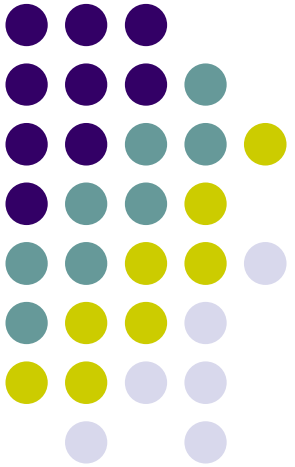
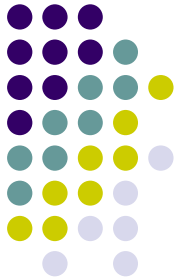
Java



Introducción a la programación Web con Java

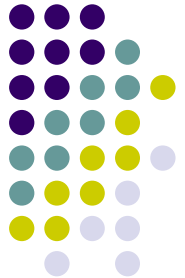
Lic. Claudio Zamoszczyk
claudio@honou.com.ar

Contenidos



- Introducción
- HTML
- JSP – Servlets

Plataformas

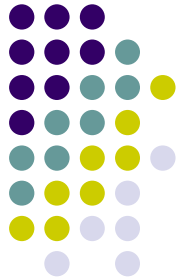


- **Java ME (Java Platform, Micro Edition) o J2ME — orientada a entornos de limitados recursos, como teléfonos móviles, PDAs (Personal Digital Assistant), etc.**

- **Java SE (Java Platform, Standard Edition) o J2SE — para entornos de gama media y estaciones de trabajo (PC de escritorio).**

- **Java EE (Java Platform, Enterprise Edition) o J2EE — orientada a entornos distribuidos empresariales o de Internet.**

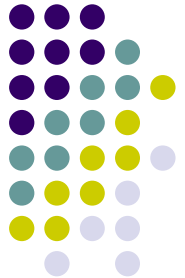
Introducción a las App. Web



La idea fundamental de los navegadores, es que muestren documentos escritos en HTML que han obtenido de un servidor Web. Estos documentos HTML habitualmente presentan información de forma estática, sin más posibilidad de interacción con ellos.

El modo de crear los documentos HTML ha variado a lo largo de la corta vida de las tecnologías Web pasando desde las primeras páginas escritas en HTML almacenadas en un archivo en el servidor Web hasta aquellas que se generan en tiempo de ejecución como respuesta a una acción del cliente y cuyo contenido varía según las circunstancias.

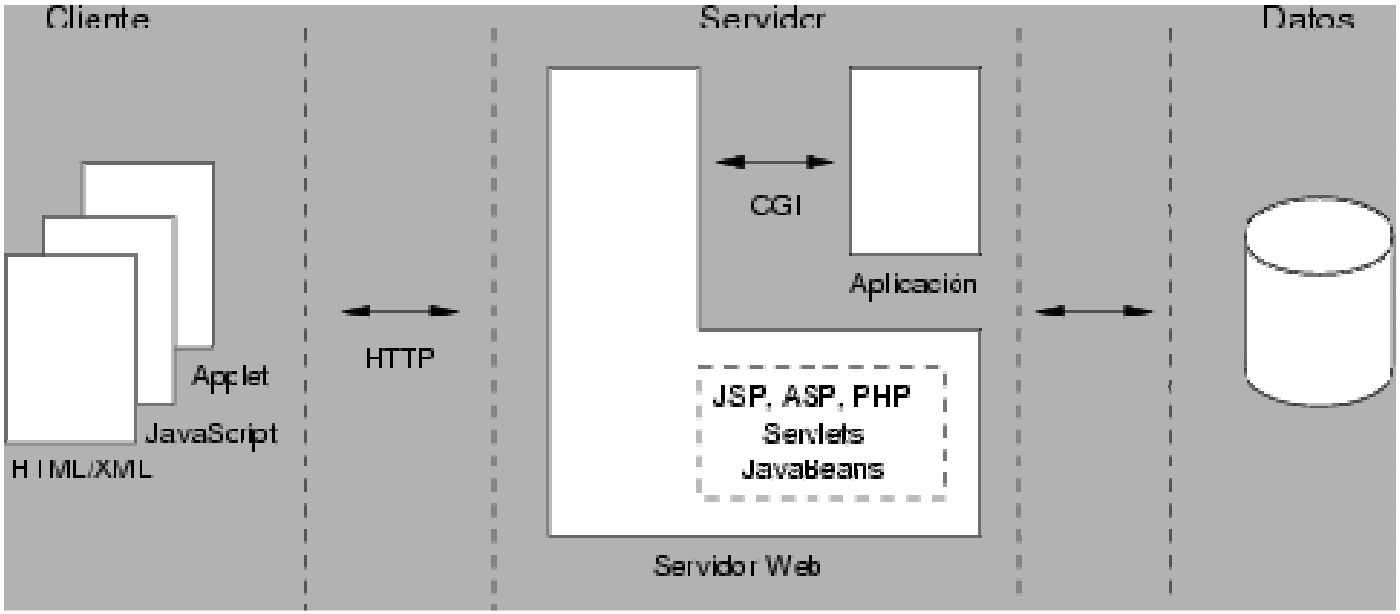
Introducción a las App. Web



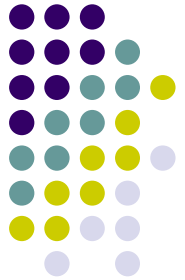
Además, el modo de generar páginas dinámicas ha evolucionado, desde la utilización del CGI ,*Common Gateway Interface*, hasta los *servlets* pasando por tecnologías tipo *JavaServer Pages*. Todas estas tecnologías se encuadran dentro de aquellas conocidas como *Server Side*, ya que se ejecutan en el servidor web.

Otro aspecto que completa (¿complica?) el panorama son las inclusiones del lado del cliente, *Client Side*, que se refieren a las posibilidades de que las páginas lleven incrustado código que se ejecuta en el cliente, como por ejemplo JavaScript.

Introducción a las App. Web



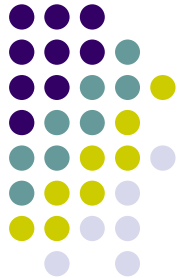
HTTP



Aunque entender el modo en que funciona HTTP no es estrictamente necesario para desarrollar aplicaciones Web, algunas nociones sobre lo que esconden esas siglas puede ayudar a desarrollarlas con más facilidad y confianza.

HTTP es un protocolo del nivel de aplicación para el intercambio de información. Es un protocolo no orientado a estado que puede ser utilizado para más propósitos que para manejar archivos HTML.

HTTP



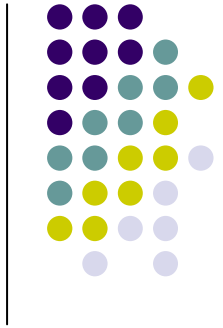
Entre las propiedades de HTTP se pueden destacar las siguientes:

- Un esquema de direccionamiento comprensible.
Utiliza el Universal Resource Identifier (URI) para localizar sitios (URL) o nombres (URN) sobre los que hay que aplicar un método. La forma general de un URL es servicio://host/archivo.ext .

- Arquitectura Cliente-Servidor.
HTTP se asienta en el paradigma solicitud/respuesta. La comunicación se asienta sobre TCP/IP. El puerto por defecto es el 80, pero se pueden utilizar otros. (Socket - ServerSocket)

- Es un protocolo sin conexión y sin estado.
Después de que el servidor ha respondido la petición del cliente, se rompe la conexión entre ambos. Además no se guarda memoria del contexto de la conexión para siguientes conexiones.

HTTP

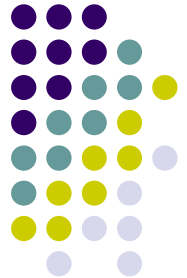


-**Está abierto a nuevos tipos de datos.**

HTTP utiliza tipos MIME (Multipart Internet Mail Extension) para la determinación del tipo de los datos que transporta. Cuando un servidor HTTP transmite información de vuelta a un cliente, incluye una cabecera que le indica al cliente sobre los tipos de datos que componen el documento. De la gestión de esos datos se encargan las utilidades que tenga el cliente (imágenes, vídeo, etc.)

Una transacción HTTP está compuesta por una cabecera, y opcionalmente, por una línea en blanco seguida de los datos. En la cabecera se especifica tanto la acción solicitada en el servidor, como los tipos de datos devueltos o un código de estado.

Interacción entre el Servidor y el Cliente

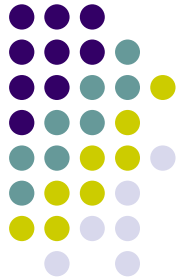


Durante la comunicación entre el cliente y el servidor HTTP en el que el cliente solicita el documento doc1.html al servidor se intercambian la siguiente transacción HTTP:

GET /doc1.html HTTP/1.1
Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent: firefox 1.02
From:200.45.124.200

El método GET indica el archivo que el cliente solicita y la versión de HTTP. El cliente también muestra una lista de los tipos MIME que puede aceptar como retorno, además de identificar el *browser* que utiliza (para que el servidor pueda optimizar la salida para el tipo particular de navegador) y su dirección.

Interacción entre el Servidor y el Cliente



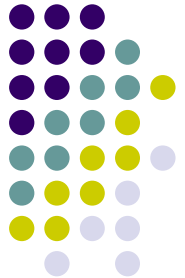
El servidor responde mandando la siguiente transacción HTTP:

HTTP/1.0 200 OK
Date: Friday, 23-Feb-01 16:30:00 GMT
Server: Apache/1.1.1
Content-type: text/html
Content-length: 230

<HTML><HEAD><TITLE> ..... </HTML>

En este mensaje el servidor utiliza la versión 1.1 de HTTP, y manda el código de estado 200 para indicar que la petición del cliente ha sido procesada satisfactoriamente. También se identifica como un servidor Apache. Indica al cliente que el contenido del documento es texto en formato HTML y que tiene una longitud de 230 bytes.

Métodos de petición (GET)



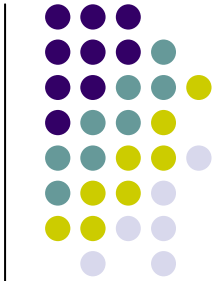
La primera línea de una petición contiene los comandos HTTP, conocidos como métodos. Existen varios, pero los más conocidos y utilizados son tres: GET y POST.

El método GET se utiliza para recuperar información identificada por un URI por parte de los navegadores. El método GET se utiliza para pasar una pequeña cantidad de información al servidor en forma de pares atributo-valor añadidos al final del URI detrás de un símbolo de interrogación, ?.

GET /cgi/saludar.pl?nombre=pepe&email=pepe@xxfoxx.xx.ar HTTP/1.1

La longitud de la petición GET está limitada por el espacio libre en los *buffers* de entrada (ej 256 bytes). Por lo que para mandar una gran cantidad de información al servidor ha de utilizarse el método POST.

Métodos de petición (POST)



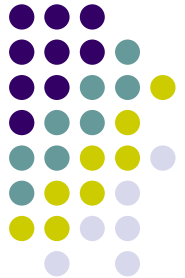
El método POST se refiere normalmente a la invocación de procesos que generan datos que serán devueltos como respuesta a la petición. Además se utiliza para aportar datos de entrada a esos programas. En este caso los pares atributo-valor son incluidos en el cuerpo de la petición.

POST /pedidos/alta.jsp HTTP/1.1
Accept: */*

nombre=pepe
email=pepe@xxxx.xx.ar

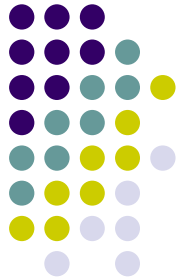
De este modo el método POST no sufre de las limitaciones de espacio y puede enviar mucha más información al servidor.

Repaso general HTML



```
<HTML>

  <HEAD>
    <TITLE>Mi primera pagina</TITLE>
  </HEAD>

  <BODY>
    <CENTER><H1>Mi Primera pagina</H1></CENTER>
    <P>Esta es mi primera pagina. Por el
    momento no se que tendra, pero dentro de poco
    pondre aqui muchas cosas interesantes.</P>
  </BODY>

</HTML>
```

Etiquetas de uso general



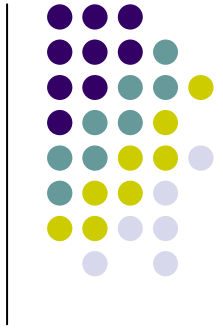
Formateo de párrafo

<P>….</P>
Sirve para delimitar un párrafo. Inserta una línea en blanco antes del texto.

<CENTER> ... </CENTER>
Permite centrar todo el texto del párrafo.

<DIV ALIGN='LEFT'> ... </DIV>
Permite justificar el texto del párrafo a la izquierda (ALIGN=‘LEFT’), derecha (RIGHT), al centro (CENTER) o a ambos márgenes (JUSTIFY)

Etiquetas de uso general



Cambiando el tipo de letra

<B> ... </B>	Pone el texto en negrita.
<I> ... </I>	Representa el texto en cursiva.
<U> ... </U>	Para subrayar algo.

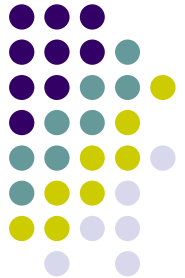
Otros elementos comunes

<HR>	Inserta una barra horizontal.
 	Salto de línea.
<!-- ... -->	Comentarios.

Links

```
<a href="/html/noticias/">Noticias</a>
```

Etiquetas de uso general



Imágenes

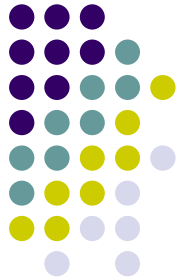
<IMG SRC=‘archivo_grafico’ ALT="descripcion">

<IMG SRC=../graficos/dwnldns.gif" ALT=“Ejemplo" WIDTH=’88’ HEIGHT=’31’>

Imagen con link

<A HREF=‘http://www.google.com’><IMG SRC=‘/img/logo.jpg’ BORDER=‘0’></A>

Etiquetas de uso general



Tipo de letras, tamaño y color

Color

```
<FONT color='red'>Estoy en rojo</FONT>
<FONT COLOR='#FF0000'>Tengo Color</FONT>
```

Tamaño

```
<FONT SIZE='2'>Tamaño 2</FONT>
```

Tipo

```
<FONT SIZE='2' FACE='Helvetica,Arial,Times'>No sé como voy a salir exactamente</FONT>
```

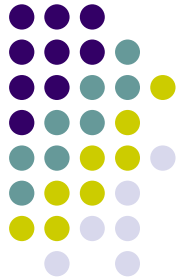
Etiquetas de uso general

Tablas

```
<TABLE>
 <TR>
  <TD>1,1</TD>
  <TD>1,2</TD>
  <TD>1,3</TD>
 </TR>
 <TR>
  <TD>2,1</TD>
  <TD>2,2</TD>
  <TD>2,3</TD>
 </TR>
</TABLE>
```

1,1	1,2	1,3
2,1	2,2	2,3

Etiquetas de uso general



<TABLE BORDER=‘1’ WIDTH=‘50%’ ALIGN=‘CENTER’>

BORDER: Especifica el grosor del borde que se dibujará alrededor de las celdas. Por defecto es cero, lo que significa que no dibujará borde alguno.

CELLSPACING: Define el número de pixels que separarán las celdas.

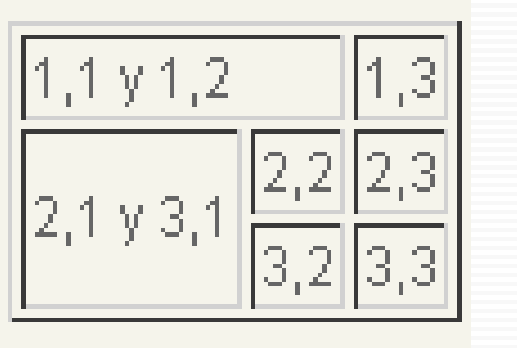
CELLPADDING: Especifica el número de pixels que habrá entre el borde de una celda y su contenido.

WIDTH: Especifica la anchura de la tabla. Puede estar tanto en pixels como en porcentaje de la anchura total disponible para él (pondremos "100%" si queremos que ocupe todo el ancho de la ventana del navegador).

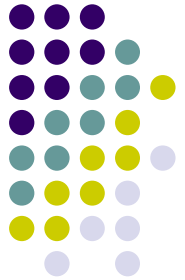
ALIGN: Alinea la tabla a izquierda (LEFT), derecha (RIGHT) o centro (CENTER).

Etiquetas de uso general

```
<TABLE BORDER='1'>
 <TR>
   <TD COLSPAN=2>1,1 y 1,2</TD>
   <TD>1,3</TD>
 </TR>
 <TR>
   <TD ROWSPAN=2>2,1 y 3,1</TD>
   <TD>2,2</TD>
   <TD>2,3</TD>
 </TR>
 <TR>
   <TD>3,2</TD>
   <TD>3,3</TD>
 </TR>
</TABLE>
```



¿Preguntas?



Ejercicio

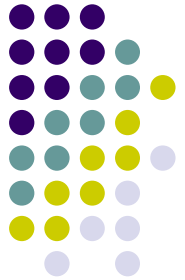
Realizar una pagina HTML que muestre los datos personales de una persona. (utilizar tablas, imágenes, etc.)

Datos Personales

Esta es una pagina …..

Nombre:	xxxxx
Telefono:	xxxxx
E-mail:	xxxx@xxxx.com

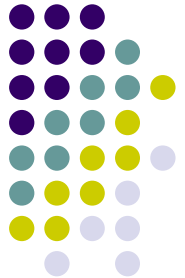
Introducción JSP - Servlets



Los Servlets y Java Server Pages (JSP) son dos métodos de creación de páginas web dinámicas en servidor usando el lenguaje Java.

En ese sentido son similares a otros métodos o lenguajes tales como el PHP, los CGIs (common gateway interface), o ASP (Active Server Pages). Sin embargo, se diferencian de ellos en otras cosas.

Introducción JSP - Servlets

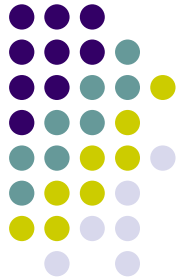


Para empezar, los Servlets y JSP se ejecutan en una máquina virtual Java, lo cual permite que, en principio, se puedan usar en cualquier OS y Hardware, siempre que exista una máquina virtual Java.

Cada Servlet/JSP se compila y se ejecuta en su propio hilo de ejecución, es decir, en su propio contexto

En el servidor existe una única instancia de cada Servlet o JSP que es compartida por todos por igual.

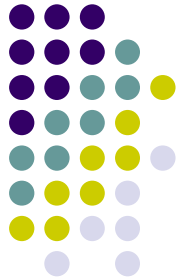
Introducción JSP - Servlets



Este modelo plantea una reutilización constante de las instancias favoreciendo la performance y el consumo de memoria.

Muy diferente al modelo de PHP, ASP, etc, donde la pagina es re procesada por cada petición.

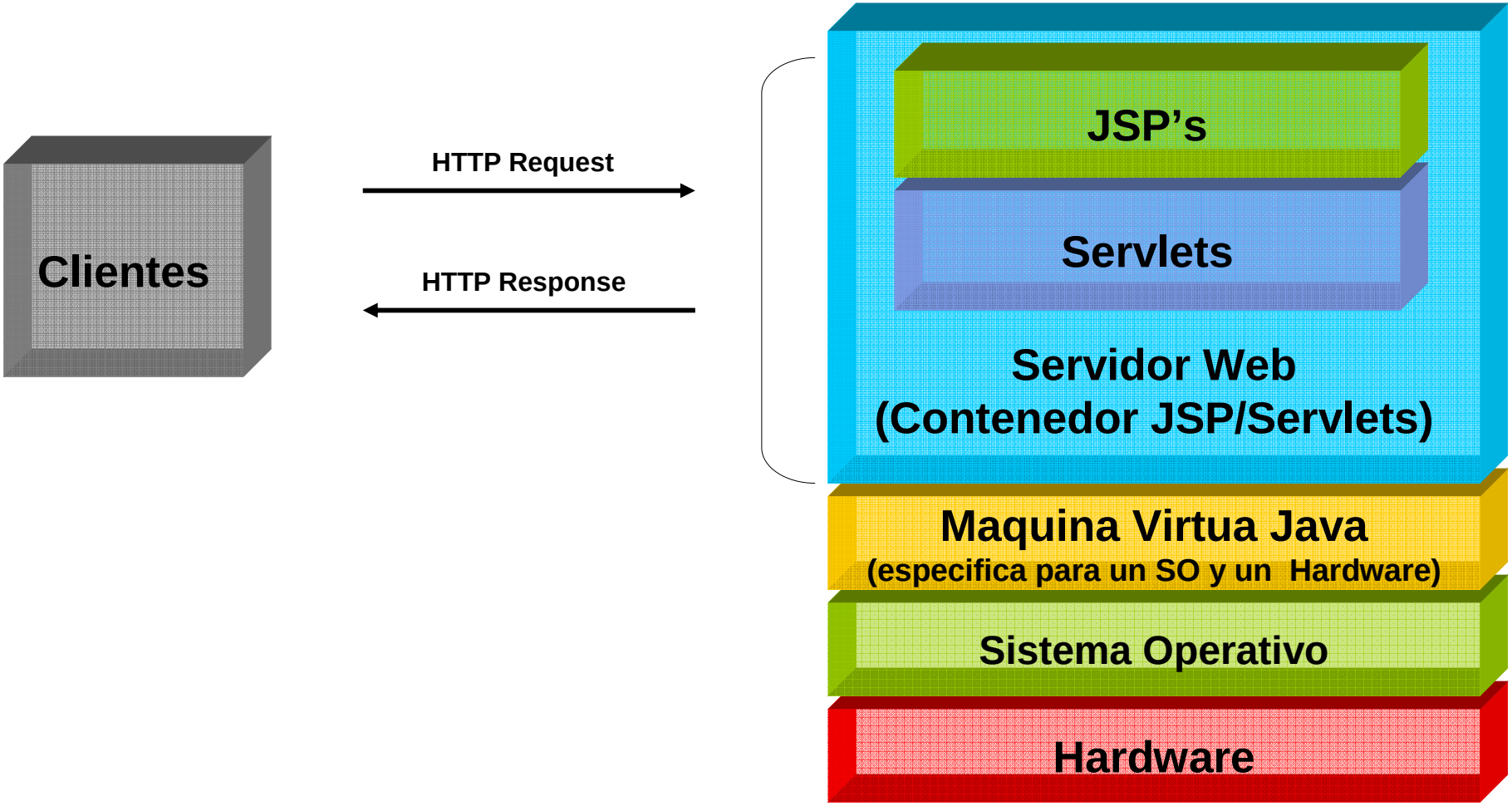
Introducción JSP - Servlets



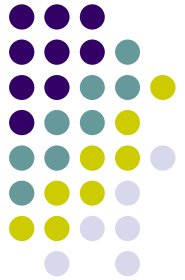
Las paginas JSP son en realidad Servlets: una pagina JSP se compila y se crea una clase que se empieza a ejecutar en el servidor como un Servlet.

La principal diferencia entre los Servlets y JSP es el enfoque de la programación: un JSP es una página Web con etiquetas especiales y código Java incrustado entre html, mientras que un Servlet es un programa (clase) que recibe peticiones y genera a partir de ellas una página web (out.println……)

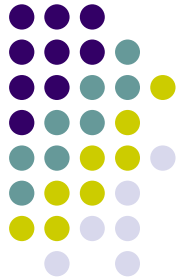
Introducción JSP - Servlets



¿Preguntas?



Introducción JSP

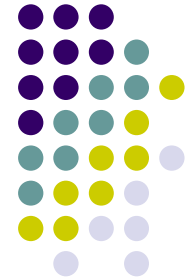


JavaServer Pages (JSP) nos permiten separar la parte dinámica de nuestras páginas Web del HTML estático.

Simplemente escribimos el HTML regular de la forma normal, usando cualquier herramienta de construcción de paginas Web que usemos normalmente.

Encerramos el código de las partes dinámicas en unas etiquetas especiales, la mayoría de las cuales empiezan con "<%" y terminan con "%>".

Introducción JSP – hola.jsp



```
<%@ page language="java" contentType="text/html" %>
<%! int contador=0; %>
<html>
<head>
<title>Hola - Intro JSP</title>
</head>
<%
contador = contador + 1;
%>
<body bgcolor="white">
Hola, mundo. Te lo repito  <%= contador %>
<% if (contador == 1) { %>
 vez
<% } else { %>
 veces
<% } %>
</body>
</html>
<%-- mi comentario Hola!!!  --%>
```

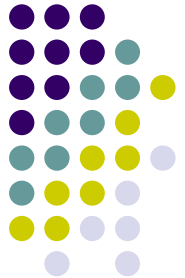
Declaración

Directiva Page

Scriptlets

Expresión

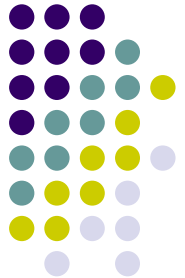
Introducción JSP – hola.jsp



```
<%@ page language="java" contentType="text/html" %>
<%! int contador=0; %>
<html>
<head>
<title>Hola - Intro JSP</title>
</head>
<%
contador = contador + 1;
%>
<body bgcolor="white">
Hola, mundo. Te lo repito  <%= contador %>
<% if (contador == 1) { %>
 vez
<% } else { %>
 veces
<% } %>

</body>
</html>
```

```
public class xxxxxx extends ServletXXX{

  private int contador=0;

  public void serviceXXX() {

    out.println("<html>");
    out.println("<head>");
    out.println("<title>Hola - Intro JSP</title>");
    out.println("</head>");

    contador = contador + 1;

    out.println("<body bgcolor='white'>");
    out.println("Hola, mundo. Te lo repito " + contador);
    if (contador == 1) {
      out.println("vez");
    } else {
      out.println("veces");
    }
    out.println("</body>");
    out.println("</html>");
    .....
  }
}
```

Directiva Page

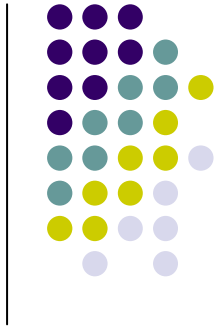


Para empezar, la línea **<%@ page language="java" contentType="text/html" %>** incluye una directiva, que son órdenes que se ejecutan antes de que se comience a procesar la pagina JSP, y modifican de alguna forma el resultado del mismo.

Todas las directivas en JSP se indican con una @ después del comienzo de la orden JSP (<%). En el ejemplo, le indicamos que la página que se va a ejecutar está en el lenguaje Java, y que el contenido que va a generar es de tipo text/html.

El estándar JSP, a pesar de su nombre, no está limitado a un solo lenguaje, aunque en la práctica se usa casi siempre Java.
En cuanto al contenido, JSP trabaja, por defecto, con HTML, pero se podrían generar páginas en otros estándares tales como el WML que se usa en los móviles WAP, o XML.

Directiva Page

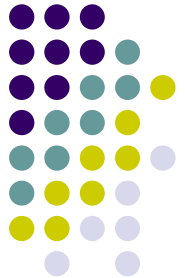


La directiva page puede contener mas parámetros como ser:

```
<%@ page language="java" contentType="text/html"
info="Mi primera página en JSP" import="java.util.*"
errorPage='errorQueTeKgas.jsp' %>
```

import="package.class"
contentType="MIME-Type"
isThreadSafe="true|false"
session="true|false"
buffer="sizekb|none"
autoflush="true|false"
info="message"
errorPage="url"
isErrorPage="true|false"
language="java"

Directiva Page

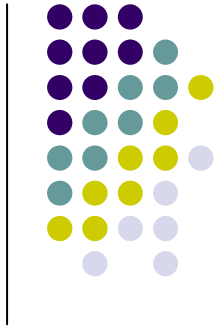


import="**package.class"** o import="**package.class1,...,package.classN"**. Esto nos permite especificar los paquetes que deberían ser importados. Por ejemplo: <%@ page import="java.sql.*" %> El atributo import es el único que puede aparecer múltiples veces.

contentType="**MIME-Type"** o contentType="**MIME-Type; charset=Character-Set"** Esto especifica el tipo MIME de la salida. El valor por defecto es text/html.

isThreadSafe="true|false". Un valor de true (por defecto) indica un procesamiento del servlet normal, donde múltiples peticiones pueden procesarse simultáneamente con un sólo ejemplar del servlet, bajo la suposición que del programador sincroniza las variables de acesso concurrente. Un valor de false indica que el servlet debería implementar SingleThreadModel, con peticiones enviadas serialmente o con peticiones simultáneas siendo entregadas por ejemplares separados del servelt.

Directiva Page

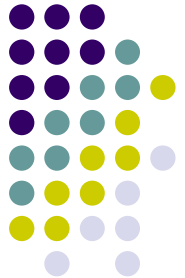


session="true|false". Un valor de true (por defecto) indica que la variable predefinida session (del tipo HttpSession) debería unirse a la sesión existente si existe una, si no existe se debería crear una nueva sesión para unirla. Un valor de false indica que no se usarán sesiones, y los intentos de acceder a la variable session resultarán en errores en el momento en que la página JSP sea traducida a un servlet.

buffer="**sizekb|none"**. Esto especifica el tamaño del buffer para el JspWriter out. El valor por defecto es específico del servidor, debería ser de al menos 8kb.

autoflush="true|false". Un valor de true (por defecto) indica que el buffer debería desacargase cuando esté lleno. Un valor de false, raramente utilizado, indica que se debe lanzar una excepción cuando el buffe se sobrecargue. Un valor de false es ilegal cuando usamos buffer="none".

Directiva Page

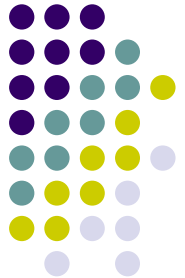


info="**message**". Define un string que puede usarse para ser recuperado mediante el método getServletInfo.

errorPage="**url**". Especifica una página JSP que se debería procesar si se lanzará cualquier Throwable pero no fuera capturado en la página actual.

isErrorPage="true|false". Indica si la página actual actúa o no como página de error de otra página JSP. El valor por defecto es false.

Declaración

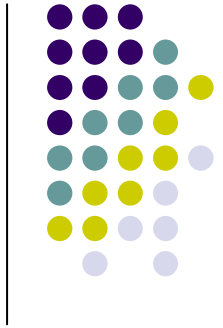


La línea **<%! int contador=1; %>** es una declaración; la admiración (!) se usa para indicar declaraciones de variables globales, es decir, variables persistentes de una llamada a otra; y es compartida por todas las llamadas a una página.

Una declaración en JSP, puede definirse como una definición de variables y métodos a nivel de clase que son usadas en la página.

La declaración se ejecutará la primera vez que se llame a la página, y se volverá a ejecutar cada vez que se recompile la página (o se rearranque el servidor); el efecto que tendrá esta declaración en el ejemplo será un contador de visitas, que se incrementará cada vez que se visite.

Declaración

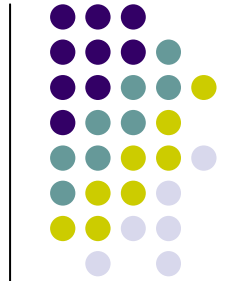


```
<%!
String strCadena = "x";
int intContador = 0;
%>
```

Por lo demás, la declaración es una declaración normal en Java; se pueden incluir tantas declaraciones como se quieran, no necesariamente al principio de la página. También podemos definir cualquier método para ser llamado desde cualquier parte de la pagina JSP

```
<%!
public String calcularElMonto(double monto) {
....
....
}
%>
```

Scriptlets



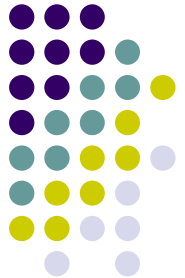
Los scriptlets usan la sintaxis JSP normal, <% y %>; y dentro, se puede incluir cualquier código Java que se quiera. En este caso es una sentencia if, y lo único relevante es que el código Java está mezclado con el texto; en este caso, se incluiría condicionalmente un texto u otro en la página de salida.

```
<% if (contador == 1) { %>
 vez
<% } else { %>
 veces
<% } %>
```

Sin embargo, de esta forma, se pueden incluir más fácilmente todo tipo de estructuras HTML, sin necesidad de meter montones de out.println.

```
<% for (int i=0;i<5;i++) { %>
    <BR>El valor del contador es <%=i%>
<% } %>
```

Expresión



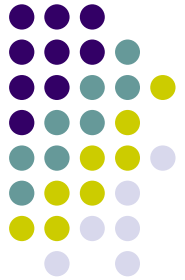
Todo lo que comience por <%= es una expresión JSP; el efecto que tiene es la evaluación de la expresión y la impresión del resultado. En este caso, se ejecuta la expresión y luego se imprime el valor. Las expresiones en JSP pueden contener sólo expresiones Java que devuelvan un valor, y no hace falta que se terminen con ;, como en el caso de las declaraciones.

<%= contador %>

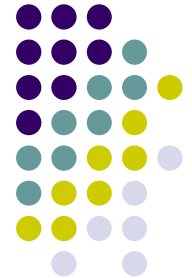
La expresión Java es evaluada, convertida a un string, e insertada en la página.

<% out.println(“El numero es: " + contador); %>

¿Preguntas?



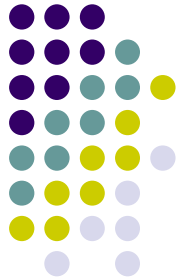
Objetos implicitos



El motor JSP nos ofrece instancias de un conjunto de clases. Son objetos ya establecidos, que no tenemos más que usar (no hay que instanciarlos). Deben utilizarse dentro del código Java. Algunos objetos implícitos:

request
response
session
application
pageContext
out
config

Objetos implicitos



request: contiene información de datos enviados al servidor a través de una página web por parte del cliente. En el se pueden encontrar cookies, parámetros existentes en el queryString, información de formularios, etc. Es del tipo HttpServletRequest.

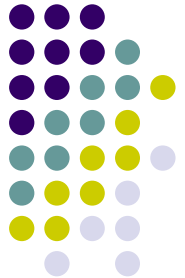
Por ejemplo el método *getParameter()* del objeto *request* recoge valores enviados por medio de un formulario o una URL. Así podemos recuperar los valores o contenido de los campos de un fórmulario de una página web, ejemplo:

```
<%
      String nombre =  request.getParameter("nombre");
%>
```

Este ejemplo podría recuperar el valor del campo nombre de la siguiente URL.

http://127.0.0.1/objetorequest/requestest.jsp?nombre=Pedro

Objetos implicitos

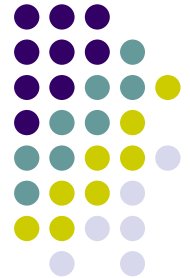


http://127.0.0.1/ejemplo.jsp?parametro=12345

Ejemplo:

```
<html>
<body>
   Su IP: <%=request.getRemoteAddr()%>
   <br>
   Su nombre de host: <%= request.getRemoteHost() %>
   <br>
   Valor del parámetro:
   <%= request.getParameter("parametro") %>
</body>
</html>
```

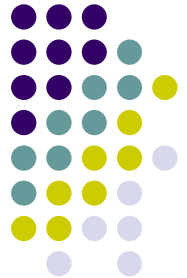
Objetos implicitos



response: permite trabajar con la respuesta que es enviada al cliente, en ella podemos manejar el tipo de salida, contenido, manejo de cabeceras, códigos de estado, cookies, entre otras cosas. Es del tipo HttpServletResponse.

```
<%
if(cliente == null) {
    response.sendRedirect(“/clientes/altacliente.jsp")
}
%>
```

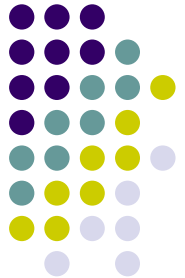
Objetos implicitos



out: Es un objeto de la clase JspWriter, es el que nos permite acceder a la salida del navegador desde los scriplet.
Ejemplo:

```
<%
   out.println(“<b>cadena</b>");
   out.println(“1234");
      %>
```

Objetos implicitos

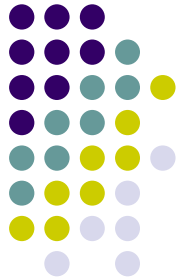


aplication: Es un objeto de la clase ServletContext. Este objeto es común para toda la aplicación web y, entre otras cosas, nos permite almacenar información que será accesible desde todas las páginas de la aplicación web.

Para guardar y recuperar valores:

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

Objetos implicitos

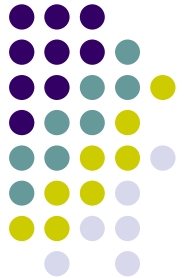


session: Es un objeto de la clase HttpSession. Nos permite acceder a la sesión asociada a la petición. A través de este objeto podemos, entre otras cosas, guardar objetos que serán accesibles desde cualquier JSP de la sesión o invalidarla. La sesión es única por cada usuario conectado al servidor con el mismo navegador.

Para guardar y recuperar información usaremos:
Object session.getAttribute("clave");
void session.setAttribute("clave", Object objeto);

Y para invalidar la sesión:
void session.invalidate();

Objetos implicitos



pageContext: Es un objeto del tipo javax.servlet.jsp.PageContext y es la union de los direfentes objetos vistos anteriormente (aplication, session ,request, etc).

config: Es un objeto de la clase ServletConfig. Permite acceder a parámetros de
inicialización del servlet y a su contexto. Este tema será tratado mas adelante en
el curso.

¿Preguntas?