Universidad Autónoma del Estado de México
Centro Universitario UAEM Valle de México

Ingeniería en Computación

UNIDAD DE APRENDIZAJE:

LENGUAJE DE PROGRAMACIÓN VISUAL

TEMA:

SERVLETS Y JAVA SERVER PAGES

Elaboró: Dr. en C. Héctor Rafael Orozco Aguirre

Agosto de 2019

Universidad Autónoma del Estado de México
UAEM

Secretaría de Docencia
Dirección de Estudios Profesionales

PROGRAMA DE ESTUDIO POR COMPETENCIAS

LENGUAJE DE PROGRAMACIÓN VISUAL

I. IDENTIFICACIÓN DEL CURSO

Espacio Educativo: Facultad de Ingeniería

Licenciatura: Ingeniería en Computación

Año de aprobación por el Consejo Universitario:

Área de docencia: Programación e Ingeniería de Software

Aprobación por los HH. Consejos Académico y de Gobierno

Fecha:

Programa elaborado por: Ing. Ma. Del Consuelo Mañon Salas

Programa revisado por: Integrantes de la Academia de Programaci e Ingeniería de Software

Fecha de elaboración : Enero de 2006

Clave	Horas de teoría	Horas de práctica	Total de horas	Créditos	Tipo de curso	Núcleo de formación
L41091	2	1	3	5	Curso-Laboratorio	Integral

Unidad de Aprendizaje Antecedente
Programación Estructurada

Unidad de Aprendizaje Consecuente
Ninguna

Programas educativos o espacios académicos en los que se imparte:
Licenciatura en Ingeniería en Computación (Facultad. de Ingeniería, Centros Universitarios: Atlacomulco, Ecatepec, Texcoco, Valle de Chalco, Valle México, Valle de Teotihuacán, Zumpango)

Universidad Autónoma del Estado de México

Secretaría de Docencia

Dirección de Estudios Profesionales

DISTRIBUCIÓN DE LAS UNIDADES DE APRENDIZAJE OPTATIVAS

	PERIODO 1	PERIODO 2	PERIODO 3	PERIODO 4	PERIODO 5	PERIODO 6	PERIODO 7	PERIODO 8	PERIODO 9	PERIODO 10
OPTATIVAS – Núcleo básico			Análisis de Fourier 2 0 2 4	Ética 2 0 2 4						
			Cálculo numérico 3 0 3 6	Sociedad e Ingeniería 2 0 2 4						
			Simulación 1 2 3 4	Lectura y redacción 0 2 2 2						
			Inferencia estadística 2 1 3 5	Química general 3 1 4 7						
			Variable compleja 3 0 3 6							
Línea 4: Desarrollo de software de aplicación				Métricas de software 2 2 4 6	Análisis de lenguajes de programación 2 2 4 6	Lenguaje de programación estructurado 2 1 3 5	Lenguaje de programación visual 2 1 3 5	Lenguaje de programación orientado a objetos 2 1 3 5	Minería de datos 2 2 4 6	Desarrollo multimedia 1 2 3 4
Línea 2: Redes y comunicaciones						Instalaciones y equipos 2 1 3 5	Tipos y configuraciones 2 1 3 5	Auditoría de redes 1 2 3 4	Servicios de internet 1 2 3 4	Interconexión de redes 1 2 3 4

Propósito de la Unidad de Aprendizaje

- El alumno conocerá la estructura de un lenguaje de programación orientado a objetos, el cual explotará como herramienta para el diseño y elaboración de páginas WEB.

Contenido

- Servlets Java:
  - Definición, características y ventajas
  - Tareas, cuándo y por qué usarlos
  - Estructura, ciclo de vida y anatomía
  - Interfaces ServletRequest y ServletResponse
  - Ejemplo

Contenido

- Java Server Pages:
  - Definición, funcionamiento y beneficios
  - Contenido y elementos
  - Ejemplos

Contenido

- IDE de desarrollo
- Motor de Servlet
- JSP vs Servlet

Guion explicativo

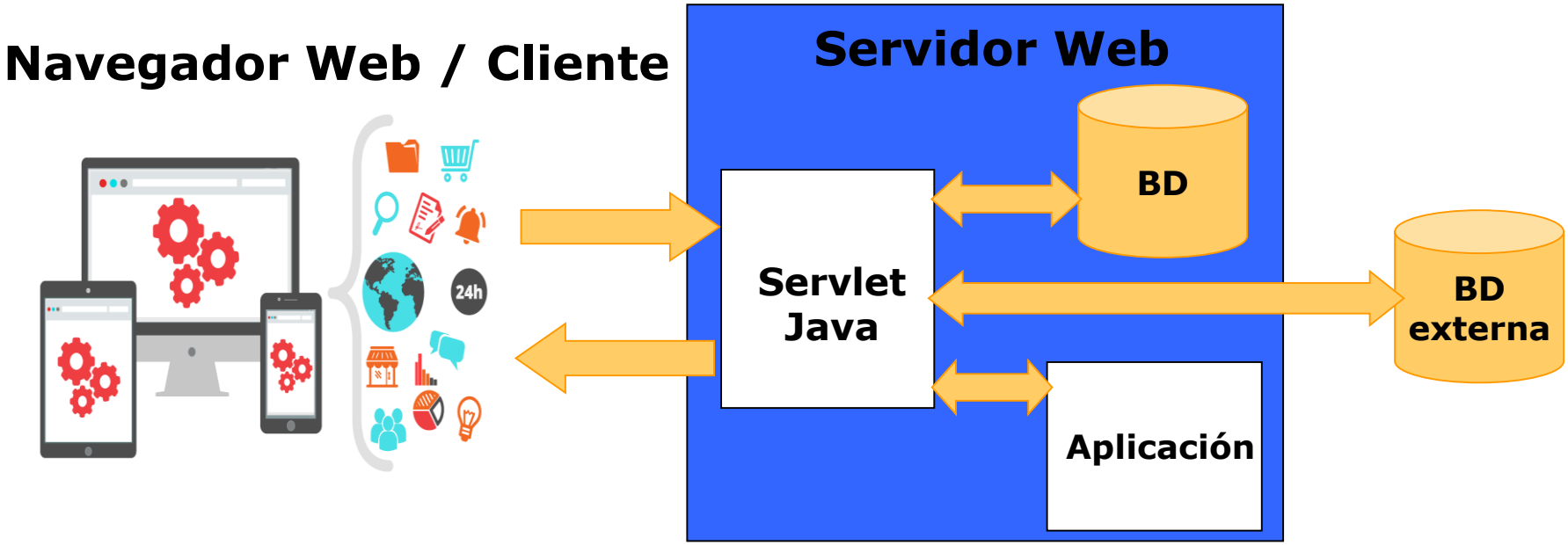
- Esta presentación tiene como fin dar a conocer a los alumnos los siguientes aspectos:
  - ¿Qué es un servlet Java?
  - Creación de servlets en Java.
  - ¿Qué son las Java Server Pages (JSP)?
  - Creación de JSP en Java.
  - IDE de desarrollo, motor de Servlet y comparativa.

Guion explicativo

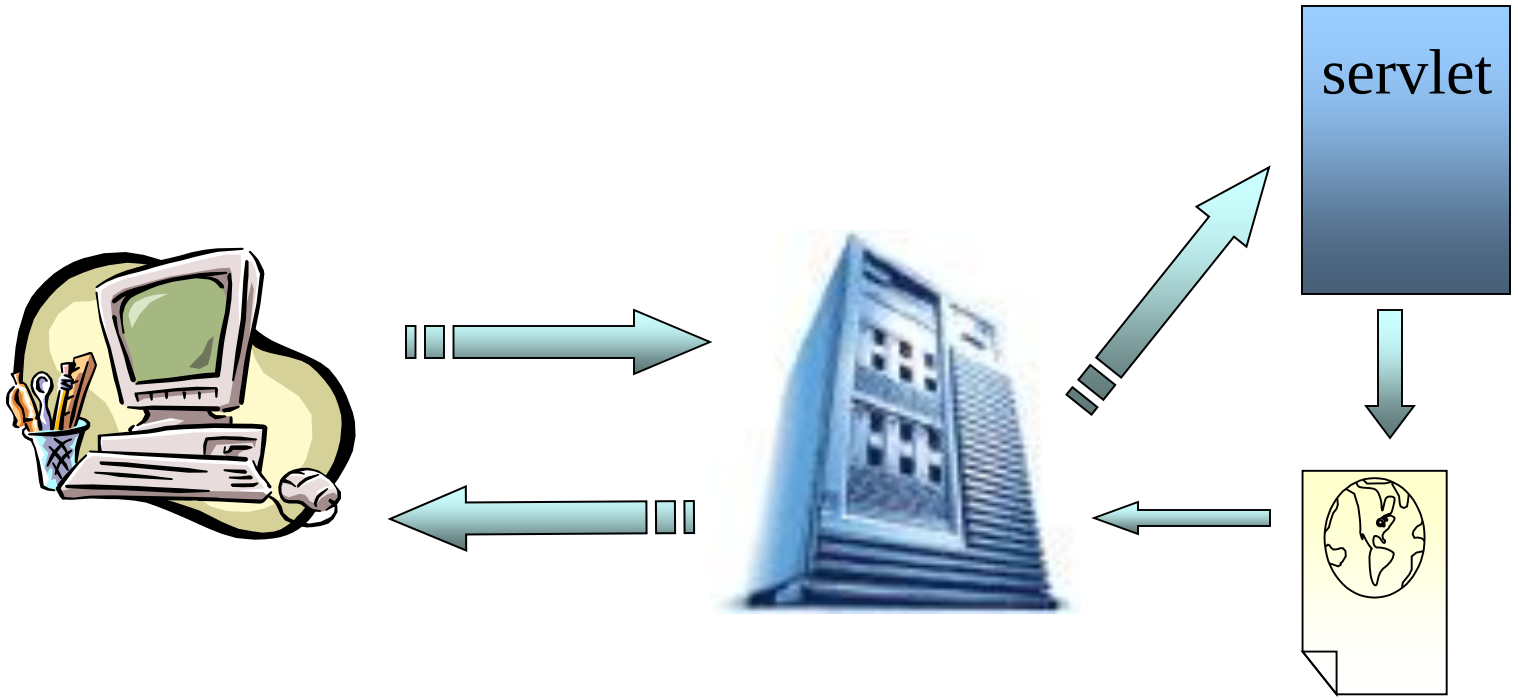
- El contenido de esta presentación contiene elementos de competencia (temas) de interés contenidos en la Unidad de Aprendizaje de Lenguaje de Programación Visual, en específico de las Unidades de Competencia II y III.

- Las diapositivas deben explicarse en orden, y deben revisarse aproximadamente en 6 horas, además de realizar preguntas y dejar prácticas al grupo sobre el contenido mostrado.

Servlets Java



Servlets Java



¿Qué son los Servlets Java?

- Fueron introducidos por Sun en 1996 como pequeñas aplicaciones Java para añadir funcionalidad dinámica a los servidores web.
- Los Servlets son la respuesta de la tecnología Java a la programación CGI, reciben una petición del cliente y generan los contenidos apropiados para su respuesta, aunque el esquema de funcionamiento es diferente al de CGI.
- Son programas que se ejecutan en un servidor Web y construyen páginas Web dinámicamente.

Características de los Servlets

- Son independientes del servidor utilizado y de su sistema operativo, lo que quiere decir que a pesar de estar escritos en Java, el servidor puede estar escrito en cualquier lenguaje de programación, obteniéndose exactamente el mismo resultado que si lo estuviera en Java.

- Pueden llamar a otros servlets, e incluso a métodos concretos de otros servlets. De esta forma se puede distribuir de forma más eficiente el trabajo a realizar. De igual forma, los servlets permiten redireccionar peticiones de servicios a otros servlets (en la misma máquina o en una máquina remota).

Características de los Servlets

- Pueden obtener fácilmente información acerca del cliente (la permitida por el protocolo HTTP), tal como su dirección IP, el puerto que se utiliza en la llamada, el método utilizado (GET, POST, etc.), etc.

- Permiten además la utilización de cookies y sesiones, de forma que se puede guardar información específica acerca de un usuario determinado, personalizando de esta forma la interacción cliente-servidor. Una clara aplicación es mantener la sesión con un cliente.

Características de los Servlets

- Pueden actuar como enlace entre el cliente y una o varias bases de datos en arquitecturas cliente-servidor de 3 capas (si la base de datos está en un servidor distinto).

- Pueden realizar tareas de proxy para un applet. Debido a las restricciones de seguridad, un applet no puede acceder directamente por ejemplo a un servidor de datos localizado en cualquier máquina remota, pero el servlet sí puede hacerlo de su parte.

Características de los Servlets

- Al igual que los programas CGI, los servlets permiten la generación dinámica de código HTML dentro de una propia página HTML. Así, pueden emplearse servlets para la creación de contadores, banners, etc.

Ventajas de los Servlets

- Los Servlets Java destacan sobre otras tecnologías del tipo CGI tradicional en lo siguiente:
  - Eficiencia.
  - Conveniencia.
  - Potencia.
  - Portable.
  - Barato.

Tareas encomendadas a un Servlet

- Leer los datos enviados por un usuario
  - Usualmente de formularios en páginas Web
  - Pueden venir de applets de Java o programas cliente HTTP.
- Buscar cualquier otra información sobre la petición que venga incluida en esta
  - Detalles de las capacidades del navegador, cookies, nombre del host del cliente, etc.

Tareas encomendadas a un Servlet

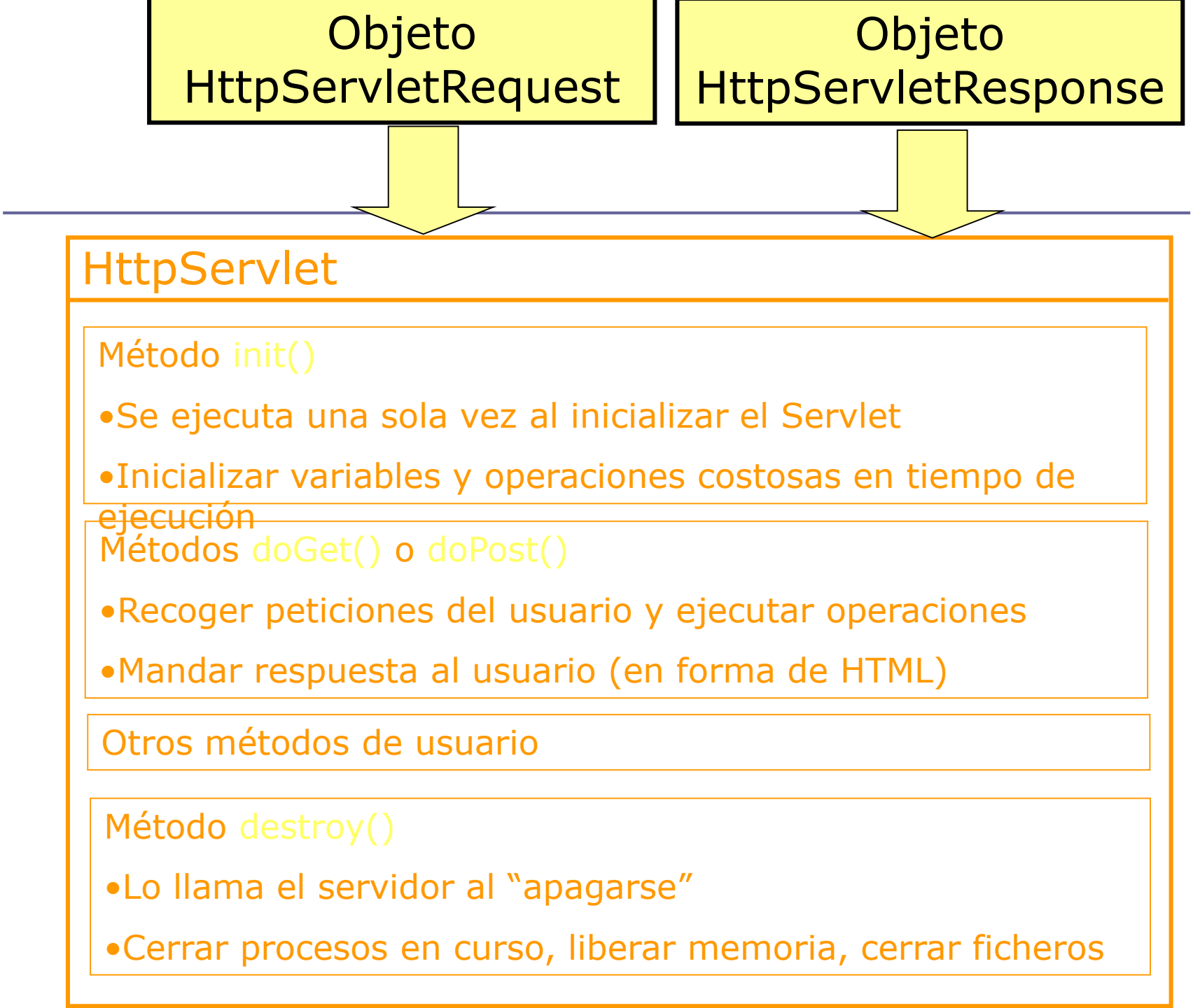
- Generar los resultados
  - Puede requerir consultas a Base de Datos, invocar a otras aplicaciones, computar directamente la respuesta, etc.
- Dar formato a los resultados en un documento
  - Incluir la información en una página HTML
- Establecer los parámetros de la respuesta HTTP
  - Decirle al navegador el tipo de documento que se va a devolver, establecer las cookies, etc.
- Enviar el documento al cliente

¿Cuándo y por qué usar Servlets?

- Muchas peticiones desde navegador se satisfacen retornando ***documentos HTML estáticos***, es decir, que están en ficheros
- En ciertos casos, es necesario generar las páginas HTML para cada petición:
  - **Página Web basada en datos enviados por el cliente**
    - Motores de búsqueda, confirmación de pedidos
  - **Página Web derivada de datos que cambian con frecuencia**
    - Informe del tiempo o noticias de última hora
  - **Página Web que usa información de bases de datos corporativas u otras fuentes del la parte del servidor**
    - Comercio electrónico: precios y disponibilidades

Estructura de un HttpServlet

```
import java.io.*;
//Se importan los paquetes con las clases para Servlets y HttpServlets
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletTemplate extends HttpServlet{

  //El método doGet responde a peticiones mediante el método GET
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                                          throws ServletException, IOException {

    // El objeto "request" se usa para leer los "HTTP headers" que llegan
    // (p.e. Cookies) y los datos de formularios HTML enviados por el usuario

    // El objeto "response" se usa para especificar "HTTP status codes" y
    // "HTTP headers" de la respuesta (p.e. El tipo de contenido, cookies, etc.)

    PrintWriter out = response.getWriter();
    // El objeto "out" se usa para enviar contenido al navegador

  }

  //El método doPost responde a peticiones mediante el método POST
  public void doPost(HttpServletRequest request, HttpServletResponse response)
                                          throws ServletException, IOException {
    doGet(request, response);
  }
}
```


Objeto HttpServletRequest
Objeto HttpServletResponse
HttpServlet
Método init()
•Se ejecuta una sola vez al inicializar el Servlet
•Inicializar variables y operaciones costosas en tiempo de ejecución
Métodos doGet() o doPost()
•Recoger peticiones del usuario y ejecutar operaciones
•Mandar respuesta al usuario (en forma de HTML)
Otros métodos de usuario
Método destroy()
•Lo llama el servidor al "apagarse"
•Cerrar procesos en curso, liberar memoria, cerrar ficheros

Ciclo de vida de los Servlets

- El paquete javax.servlet provee interfaces y clases que permiten escribir y compilar servlets.
- No viene con el j2sdk, es necesario bajar el .jar respectivo y hacerlo visible para compilar los programas, o bien, hacer una instalación del j2EE junto con el J2SE.
- Por su parte, tambien los servidores deben poder saber recibir requerimientos para servlets y saber interpretarlos.
- Cada servidor de servlets tiene sus propias reglas. Uno de los más usados es Apache Tomcat.
- En la mayoría de los casos hay que ponerlos en un directorio específico dentro del directorio de aplicaciones Web del servidor.

Anatomía de un Servlet

- Un nuevo tipo (clase) de servlet es especificado extendiendo la clase HttpServlet
- Existen métodos predefinidos:
  - `init()` es llamado por el servidor web cuando el servlet se "sube" la primera vez (el momento en que sucede esto puede variar, dependiendo del servidor).
  - `doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException`

  se llama cuando el servlet es invocado con una petición Http GET, que es la normal cuando se contacta un serv.
  - `doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException`

  se ejecuta cuando el servlet fue invocado con una petición Http POST

Anatomía de un servlet

- Una petición GET se genera siempre cuando una petición http es ingresada en el navegador Web.
- Cuando el servlet se llama por primera vez, se inicializa y 4-6 threads son levantados para atender a cleintes en paralelo cuando vayan apareciendo, Para ahorrar tiempo
- En la mayoría de los casos los servlets son contactados a través de forms en páginas html. En estos casos es posible especificar en los parámetros una petición POST

Interfaz ServletRequest

- HttpServletRequest es la clase de uno de los parámetros con que el servidor llama a los métodos del servlet que implementa la interfaz ServletRequest provee acceso a:
  - Información que viene del cliente, como por ejemplo nombre de los parámetros pasados y sus valores, el protocolo usado, el nombre del computador del cliente y del servidor que lo atiende, etc.
  - El input stream, ServletInputStream. Los Servlets usan este input stream para recibir datos de los clientes que envían con protocolos de aplicaciones como los métodos POST y PUT de HTTP.

Interfaz ServletResponse

- HttpServletResponse es la clase con el que servidor llama a los métodos del servlet. Implementa la interfaz ServletResponse la cual da al servlet métodos para responder al cliente:
  - Establecer el tipo MIME de la respuesta que se le mandará al cliente
  - El output stream ServletOutputStream y un Writer con el cual van a mandar datos al clientes.

Ejemplo de un Java Servlet

index.html

Source | History

```
<!DOCTYPE html>
<html>
    <head>
        <title>Denos su opini&oacute;n</title>
    </head>
    <body>
        <h2>Opini&oacute;n acerca de  este sitio web</h2>
        <form action = "ServletOpinion" method = "post">
            Nombre: <input type = "text" name = "nombre" size = 15><br>
            Apellidos: <input type = "text" name = "apellidos" size = 30><p>
            Opini&oacute;n de este sitio Web<br>
            <input type = "radio" checked = "checked" name = "opinion"
                value = "buena">Buena<br>
            <input type = "radio" name = "opinion"
                value = "regular">Regular<br>
            <input type = "radio" name = "opinion"
                value = "mala">Mala<p>
            Comentarios <br>
            <textarea name = "comentarios" rows = 6 cols = 40></textarea>
            <br>
            <input type = "submit" name = "botonenviar" value = "enviar">
            <input type = "reset" name = "botonlimpiar" value = "limpiar">
        </form>
    </body>
</html>
```

ServletOpinion.java

Source | History

```java
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ServletOpinion extends HttpServlet {
    private String nombre=null;
    private String apellidos=null;
    private String opinion=null;
    private String comentarios=null;

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        System.out.println("Iniciando ServletOpinion...");
    }

    @Override
    public void destroy() {
       System.out.println("No hay nada que hacer...");
    }
```

ServletOpinion.java

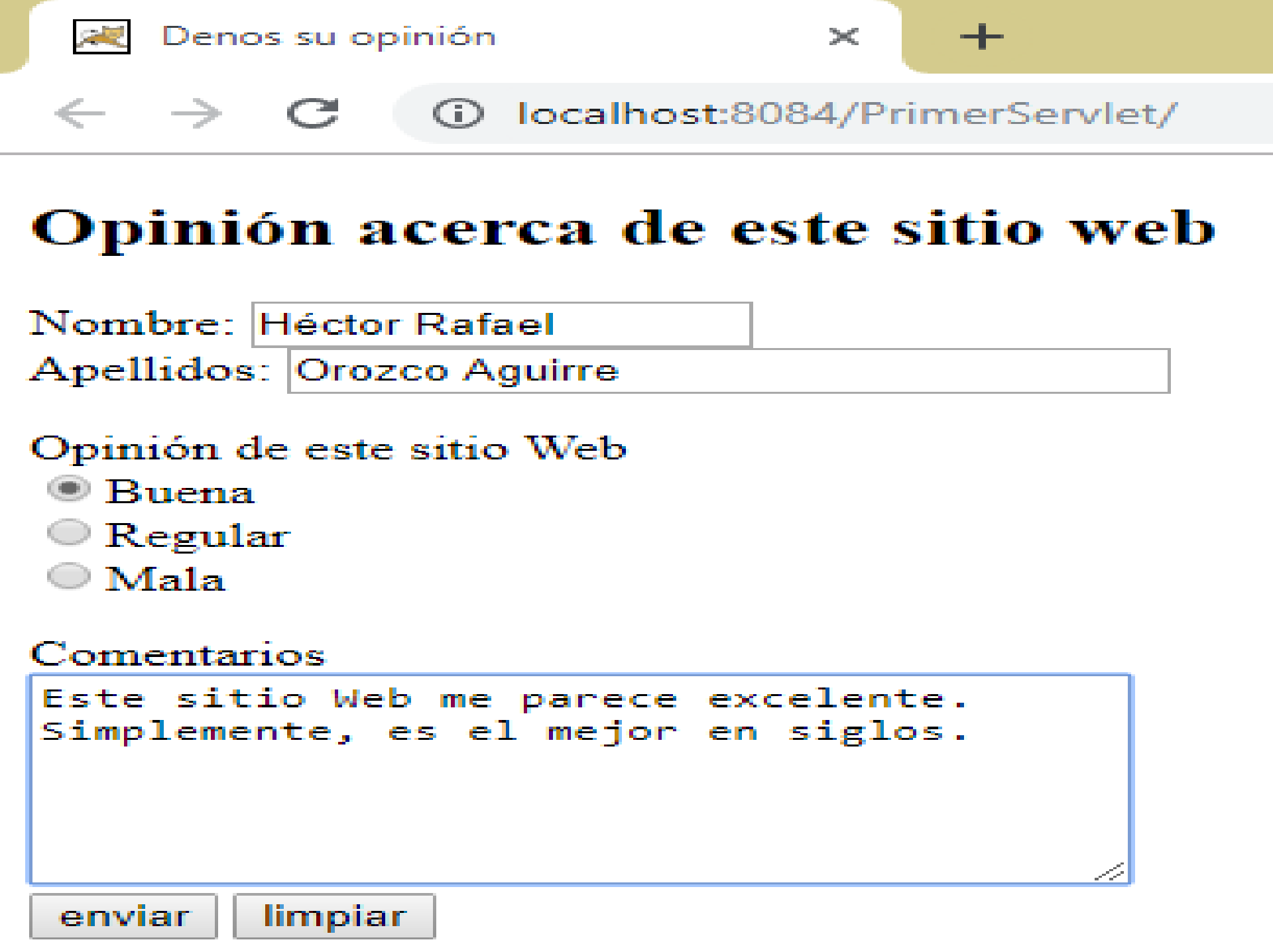
Source | History

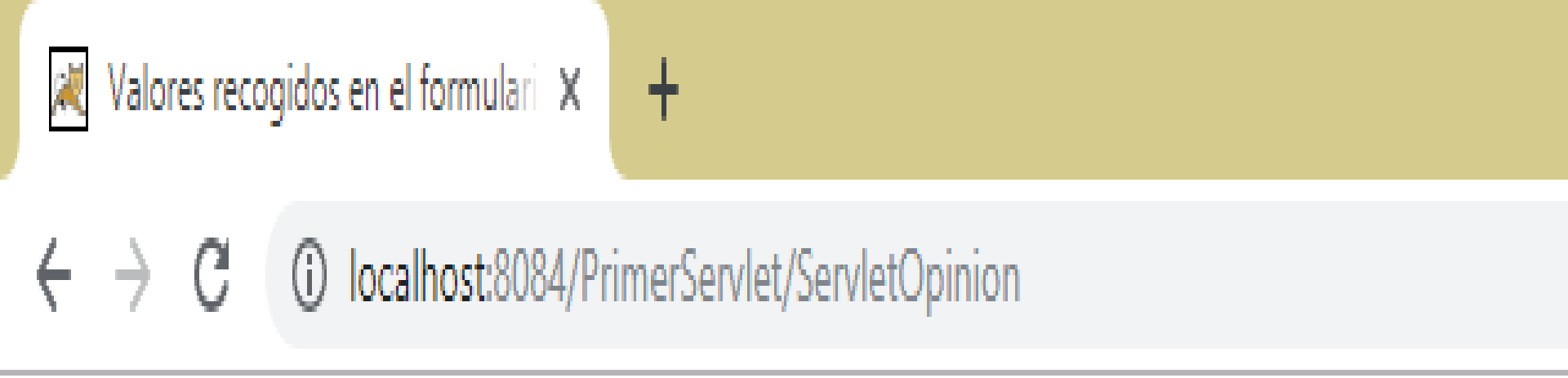
```
25
26      @Override
        public void doPost (HttpServletRequest req, HttpServletResponse resp)
28                          throws ServletException, IOException {
29          nombre=req.getParameter("nombre");
30          apellidos=req.getParameter("apellidos");
31          opinion=req.getParameter("opinion");
32          comentarios=req.getParameter("comentarios");
33          devolverPaginaHTML(resp);
34      }
35
36      public void devolverPaginaHTML(HttpServletResponse resp) {
37          resp.setContentType("text/html");
38          PrintWriter out = null;
39          try {
40              out=resp.getWriter();
41          } catch (IOException io) {
42                System.out.println("Se ha producido una excepcion");
43            }
            out.println("<html>");
45          out.println("<head>");
46          out.println("<title>Valores recogidos en el formulario</title>");
47          out.println("</head>");
48          out.println("<body>");
49          out.println("<b><font size=+2>Valores recogidos del ");
50          out.println("formulario: </font></b>");
```

tOpinion.java

History

```
        out.println("<p><font size=+1><b>Nombre: </b>"+nombre+"</font>");
        out.println("<br><fontsize=+1><b>Apellido: </b>"
                    + apellidos+"</font><b><font size=+1></font></b>");
        out.println("<p><font size=+1> <b>Opini&oacute;n: </b><i>" + opinion
                    + "</i></font>");
        out.println("<br><font size=+1><b>Comentarios: </b>" + comentarios
                    + "</font>");
        out.println("</body>");
        out.println("</html>");
        out.flush();
        out.close();
    }

    @Override
    public String getServletInfo() {
        return "Este servlet lee los datos de un formulario"
        + " y los muestra en pantalla";
    }
}
```


Denos su opinión
localhost:8084/PrimerServlet/
Opinión acerca de este sitio web
Nombre: Héctor Rafael
Apellidos: Orozco Aguirre
Opinión de este sitio Web
Buena
Regular
Mala
Comentarios
Este sitio Web me parece excelente.
Simplemente, es el mejor en siglos.
enviar
limpiar

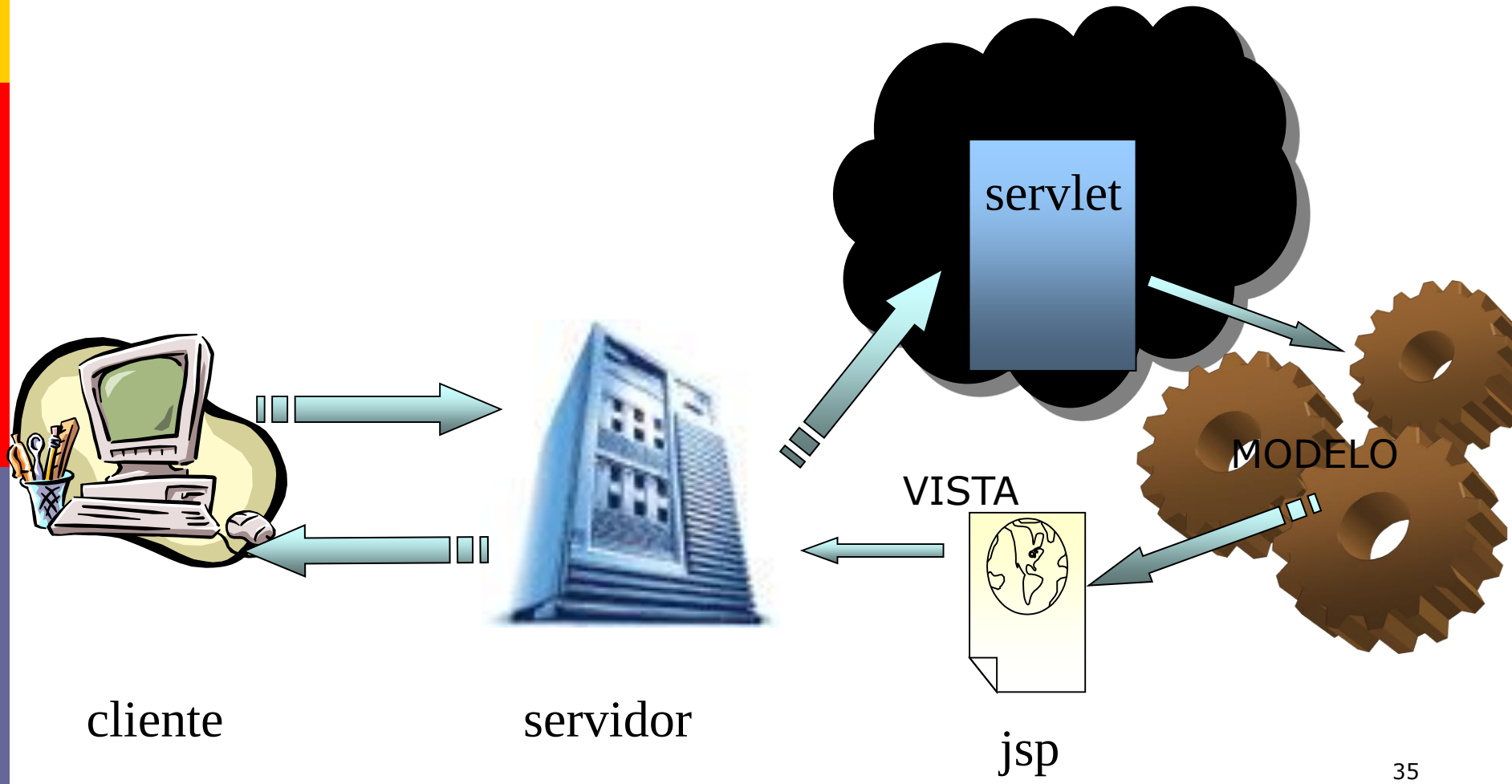


Valores recogidos del formulario:

Nombre: Héctor Rafael
Apellido: Orozco Aguirre

Opinión: *buena*
Comentarios: Este sitio Web me parece excelente. Simplemente, es el mejor en siglos.

Java Server Pages (JSP)



¿Qué es JSP?

- JSP es una tecnología basada en Java que simplifica el proceso de desarrollo de sitios web dinámicos. Las *JavaServer Pages* son archivos de texto (normalmente con extensión *.jsp*) que sustituyen a las páginas HTML tradicionales.

- Los archivos JSP contienen etiquetas HTML y código embebido que permite al diseñador de la página web acceder a datos desde código Java que se ejecuta en el servidor.

¿Qué es JSP?

- Cuando la página JSP es requerida por un navegador a través de una petición HTTP, las etiquetas HTML permanecen inalterables, mientras que el código que contiene dicha página es ejecutado en el servidor, generando contenido dinámico que se combina con las etiquetas HTML antes de ser enviado al cliente.

- Este modo de operar implica una separación entre los aspectos de presentación de la página y la lógica de programación contenida en el código.

¿Qué es JSP?

- JSP se implementa utilizando la tecnología Servlet. Cuando un servidor web recibe una petición de una página *.jsp*, la redirecciona a un proceso especial dedicado a manejar la ejecución de servlets (*servlet container*). En el contexto de JSP, este proceso se llama *JSP container*.

- Aunque las JSP están basadas en la tecnología de servlets, la programación de las mismas es bastante mas sencilla que un servlet, ya que se trabaja directamente sobre la página web que se devolverá al cliente.

¿Cómo funcionan las páginas JSP?

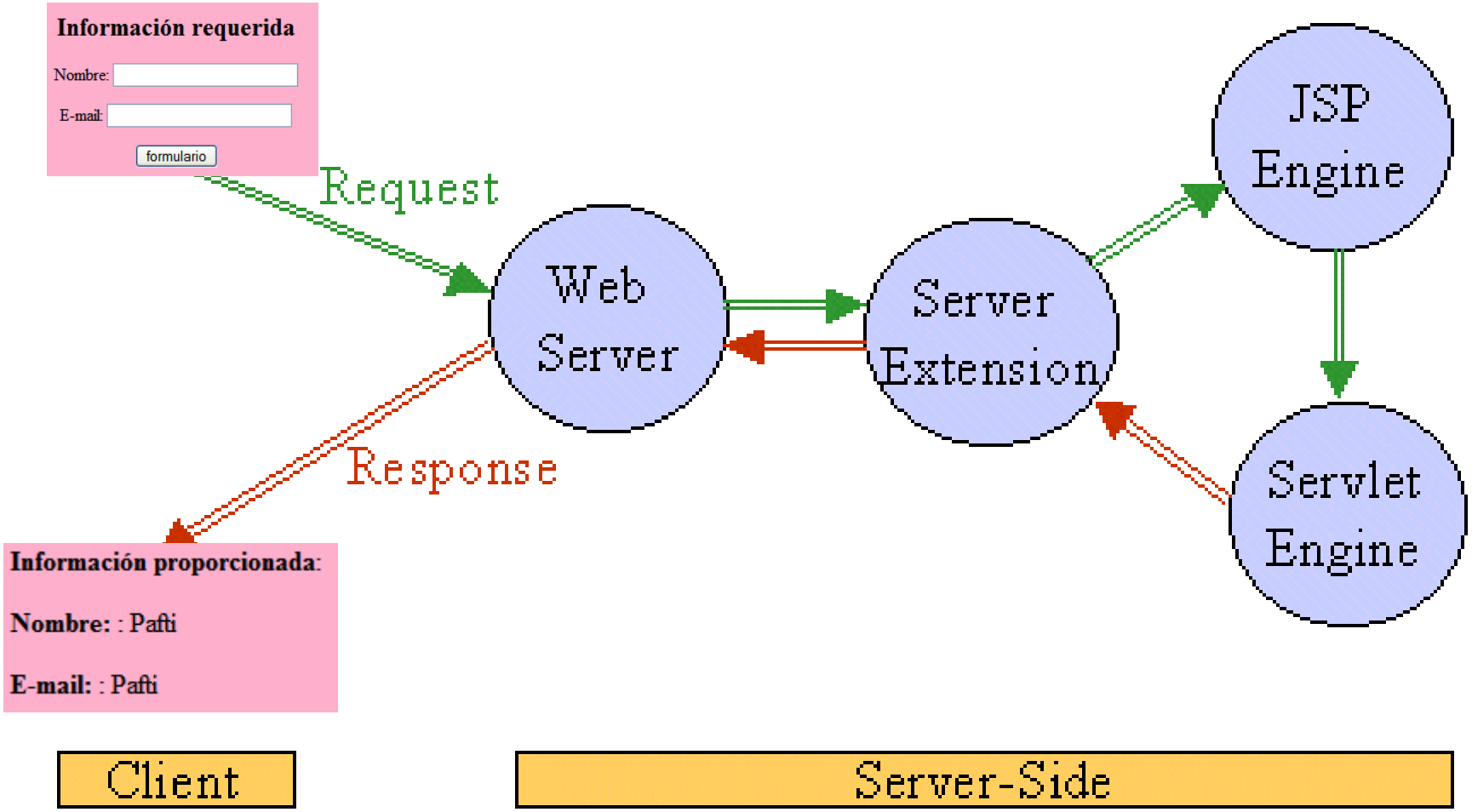
- JSP es una tecnología poderosa utilizada para generar HTML de forma dinámica a petición del usuario. El motor de JSP (que es un servlet realmente) estará en un servidor que recibirá peticiones del usuario. Estas peticiones llegarán del cliente, se analizarán y el servidor dará la respuesta adecuada. Para realizar este simple proceso los pasos que se siguen son:
  - El cliente envía una petición de página (solicita algo).
  - El motor de JSP compila la página en un servlet (crea un servlet).
  - El servlet generado se compila y se carga.
  - El servlet generado se encarga de analizar la petición y generar una respuesta.

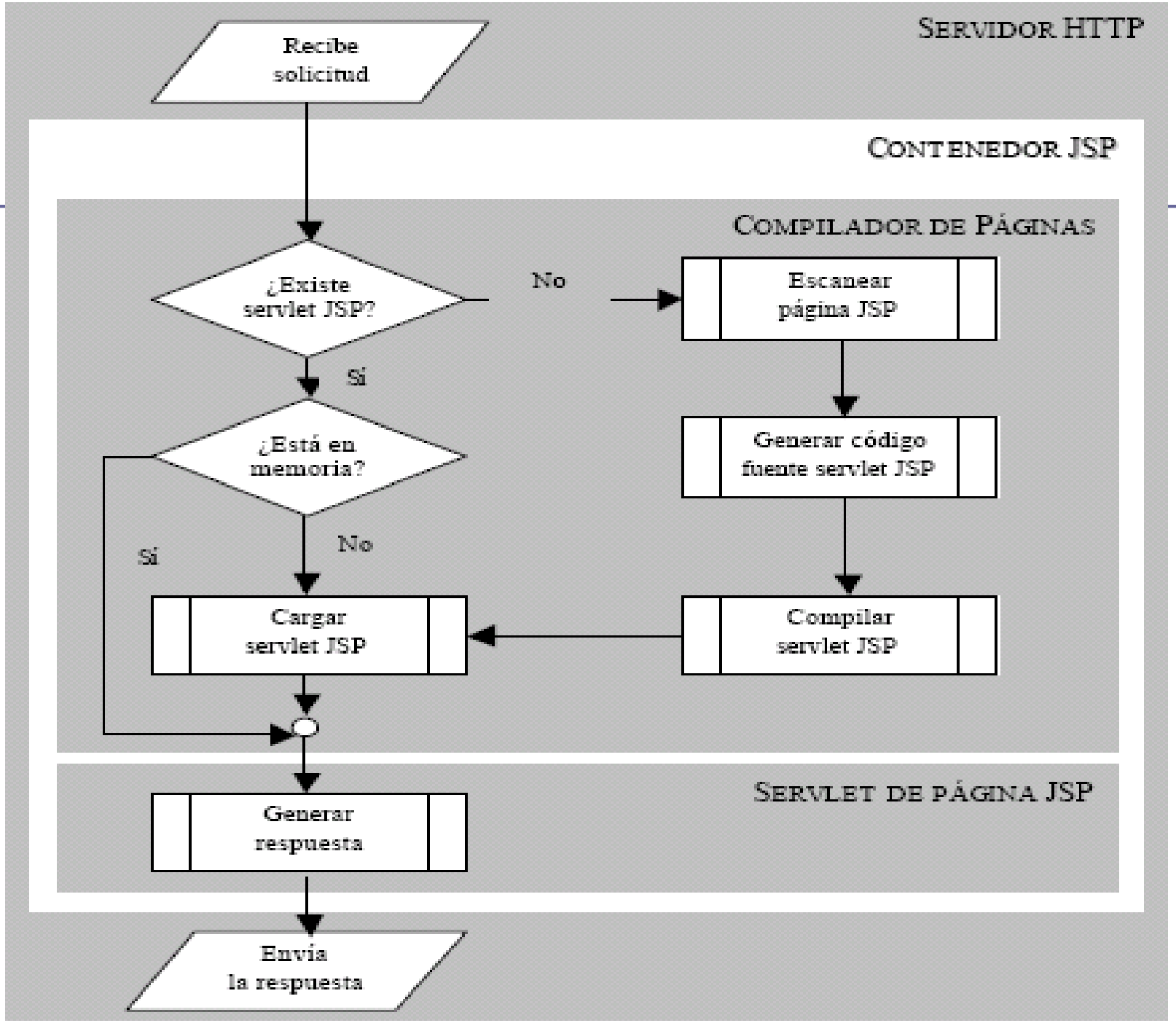
¿Cómo funcionan las páginas JSP?

- Lo anterior dice que cuando se crea una página JSP, la primera vez que se accede a la misma, esta se traduce a un Servlet que se crea y se compila, debido a esto la primera vez que se accede a la página JSP, esta tarda un rato en ser cargada, pero el resto de accesos son mucho mas rápidos debido a que el Servlet ya esta creado.

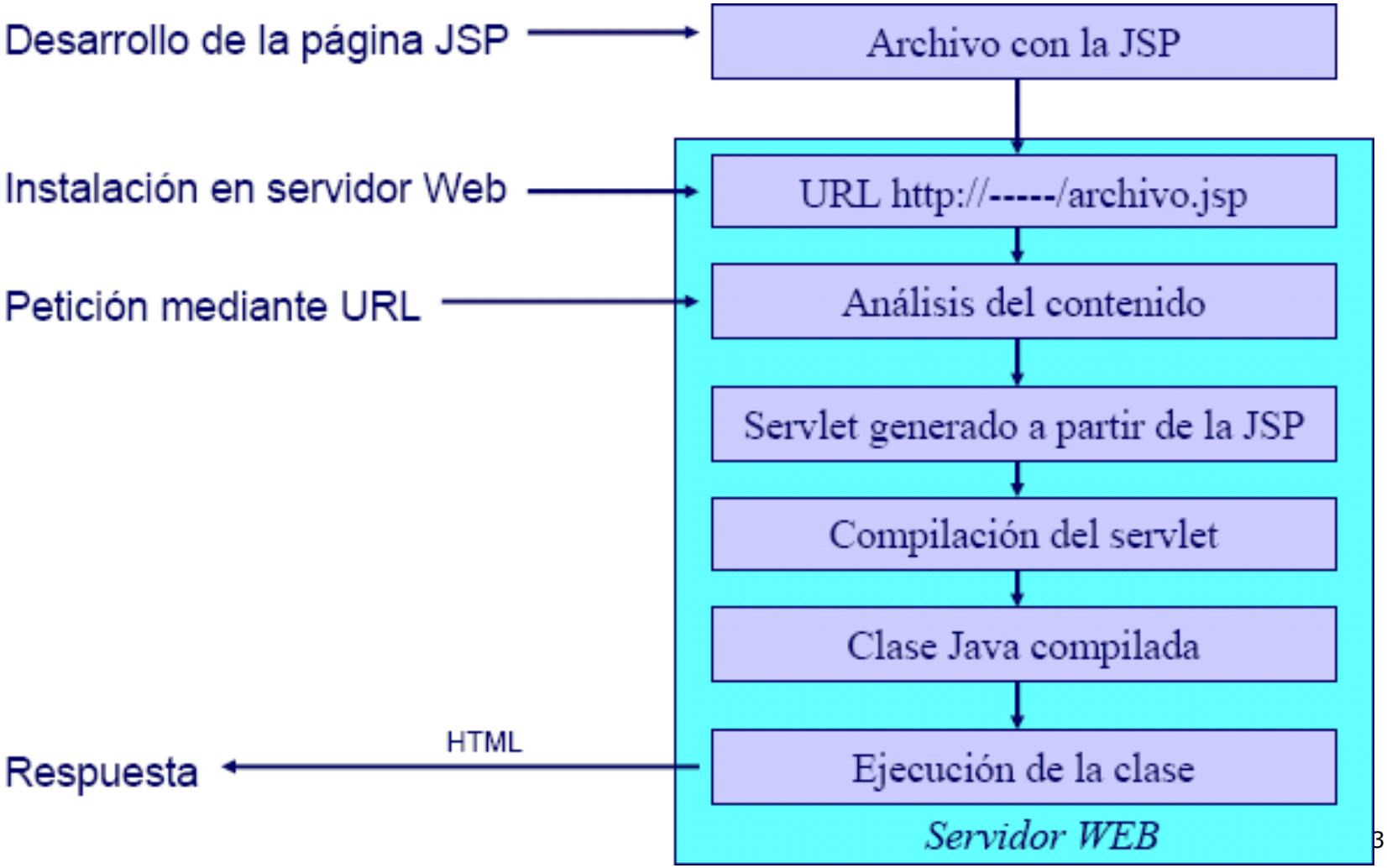
- El servidor realiza el proceso anterior siempre que una pagina sea nueva o haya sido modificada. Si no es así y ya ha sido solicitada previamente, invoca el Servlet que debe de estar generado, lo cuál es muy eficiente.

¿Cómo funcionan las páginas JSP?




SERVIDOR HTTP
Recibe solicitud
CONTENEDOR JSP
COMPILADOR DE PÁGINAS
¿Existe servlet JSP?
No
Escanear página JSP
Si
¿Está en memoria?
Generar código fuente servlet JSP
Si
No
Cargar servlet JSP
Compilar servlet JSP
SERVLET DE PÁGINA JSP
Generar respuesta
Envia la respuesta

¿Cómo funcionan las páginas JSP?



Beneficios que aporta JSP

- **Mejoras en el rendimiento**
  - Utilización de hilos Java para el manejo de peticiones.
  - Manejo de múltiples peticiones sobre una página .jsp en un instante dado.
  - El contenedor servlet puede ser ejecutado como parte del servidor web.
  - Facilidad para compartir recursos entre peticiones (hilos con el mismo padre: *servlet container*).

Beneficios que aporta JSP

- **Soporte de componentes reutilizables**
  - Creación, utilización y modificación de JavaBeans del servidor.
  - Los JavaBeans utilizados en páginas *.jsp* pueden ser utilizados en servlets, applets o aplicaciones Java.

Beneficios que aporta JSP

- **Separación entre el código de presentación y el código de implementación**
  - Cambios realizados en el código HTML relativos a cómo son mostrados los datos, no interfieren en la lógica de programación y viceversa.

Beneficios que aporta JSP

- **División del trabajo**
  - Los diseñadores de páginas pueden centrarse en el código HTML y los programadores en la lógica del programa.
  - Los desarrollos de la página pueden hacerse independientemente.
  - Las frecuentes modificaciones de una página se realizan más eficientemente.

Componentes de una página JSP

- **Contenido de una página JSP**
  - Código HTML
  - Elementos JSP
    - Etiquetas específicas de comienzo y fin
  - Combinación de HTML y JSP

- **Elementos JSP**
  - Comentarios
  - Objetos implícitos
  - Directivas
  - Secuencias de comandos (expresiones, *scriptlets* y declaraciones)
  - Acciones

Comentarios

- Dos tipos:
  - Sólo visibles en página JSP

    <%-- *Comentario JSP oculto en HTML generado* **--%>**

  - Visibles en HTML generado

    <!– *Incluido en el HTML generado* **--%>**

Objetos implícitos

- **Generación del servlet a partir de JSP**
  - Creación del método _jspService()
  - Se escribe el esqueleto principal del método incorporando:
    - Scriptlets
    - HTML
    - Expresiones

Objetos implícitos

- Se inicializa el contexto del servlet con una serie de variables
  - request
  - response
  - pageContext
  - session
  - aplication
  - out
  - config
  - page
  - exception

Objetos implícitos

- **request**: Es el HttpServletRequest asociado con la petición, y permite obtener los parámetros de la petición (mediante getParameter), el tipo de petición (GET, POST, HEAD, etc.), y las cabeceras HTTP entrantes (cookies, Referer, etc.).

- Estrictamente hablando, se permite que la petición sea una subclase de ServletRequest distinta de HttpServletRequest, si el protocolo de la petición es distinto del HTTP. Esto casi nunca se lleva a la práctica.

Objetos implícitos

- **response:** Es el HttpServletResponse asociado con la respuesta al cliente. Como el stream de salida tiene un buffer, es legal seleccionar los códigos de estado y cabeceras de respuesta, aunque no está permitido en los servlets normales una vez que la salida ha sido enviada al cliente.

Objetos implícitos

- **out**: Este es el PrintWriter usado para enviar la salida al cliente. Se puede ajustar el tamaño del buffer, o incluso desactivar el buffer, usando el atributo buffer de la directiva page.

- Se usa casi exclusivamente en scriptlets ya que las expresiones JSP obtienen un lugar en el stream de salida, y por eso raramente se refieren explícitamente a out.

Objetos implícitos

- **session**: Este es el objeto `HttpSession` asociado con la petición. Las sesiones se crean automáticamente, por esto, esta variable se une incluso si no hubiera una sesión de referencia entrante.

- La única excepción es usar el atributo `session` de la directiva `page` para desactivar las sesiones, en cuyo caso los intentos de referenciar la variable `session` causarán un error en el momento de traducir la página JSP a un servlet.

Objetos implícitos

- **application**: es el ServletContext obtenido mediante getServletConfig().getContext().

- **pageContext**: sirve para encapsular características de uso específicas del servidor como JspWriters de alto rendimiento. La idea es que, si se tiene acceso a ellas a través de esta clase, el código seguirá funcionando en motores servlet/JSP "normales".

Objetos implícitos

- **page**: es sólo un sinónimo de this, y no es muy útil en Java. Fue creado como situación para el día en que los lenguajes de script puedan incluir otros lenguajes distintos de Java.

- **config**: es el objeto ServletConfig.

Objetos implícitos

Objeto	Contexto	Descripción	Ámbito
`request`	`HttpServletRequest`	Invocación del servicio	Petición
`response`	`HttpServletResponse`	Respuesta a la petición	Página
`pageContext`	`jsp.PageContext`	Características de la página dependientes de la implementación, espacios de nombres y otras facilidades	Página
`session`	`http.HttpSession`	Conserva el estado de la session	Sesión
`application`	`ServletContext`	Contexto del servletde configuración	Aplicación
`out`	`jsp.JspWriter`	Flujo de salida	Página
`config`	`ServletConfig`	Configuración del servletdel JSP	Página
`page`	`Object`	Página que procesa la petición en curso	Página

Directivas

- **¿Qué son?**
  - Instrucciones dirigidas al contenedor de JSP
  - Describen aspectos del código a generar
  - Forma genérica

    <%@ *nombre-directiva* [*atributo*=“*valor*” *atributo*=“*valor*” ...] **%>**
- **Directivas en la especificación JSP 1.1**
  - page
  - include
  - taglib

Directiva page

- **Directiva de página**
  - Objetivo
    - Especificar atributos para toda la página JSP
  - Sintaxis:

    <%@ page [*atributo*="*valor*" *atributo*="*valor*" ...] **%>**
  - Puede haber más de una directiva *page* en un archivo
  - Ningún atributo puede especificarse más de una vez
    - Excepto el atributo **import**
  - Equivalente XML

```
<jsp:directive.page atributo="valor" …\>
```

Directiva page

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

Directiva include

- **Directiva de inclusión**
  - Objetivo
    - Fusionar en el momento de la traducción, el contenido de otro archivo con el actual
    - Similar a **#include** en el preprocesador de C
  - Sintaxis:

    <%@ include file="url *de archivo*" **%>**
  - Equivalente XML

    ```
    <jsp:directive.include file="url de archivo"\>
    ```

Directiva taglib

- **Directiva de biblioteca de etiquetas**
  - Objetivo
    - Personalización de acciones mediante nuevas etiquetas
  - Sintaxis:

    <%@ taglib uri="*tagLibraryURI*" **prefix**="*tagPrefix*" **%>**
  - Atributos:
    - *tagLibraryURI*: Dirección URL de un Descriptor de biblioteca de etiquetas (TLD)
    - *tagPrefix*: Prefijo único usado para identificar las etiquetas personalizadas

Expresiones

- **Objetivo**
  - Proporcionar un medio para acceder al valor de una variable o expresión Java
  - Enlazar el valor con el HTML de la página

- **Sintaxis**

  <%= *expresión* **%>**

- **Equivalencia en un servlet**

  out.print(*expresión*)

- **Equivalente XML**

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

Scriptlets

- **¿Qué son?**
  - Conjunto de una o más sentencias Java
  - Una página puede tener varios Scriptlets
- **Sintaxis**

  <% *instrucción; [instrucción; ...]* **%>**
- **¿Qué genera? y ¿Cómo se trata?**
  - Se incluye textualmente en el cuerpo del método _jspService() del servlet generado
- **Equivalente XML**

```
<jsp:scriptlet>código</jsp:scriptlet>
```

Declaraciones

- **¿Qué son?**
  - Instrucciones Java que se incorporarán en el servlet fuera del método _jspService()
  - Diferencia principal con los scriplets
    - No tienen acceso a los objetos implícitos
    - A los métodos hay que pasárselos como argumentos
- **Sintaxis**

  <%! *instrucción; [instrucción; ...]* **%>**
- **¿Para qué se usan?**
  - Declaración de clases, variables de instancia, métodos o clases internas
- **Equivalente XML**

  `<jsp:declaration>código</jsp:declaration>`

Acciones

- **¿Qué son?**
  - Elementos de alto nivel que crean, modifican y/o utilizan otros objetos
  - Se codifican usando XML
- **Sintaxis**

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

Acciones

- **Acciones estándar**
  - <jsp:useBean>
  - <jsp:setProperty>
  - <jsp:getProperty>
  - <jsp:include>
  - <jsp:forward>
  - <jsp:param>
  - <jsp:plugin>

Acciones

- **Acción jsp:include**

```
<jsp:include page="URL relativa" flush="true"/>
```

  - Incluye un fichero en el momento en que la página es solicitada.
  - **Nota:** en algunos servidores, el fichero incluido debe ser un fichero HTML o JSP, según determine el servidor (normalmente basado en la extensión del fichero).

Acciones

- **Acción jsp:useBean**

<jsp:useBean atributo=valor ..../>

  - Encuentra o construye un Java Bean.
  - Los posibles atributos son:
    - id="name"
    - scope="page|request|session|application"
    - class="package.class"
    - type="package.class"
    - beanName="package.class"

Acciones

- **Acción jsp:setProperty**

<jsp:setProperty atributo=valor ..../>

  - Selecciona las propiedades del bean, bien directamente o designando el valor que viene desde un parámetro de la petición.

  - Los atributos legales son:
    - name="nombre del bean"
    - property="nombre de la propiedad|*"
    - param="nombre del parámetro"
    - value="valor"

Acciones

- **Acción jsp:getProperty**

```
<jsp:getProperty name="propiedad" value="valor"/>
```

  - Recupera y saca las propiedades del Bean.

- **Acción jsp:forward**

```
<jsp:forward page="URL relativa"/>
```

  - Reenvía la petición a otra página.

Acciones

- **Acción jsp:plugin**

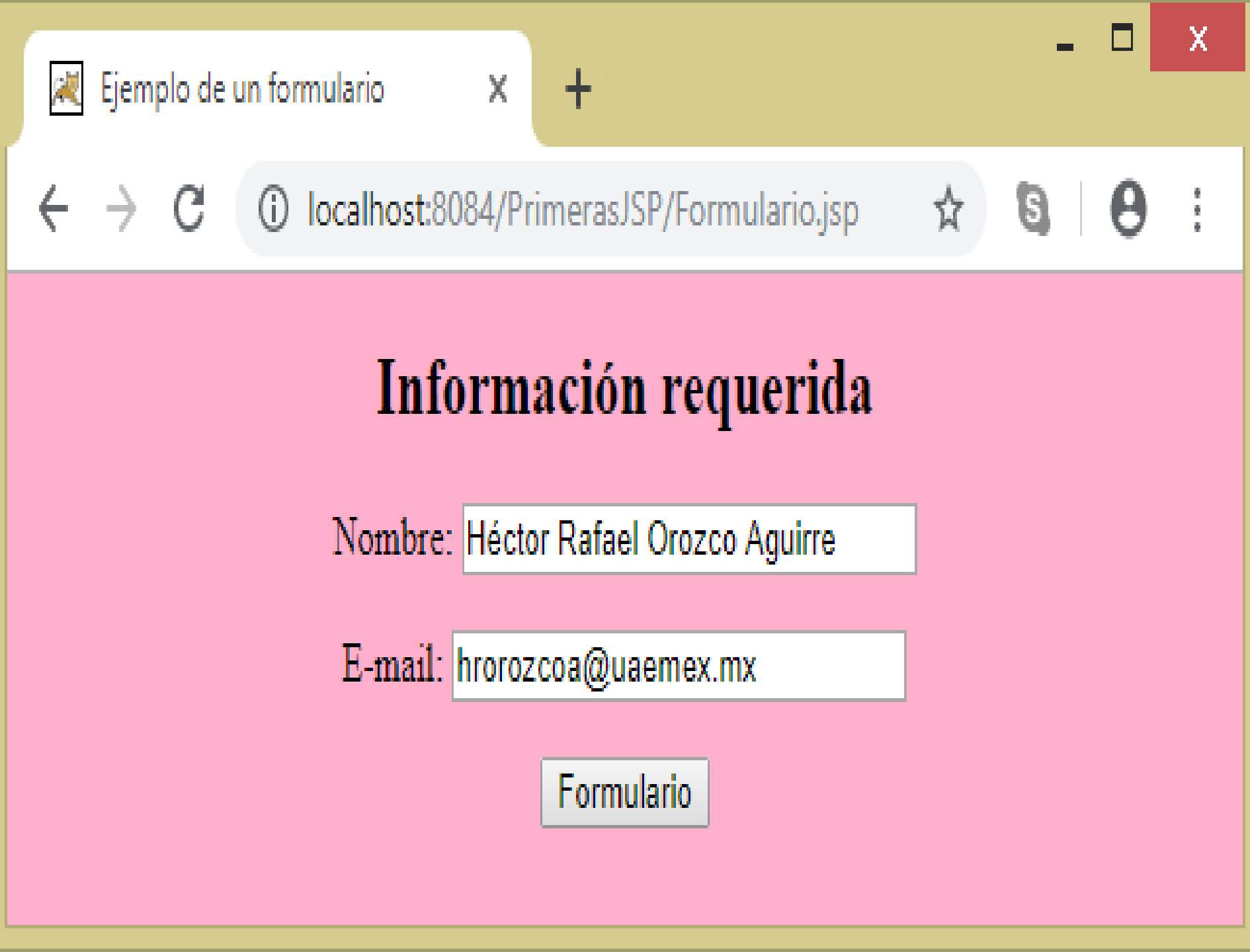
```
<jsp:plugin atributo="valor"*> .... </jsp:plugin>
```

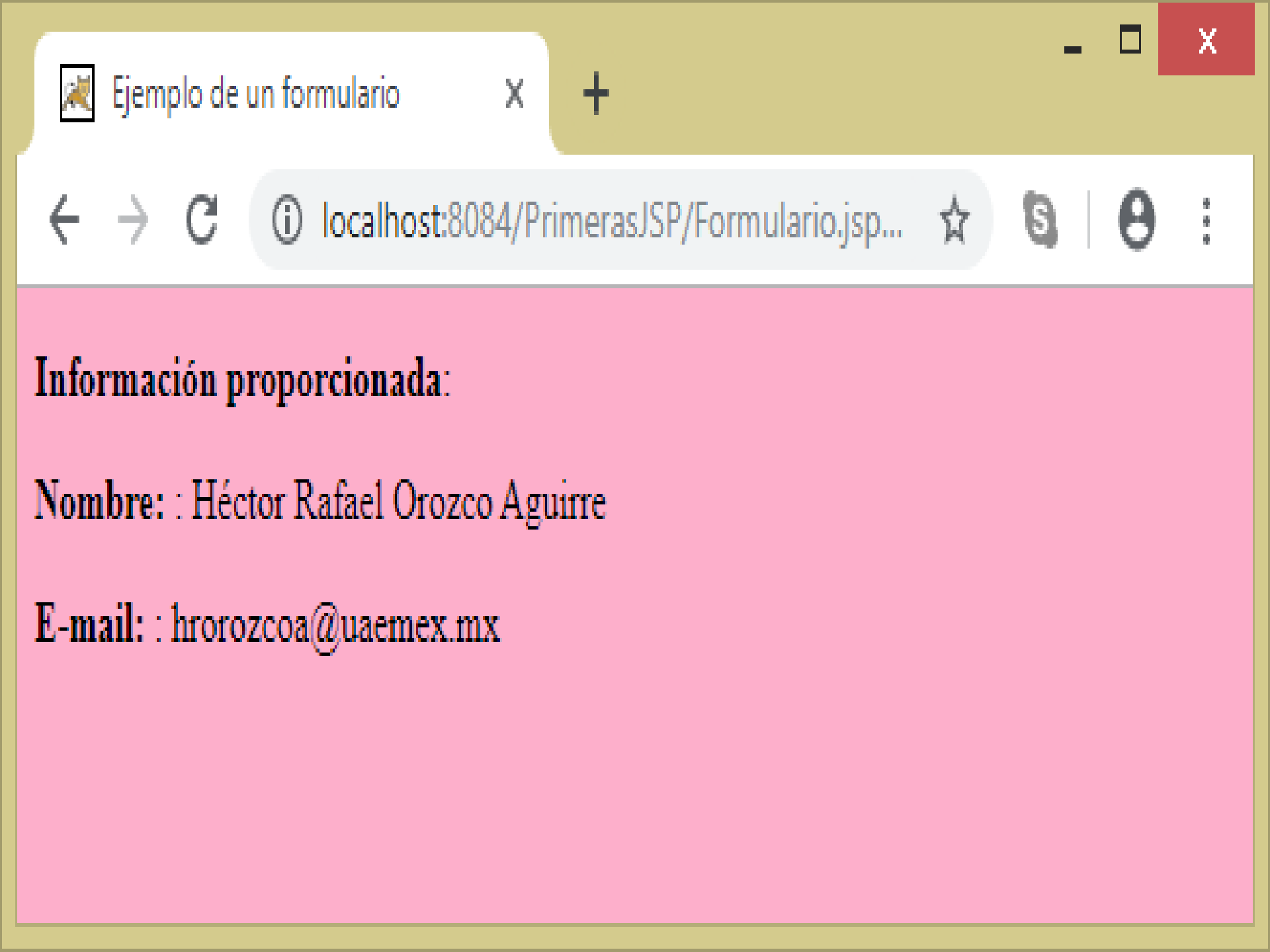
  - Genera etiquetas OBJECT o EMBED, apropiadas al tipo de navegador, pidiendo que se ejecute un applet usando el Java Plugin.

Formulario.jsp

Source | History

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>Ejemplo de un formulario</title>
    </head>
    <body bgcolor="#fdafcb">
      <% if (request.getParameter("nombre")== null
            && request.getParameter("email")  == null) { %>
        <center>
            <h2>Informaci&oacute;n requerida</h2>
            <form method="get" action="Formulario.jsp">
                <p> Nombre: <input type="text" name="nombre" size=26>
                <p> E-mail: <input type="text" name="email" size=26>
                <p> <input type = "submit" name = "enviar" value = "Enviar">
            </form>
        </center>
      <% } else { %>
        <% String nombre, email; %>
        <%  nombre = request.getParameter("nombre");
            email = request.getParameter("email"); %>
      <p>
      <b>Informaci&oacute;n proporcionada</b>:
      <p>
        <b>Nombre: </b>: <%= nombre %><p>
        <b>E-mail: </b>: <%= email %>
      <% } %>
  </body>
</html>
```

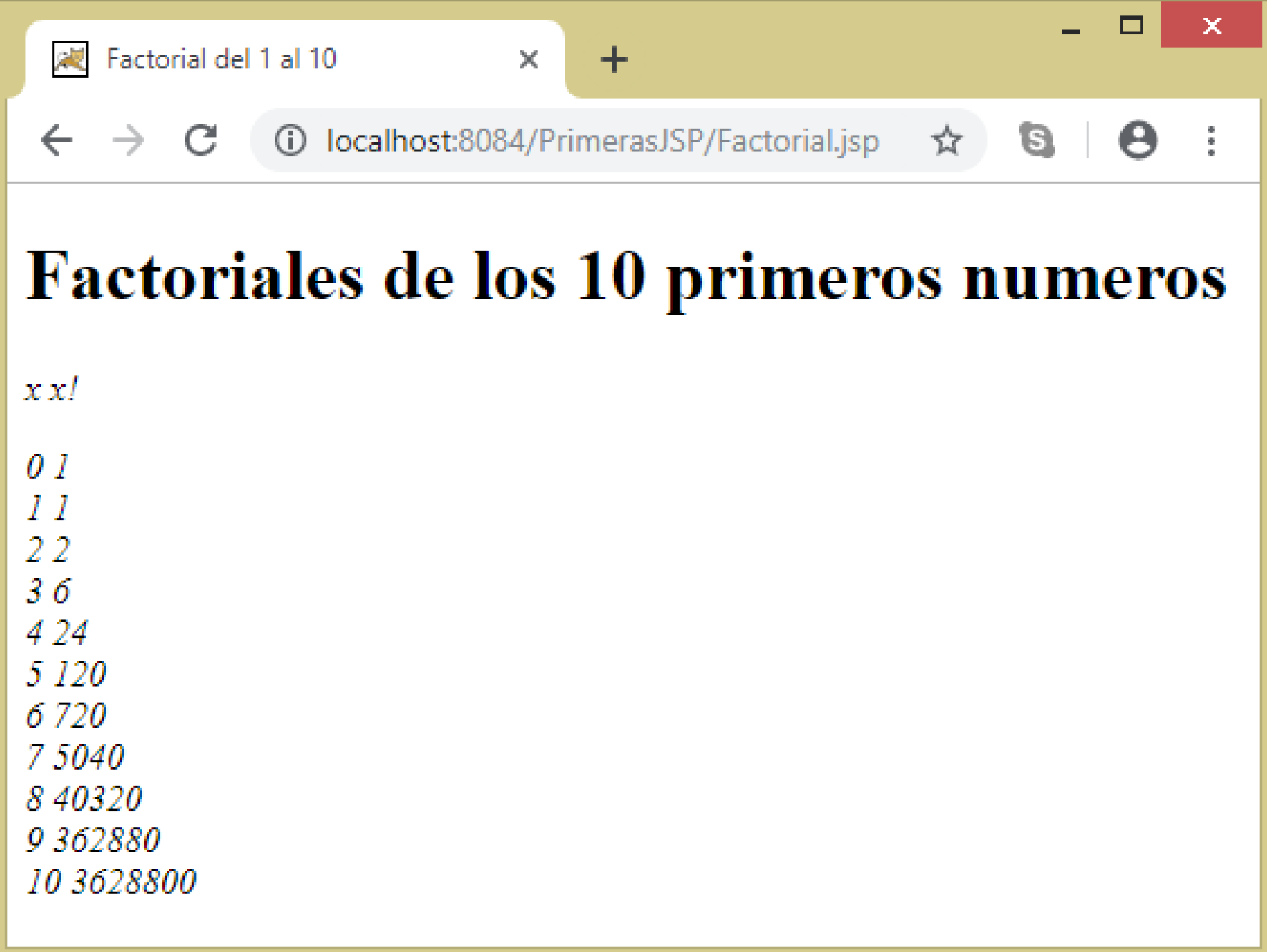

Ejemplo de un formulario
localhost:8084/PrimerasJSP/Formulario.jsp
Información requerida
Nombre: Héctor Rafael Orozco Aguirre
E-mail: hrorozcoa@uaemex.mx
Formulario


Ejemplo de un formulario
localhost:8084/PrimerasJSP/Formulario.jsp...
Información proporcionada:
Nombre: : Héctor Rafael Orozco Aguirre
E-mail: : hrorozcoa@uaemex.mx

Factorial.jsp

Source | History

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>Factorial del 1 al 10</title>
    </head>
    <body>
        <%! public long factorial (long numero)
        {
            if (numero == 0) return 1;
            else return numero * factorial(numero - 1);
        } %>
        <h1>Factoriales de los 10 primeros numeros</h1>
        <p><i>x    x!<i></p>
        <% for (long x = 0; x <=10; ++x) { %>
                <%= x %>    <%= factorial (x) %><br>
        <% } %>
    </body>
</html>
```


Factorial del 1 al 10
localhost:8084/PrimerasJSP/Factorial.jsp
Factoriales de los 10 primeros numeros
x x!
0 1
1 1
2 2
3 6
4 24
5 120
6 720
7 5040
8 40320
9 362880
10 3628800

IDE de desarrollo

- Alternativa a la línea de comandos
- Integración de herramientas:
  - Compilación
  - Depuración (necesitan el SDK)
  - Ant, CVS…
- Modulares: ampliables mediante plugins
- Ayuda a la programación

IDE de desarrollo

- Eclipse:
  - Open Source, impulsado por IBM
  - meta-IDE: sirve para muchas cosas
  - SWT: sustituye a AWT y Swing

- NetBeans:
  - Open Source, desarrollado por Su
  - Generación automática de plantillas
  - Java 100% estándar

IDE de desarrollo

- BlueJ (*http://www.bluej.org/*) - Software libre
- jCreator (*http://www.jcreator.com/*) - Software propietario, versión gratuita disponible.
- Sun Java Studio (*http://wwws.sun.com/software/sundev/jde/*) - Software propietario, extensión de netBeans.
- JBuilder (*http://www.borland.com/jbuilder/*) - Software propietario
- IntelliJ IDEA

Motor de Servlet

- Aplicación que *contiene* la aplicación java
- Necesario para *ejecutar* los servlet y jsp
- Productos
  - Apache Tomcat
  - BEA WebLogic
  - IBM WebSphere
  - Sun/Netscape IPlanet
  - Macromedia JRun
  - …

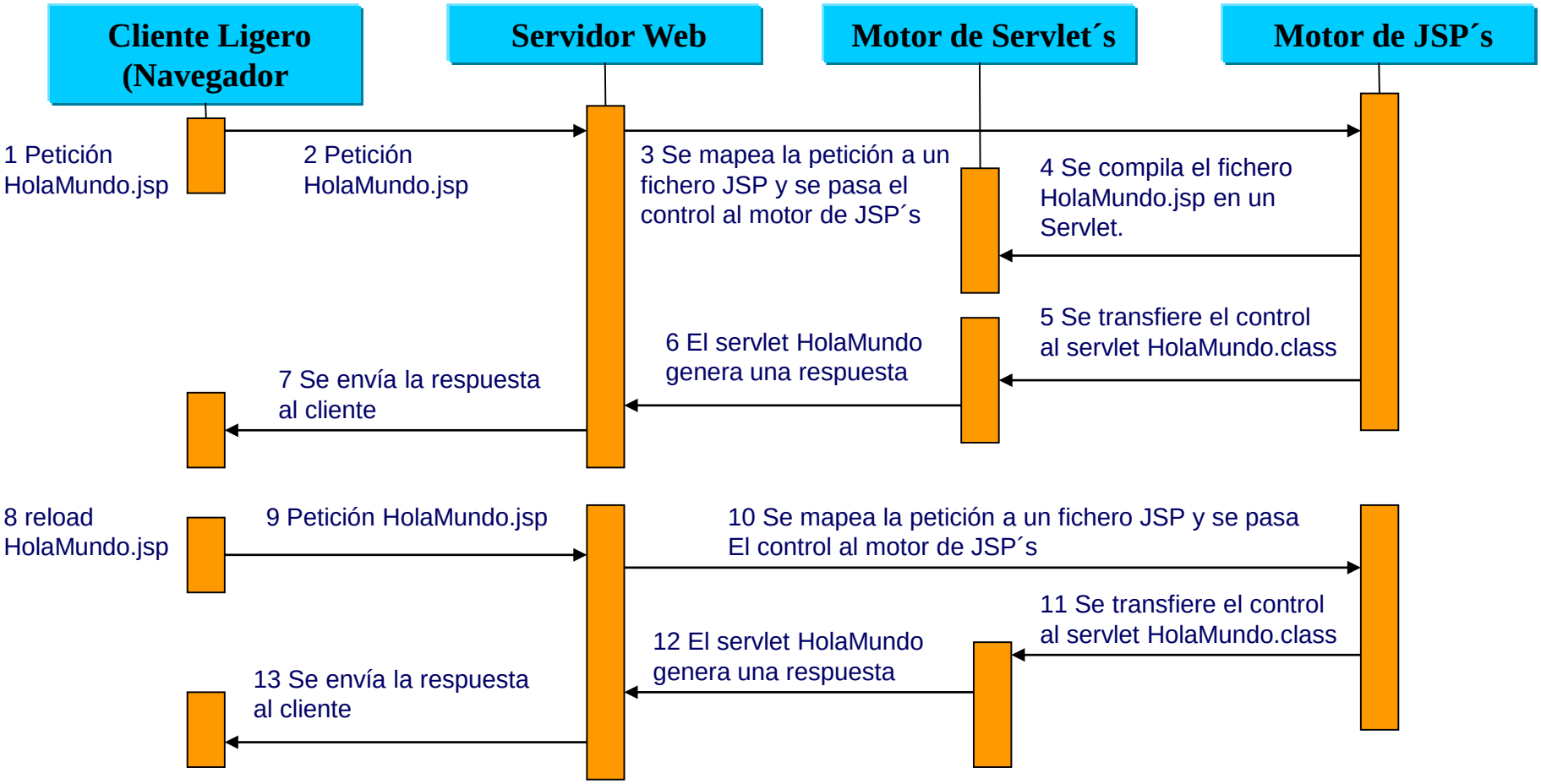
JSP vs Servlet

- Servlets: Java con HTML embebido
- JSPs: HTML con Java embebido

- Semejanzas:
  - JSP son una extensión de Servlets. No aporta funcionalidades nuevas
  - Un JSP compilado es un Servlet
  - Misma función: construir contenido dinámico

- Diferencias:
  - JSP separa más claramente el diseño de la lógica
  - Más sencillo modificar código HTML que miles de sentencias println

JSP vs Servlet



Referencias

- Wolf, D., & Henley, A. J. (2017). *Java EE Web Application Primer: Building Bullhorn: A Messaging App with JSP, Servlets, JavaScript, Bootstrap and Oracle*. Apress.
- Kurniawan, B. (2015). *Servlet & JSP: A Tutorial*. Brainy Software Inc.
- Sierra, F. J. C. (2015). *JAVA. Interfaces gráficas y aplicaciones para Internet* (Vol. 14). Grupo Editorial RA-MA.
- Morales, M. S. (2012). *Manual de Desarrollo Web basado en ejercicios y supuestos prácticos*. Lulu. com.
- Groussard, T. (2010). *Java enterprise edition: desarrollo de aplicaciones web con JEE 6*. Ediciones Eni.