"...]>

  </nombre_etiqueta> ó

  <nombre_etiqueta [atr="valor" atr="valor"...] />

- Tenemos siete tipos de acciones diferentes:
  - useBean
  - setProperty
  - getProperty
  - include
  - forward
  - param
  - plugin

5.1. ETIQUETA USEBEAN

- Si quiere interactuar con un componente JavaBeans utilizando las etiquetas estándar en una página de JSP, primero debe declarar el bean.
- Para ello se utiliza la etiqueta estándar useBean.
- La sintaxis de la etiqueta useBean es:

```
<jsp:useBean id="NombreBean"
        scope="page | request | session | application"
        class="NombreClase" />
```

- El atributo **id** especifica el nombre de atributo del bean. El atributo **scope** indica dónde se almacena el bean. Si no se especifica, scope adopta el valor predeterminado page. El atributo **class** especifica el nombre de clase completo.

5.1. ETIQUETA USEBEAN

Para una declaración useBean de lo siguiente:

```
<jsp:useBean id="miBean" scope="request"
             class="sl314.beans.CustomerBean" />
```

el código Java equivalente podría ser así:

```
CustomerBean miBean =
      (CustomerBean)request.getAttribute("miBean");
if( miBean == null ) {
  miBean = new CustomerBean();
  request.setAttribute("miBean", miBean);
}
```

5.2. ETIQUETA SETPROPERTY

- La etiqueta setProperty sirve para almacenar datos en la instancia de JavaBeans. La sintaxis de la etiqueta setProperty es:

```
<jsp:setProperty name="NombreBean" property_expression />
```

- El atributo **name** especifica el nombre de la instancia de JavaBeans. Debe coincidir con el atributo id utilizado en la etiqueta useBean.
- property_expression puede ser así:
  - property="*"
  - property="NombrePropiedad"
  - property="NombrePropiedad" param="NombreParámetro"
  - property="NombrePropiedad" value="ValorPropiedad"

5.2. ETIQUETA SETPROPERTY

- El atributo **property** especifica la propiedad dentro del bean que se definirá. Por ejemplo, para definir la propiedad email en el bean del cliente, puede utilizar lo siguiente:

  <jsp:setProperty name="cust" property="email" />

- Esta acción recupera el valor del parámetro de solicitud email y emplea este valor en el método set del bean. El código Java equivalente sería como éste:
  - cust.setEmail(request.getParameter("email"));

5.2. ETIQUETA SETPROPERTY

- El atributo **param** se puede suministrar si el nombre del parámetro de solicitud es distinto al nombre del parámetro de bean.
- Por ejemplo, si el campo del formulario para la dirección de correo electrónico fuera emailAddress, podría definir la propiedad named email del bean mediante:

```
<jsp:setProperty name="cust" property="email"
                 param="emailAddress" />
```

- El código Java equivalente sería como éste:

```
cust.setEmail(request.getParameter("emailAddress"));
```

5.2. ETIQUETA SETPROPERTY

- El atributo **value** puede utilizarse para suministrar el valor que debe emplearse en el método set. Por ejemplo, el valor se podría codificar rígidamente en la página de JSP:

```
<jsp:setProperty name="cust" property="email"
                 value="joe@host.com" />
```

- Los valores de los atributos se pueden especificar con expresiones, que se evalúan en tiempo de ejecución. Por ejemplo:

```
<jsp:setProperty name="cust" property="email"
  value='<%= someMethodToGetEmail() %>' />
```

- El carácter de **asterisco** (*) sirve para especificar todas las propiedades del bean.

5.3. ETIQUETA GETPROPERTY

- La etiqueta getProperty sirve para recuperar una propiedad de una instancia de JavaBeans y mostrarla en el flujo de salida.
- La sintaxis de la etiqueta getProperty es:

```
<jsp:getProperty name="NombreBean"
property="NombrePropiedad" />
```

- El atributo name especifica el nombre de la instancia de JavaBeans, mientras que el atributo property determina la propiedad utilizada con el método get.

5.3. ETIQUETA GETPROPERTY

- Para un uso de getProperty como el siguiente:

  <jsp:getProperty name="cust" property="email" />

- el equivalente en lenguaje Java sería así:

  out.print(cust.getEmail());

- La etiqueta getProperty brinda un mecanismo práctico para mostrar las propiedades de una instancia de JavaBeans evitando el código de scriptlet.

5.4. ETIQUETA INCLUDE

- Esta acción permite insertar un archivo estático o dinámico en la página que está siendo generada por el motor JSP.
- Su sintaxis es:

```
<jsp:include page="url" flush="true">
        <jsp:param ... />
        <jsp:param ... />
</ jsp:include>
```

5.4. ETIQUETA INCLUDE

Es importante distinguir entre directiva include y acción include:

- Directiva <%@ include file="Nombre fichero" /> se añade el código al servlet que se genera para la página en tiempo de compilación y se incluye el contenido existente en el momento inicial.
- Acción <jsp:include> no se añade código al servlet, sino que se invoca al objeto en tiempo de ejecución y se ejecuta el contenido existente en el momento de la petición.

5.5. ETIQUETA FORWARD

- Esta etiqueta permite que la petición sea redirigida a otra página JSP, a otro servlet o a otro recurso estático para que lo procese.
- Cuando el motor JSP encuentra esta etiqueta, la petición se pasa directamente al otro recurso, sin procesar el resto de la página que contenía la acción forward.
- Esta acción es muy útil cuando se quiere separar la aplicación en diferentes vistas, dependiendo de la petición interceptada. Sintaxis:

```
<jsp: forward page="url" >
    <jsp:param ... />
    <jsp:param ... />
</ jsp: forward >
```

5.6. ETIQUETA PARAM

- Este elemento es utilizado dentro de otras acciones para proporcionar información adicional de la forma clave/valor.
- Sirve para pasar parámetros a un objeto. Asocia un valor a un nombre y pasa la asociación a otro recurso invocado con <jsp:include>, <jsp:forward> o <jsp:plugin>. Su sintaxis es:

```
<jsp:param name="nombreParametro"
value="{valorParametro | <%= expresion %>}" />
```

- Los dos atributos son obligatorios.
- El primero corresponde al nombre del parámetro y el segundo con el valor que se le asigna, que puede ser una expresión JSP que se evalúa en el momento de realizar la petición de la página.

5.7. ETIQUETA PLUGIN

- Esta acción genera código HTML (las etiquetas object o embed) específico al navegador al que va dirigida la página JSP, que provocará la descarga del software plug-in (en caso de que sea necesario) correspondiente al navegador en el cual se intenta ejecutar un applet o JavaBean.
- Esta acción permite que la página JSP incluya un bean o un applet en la página cliente. Su sintaxis es:

```
<jsp:plugin
        type="bean|applet"
        code="nombre de la clase"
        codebase="directorio donde está el .class"
        {align="bottom|top|middle|left|right"}
        {archive="fichero jar donde se ha almacenado la clase"}
```

5.7. ETIQUETA PLUGIN

{height="altura en pixeles"}
{hspace="espacio horizontal en pixeles"}
{jreversion="numeroversionJRE"}
{name="nombre de componente"}
{vspace="espacio vertical en pixeles"}
{width="anchura en pixeles"}
{nspluginurl="URL del plugin Netsacape"}
{iepluginurl="URL del plugin Explorer"} >

```
<jsp:params>
    <jsp:param ... />
</jsp:params>
<jsp:fallback>Problema con el plugin </ jsp:fallback>
</ jsp:plugin>
```

6. OBJETOS IMPLICITOS DE JSP

- El motor de JSP proporciona acceso a las siguientes variables en etiquetas de scriptlet y de expresión.
- Estas variables representan objetos de uso habitual con los servlets que los desarrolladores de páginas de JSP pueden necesitar.
- Por ejemplo, puede recuperar datos de parámetros de un formulario HTML con la variable request, que representa al objeto HttpServletRequest.
- En la siguiente tabla se recogen los objetos implícitos de JSP.

6. OBJETOS IMPLICITOS DE JSP

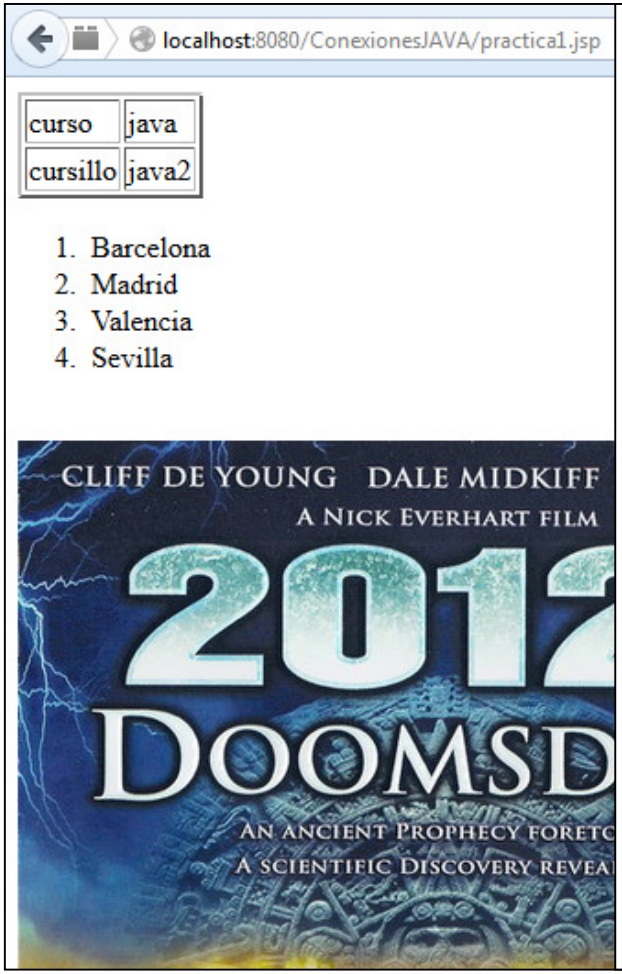
Nombre de la variable	Descripción
request	El objeto HttpServletRequest asociado a la solicitud
response	El objeto HttpServletResponse asociado a la respuesta que se devuelve al navegador
out	El objeto JspWriter asociado al flujo de salida de la respuesta
session	El objeto HttpSession asociado a la sesión para el usuario especifico de la solictud. Esta variable sólo es significativa si la página de JSP interviene en una sesión HTTP
application	El objeto ServletContext para la aplicación web
config	El objeto ServletConfig asociado al servlet para esta página de JSP
pageContext	El objeto pageContext que encapsula el entorno de una sola solicitud para esta página de JSP
page	La variable page equivale a la variables this en el lenguaje Java
exception	El objeto Throwable generado por otra página de JSP. Esta variables sólo está disponible en una página de error de JSP.

RECUERDA QUE...

- Las páginas jsp son las vistas de nuestra aplicación.
- Podemos incluir código java en ellas utilizando diferentes formatos: scriptlets, etiquetas, directivas, objetos implícitos, ...etc.
- Una página JSP se traduce en un servlet en tiempo de ejecución.

Ejemplo 1. Prueba con jsp

Primer ejercicio con jsp.

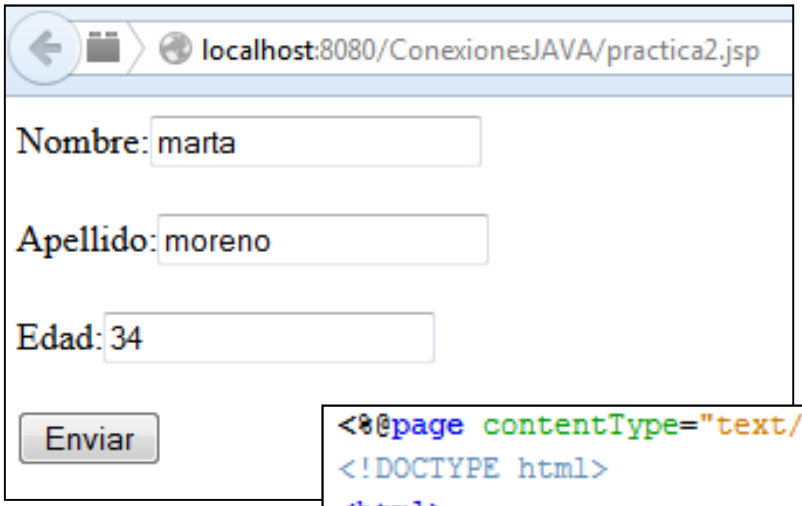



```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%! String ponerimagen()
    {
        StringBuffer buffer = new StringBuffer();
        buffer.append("<ol>");
        buffer.append("<li> Barcelona");
        buffer.append("<li> Madrid");
        buffer.append("<li> Valencia");
        buffer.append("<li> Sevilla");
        buffer.append("</ol>");
        buffer.append("<br>");
        buffer.append("<img src=http://thefauxistinternational.files."
                + "wordpress.com/2009/09/2012-doomsday-election-presidentielle.jpg>");
        return buffer.toString();
    }
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Practica 1</title> </head>
<body>
    <table border=2>
            <tr><td>curso<td>java
            <tr><td>cursillo<td>java2
    </table>

    <%=ponerimagen()%>
</body>
</html>
```

Ejemplo 2. Paso de parámetros entre páginas

Crear un formulario con las siguientes características. Los datos serán recogidos por un fichero jsp




```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<body>
    <form method=get action=http://localhost:8080/ConexionesJAVA/practica21.jsp>
        Nombre:<input type=text name=nombre><br><br>
        Apellido:<input type=text name=apellido><br><br>
        Edad:<input type=text name=edad><br><br>
        <input type=submit value=Enviar>
    </form>
</body>
</html>
```

Ejemplo 2. Paso de parámetros entre páginas

El fichero jsp recoge los datos pasados por POST y los muestra por pantalla

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
<head><title>Practica21.jsp</title></head>
<body>
<%
    String nombre = request.getParameter("nombre");
    String apellido = request.getParameter("apellido");
    String edad = request.getParameter("edad");
    out.println("Name: <b>" + nombre + "</b><br>");
    out.println("Apellido: <b>"+ apellido +  "</b><br>");
    out.println(" Tus datos han sido enviados correctamente");
%>
</body>
</html>
```