


Laboratorio V

Introducción a JSP

Ing. Gregorio Nicolás Tkachuk




Objetivos

- ▶ Que el estudiante logre:
 - Conocer la plataforma Java para el desarrollo de aplicaciones Web.
 - Conocer las directivas de JSP para crear paginas dinámicas.
 - Aprender los pasos para llevar a cabo la implementación de una aplicación Web desarrollada en Java.
- 


Arquitectura Java

- ▶ Servidor: Apache Tomcat
 - ▶ Plataforma: JEE (Java Enterprise Edition)
 - ▶ Lenguaje: Java, Servlets/JSP (Java Server Page)
 - ▶ Base de Datos: MySQL, SQL Server, PostgreSQL
- 

Componentes Web de Java

- **Servlets**
 - **JSP (Java Server Pages)**
 - **JavaBeans**
- 

¿Que es un Servlet?

- ▶ Un servlet es un componente web basado en tecnología Java, gestionado por un contenedor que sirve para generar contenido dinámico.
 - ▶ Contenedor o motor de servlets son extensiones de servidor web que permiten ejecutar servlets.
 - ▶ Los servlets interactúan con los clientes web a través del paradigma request / response que debe ser implementado por el contenedor.
- 

¿Que es una JSP (Java Server Page)?

Es una tecnología basada en el lenguaje Java que permite incorporar contenido dinámico a las páginas web. Los archivos JSP combinan HTML con etiquetas especiales y fragmentos de código Java.

JSP es una extensión de los servlets para facilitar la construcción de interfaces de usuario.

Una JSP es como una página HTML a la que se han añadido etiquetas específicas de JSP y código Java.

El objetivo final es separar la interfaz (presentación visual) de la implementación (lógica de ejecución)


Se guarda como un archivo con extensión .jsp

¿Que es un JavaBeans?

- ▶ Un JavaBean o bean es un componente hecho en software que se puede reutilizar y que puede ser manipulado visualmente por una herramienta de programación en lenguaje Java.
- ▶ Para ello, se define un interfaz para el momento del diseño (design time) que permite a la herramienta de programación o IDE, interrogar (query) al componente y conocer las propiedades (**properties**) que define y los tipos de sucesos (**events**) que puede generar en respuesta a diversas acciones.

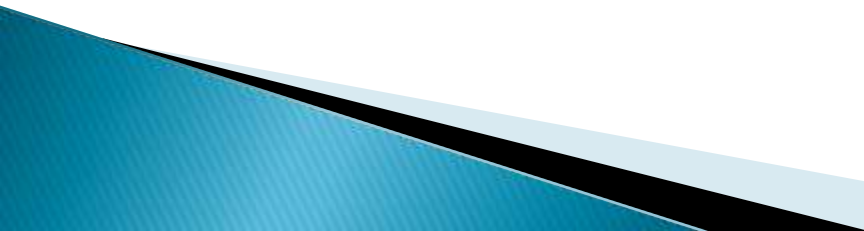
Servlet y JSP,

Un JSP es una página Web con etiquetas especiales y código Java incrustado, mientras que un servlet es un programa que recibe peticiones y genera a partir de ellas una página web.



JSP (Java Server Pages) (cont.)

¿Cómo funciona un JSP?

- ▶ Cuando el cliente solicita una página .jsp:
 - ▶ 1. La primera vez que es solicitado se traduce el código de la jsp en un servlet.
 - ▶ 2. Se compila el servlet y se carga en memoria.
 - ▶ 3. Se ejecuta el método `service()` del servlet.
- 

JSP Ejemplo de código

Código Java en páginas HTML

```
<html>
<head>
  <title>Mi primera página JSP</title>
</head>
<body>
  <h1>
    Hola, <%=request.getParameter("nombre") %>
    Hoy es: <%=new java.util.Date() %>
  </h1>
</body>

</html>
```

JSP: Procesamiento

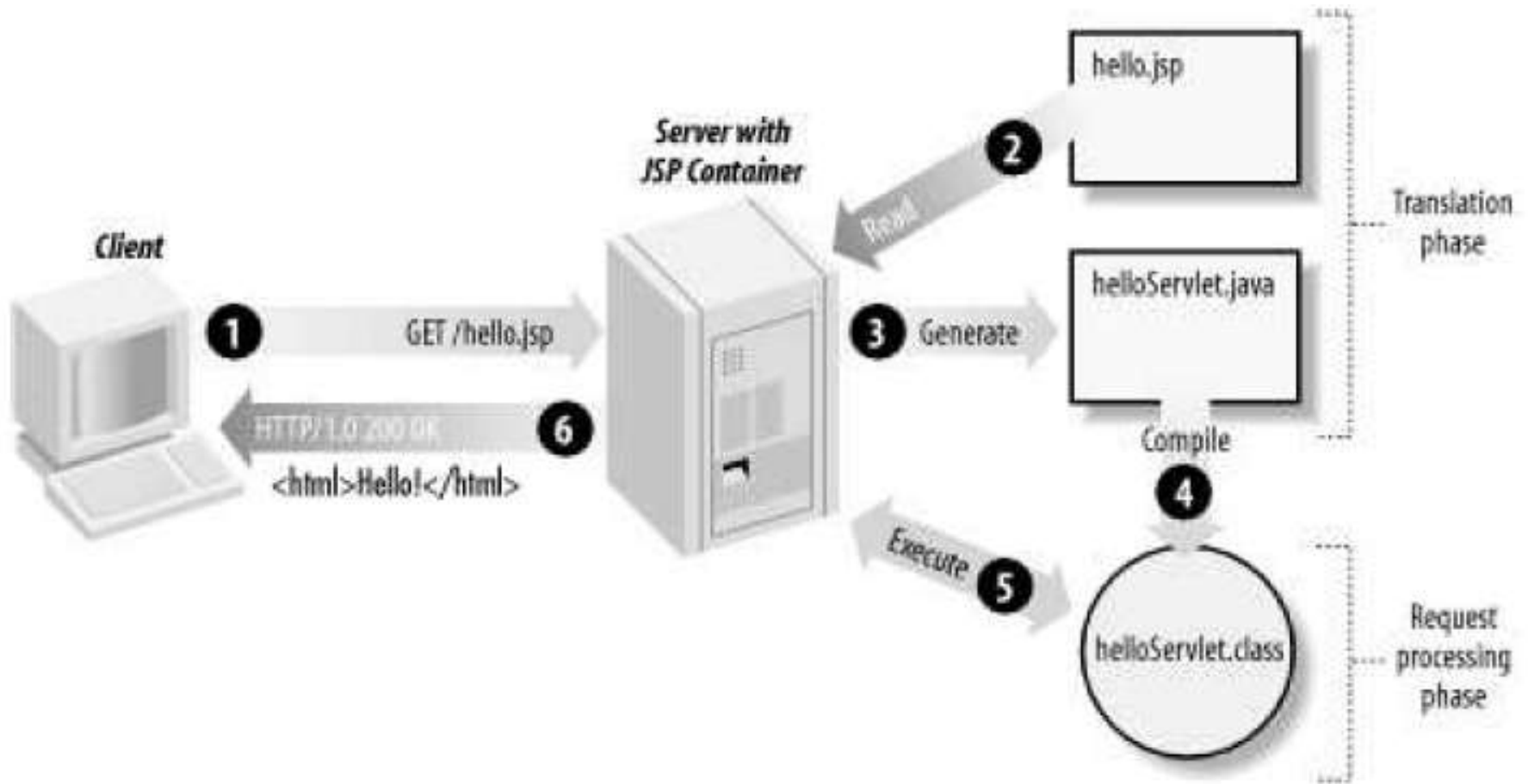
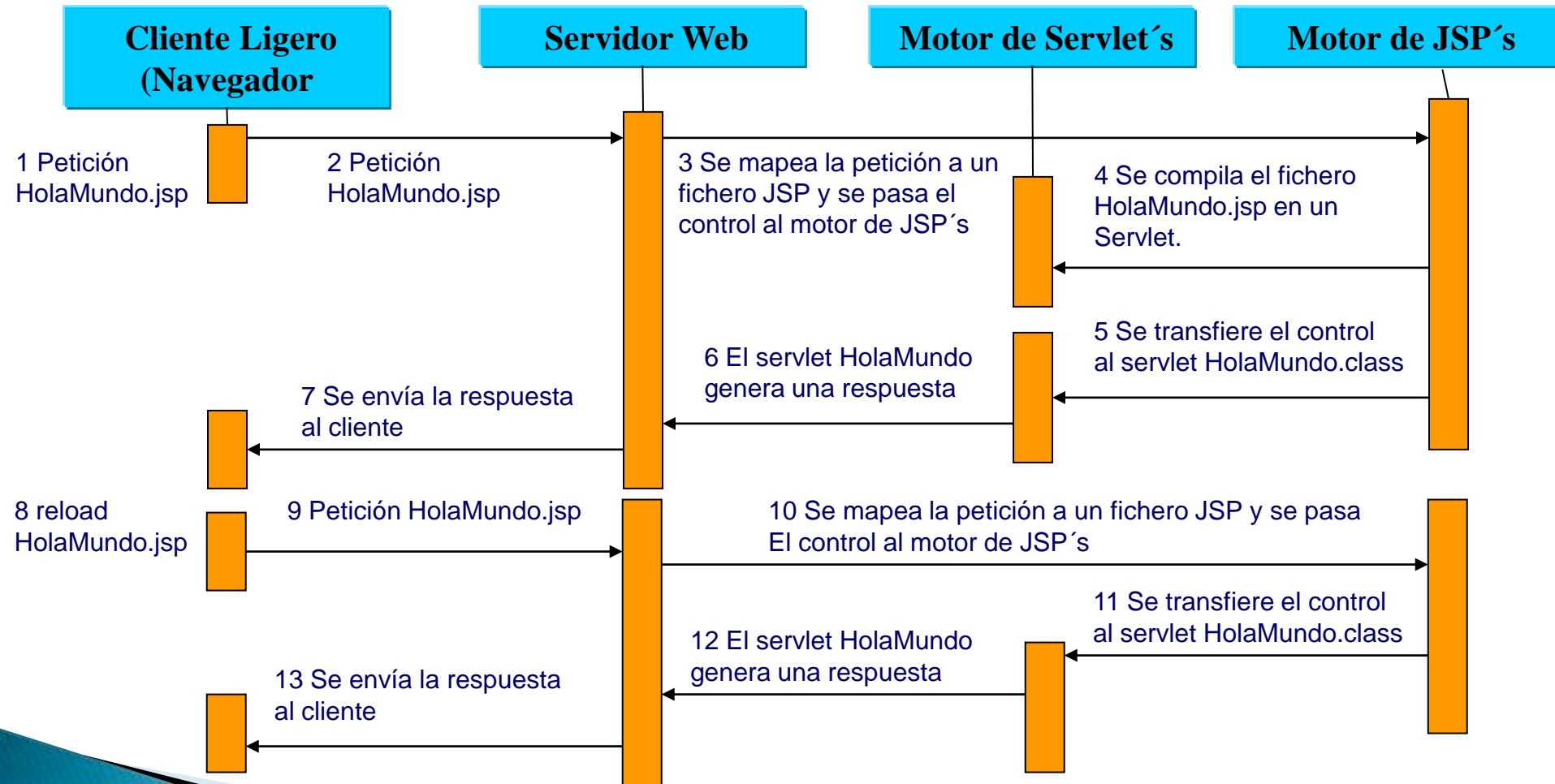


Figura 1: Petición de una página JSP

JSP: Funcionamiento



JSP: Ejemplo

Pagina JSP original:

```
<%@ page language='java'
  contentType='text/html;charset=iso-8859-1'%>
<%@ page import='java.util.Date' %>
<html>
<head>
<title>Hola Mundo</title>
</head>
<body>
Hola, esto es una pagina JSP.
<p>La hora del servidor es <%= new Date() %> </p>
</body>
</html>
```

JSP: Ejemplo (Cont.)

Tras la fase de traducción, se obtiene el siguiente Servlet:

```
package org.apache.jsp;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import org.apache.jasper.runtime.*;
// <%@ page import='java.util.Date' %>
//
import java.util.Date;
public class hola_jsp extends HttpJspBase {
private static java.util.Vector _jspx_includes;
public java.util.List getIncludes() {
return _jspx_includes;
}
public void _jspService(HttpServletRequest request,
HttpServletResponse response)
throws java.io.IOException, ServletException {
```

JSP: Ejemplo (Cont.)

Tras la fase de traducción, se obtiene el siguiente Servlet:

```
JspFactory _jspxFactory = null;
javax.servlet.jsp.PageContext pageContext = null;
HttpSession session = null;
ServletContext application = null;
ServletConfig config = null;
JspWriter out = null;
Object page = this;
JspWriter _jspx_out = null;
try {
    _jspxFactory = JspFactory.getDefaultFactory();
    // <%@ page language='java' contentType='text/html; charset=iso-8859-1'%>
    //
    response.setContentType("text/html; charset=ISO-8859-1");
    pageContext = _jspxFactory.getPageContext(this, request, response,
    null, true, 8192, true);
    application = pageContext.getServletContext();
    config = pageContext.getServletConfig();
    session = pageContext.getSession();
    out = pageContext.getOut();
    _jspx_out = out;
```

JSP: Ejemplo (Cont.)

Tras la fase de traducción, se obtiene el siguiente Servlet:

```
out.write("\n");
out.write("\n\n");
out.write("<html>\n ");
out.write("<head>\n ");
out.write("<title>Hola Mundo");
out.write("</title>\n ");
out.write("</head>\n ");
out.write("<body>\n Hola, esto es una pagina JSP.\n ");
out.write("<p>La hora del servidor es ");
// <%= new Date() %>
out.print( new Date() );
out.write("</p>\n ");
out.write("</body>\n");
out.write("</html>\n");
} catch (Throwable t) {
out = _jspx_out;
}
```

JSP: Ejemplo (Cont.)

Tras la ejecución del servlet anterior, este es el resultado:

```
<html>  
<head>  
<title>Hola Mundo</title>  
</head>  
<body>
```

Hola, esto es una pagina JSP.

```
<p>La hora del servidor es Wed Nov 06 13:25:34  
CET 2010</p>  
</body>  
</html>
```

Como puede verse, el resultado es muy similar a la pagina JSP original.

JSP: Elementos

1. Código Java

1.1. *Expresiones*

1.2. *Scriptlets*

1.3. *Declaraciones*

2. Directivas

2.1. *page*

2.2. *include*

2.3. *taglib*

3. Acciones

3.1. Inclusión de páginas

3.2. Transferencia de control

JSP: Expresiones

Son fragmentos de código Java, con la forma `<%= expresión %>` que se evalúan y se muestran en la salida del navegador.

Ejemplos:

```
<%= "Tamaño de cadena: "+cadena.length() %>
```

```
<%= new java.util.Date() %>
```

```
<%= Math.PI*2 %>
```

JSP: Scriptlets

Son fragmentos de código Java con la forma `<% código %>`, en general, podemos insertar cualquier código que pudiéramos usar dentro de una función Java. Se insertan dentro del método `service` del servlet.

Ejemplos:

```
<table>
  <% for (int i=0;i<10;i++) { %>
    <tr><td> <%=i%> </td></tr>
    <% } %>
</table>
```

JSP: Scriptlets, Comentarios

Para introducir comentarios en JSP, usaremos las marcas `<%-- comentario --%>`, dentro de un *scriptlet* o declaración podemos usar comentarios siguiendo la sintaxis de Java.

```
<%-- Comentario JSP --%>
```

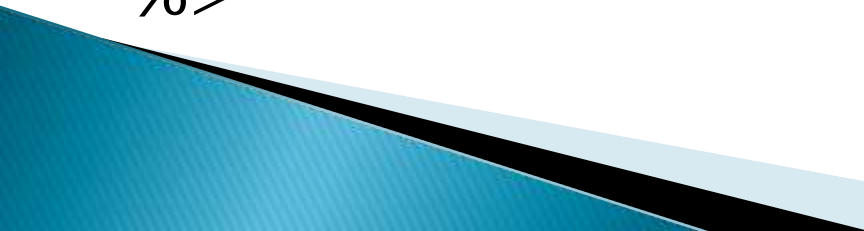
```
<!-- Comentario HTML -->
```

```
<%
```

```
    // Comentario
```

```
    /* Comentario */
```

```
%>
```



JSP: Declaraciones

Contienen declaraciones de variables o métodos, con la forma `<%! declaración %>`. Estas variables o métodos serán accesibles desde cualquier lugar de la página JSP. Hay que tener en cuenta que el servidor transforma la página JSP en un *servlet*, y éste es usado por múltiples peticiones, lo que provoca que las variables conserven su valor entre sucesivas ejecuciones.

Ejemplos:

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

JSP: Declaraciones (Cont.)

```
<%! int numeroAccesos=0; %>
<html>
<body>
<%= "La pagina ha sido accedida "+(++numeroAccesos)+ " veces desde
    el arranque del servidor" %>
<br />
<%! java.util.Date primerAcceso=new java.util.Date(); %>
<br />
    El primer acceso a la pagina se realizo en: <%= primerAcceso %>
        <%! private String ahora() {
                return ""+new java.util.Date();
            } %>
<br> <br>
    El ultimo acceso a la pagina se realizo en: <%= ahora() %>
</body>
</html>
```

JSP: Directivas

Las directivas son elementos que proporcionan información al motor JSP, e influirán en la estructura del *servlet generado*.

Hay tres tipos de directivas:

- *page*
- *taglib*
- *include*.

JSP: Directiva *page*

Se indica con la forma `<%@ page atributo="valor">`.

Tiene diversos usos, entre los cuales destacaremos:

- Importar clases. Importar código, de la misma forma que se realiza en un programa en Java, se indica con el atributo `import`.

Ejemplo:

```
<%@page import="java.io.*, miPackage.miClase"%>
```

- Indicar si la página tendrá acceso a la sesión. Se especifica con el atributo `session`.

Ejemplo:

```
<%@page session="true" import="java.util.ArrayList"%>
```

- Gestión de errores. Permite redireccionar a una página cuando se produzca un error

JSP: Directiva *include*

Permite incluir un archivo en el lugar donde se especifique, simplemente copia el contenido del archivo byte a byte, siendo el resultado similar a si copiáramos el texto del archivo incluido y lo pegáramos en el JSP.

Ejemplo:

```
<html>  
<head>  
  <%@ include file="titulo.txt"%>  
</head>  
<body>  
  <%@ include file="cuerpoPagina.jsp"%>  
</body>  
</html>
```

JSP: Directiva *taglib*

Se emplea para indicar que se van a emplear librerías de etiquetas.

Ejemplo:

```
<%@ taglib prefix="c"  
uri="http://java.sun.com/jsp/jstl/core" %>
```

JSP: Acciones

Las acciones tienen la forma

`<jsp:accion [parámetros]/>`, y tienen diversos usos, entre los que destacan:

- ▶ *inclusión de páginas*
- ▶ *transferencia de control*

JSP: Accion Inclusión de páginas

Se realiza con la acción `<jsp:include page="pagina.jsp">`. Incluye la salida de otra página JSP en la actual, al contrario que con la directiva `<%@include file="fichero.ext"%>` la página incluida se *ejecuta y su salida se inserta en la página que la incluye, con la directiva se incluye el contenido del archivo (no su salida) y se ejecuta conjuntamente con la página principal.*

La página incluida tiene acceso a los parámetros enviados a la principal, y podemos enviarle nuevos parámetros con la subetiqueta `<jsp:param name="nombre" value="valor"/>`.

Ejemplo:

```
<html> <head>
    <jsp:include page="cabecera.jsp"/>
</head>
<body>
    <jsp:include page="cuerpo.jsp">
    <jsp:param name="tipo" value="paginaPrincipal"/>
    </jsp:include>
</body>
</html>
```

JSP: Acción Transferencia de control

Se realiza con la acción

`<jsp:forward page="pagina.jsp"/>`. La petición es redirigida a otra página, y la salida de la actual se descarta. La página a la que se redirige tiene acceso a los parámetros pasados a la actual, y es posible el envío de nuevos parámetros.

Ejemplo:

```
<jsp:forward page="principal.jsp">
```

```
<jsp:param name="titulo" value="Principal"/>
```

```
</jsp:forward>
```

JSP: Objetos implícitos

En JSP disponemos de algunos objetos implícitos, que nos permitirán acceder a diferente información y realizar diversas acciones. En JSP tenemos los siguientes objetos implícitos:

- *request*
- *response*
- *out*
- *session*
- *application*
- *config*
- *pagecontext*
- *page*

JSP: request

Es un objeto de la clase `HttpServletRequest`, su uso principal es el acceso a los parámetros de la petición. Se destacan las siguientes funciones:

- ***String* `getParameter(String name)`**

Devuelve el valor de un parámetro.

- ***Enumeration* `getParameterNames()`**

Devuelve una enumeración con los nombres de todos los parámetros de la petición.

- ***String[]* `getParameterValues(String name)`**

Los parámetros pueden tener valor múltiple, con esta función recuperamos un array con todos los valores para un nombre dado.

- ***String* `getRemoteAddr()`**

Devuelve la IP del host desde el que se realiza la petición

- ***String* `getRemoteHost()`**

Devuelve el nombre del host desde el que se realiza la petición.

JSP: request (Cont.)

Ejemplo:

```
<html>
```

```
<body>
```

```
<form>
```

```
    <input type="text" name="parametro" /> <input  
    type="submit" />
```

```
</form> <br>
```

```
Su IP: <%=request.getRemoteAddr()%> <br>
```

```
Su nombre de host: <%=request.getRemoteHost()%>
```

```
<br>
```

```
Valor del parámetro:
```

```
<%=request.getParameter("parametro")%>
```

```
</body>
```

```
</html>
```

JSP: response

Es un objeto de la clase `HttpServletResponse`, que asiste al *servlet* en su generación de la respuesta para el cliente, contiene funciones para manejo de cabeceras, códigos de estado, cookies y transferencia de control.

JSP: out

Es un objeto de la clase `JspWriter`, es el que nos permite acceder a la salida del navegador desde los *scriptlet*.

Ejemplo:

```
<%  
    out.print("cadena");  
    out.println("cadena");  
%>
```

JSP: session

Es un objeto de la clase HttpSession. Nos permite acceder a la sesión asociada a la petición. A través de este objeto podemos, entre otras cosas, guardar objetos que serán accesibles desde cualquier JSP de la sesión o invalidarla.

Para guardar y recuperar información usaremos:

```
Object session.getAttribute("clave");
```

```
void session.setAttribute("clave", Object objeto);
```

Y para invalidar la sesión:

```
void session.invalidate();
```

JSP: session (Cont)

Ejemplo:

```
<%@ page session="true" %>
<% java.util.ArrayList accesos=
(java.util.ArrayList)session.getAttribute("accesos");
if (accesos==null)
    accesos=new java.util.ArrayList();
    accesos.add(new java.util.Date().toString());
    session.setAttribute("accesos", accesos);
if (request.getParameter("invalidaSesion")!=null)
    session.invalidate(); %>
```

JSP: session (Cont)

```
<html>
<body>
<form>
  <input type="submit" name="invalidaSesion" value="Invalidar
  sesión"/>
<input type="submit" value="Recargar página"/>
</form>
<br/>
  Usted accedió a esta página en los
  siguientes momentos: <br>
  <% for (int i=0;i<accesos.size();i++) { %>
  <%= accesos.get(i) %>
  <br>
  <% } %>
</body>
</html>
```

JSP: application

Es un objeto de la clase ServletContext. Este objeto es común para toda la aplicación web y, entre otras cosas, nos permite almacenar información que será accesible desde todas las páginas de la aplicación web, independientemente de la sesión.

Para guardar y recuperar valores:

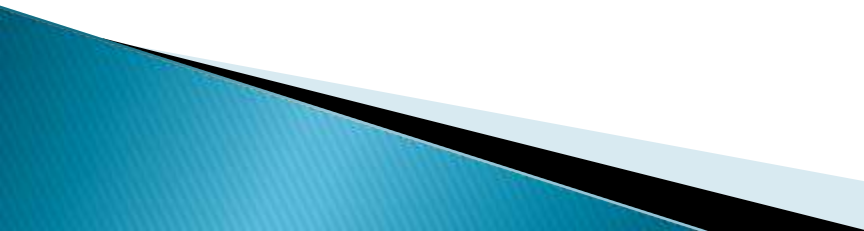
```
Object application.getAttribute("clave");  
void application.setAttribute("clave", Object  
objeto);
```

JSP: application (ejemplo)

Ejemplo:

```
<%@ page session="true" %>
```

```
<%  
java.util.Hashtable direcciones=  
    (java.util.Hashtable)application.  
        getAttribute("direcciones");  
if (direcciones==null)  
    direcciones=new java.util.Hashtable();  
    direcciones.put(request.getRemoteAddr(),"");  
    application.setAttribute("direcciones", direcciones);  
%>
```



JSP: application (ejemplo cont.)

```
<html>
```

```
<body>
```

El servidor fue accedido desde las siguientes direcciones IP:

```
<br>
```

```
<%
```

```
    java.util.Enumeration e= direcciones.keys();
```

```
    while (e.hasMoreElements() {
```

```
%>
```

```
<%= e.nextElement() %>
```

```
<br>
```

```
<% } %>
```

```
</body>
```

```
</html>
```



JSP: application (ejemplo cont.)

```
<html>
```

```
<body>
```

El servidor fue accedido desde las siguientes direcciones IP:

```
<br>
```

```
<%
```

```
    java.util.Enumeration e= direcciones.keys();
```

```
    while (e.hasMoreElements() {
```

```
%>
```

```
<%= e.nextElement() %>
```

```
<br>
```

```
<% } %>
```

```
</body>
```

```
</html>
```



JSP: pageContext

Es un objeto de la clase PageContext. Entre otras cosas, nos permite almacenar información localmente a la página.

Para guardar y recuperar valores:

```
Object pageContext.getAttribute("clave");
```

```
void pageContext.setAttribute("clave", Object objeto);
```

También podemos usar PageContext para almacenar y recuperar información en sesión y en aplicación:

Almacenar en contexto de página:

```
PageContext.setAttribute("clave", obj, PageContext.PAGE_SCOPE);
```

```
PageContext.setAttribute("clave", obj);
```

JSP: pageContext (cont.)

Almacenar en contexto de sesión:

```
PageContext.setAttribute("clave",obj, PageContext.SESSION_SCOPE);  
session.setAttribute("clave", objeto);
```

Almacenar en contexto de aplicación:

```
PageContext.setAttribute("clave",obj,PageContext.APPLICATION_SCOPE);  
application.setAttribute("clave",objeto);
```

JSP: Comunicación entre formularios HTML y páginas JSP

Breve repaso sobre formularios

Como pasar parámetros entre formularios de paginas HTML y JSP.



JSP: Conceptos básicos

Un formulario HTML tiene la forma:

```
<form action="destino" method="método">
```

Elementos de formulario

```
</form>
```

En *destino* especificaremos la página que recibe los datos del formulario (p.e. *procesaformulario.jsp*), en el atributo *method* podemos indicar dos valores diferentes *GET* y *POST*. Si no se especifica el valor de los atributos, los valores por defecto son la página actual para *action* y *GET* para *method*

GET y POST

Usando GET, la información se codifica directamente en la URL, con la forma:

`http://url?param1=valor1¶m2=valor2...¶mN=valorN`

Desventajas de GET:

No se puede enviar grandes cantidades de información.

El servidor o el navegador guardan en caché la página llamada.

los logs del servidor y el historial del navegador guardarán el acceso incluyendo los parámetros.

Con POST la información se envía directamente al servidor, no se codifica en la URL, y además permite el envío de grandes cantidades de información, como podrían ser archivos.

JSP: Elementos de formulario

Se indican con las etiquetas HTML:

```
<input type="tipo" name="nombre" value="valor" />
```

```
<textarea name="nombre" />Contenido por defecto</textarea>
```

```
<select name="nombre">
```

```
    <option value="valorOpcion">Texto  
    opcion</option>
```

```
    [...]
```

```
</select>
```

Para enviar los datos usamos el tipo submit.

```
<input type="submit" />
```

JSP: Campos de texto

Los tipos que se envían como texto simple son text y password para <input>, y el elemento <textarea>.

Ejemplo:

```
<form action="pagina.jsp">
<input type="text" name="parametro1"
value="valor por defecto"/>
<br>
<input type="password" name="clave"/>
<br>
<textarea name="parametro2">Texto por
defecto</textarea>
<br>
<input type="submit"/>
</form>
```

JSP: Campos de texto

Ejemplo del archivo pagina.jsp:

Valor de parametro1:

```
<%= request.getParameter("parametro1") %>
```

```
<br>
```

Valor de parametro2:

```
<%= request.getParameter("parametro2") %>
```

```
<br>
```

Valor de parametro 'clave':

```
<%= request.getParameter("clave") %>
```

Ejemplo de uso de objetos implícitos

Aplicación que pide el nombre al usuario y le devuelve un saludo.

Utiliza un fichero HTML como formulario que pide los datos al cliente y se los pasa a una página JSP que muestra el saludo con éstos datos. El paso de los datos del formulario al JSP se realiza a través de un objeto implícito: el objeto request.

JSP: Ejemplo

```
< -- Pagina HTML que pide los datos al Cliente. -- >
```

```
<HTML>
```

```
<head>
```

```
  <title> Formulario de petición de nombre </title>
```

```
</head>
```

```
<body>
```

```
  <h1> Formulario de petición de nombre </h1>
```

```
  <!-- Se envía el formulario al JSP "saludo.jsp" -->
```

```
<form method="post" action="saludo.jsp" >
```

```
  <p> Por favor, introduce tu nombre:
```

```
  <input type="text" name="nombre">
```

```
  </p>
```

```
  <p> <input type="submit" value="enviar información"> </p>
```

```
</form>
```

```
</body>
```

```
</HTML>
```

JSP: Ejemplo

```
< -- Pagina JSP que procesa los datos recibidos y lo muestra -- >
```

```
<HTML>
```

```
<head><title> Saludo al cliente </title></head>
```

```
<body>
```

```
<h1>Saludo al cliente</h1>
```

```
<!-- Los parámetros que le pasa el cliente en la petición se  
obtienen del objeto implícito request -->
```

```
<%
```

```
String nombre = request.getParameter("nombre");
```

```
out.println("Encantado de conocerle, " + nombre);
```

```
%>
```

```
<!-- Al evaluarse el código, hay que escribir explícitamente en  
la salida (obj. implícito out) -->
```

```
</body>
```

```
</HTML>
```

Bibliografía

- Deitel, Paul J.
Dietel, Harvey M. ***AJAX Rich Internet Applications y Desarrollo Web para Programadores***
Editorial Anaya Multimedia, 2009
- Duckett, Jon ***Fundamentos Programación Web con HTML XHTML y CSS***
Editorial Anaya Multimedia, 2009
- Jaworski, Jamie
Perrone, Paul J. ***Seguridad en Java Edición Especial***
Editorial Pearson Alhambra, 2001
- Maruyama, Hiroshi
Tamura Kent, Uramoto
Naohiko ***Creación de Sitios Web con XML y Java***
Editorial Prentice-Hall, 2000
- Programación Web www.lawebdelprogramador.com/
- Programación Web www.programacionweb.net/

Preguntas ?

