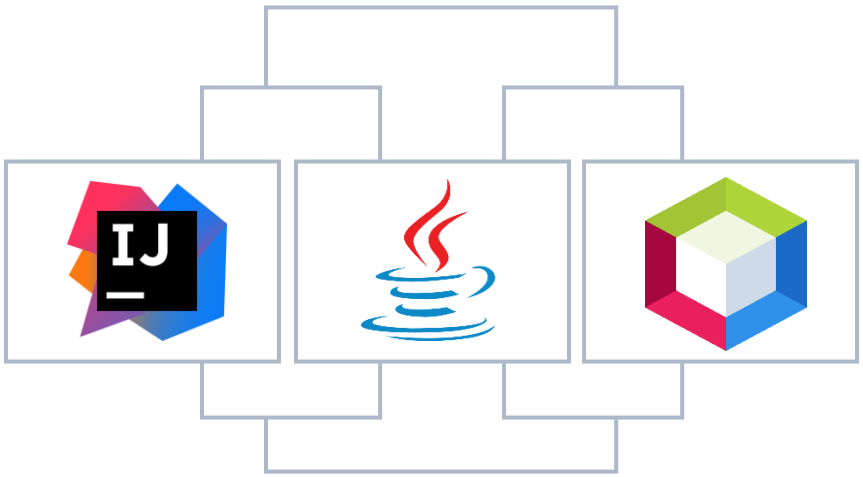
# Universidad Peruana Los Andes

## Facultad de Ingeniería

### Escuela Profesional de Ingeniería de Sistemas y Computación




# Desarrollo de Aplicaciones I

Mg. Ing. Raúl Fernández Bejarano

2024

# 1. ¿Qué se necesita para hacer una aplicación que funciona en la red?

## 1.1 Objetivo

Ubicar los componentes esenciales de una aplicación Web.

## 1.2 ¿Qué es una aplicación Web?

El término Web proviene del inglés, y significa red o malla, este término ha sido adoptado para referirse al internet. Una aplicación Web es un conjunto de páginas que funcionan en internet, estas páginas son las que el usuario ve a través de un navegador de internet (Internet Explorer de Microsoft, Chrome, Mozilla Firefox, etc.) y están codificadas en un lenguaje especial. Existen varios tipos de páginas Web: HTML, JSPs, XML… En la primera parte de este curso trabajaremos con las JavaServer Pages (JSPs). Las páginas JSP se ejecutan en una máquina virtual de Java, el resultado de la ejecución es código HTML listo para correr en el navegador. Las JSP constituyen la interfaz de la aplicación con el usuario.

Las aplicaciones Web se almacenan en un servidor, el cual es una computadora que se encarga de que éstas sean accesibles a través de internet. Como se ilustra en la Figura I-1, una aplicación Web corre en un *servidor* bajo el control de un software especial, al cual se le llama también *servidor*. Para evitar confusiones es importante aclarar que el *software servidor* corre en una *computadora servidor*.

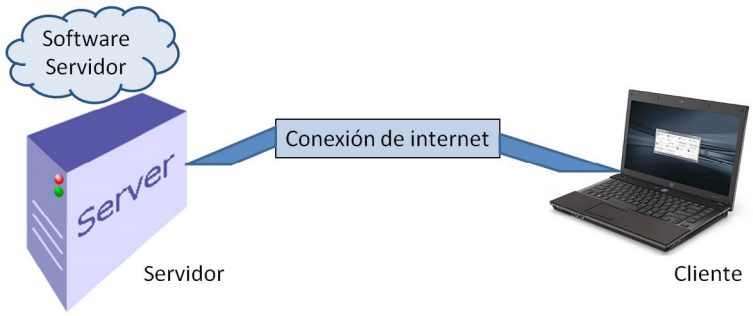



**Figura I-1. Una aplicación web funciona con una computadora-servidor y una o varias computadoras-cliente conectadas a través de internet**

Uno de los *software servidores* más ampliamente utilizados es el *Apache Tomcat*, y es el que usaremos en este curso, es de distribución libre. *GlassFish* es otro software servidor que también es muy utilizado.

Es muy común que las aplicaciones Web hagan uso de una base de datos ubicada en la computadora-servidor, los manejadores de bases de datos más populares son Oracle y MySQL. En este curso utilizaremos MySQL, porque es gratuito. El manejador de base de datos permite que varios clientes compartan la información, éste es uno de los aspectos más útiles de las aplicaciones web, ya que permite el comercio en línea (tiendas virtuales, reservaciones de hoteles, vuelos, etc.) y facilita la organización en las instituciones (solicitudes en línea, sistemas de inscripción, control de préstamos bibliotecarios, etc.), como se ilustra en la Figura I-2.

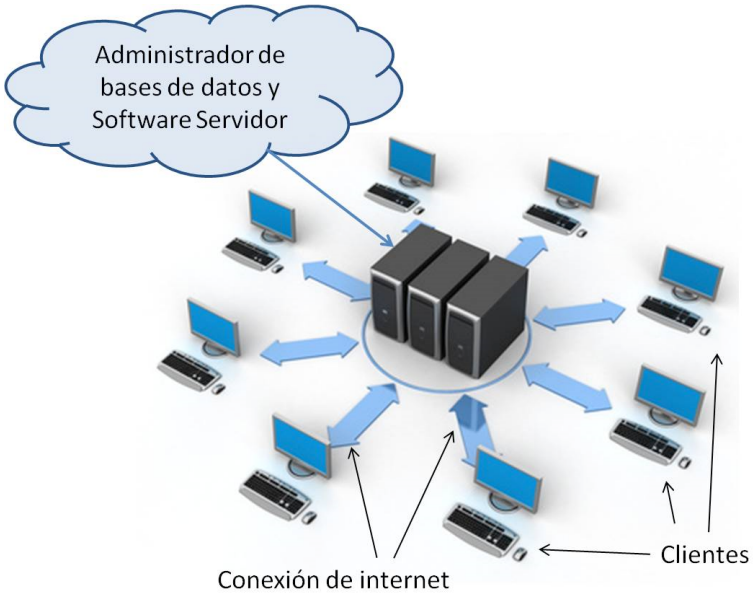



**Figura I-2. Gracias al administrador de base de datos que corre en el servidor, las computadoras-cliente comparten una misma información.**

Finalmente, al "cerebro" de la aplicación web, se le conoce como la *lógica del negocio* y ésta se le puede codificar de diversas formas (C#, PHP, .NET, JavaScript,…) en el presente curso utilizaremos *Java* para hacer *Servlets* y *clases Java*. Dos de los ambientes de desarrollo integrado más utilizados para el desarrollo de aplicaciones web son NetBeans y Eclipse, ya que son gratuitos. En el curso utilizaremos NetBeans.

## 1.3 Peticiones y respuestas en las páginas Web

El software servidor y el navegador del cliente se comunican por medio de un protocolo llamado HiperText Transfer Protocol (HTTP). El navegador hace la petición de una página Web al servidor enviándole un mensaje conocido como *petición HTTP (request)*, la cual incluye el nombre de un archivo *.html, y el servidor contesta a esta petición con un mensaje conocido como *respuesta HTTP (response)*.

En el caso de las ***páginas Web estáticas***, el servidor proporciona en la respuesta HTTP el documento *.html que el navegador solicitó. El usuario que visualiza una página Web estática, no puede hacer modificaciones en ésta. Cuando el usuario da clic en la liga a otra página, se envía otra petición HTTP pero ahora con el nombre del archivo de la otra página que se desea visualizar. Otra manera de pedir una página diferente es escribiendo directamente en el navegador su dirección Web.

En el caso de las ***páginas web dinámicas***, el servidor pasa la petición HTTP generada por el navegador a una *aplicación Web,* la cual procesa la información que contiene la petición. La respuesta que genera la aplicación se envía al servidor, quien contesta al navegador con una respuesta HTTP, como se ilustra en la Figura I-3. La respuesta que recibe el usuario no es siempre la misma, sino que depende de la información que éste proporciona, por eso se dice que la página es dinámica.

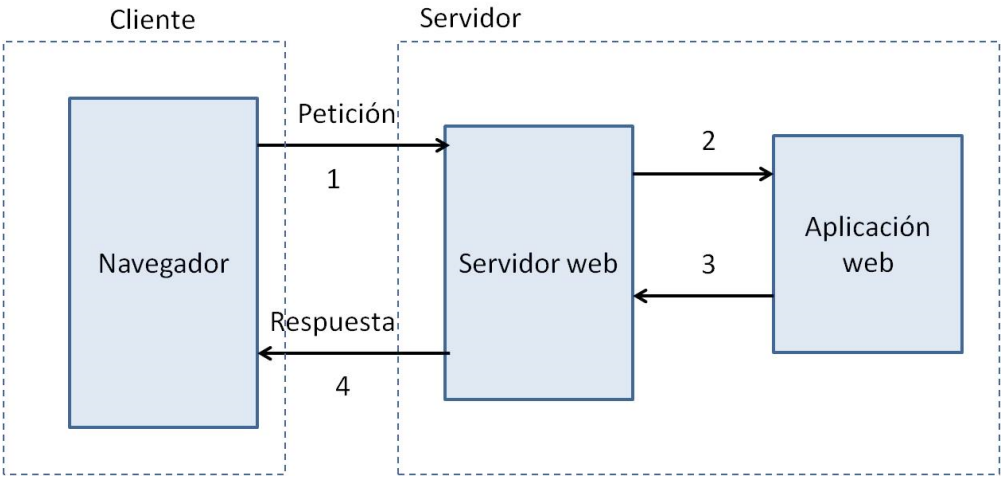



**Figura I3. El servidor web turna el procesamiento de las páginas web dinámicas a una aplicación web**

## 1.4 ¿Qué es una JSP?

Una página JSP (*JavaServer Page*) es una página HTML a la que se le incrusta código Java. En el capítulo II se da una introducción al código HTML para los lectores que no están familiarizados con éste. El código Java se incrusta entre los siguientes indicadores **<%** y **%>**. En el capítulo III trabajaremos con las JSP.

## 1.5 ¿Qué es un Servlet?

Un *servlet* es una clase Java (hija de la clase `HttpServlet`) y corre en el servidor. Su nombre se deriva de la palabra *applet*. Anteriormente se utilizaban los *applets*, que eran pequeños programas, escritos en Java, que corrían en el contexto del navegador del cliente, sin embargo, desde que Microsoft Explorer suspendió su mantenimiento, los *servlets* substituyeron a los *applets*, sólo que los *servlets* no tienen una interfaz gráfica. Un *servlet* da servicio a las peticiones de un navegador Web, es decir, recibe la petición, la procesa y devuelve la respuesta al navegador. Un *servlet* es una clase Java en la que se puede incrustar código HTML. Como los *servlets* están escritos en Java, son tan portables como cualquier aplicación Java, es decir, pueden funcionar sin necesidad de cambios en diferentes servidores.

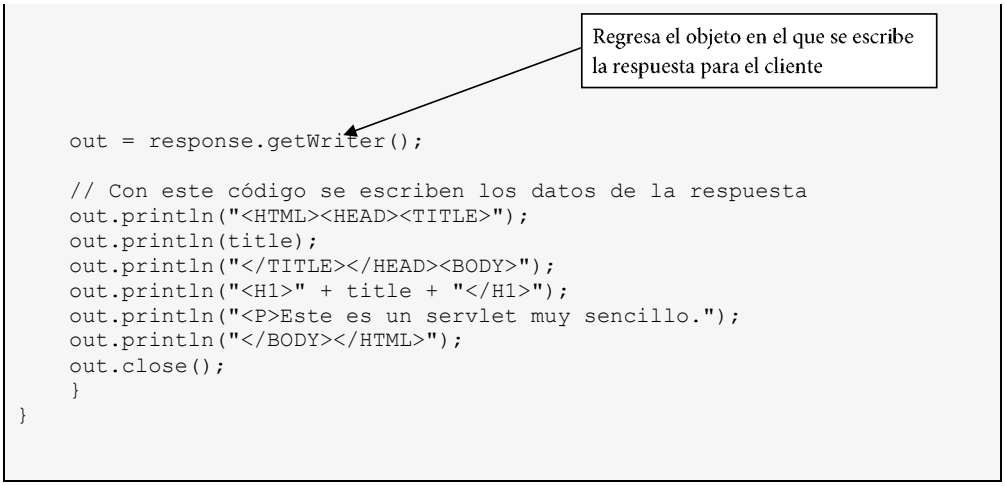
A manera de ejemplo, presentamos el código de un *servlet* muy sencillo, el cual genera como respuesta una página HTML que despliega un breve mensaje.

El objeto `request` se usa para leer los datos que están en los formularios que envía el navegador. El objeto `response` se usa para especificar códigos y contenidos de la respuesta.

```
public class EjemploServlet extends HttpServlet {

    // Escribe una página Web sencilla

    public void doGet (HttpServletRequest request,
                       HttpServletResponse response)
                       throws ServletException, IOException {

    PrintWriter out;
    String title = "Ejemplo de un servlet";


    response.setContentType("text/html"); // Regresa texto y html
```

Recibe los parámetros enviados por el navegador

Tipo de información que regresa el *servlet*




```
    out = response.getWriter();

    // Con este código se escriben los datos de la respuesta
    out.println("<HTML><HEAD><TITLE>");
    out.println(title);
    out.println("</TITLE></HEAD><BODY>");
    out.println("<H1>" + title + "</H1>");
    out.println("<P>Este es un servlet muy sencillo.");
    out.println("</BODY></HTML>");
    out.close();
    }
}
```

Un servlet puede generar su resultado consultando a una base de datos, invocando a otra aplicación o computando directamente la respuesta. También puede dar formato al resultado generando una página HTML, y enviar al cliente un código ejecutable.

## 1.6 La arquitectura Modelo-Vista-Controlador

La arquitectura Modelo-Vista-Controlador (MVC), es un patrón que organiza la aplicación en tres partes independientes:

- ***La vista***.- Son los módulos SW involucrados en la interfaz con el usuario, por ejemplo, las páginas de internet que se despliegan en la computadora del usuario.
- ***El controlador***.- Es el software que procesa las peticiones del usuario. Decide qué modulo tendrá el control para que ejecute la siguiente tarea.
- ***El modelo***.- Contiene el núcleo de la funcionalidad, es decir, ejecuta la "lógica del negocio". Se le llama *lógica del negocio* a la forma en la que se procesa la información para generar los resultados esperados. El modelo se conecta a la base de datos para guardar y recuperar información.
-

El patrón MVC convierte la aplicación en un sistema modular, lo que facilita su desarrollo y mantenimiento. En la Figura I-4 se ilustran las tres partes del modelo MCV.

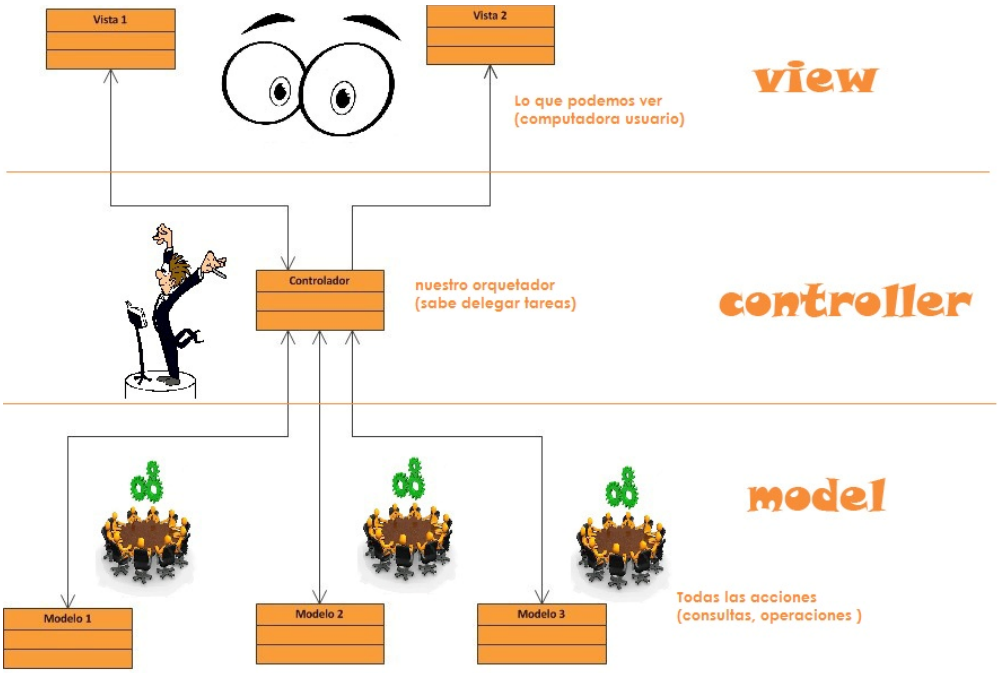



**Figura I-4. Arquitectectura Modelo-Vista-Controlador (MVC)**

En las aplicaciones Web más sencillas de este curso, sólo separamos la vista (con páginas JSP) y el procesamiento de los datos (con *servlets*). Sin embargo, para aplicaciones Web un poco más avanzadas utilizamos el patrón MVC. La vista la implementaremos con páginas JSP, el controlador con *servlets* y el modelo con clases Java(ver capítulo V).

Al diseñar y codificar los módulos MVC, es importante que se haga una buena división de las tareas. Por ejemplo, las páginas JSP no deben incluir tareas de procesamiento de datos y los *servlets* no deben contener código para la presentación de las páginas. La división de tareas entre el controlador y el modelo es la más difícil. Mantener una total independencia entre los módulos es a veces imposible, sin embargo es el objetivo ideal.

## 1.7 Frameworks

Un *framework* se traduce al español como *marco de trabajo*, y es un esqueleto para el desarrollo de una aplicación. Los *frameworks* definen la estructura de la aplicación, es decir, la manera en la que se organizan los archivos e, inclusive, los nombres de algunos de los archivos y las convenciones de programación. Como el framework proporciona el esqueleto que hay que

rellenar, el programador ya no se preocupa por diseñar la estructura global de la aplicación. Como la información está estandarizada es más sencillo el trabajo colaborativo, y el mantenimiento de las aplicaciones, porque la estructura de la aplicación es bien conocida.

Podemos ver a un *framework* como una estructura de software que tiene componentes personalizables e intercambiables y constituye una aplicación genérica incompleta y configurable, a la que se le añaden las últimas piezas para construir una aplicación concreta.

La mayoría de los *frameworks* para desarrollo Web implementan el patrón MVC con algunas variantes. Entre los *frameworks* más conocidos están:

- JavaServer Faces
- Struts
- Spring Web MVC

En la segunda parte de este libro aprenderemos a hacer aplicaciones Web con *JavaServer Faces*.

## 1.8 Preparación del ambiente para desarrollar una aplicación wWeb

### 1.8.1 Para instalar el software servidor *Apache Tomcat*

*Apache Tomcat* es un software que actúa como *contenedor de servlets*, es decir, recibe las peticiones de las páginas Web y redirecciona estas peticiones a un objeto de clase *servlet*.

Apache Tomcat puede descargarse de http://tomcat.apache.org/

Descargar el zip (32 o 64bit) y descomprimirlo para iniciar la instalación.

### 1.8.2 Para instalar *NetBeans*

NetBeans es un entorno de desarrollo en el que los programadores puedan escribir, compilar, depurar y ejecutar programas. Éste se puede descargar de:

https://netBeans.org/downloads/index.html

Para poder instalar NetBeans es necesario tener previamente instalado el JDK (*JavaDevelopment Kit*), ya que éste es la plataforma en la que se ejecuta *NetBeans*.

Cuando se ejecuta el instalador de *NetBeans*, es recomendable seleccionar los dos servidores: **Apache Tomcat** y **Glassfish**, que son con los que trabaja *NetBeans*. Esto se hace con el boton "*Customize*", como se indica en la Figura I-5.

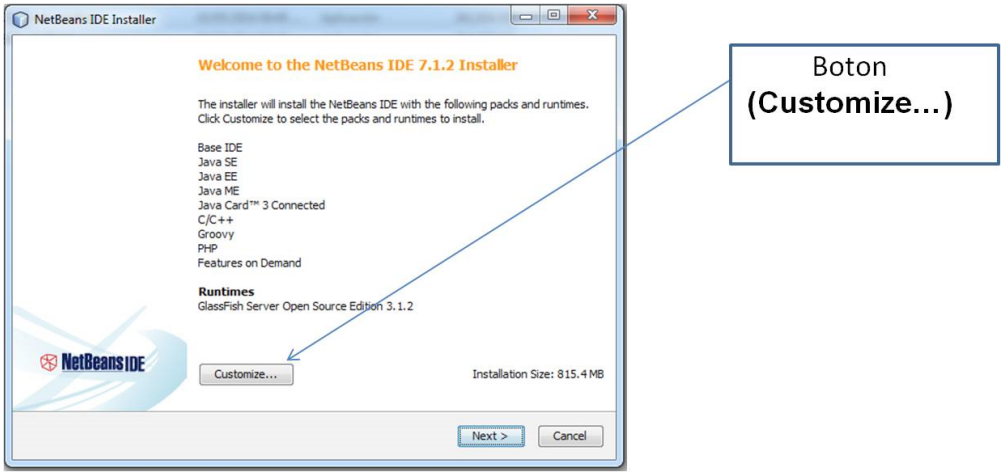



**Figura I-5 El botón para personalizar la instalación**

La selección de los dos servidores se ilustra en la Figura I-6.

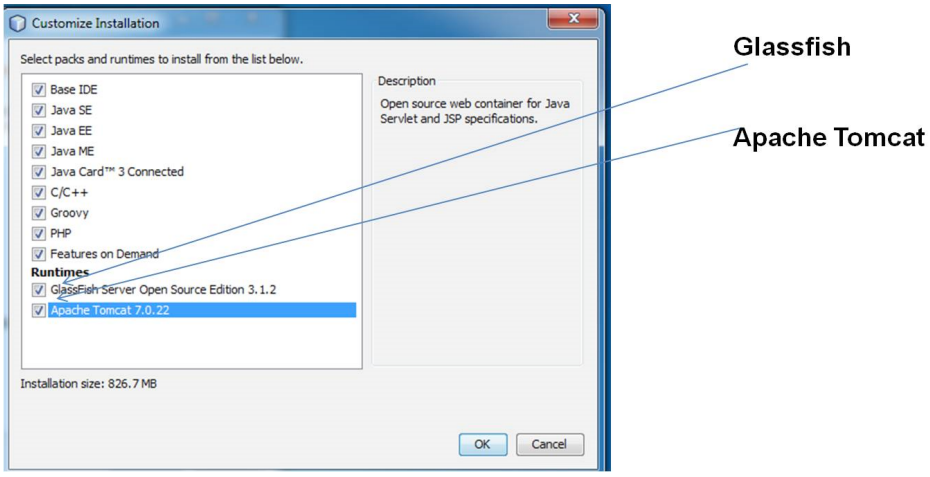



**Figura I-6. Selección de los dos servidores**

En el capítulo III se explica cómo crear una aplicación Web con *NetBeans*.

## 1.9 Iniciando un proyecto Web con NetBeans

Para iniciar una aplicación Web se selecciona File →New Project→JavaWeb, como se ilustra en la Figura I-7.

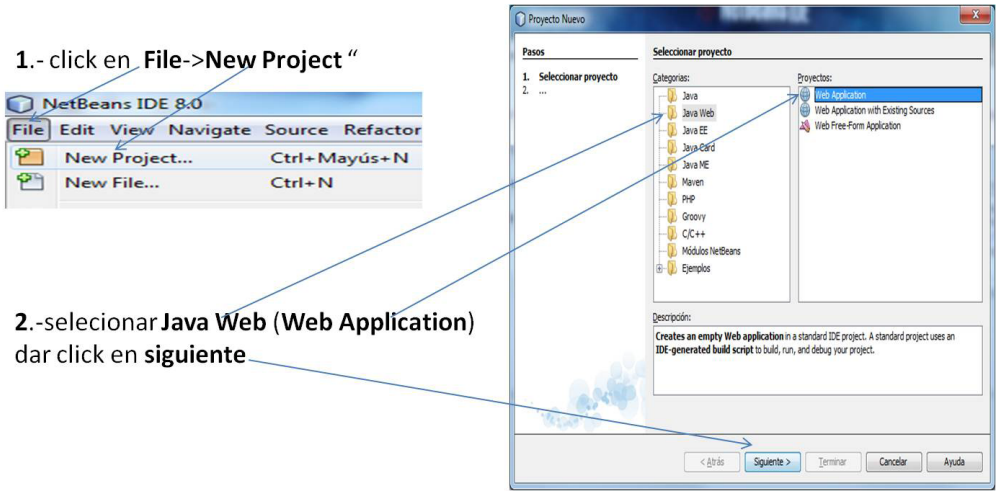



**Figura I-7. Inicio de una aplicación web en NetBeans**

Es muy importante seleccionar correctamente el servidor que tenemos instalado, en este caso Tomcat, como se ilustra en la Figura I-8.

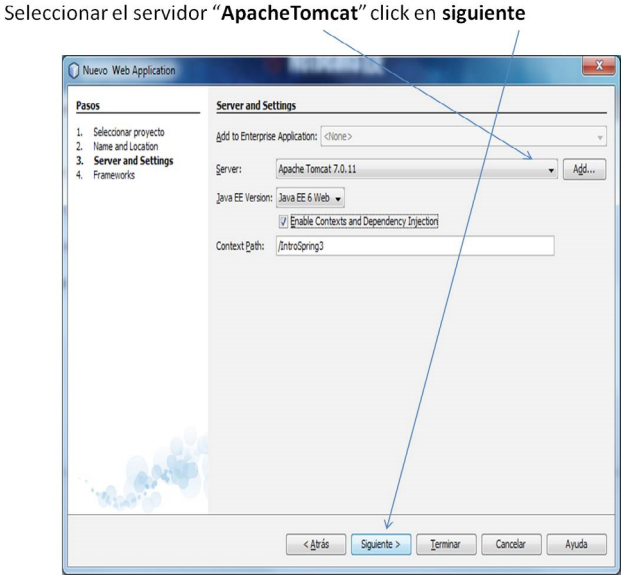



**Figura I-8. Selección del servidor Tomcat**

Para una aplicación Web sencilla no es necesario seleccionar un *framework*. Cuando se selecciona el botón "Terminar", *NetBeans* crea automáticamente la estructura de un proyecto Web. Esta estructura se muestra en la Figura I-9.

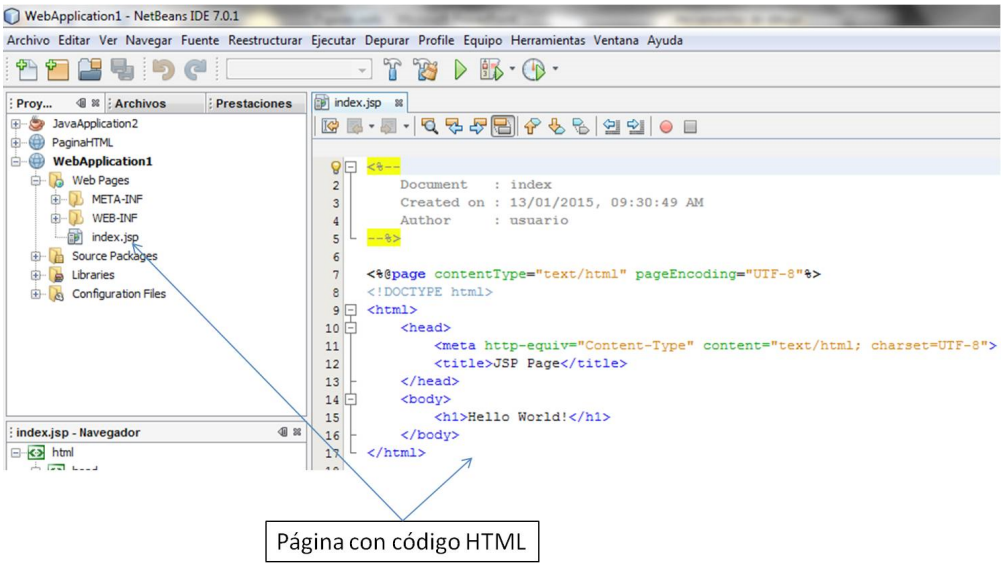



**Figura I-9. Estructura de un proyecto Web en NetBeans**

El archivo *index.jsp* contiene código HTML. Por *default* la página *index.jsp* sólo contiene el saludo "Hello World!". En el siguiente capítulo aprenderemos a trabajar con código HTML y a visualizar las páginas en el navegador.