

JAVA WEB

DESARROLLO DE APLICACIONES WEB CON JAVA

MODULO I :Arquitectura de una aplicación WEB



Aplicaciones WEB con JAVA

- Introducción a las aplicaciones WEB con JAVA
- Modelo MVC
- Configuración del entorno de desarrollo

Introducción a las aplicaciones WEB con JAVA

Una aplicación WEB puede definirse como una herramienta de software que se utiliza accediendo a un servidor WEB por medio de Internet o de una intranet a través de un navegador web.



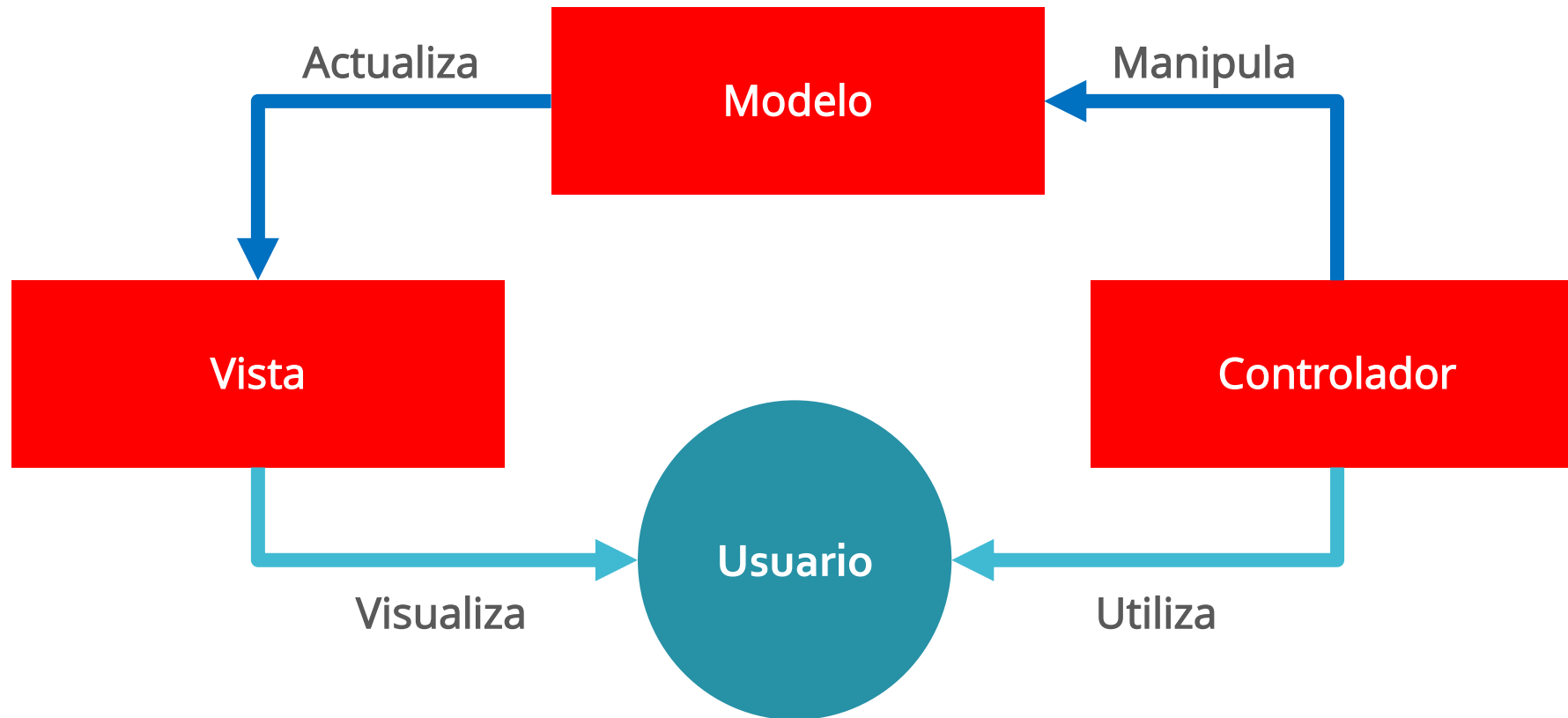
Arquitectura de una aplicación WEB

Podemos separar una aplicación para WEB en tres capas



Arquitectura de aplicaciones Modelo - Vista - Controlador

Es una arquitectura que provee un mecanismo que posibilita separar los datos (el modelo) de la forma en que estos serán visualizados (la vista).



Definición formal del Modelo - Vista - Controlador

Modelo: Representa **las reglas de negocio** de nuestra aplicación. Se encarga de mantener la persistencia de los datos, guardando o recuperando la información independiente de medio utilizado (ficheros XML, bases de datos, etc...).

Vista: Representa los componentes que muestran la interfaz de la aplicación, mostrando la información obtenida a partir del modelo, de manera que el usuario pueda visualizarla. Básicamente las vistas contienen el **código de presentación** que se va a enviar al navegador.

Controlador: Representa los componentes que se encargan de la interacción del usuario, actuando de intermediario entre el usuario, el modelo y la vista. El controlador recoge las **peticiones** del usuario, interacciona con el modelo y finalmente selecciona que vista es la adecuada para mostrar los datos en cuestión.

Ventajas de Modelo - Vista - Controlador

- Escalabilidad: Se puede dividir la lógica de negocio del diseño, haciendo el proyecto más escalable.
- Reutilización de los componentes.
- Facilidad para la realización de pruebas unitarias de los componentes.

Sitios Web dinámicos y estáticos

Sitio estático: Son páginas enfocadas principalmente a mostrar una información permanente, se crean mediante el lenguaje HTML, que NO permite grandes libertades para crear efectos o funcionalidades más allá de los enlaces.

Sitio dinámicos: Se construyen haciendo uso de otros lenguajes de programación, siendo Java el que utilizaremos en este curso. Específicamente *Servlets y JSPs*

Web dinámica con Java

¿Porqué usar Servlets y JSPs?

Las aplicaciones tradicionales desarrolladas en Java utilizando prácticos y vistosos componentes Swing (Widgets) son excelentes para cierto alcance de nuestra aplicación.

¿Qué pasaría si nuestra aplicación necesita llegar a un considerable numero de usuarios repartidos en diversas partes del planeta?

El objetivo sería desarrollar un sitio web capaz de alojar el contenido y funcionalidad que nuestros usuarios requieran.

Y... ¿Dónde se ejecuta mi aplicación Web?

Las aplicaciones que nosotros construiremos se ejecutarán sobre la World Wide Web o simplemente Web.

La web es una plataforma que nos permite acceder a cierta información a través de internet. Esta plataforma está construida sobre **internet**.

La **Web** utiliza un *protocolo* de comunicación para establecer enlaces entre dispositivos. **WWW**

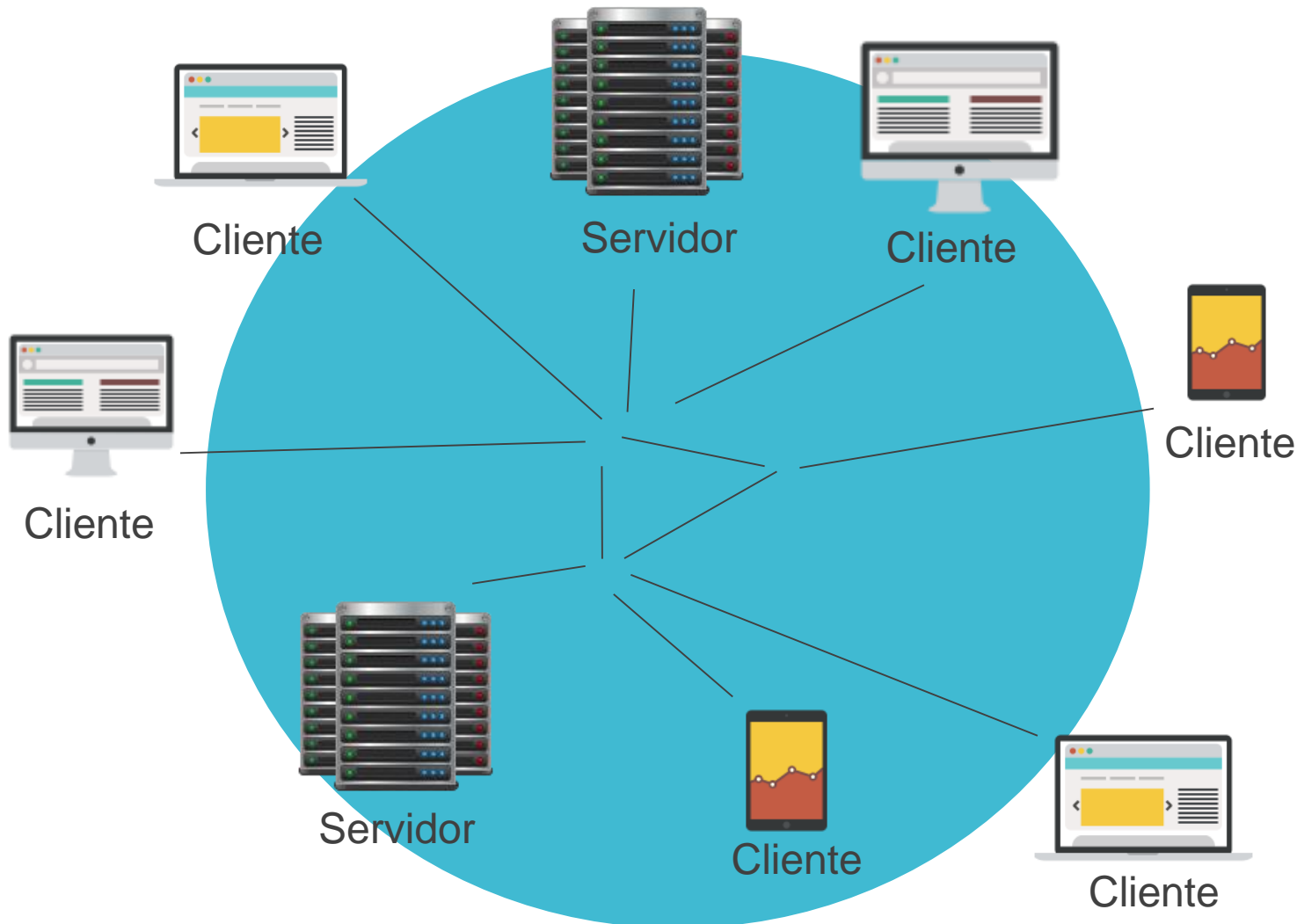
A este protocolo se le conoce como **HTTP** (Hypertext Transfer Protocol o Protocolo de transferencia de hipertexto). 

¿Entonces, qué es Internet?

Internet es una red masiva, es la red de redes. Internet conecta millones de ordenadores en todo el mundo a través de una red que permite que cualquier computador pueda comunicarse con otro sin importar en qué lugar del planeta se encuentren, siempre y cuando los dos estén conectados a Internet.



¿Cómo trabaja mi aplicación Web?



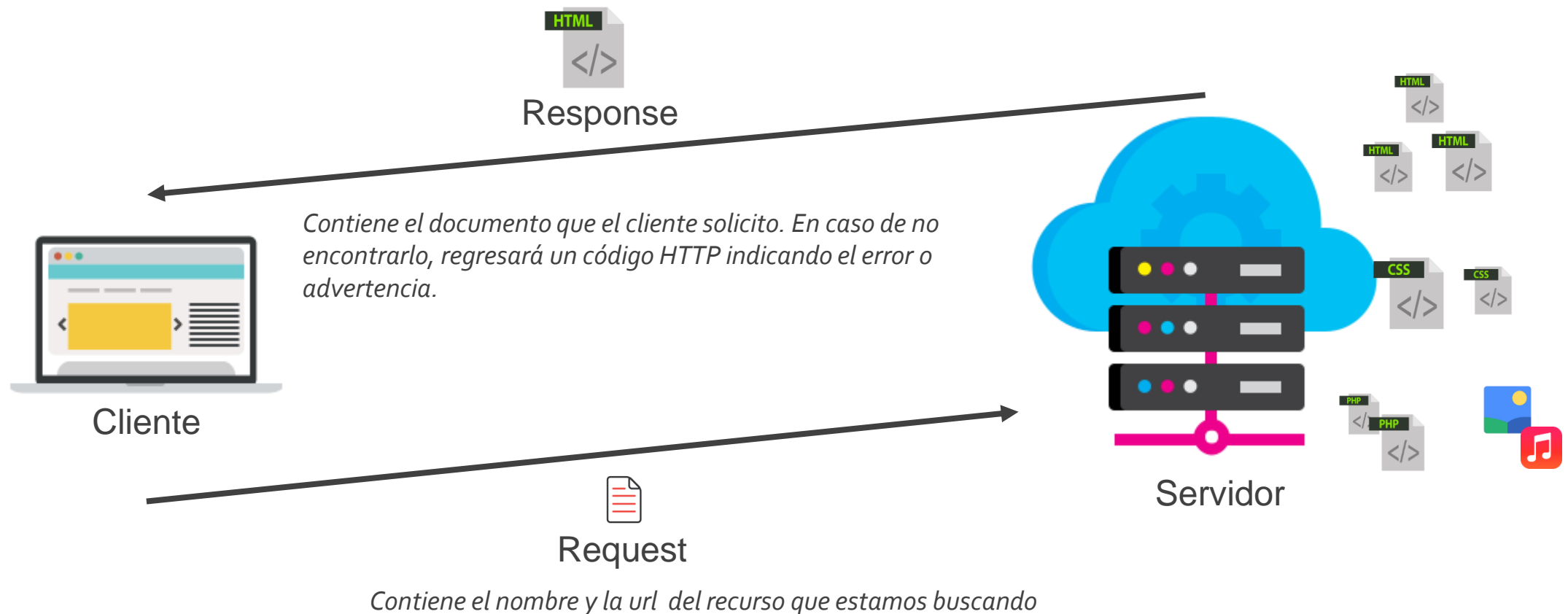
La web consiste en cientos de millones de dispositivos interconectados entre sí.

Estos dispositivos pueden ser: **Clientes y Servidores** conectados mediante cables o conexiones inalámbricas

Nuestro objetivo es construir una aplicación web que sea accesible por clientes distribuidos alrededor del mundo.

¿Qué función tiene un servidor de aplicaciones?

Un servidor de aplicaciones toma una **petición** (Request) de un cliente y regresa una **respuesta** (Response) al cliente.

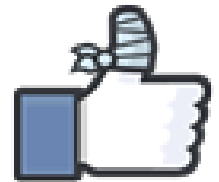


¿Qué función tiene un servidor de aplicaciones?

Un navegador web le permite al usuario realizar una petición de un recurso al servidor.

El servidor web recibe la petición, encuentra el recurso solicitado y regresa algo al usuario. El recurso solicitado puede ser: Una pagina HTML, una imagen, audio, video, texto, un PDF, etc. Realmente no importa el tipo de archivo que el cliente solicite, el servidor realizará una búsqueda y retornará algo...

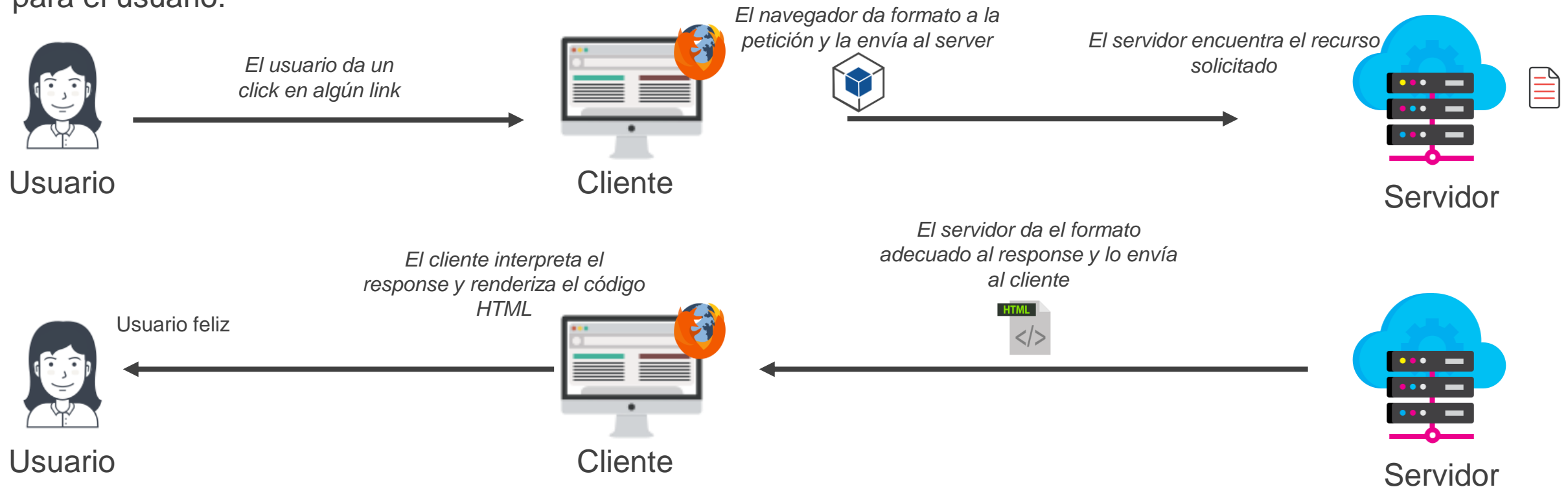
Sin embargo, en ocasiones, el recurso solicitado puede que no se encuentre en la ruta especificada o no exista en el servidor, entonces nos arrojará un código HTTP muy común, el código 404 (Not found).



¿Qué es un cliente y cómo funciona?

Un cliente Web le permite al usuario realizar peticiones por recursos al servidor Web y le muestra al usuario el resultado de la petición.

El navegador web es la pieza de software que sabe como comunicarse con el servidor.
Otro gran trabajo que hace el navegador web para el usuario es el de interpretar el código HTML y renderizarlo para el usuario.



HTTP

HTTP es un protocolo de red que depende de TCP / IP para poder emitir una petición y una respuesta completa de un punto De una red a otro.

HTTP se ejecuta en la capa mas alta de TCP / IP .

TCP -> Es responsable de asegurarse que un archivo que proviene de un nodo de una red llegue completo a otro nodo de Destino. En el proceso, el archivo puede partirse en tramas cuando es enviado.

IP -> Es otro protocolo que se encarga de mover o direccionar estas tramas (paquetes) provenientes de un host a su destino.

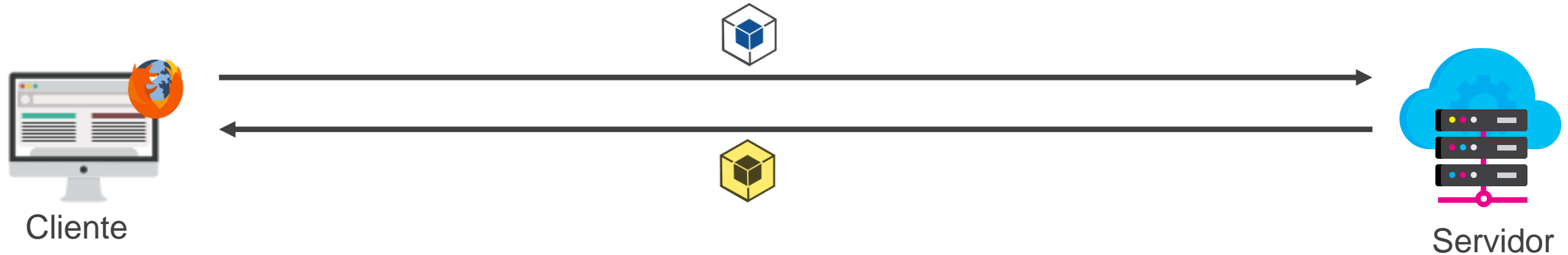
| METHOD | SERVLET METHOD | PURPOSE |
|---------|----------------|--|
| GET | doGet () | Retrieves the resource at the specified URL |
| HEAD | doHead () | Identical to GET, except only the headers are returned |
| POST | doPost () | Typically used for web form submission |
| PUT | doPut () | Stores the supplied entity at the URL |
| DELETE | doDelete () | Deletes the resource identified by the URL |
| OPTIONS | doOptions () | Returns which HTTP methods are allowed |
| TRACE | doTrace () | Used for diagnostic purposes |

HTTP Peticiones

Las peticiones HTTP se caracterizan por:

Request

- El cuerpo de un **request** se caracteriza por:
- Método HTTP (La acción que implementar)
 - La dirección del recurso solicitado (URL)
 - Parámetros



El cuerpo de un **response** se caracteriza por:

Response

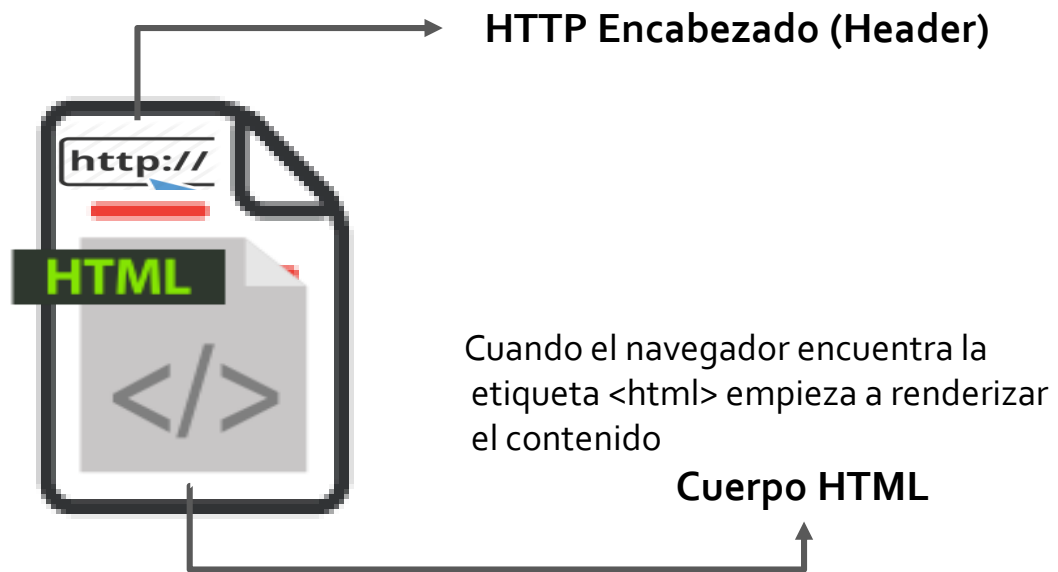
- Un código de status
- Tipo de contenido (texto, imágenes, HTML, etc.)
- El contenido (La pagina HTML actual, imágenes, etc)

Response

Una respuesta HTTP puede contener código HTML.

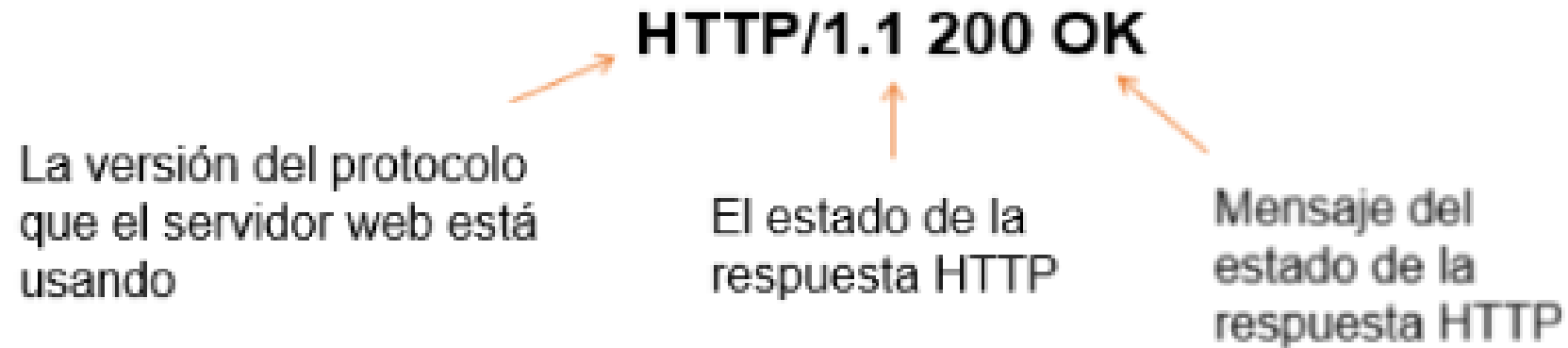
HTTP agrega información en forma de encabezado en todo contenido que devuelve.

Un navegador web utiliza el contenido del encabezado como información auxiliar en el renderizado del código HTML.



Anatomía de un response

la información de la cabecera indica al navegador sobre el protocolo que se utiliza. El cuerpo contiene el contenido.



Cabeceras de la respuesta.

| | |
|----------------------------------|---|
| Set-Cookie | JSESSIONID=0AAB6C8DE415E2E5F307CF334BFCAC1;Path=/testEL |
| Content-Type | text/html |
| Content-Length | 397 |
| Date | Wed, 19 Nov 2003 03:25:40 GMT |
| Server | Apache-Coyote/1.1 |
| Connection | Close |
| <html></html> | El contenido es renderizado |

Request

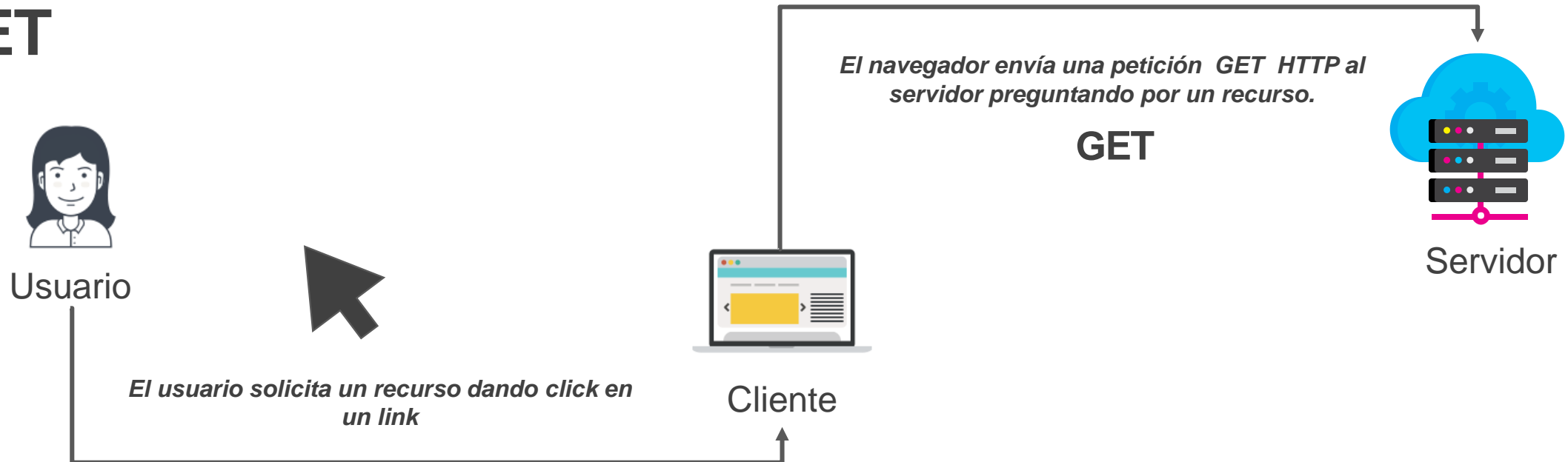
Lo primero que vamos a encontrar en un **request** (petición) es un **método** de envío:

Estos métodos pueden ser: **GET** o **POST**

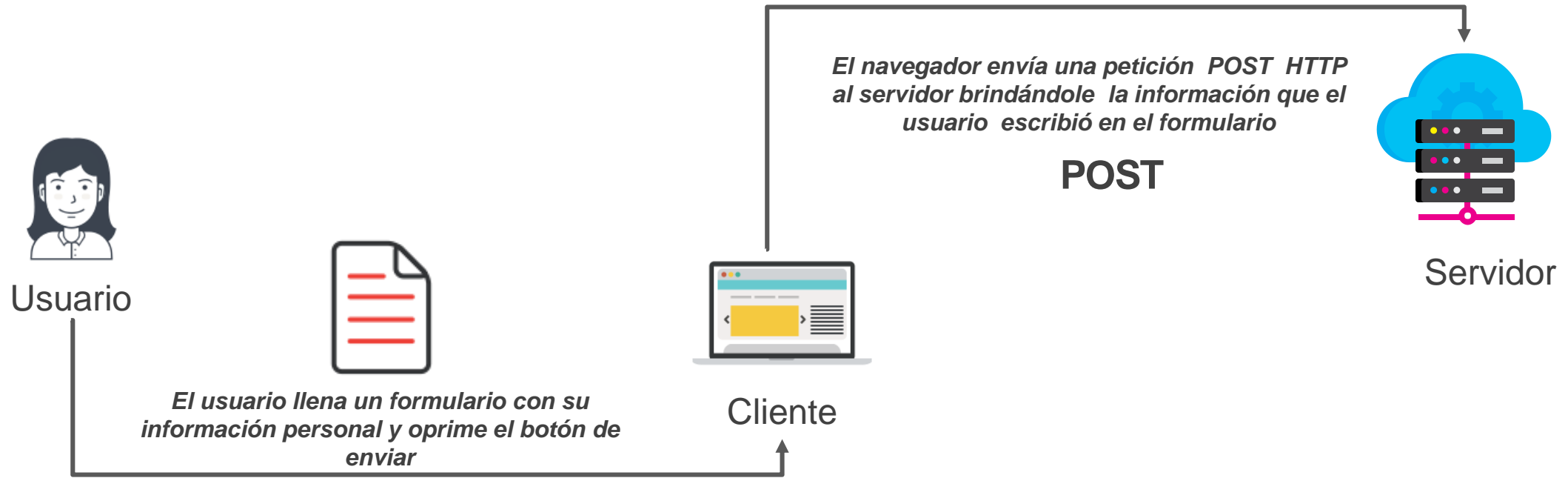
El nombre del método le indica al servidor que tipo de petición se va a realizar y el formato específico que va a contener el resto del mensaje.

El protocolo HTTP tiene diversos métodos pero los mas comunes son : **GET** y **POST**.

GET

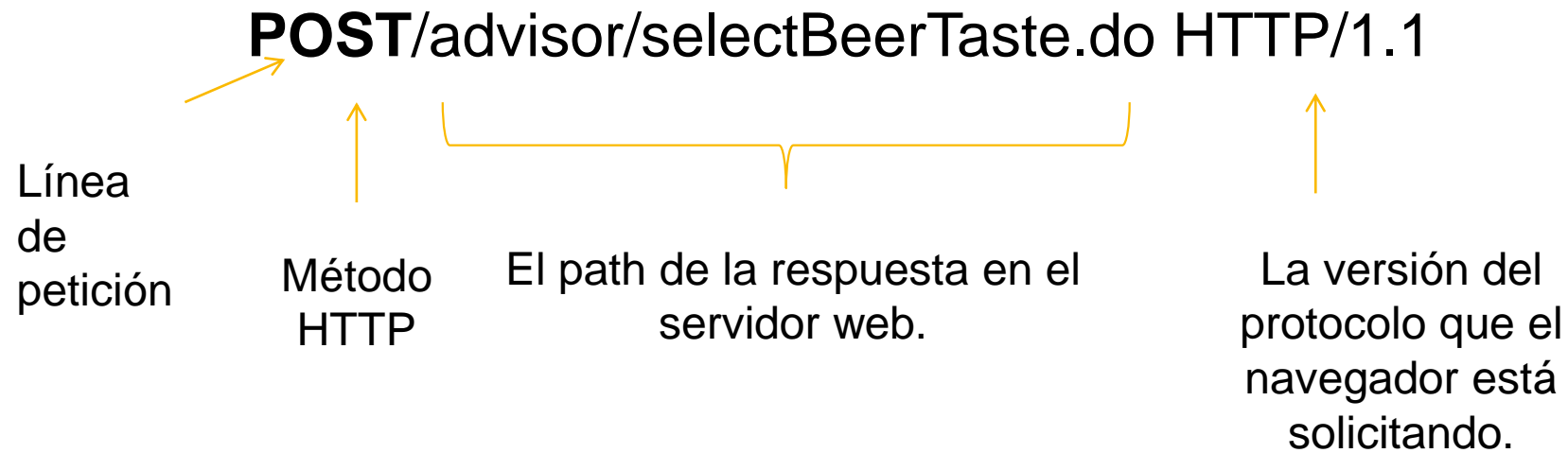


POST



Anatomía de un request

La solicitud Http Post está diseñada para ser utilizada por el navegador para hacer una petición compleja en el servidor.



Cabeceras de la petición.

| | |
|------------------------|--|
| Host | www.wikedlysmart.com |
| User-Agent | Mozilla 5.0 (Macintosh;U;PPC MacOS X Mach-O;en-US;rv:1.4) Gecko/20030624 Netscape 7.1 |
| Accept | text/xml.application/xml.application/xhtml+xml;text/html;q=0.9;text/plain;q=0.8.video/x-mng.image/png.image/jpeg.image/gif;q=0.2,*/*;q=0.1 |
| Accept-Language | en-us,en;q=0.5 |
| Accept-Encoding | gzip.deflate |
| Accept-charset | ISO-8859-1,utf-8;q=0.7,*;q=0.7 |
| Keep-Alive | 300 |
| Connection | Keep-alive |
| color=dark&taste=malty | Cuerpo del mensaje |

HTML y lenguajes de marcado

| HTML etiquetas principales | |
|----------------------------|---------------------------|
| <!-- --> | Línea de comentarios |
| <a> | Ancla |
| <body> | Define el cuerpo del HTML |
| | Salto de línea |
| <center> | Centrar |
| <form> | Define un formulario |
| <h1> | Encabezado principal |
| <head> | Cabecera del documento |
| <html> | Define el documento html |
| <input> | Elemento de un formulario |
| <p> | Pàrrafo |
| <title> | Título de la página |

Introducción a los Servlets y JSPs

Servlets

Los Servlets de Java son la propuesta de la tecnología Java para el desarrollo de aplicaciones web.

Un **Servlet** es un programa que se ejecuta en un servidor web y que construye una página web que es devuelta al usuario.

Esta página, construida dinámicamente, puede contener información procedente de bases de datos, ser una respuesta a los datos introducidos por el usuario, etc.

La especificación Java EE API define a un **Servlet** como lo siguiente:

Un **Servlet** es un pequeño programa en Java que corre en un servidor Web. Los Servlets reciben y responden a las peticiones provenientes de clientes Web, usualmente a través del protocolo HTTP.

<http://docs.oracle.com/javaee/7/api/javax/servlet/Servlet.html>

Características de los Servlets

Eficiencia

Con el modelo de Servlets, la máquina virtual de Java, el entorno donde se ejecutan, se arranca al iniciar el servidor, permaneciendo arrancada durante toda la ejecución del mismo. Para atender cada petición no se arranca un nuevo proceso, sino un thread, un proceso ligero de Java, mucho más rápido (de hecho, casi instantáneo).

Facilidad de uso

El estándar de Servlets nos ofrece una magnífica infraestructura de desarrollo de aplicaciones web, proporcionándonos métodos para análisis automático y decodificación de los datos de los formularios de HTML, acceso a las cabeceras de las peticiones HTTP, manejo de cookies, seguimiento, control y gestión de sesiones, entre otras muchas facilidades.

Potencia

Los Servlets Java permiten hacer muchas cosas que son difíciles o imposibles de realizar con los CGI tradicionales. Los Servlets pueden compartir los datos entre sí, permitiendo compartir datos, conexiones a bases de datos, etc. Asimismo, pueden mantener información de solicitud en solicitud, facilitando tareas como el seguimiento de las sesiones de usuario, etc.

Portabilidad

Los Servlets están escritos en Java y se rigen por un API estándar bien documentado. Como consecuencia de ello, los Servlets pueden ejecutarse en todas las plataformas que nos ofrezcan soporte de Java Servlets, sin tener que recompilar, modificarse, etc., sean estas plataformas Apache, iPlanet, IIS, etc., y además, con independencia de sistema operativo, arquitectura hardware, etc.

Características de los Servlets

```
1  import java.io.*;
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4
5  public class Holaweb extends HttpServlet
6  {
7      public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
8      {
9          response.setContentType("text/html");
10         PrintWriter out = response.getWriter();
11         out.println(
12             "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 \" +
13             \"Transitional//EN\">\n\" +
14             "<HTML>\n\" +
15             "<HEAD><TITLE>Hola</TITLE></HEAD>\n\" +
16             "<BODY>\n\" +
17             "<H1>Hola web</H1>\n\" +
18             "</BODY></HTML>\"
19         );
20     }
21 }
```

JSPs

Las Java Server Pages (JSP) son una tecnología que nos permite mezclar HTML estático con HTML generado dinámicamente mediante código Java embebido en las páginas.

Tanto los **CGI** como los **Servlet** nos obligan a generar la página por completo desde nuestro código de programa, dificultando así las tareas de mantenimiento, diseño gráfico, comprensión del código, etc.

Los JSP, por otro lado, nos permiten crear las páginas fácilmente.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1>Hello World!</h1>
    <br/>
    La fecha es: <%= new java.util.Date() %>
  </body>
</html>
```

SERVLETS

El código HTML en
Java.

No facil para el autor

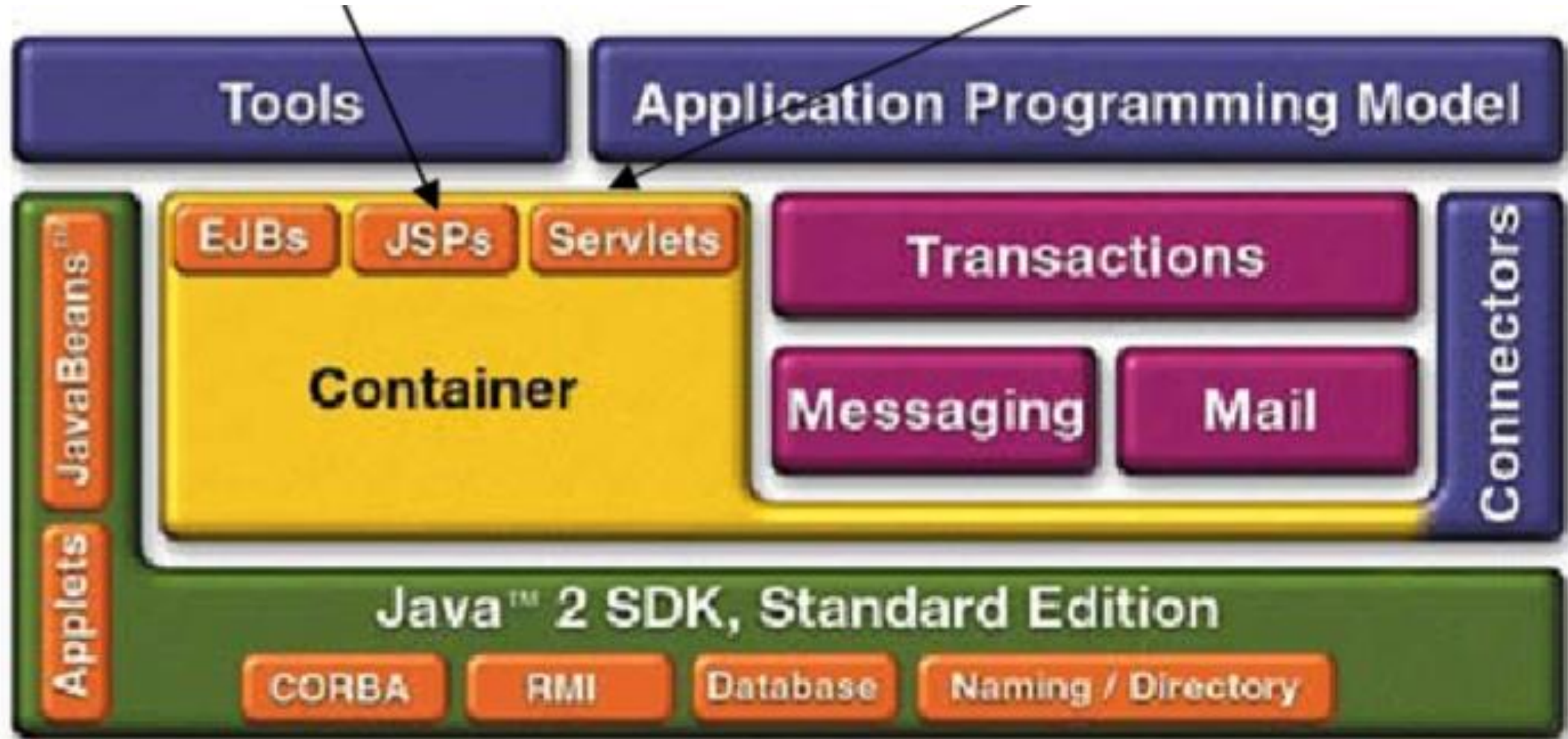
JSP

Código Java en HTML.

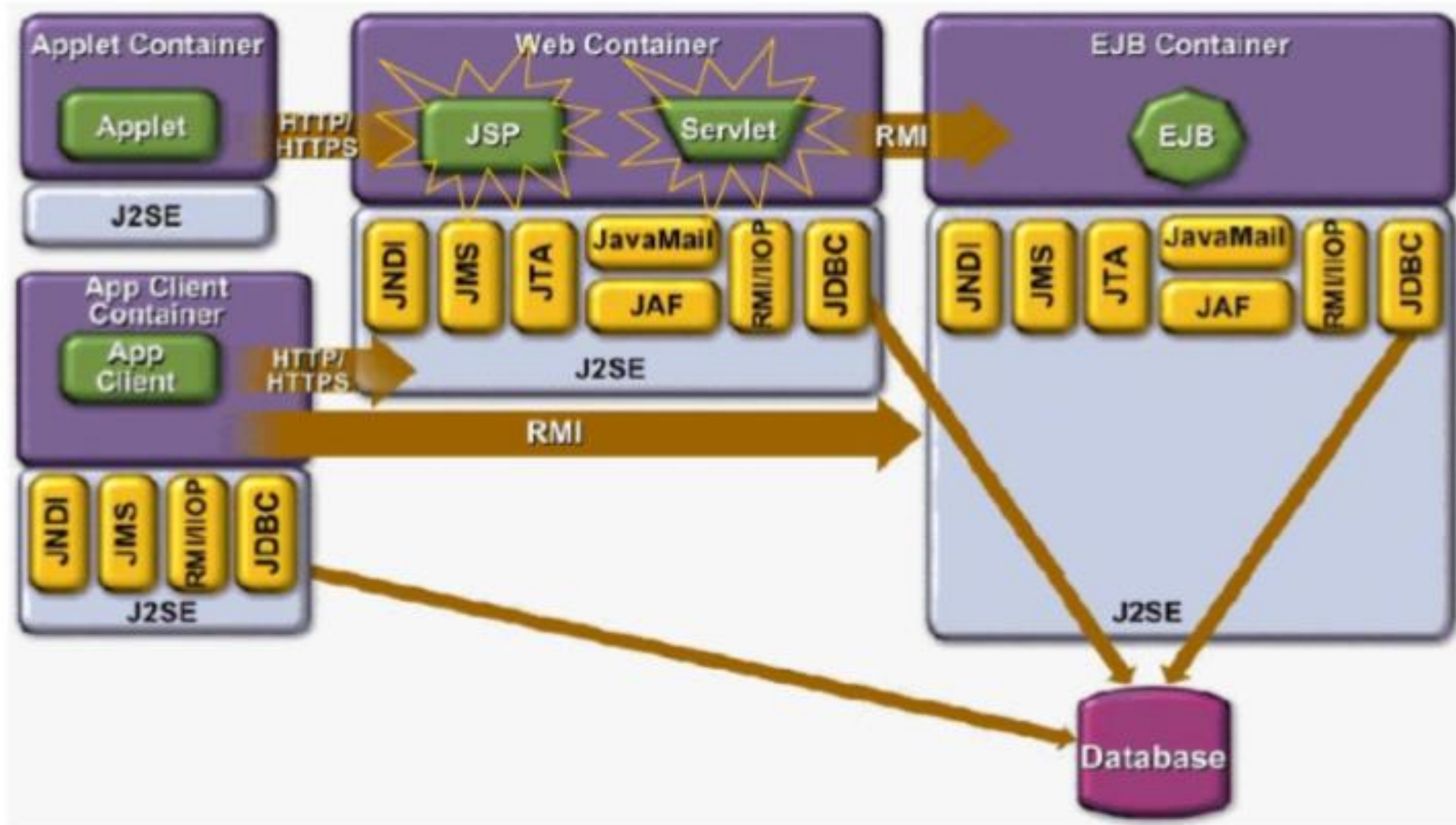
Facil para el autor.

El código es compilado
a un Servlet

JSPs y Servlets en la arquitectura de Java EE



JSPs y Servlets como componentes WEB



Configuración del entorno de trabajo

Requisitos:

1. Servidor HTTP-> Apache Tomcat.

Funciona como un **contenedor** de **servlets** desarrollado bajo el proyecto Jakarta en la Apache Software Foundation.

Es la implementación de referencia de las especificaciones de servlets 2.5 y de Java Server Pages (JSP) 2.1, especificaciones para Java Community Process, usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

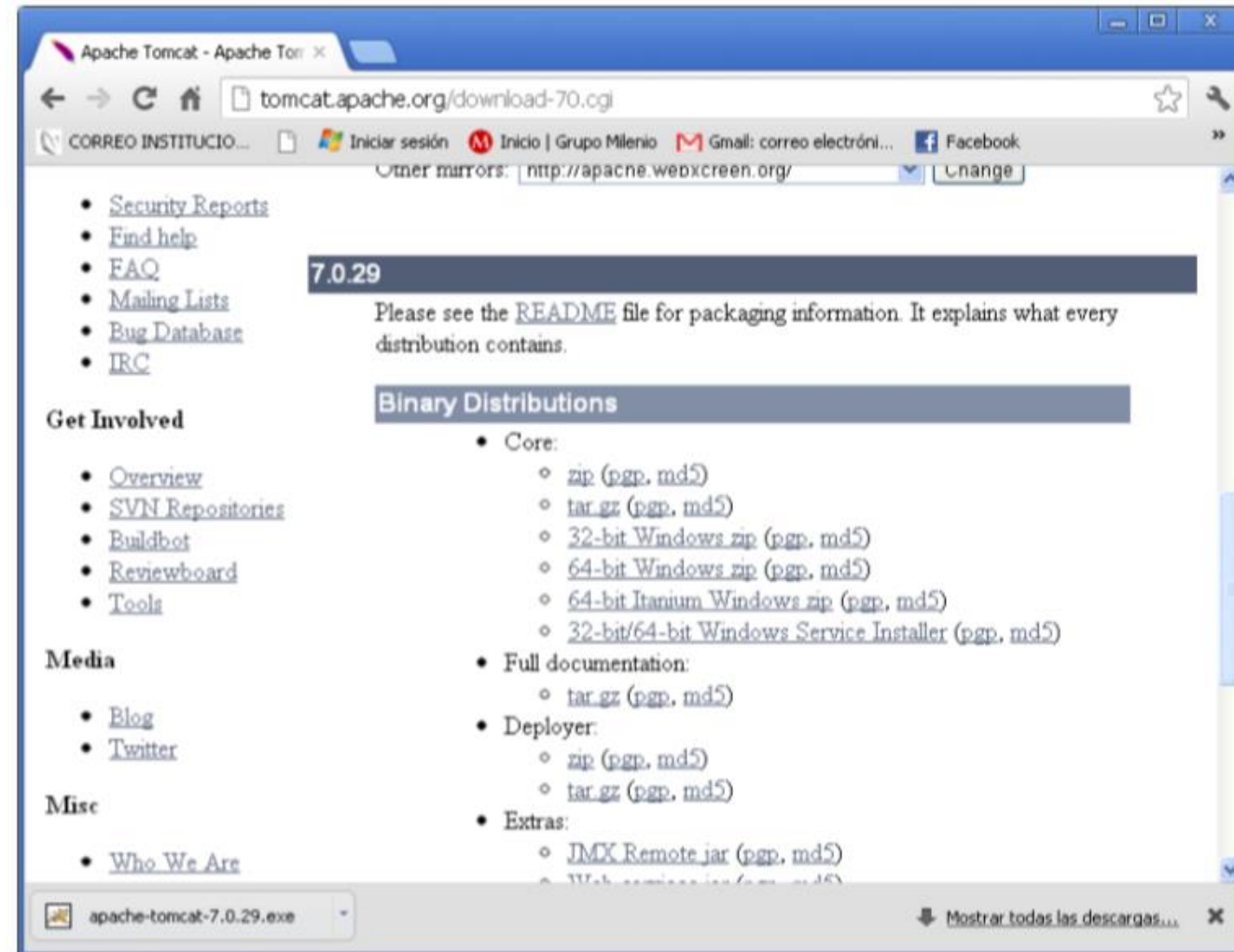
Dado que Apache Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

Configuración del entorno de trabajo

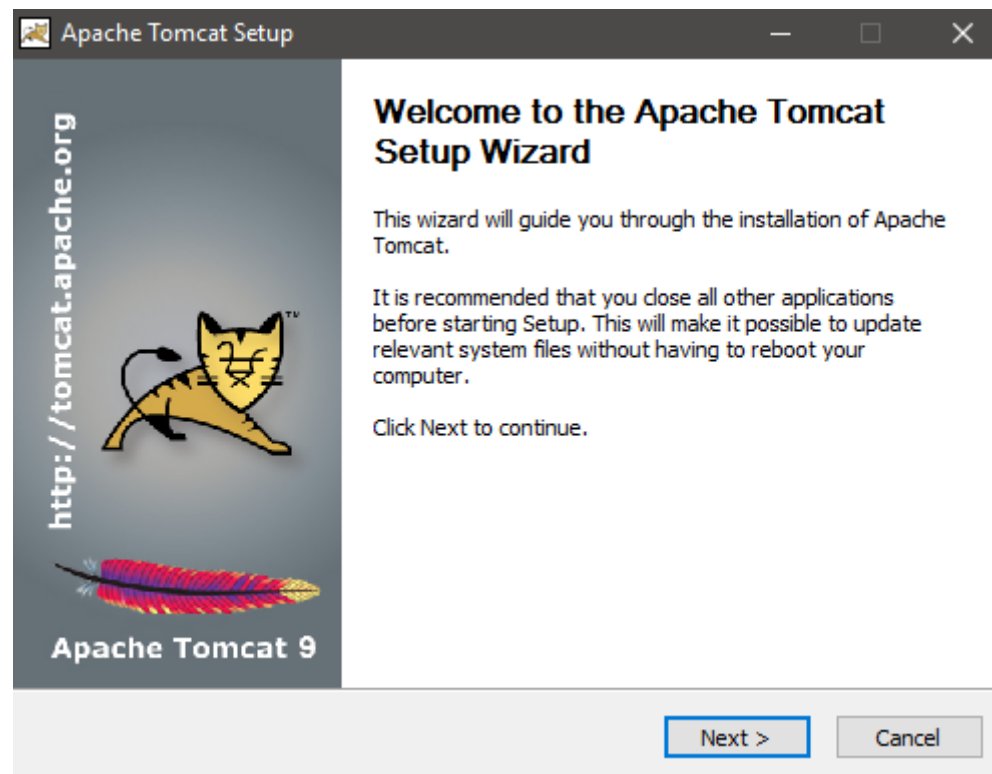
1. Vaya a la dirección: <http://tomcat.apache.org/>

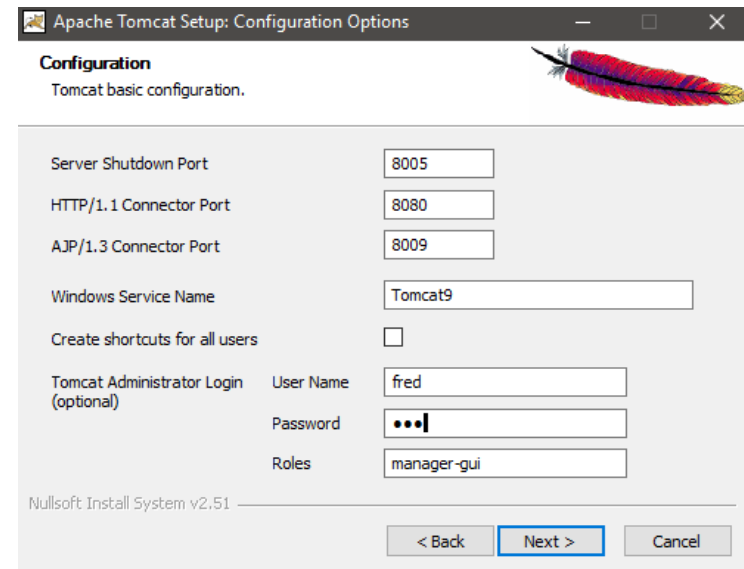
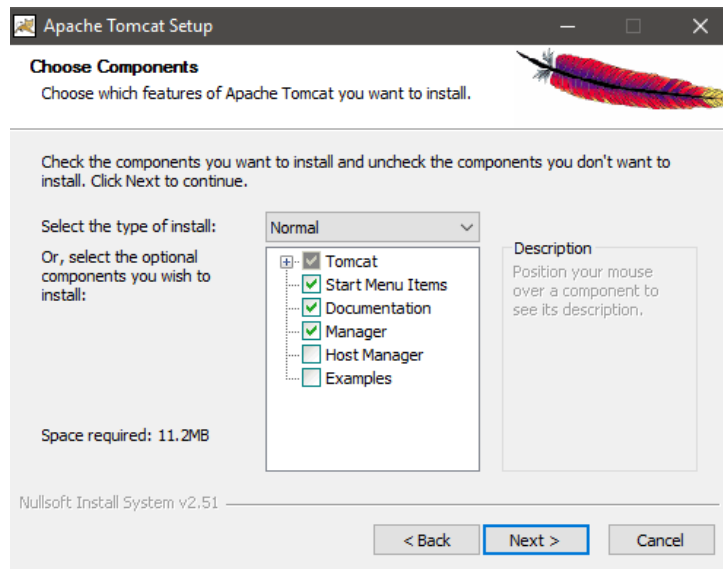
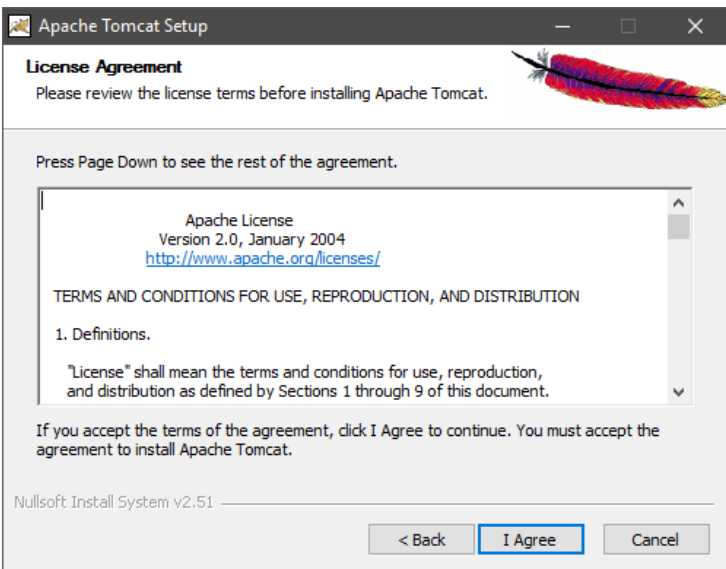


2. Elija descargar la ultima versión



3. Seleccione la distribución adecuada para el sistema operativo de su computadora.
Si es Windows descargue el programa instalador:
<http://apache.webxscreen.org/tomcat/tomcat-7/v7.0.42/bin/apache-tomcat7.0.42.exe>
4. Ejecute el programa y siga las instrucciones.
(En su momento le pedirá el nombre y contraseña de administrador, apúntelos)



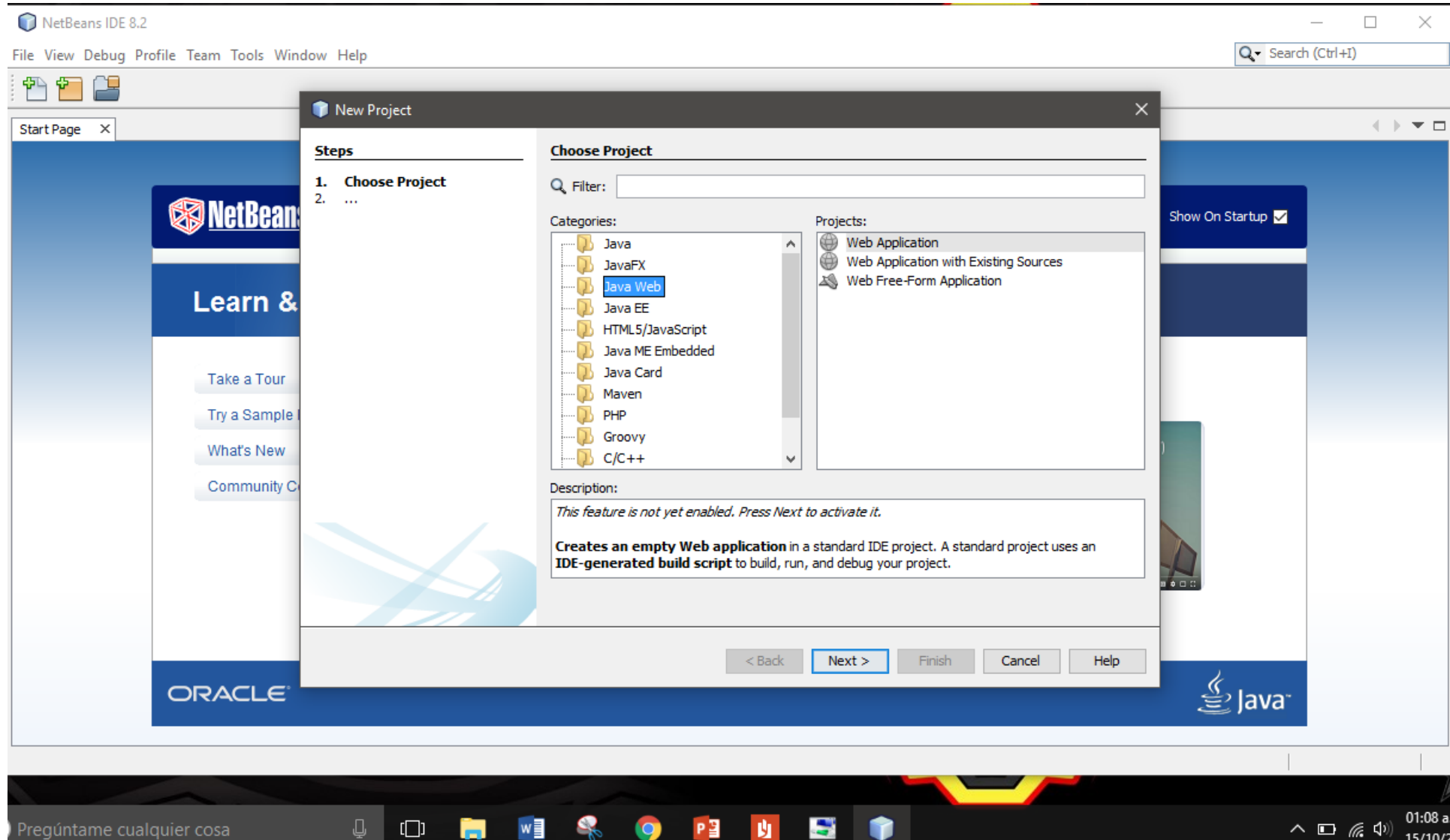


5. Pruebe su servidor: <http://localhost:8080>



6. Si puede ver la página de inicio... disfrútelo, lo ha logrado!

Configuración del primer proyecto



Start Page



Learn &

[Take a Tour](#)[Try a Sample](#)[What's New](#)[Community](#)

ORACLE



New Web Application

Steps

1. Choose Project
2. **Name and Location**
3. Server and Settings
4. Frameworks

Name and Location

Project Name:

Servlet1

Project Location:

C:\Users\fredd\Documents\NetBeansProjects

Browse...

Project Folder:

C:\Users\fredd\Documents\NetBeansProjects\Servlet1

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Browse...

Different users and projects can share the same compilation libraries
(see Help for details).

< Back

Next >

Finish

Cancel

Help

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

Add to Enterprise Application: <None>

Server:

Apache Tomcat or TomEE

Add...

Java EE Version: Java EE 7 Web

Context Path: /Servlet1

< Back

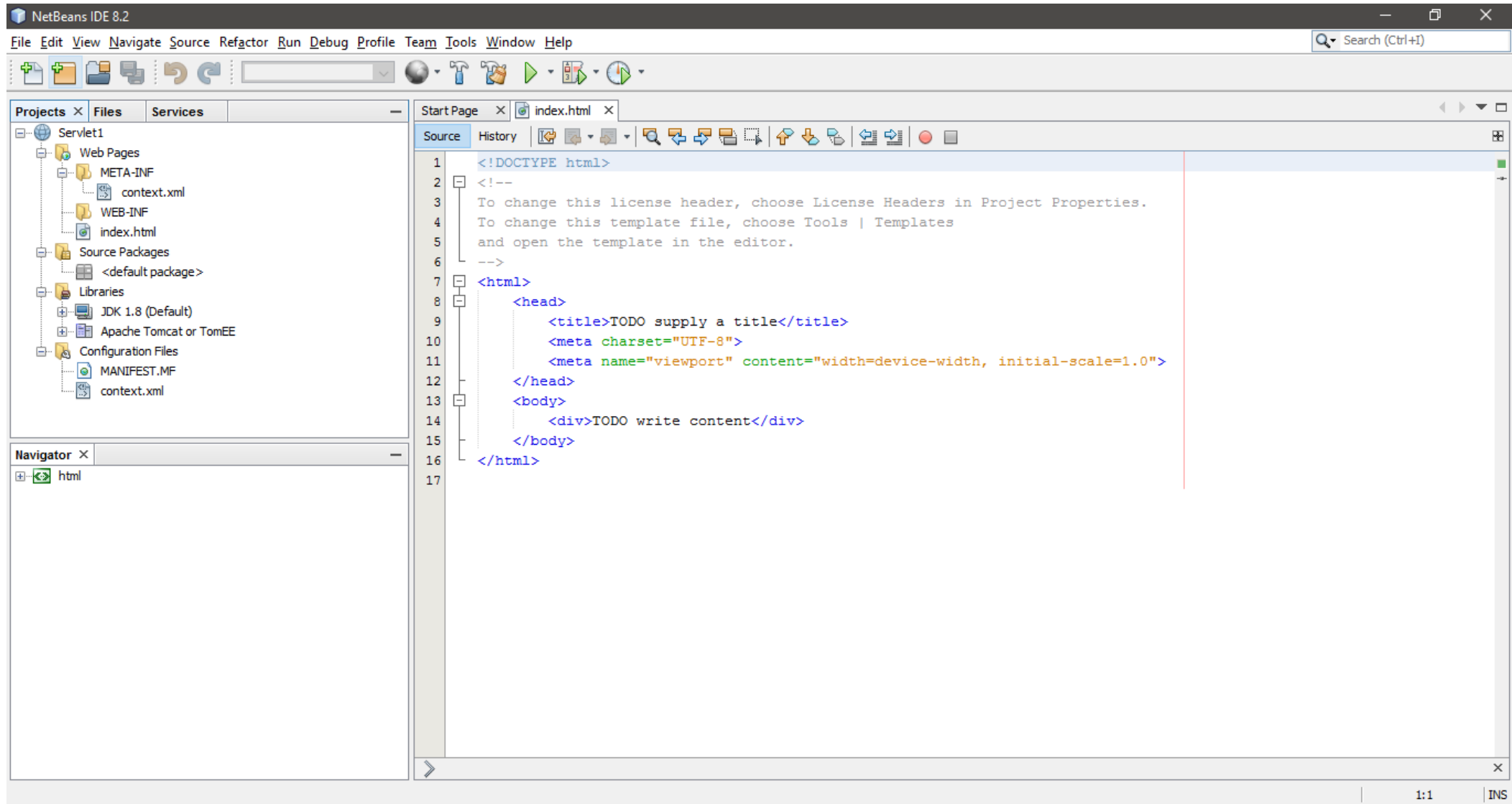
Next >

Finish

Cancel

Help

Configuración del primer proyecto



Trabajando con Servlets

Los Servlets y su herencia.

Los Servlets necesitan heredar ciertas propiedades para poder realizar comunicación vía HTTP y para ello utilizan la clase **GenericServlet**.

Por lo tanto, el paquete necesario de importación es el siguiente: **javax.servlet.http.HttpServlet**.

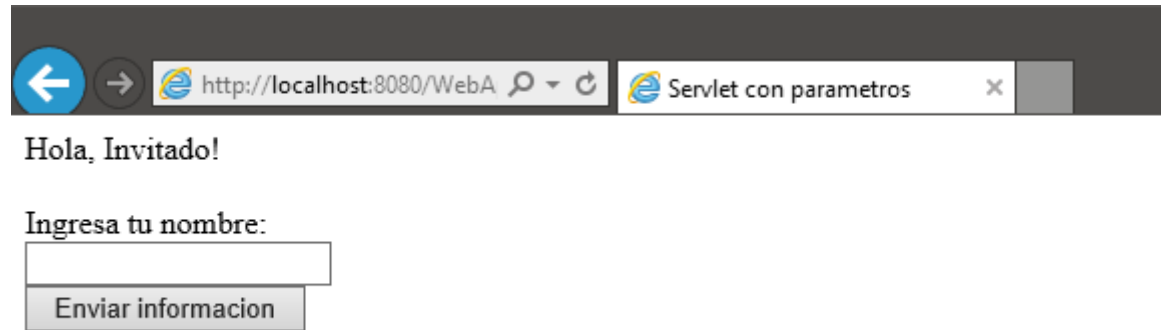
Ejercicio 1 ->

```
//Metodos de inicializacion del servlet
@Override
public void init() throws ServletException{
    System.out.println("Inicio del ciclo de vida del servlet: " + this.getServletName());
}

@Override
public void destroy(){
    System.out.println("Servlet " + this.getServletName() + "se ha detenido.");
}
```

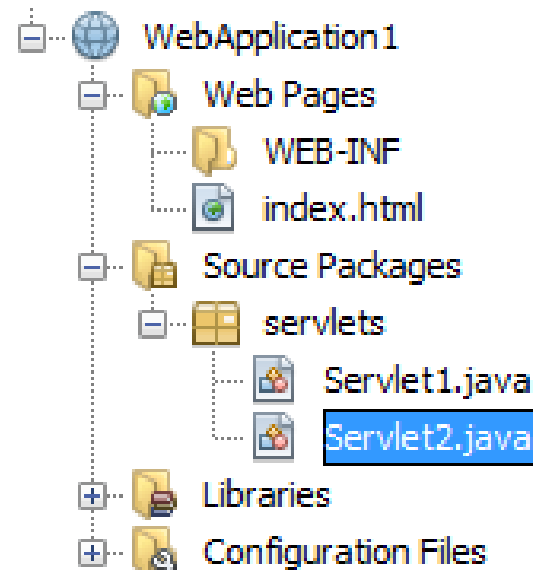
Modifique el ejercicio con la finalidad de mostrar su nombre, edad y una breve descripción mediante un servlet.

Ejercicio 2 -> Crear un segundo Servlet con el nombre de Servlet2 y crear el siguiente formulario



A screenshot of a web browser window. The address bar shows 'http://localhost:8080/WebA'. The page content includes a greeting 'Hola, Invitado!', a label 'Ingresa tu nombre:', an empty text input field, and a button labeled 'Enviar informacion'.

La estructura del proyecto es la siguiente:



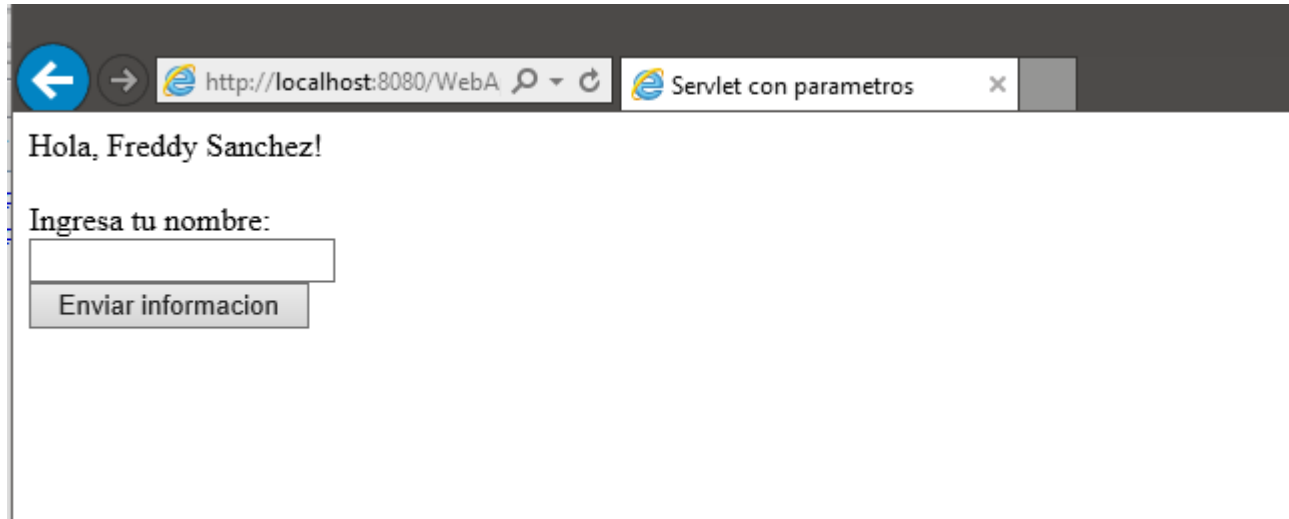
La implementación de Servlet2 :

```
private static final String DEFAULT_USER = "Invitado";
protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String user = request.getParameter("user");
    if(user == null){
        user = Servlet2.DEFAULT_USER;
    }
    response.setContentType("text/html");
    response.setCharacterEncoding("UTF-8");
    //Definimos el printwriter
    PrintWriter writer = response.getWriter();
    writer.append("<!DOCTYPE html> \r\n")
        .append("<html> \r\n")
        .append("    <head> \r\n")
        .append("        <title> Servlet con parametros </title> \r\n")
        .append("    </head>\r\n")
        .append("    <body>")
        .append("        Hola, ").append(user).append("!<br/><br/> \r\n")
        .append("        <form action = \"Servlet2\" method = \"POST\" > \r\n")
        .append("            Ingresa tu nombre: <br/> \r\n")
        .append("            <input type = \"text\" name = \"user\" /> <br/> \r\n")
        .append("            <input type = \"submit\" value = \"Enviar informacion\" />    ")
        .append("        </form> \r\n")
        .append("    </body> \r\n")
        .append("</html> \r\n");
}
```

HttpServlet methods. Click on the + sign on the left to edit the code.

Recuerde mantener un código bien organizado y comentado.

Ejecute su **servlet** y comente los resultados en clase.



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/WebA` and the page title is "Servlet con parametros". The main content area displays the text "Hola, Freddy Sanchez!". Below this, there is a form with the label "Ingresa tu nombre:" followed by a text input field. At the bottom of the form is a button labeled "Enviar informacion".

Hola, Freddy Sanchez!

Ingresa tu nombre:

Enviar informacion

JSP Para mostrar contenido

El principio ... `
` es mas fácil que `System.out.println("
 ")`

Elementos que componen un JSP

Directivas

`<%@ Esta es una directiva %>`

`<%! Esta es una declaración %>`

```
<!-- comentario HTML -->  
<%-- comentario JSP --%>
```

Directivas

<%@ Esta es una directiva %>

Las directivas se van a utilizar cuando:

- *Queremos indicar al interprete de JSP que represente a una acción (Como establecer el contentType de nuestra pagina) .
- *Vamos a importar alguna clase de java
- *Se incluyen scripts, css, etc
- *Se hace referencia a una biblioteca de tags de JSPs

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="servlets.Servlet1" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ejemplo de directiva page con atributo import
      con nueva clase dentro de un nuevo paquete</title>
  </head>
  <body>
    <font size="20" color="red">
      <%
        Servlet1 ejemplo = new Servlet1();
        out.print(ejemplo.saludar());
      %>
    </font>
  </body>
</html>
```

Declaraciones

<%! Esta es una declaración %>

Utilizamos las declaraciones de JSP para escribir variables de instancia, clases o métodos. Tener en cuenta que estas líneas de código java tienen un alcance (**scope**) únicamente al nivel del JSP. Esto quiere decir que las declaraciones están hechas con la clase JSP Servlet generada (Recordar que un JSP finalmente se convierte en un servlet).

```
<%--
    Document    : declaraciones
    Created on  : 17/10/2016, 01:05:15 AM
    Author      : fredd
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Ejemplo Declaraciones1</title>
    </head>
    <body>
        <%! double num1=2.2, num2=4.4, num3=6.0;%>
        <h2>
        <center>
            Los números a promediar son: <%=num1%>, <%=num2%> y <%=num3%><br><hr>
            <%! public double media(double n1,double n2,double n3){ return (n1+n2+n3)/3; } %>
            Media = <%=media(num1,num2,num3)%>
        </center>
        </h2>
    </body>
</html>
```


Declaraciones

<%! Esta es una declaración %>

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Ejemplo Declaraciones2</title>
    </head>
    <body>
        <h2><center>
            <%! private int Contador = 0; %>
            Número de accesos a la página desde que se cargó la primera vez: <%= ++Contador %>
        </center></h2>
    </body>
</html>
```

Expresiones

<%= Esta es una expresión %>

En JSP se utilizan las expresiones para insertar valores, obtenidos con Java, directamente a la salida que se envía al cliente o solicitante.

¿Cómo funcionan las expresiones?. Pues muy sencillo, la expresión es evaluada en el Motor de JSP, se obtiene un valor de la expresión y éste se sustituye justo donde se utiliza.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP (Expresiones)</title>
    </head>
    <body>
        <h1>
            <center>
                Hola. La fecha y la hora son <%= new java.util.Date() %>
            </center>
        </h1>
    </body>
</html>
```

Scriptlets

<% Este es un scriptlet %>

Al igual que una declaración, un scriptlet contiene código java. La diferencia radica en el alcance (scope) y aquí podremos escribir cualquier tipo de código java, desde declarar variables hasta ejecutar sentencias de control.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ejemplo Scriptlet Super Heroes</title>
  </head>
  <body>
    <%
      String[] nombres={"IronMan","Batman","Robin","Spiderman","Carmen3CT"};
      for ( int i = 0; i < nombres.length; i ++ )
      {
    %>
    <font color="blue" size="<%=i+2%>">
      a <i><%= nombres[i]%></i></font><br>
    <% } %>
  </body>
</html>
```

Scriptlets

<% Este es un scriptlet %>

Al igual que una declaración, un scriptlet contiene código java. La diferencia radica en el alcance (scope) y aquí podremos escribir cualquier tipo de código java, desde declarar variables hasta ejecutar sentencias de control.

```
<body>
  <%
    java.util.Calendar ahora = java.util.Calendar.getInstance();
    int hora = ahora.get(java.util.Calendar.HOUR_OF_DAY);
  %>
  <H1>
    <center>
      <b>Hola Fred,
        <i>
          <% if ((hora > 20) || (hora < 6)) {%>
            buenas noches.
          <% } else if ((hora >= 6) && (hora <= 12)) {%>
            buenos días.
          <% } else {%>
            buenas tardes.
          <% };%>
        </i>
      </b>
      <HR>
    </center>
  </H1>
</body>
```

Sesiones (HTTP Session)

Sesión: una **sesión** es un **período temporal ocupado por una cierta actividad**.

Esto quiere decir que, durante una determina sesión, se llevan a cabo una serie definida de tareas.

¿Y una sesión informática?

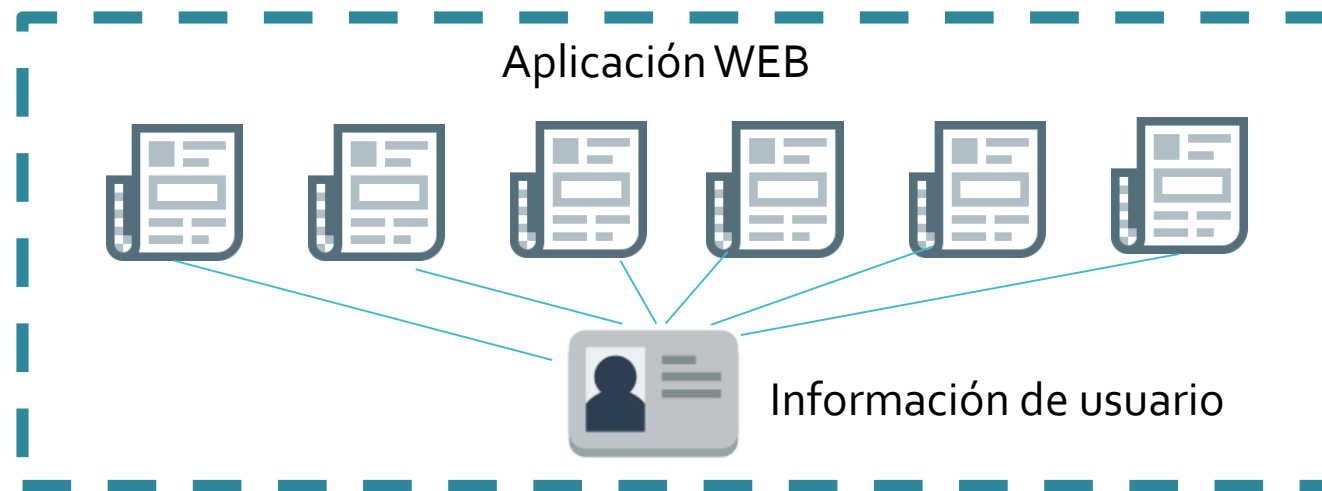
Se conoce como sesión a la **duración de una conexión** a un determinado sistema o red.

La sesión informática suele incluir el intercambio de paquetes de información entre un usuario y un servidor.

Es habitual que el usuario deba ingresar un nombre de usuario y contraseña para iniciar una sesión, en un procedimiento conocido como **log in** o **loguearse**.

¿Porqué utilizar una sesión?

Es común que cuando navegamos en cualquier aplicación WEB (Facebook, twitter, Wikipedia, foros, Correo electrónico, etc) utilicemos información sobre nosotros (Usuario) para acceder a estos sitios e Interactuar con ellos.



Paginas que conforman la aplicación Web



Usuario

Existe una peculiaridad en el protocolo HTTP y es que este no cuenta con un estado, es decir, es un protocolo sin estado (Stateless).

¿Y esto en que nos afecta a nosotros los desarrolladores?



En muchas ocasiones nos encontraremos con el problema de compartir estado (datos usuario) entre un conjunto amplio de páginas de nuestra Aplicación

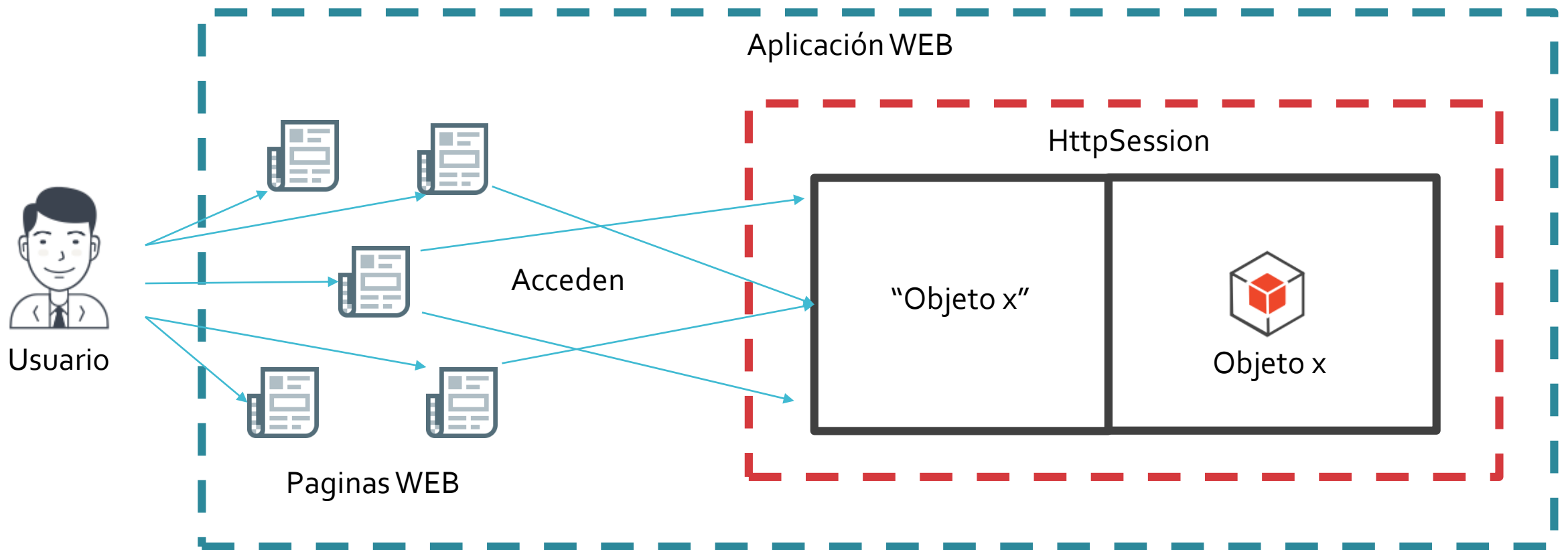
¿Java EE Me proporciona herramientas para resolver este problema?



La respuesta es:

<https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html>

Si, Java me proporciona la clase HttpSession que cuenta con una estructura De tipo diccionario (Hash Map) y nos permite almacenar cualquier tipo de objeto en ella de tal forma Que pueda ser compartido entre diferentes paginas de nuestra aplicación WEB.



Funcionamiento del mecanismo de sesión

Usuario crea sesión

- Accediendo a una pagina que contenga la funcionalidad para crear la sesión (Un Servlet puede encargarse de esto)

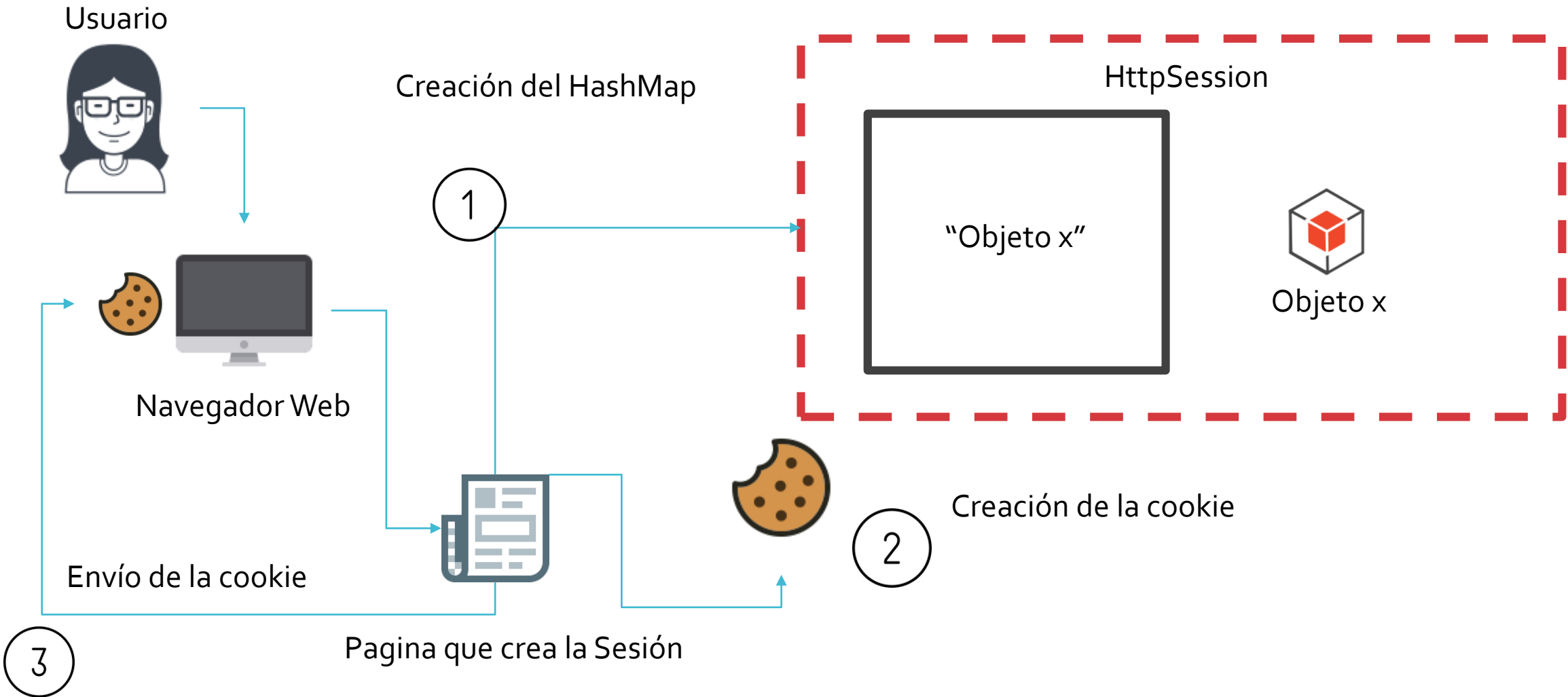
Creación de un
HashMap Vacío

- Nos permitirá almacenar la información relativa a este usuario. (Este HashMap es a nivel de servidor)

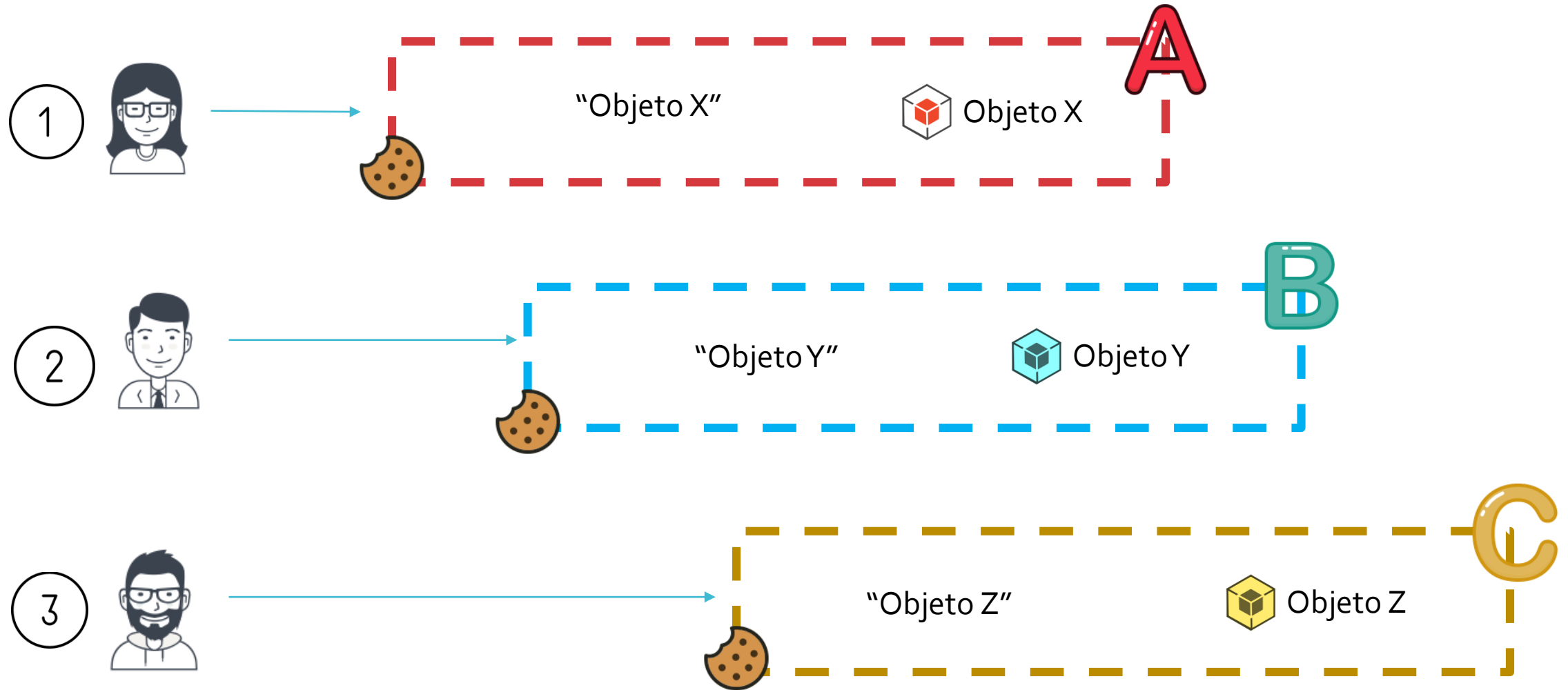
Envío de Cookie al
navegador de
usuario

- Sirve para identificar y asociar el HashMap (El HashMap puede ser consultado desde diversas paginas permitiendo que la información se comparta)

Funcionamiento del mecanismo de sesión



Las sesiones son a nivel de usuario





Ejercicio de sesiones

Sesiones y Cookies

El tiempo de vida de una sesión comienza cuando un usuario se conecta por primera vez a un sitio web pero su finalización puede estar relacionada con tres circunstancias:

1. Cuando se abandona el sitio web.
2. Cuando se alcanza un tiempo de inactividad que es previamente establecido, en este caso la sesión es automáticamente eliminada. Si el usuario siguiera navegando se crearía una nueva sesión.
3. Se ha cerrado o reiniciado el servidor.
4. <https://josephz.wordpress.com/2009/09/20/pequeno-codigo-jsp-de-sesiones-jsp-session/>