

Programación Frontend y Backend

ARQUITECTURA

SERVIDOR APLICACIONES



Servidores

Los servidores de aplicaciones permiten ejecutar aplicaciones complejas y escalables en una red o en la web.

Estas aplicaciones pueden ser desde sistemas de gestión de contenidos hasta aplicaciones empresariales que requieren un alto nivel de procesamiento y acceso a bases de datos.

Los servidores de aplicaciones proporcionan un entorno en el que las aplicaciones pueden ser desplegadas y ejecutadas, y proporcionan una serie de servicios para ayudar a las aplicaciones a interactuar con otros sistemas y servicios, como bases de datos, servidores de correo electrónico, servicios de autenticación, entre otros.



Servidores: Características

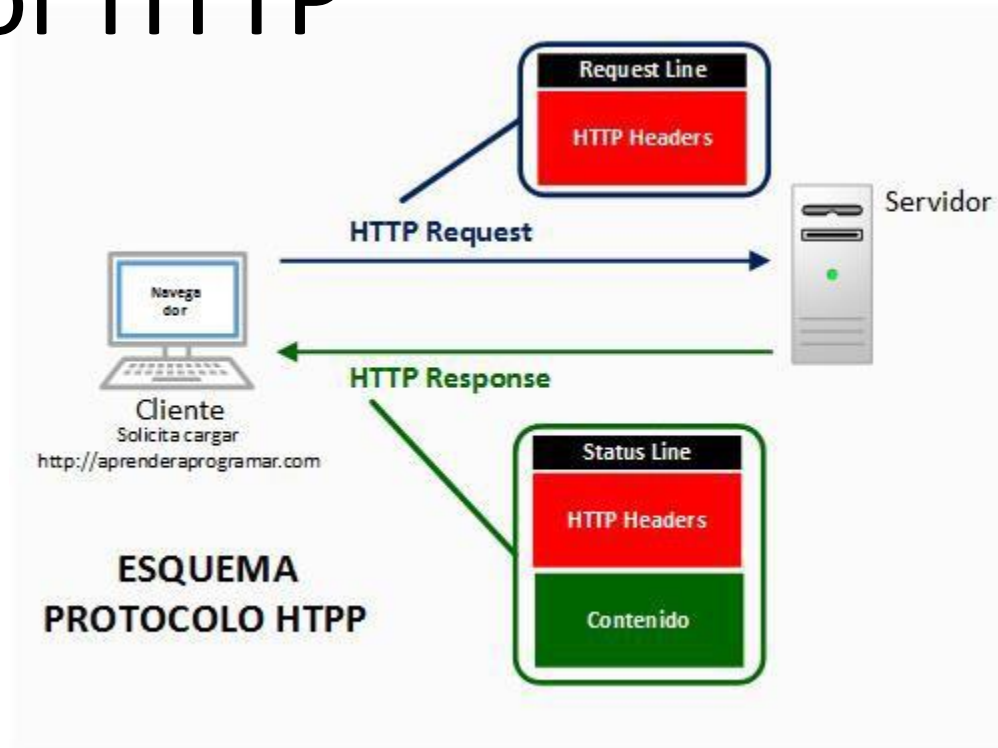
1.Escalabilidad: Los servidores de aplicaciones pueden manejar grandes volúmenes de tráfico y solicitudes, lo que los hace ideales para aplicaciones que necesitan escalar de manera efectiva para manejar un gran número de usuarios.

2.Gestión de recursos: Los servidores de aplicaciones proporcionan herramientas y servicios para administrar recursos de hardware y software, como la gestión de memoria, la administración de bases de datos y la gestión de sesiones.

3.Seguridad: Los servidores de aplicaciones proporcionan herramientas y servicios para proteger las aplicaciones de amenazas de seguridad, como ataques de denegación de servicio (DDoS), inyecciones de código malicioso y otros tipos de ataques.

4.Interoperabilidad: Los servidores de aplicaciones permiten a las aplicaciones interactuar con otros sistemas y servicios de la red, como bases de datos y servicios de autenticación, lo que las hace ideales para aplicaciones empresariales complejas que necesitan interactuar con múltiples sistemas.

Servidor HTTP



En este diagrama, el cliente es un navegador web que solicita recursos a través del protocolo HTTP, mientras que el servidor es la máquina que aloja esos recursos y los proporciona al cliente.

Protocolo HTTP

El proceso se inicia cuando el cliente envía una petición HTTP al servidor, solicitando un recurso específico, como una página web o un archivo. Esta petición incluye información como el tipo de solicitud (GET, POST, etc.), la URL del recurso y cualquier otro dato que sea necesario para completar la solicitud.

El servidor recibe la petición y la procesa, buscando el recurso solicitado y devolviendo una respuesta HTTP al cliente. Esta respuesta incluye información como el código de estado HTTP (200 para indicar que se ha encontrado el recurso solicitado, o un código de error en caso contrario), el tipo de contenido (HTML, imagen, archivo de audio, etc.) y los datos del recurso solicitado en sí.

El cliente recibe la respuesta HTTP y la procesa, mostrando los datos al usuario final en el caso de que la petición haya sido satisfactoria. En caso contrario, el navegador puede mostrar un mensaje de error al usuario.

Ejercicio: [Ver cabecera http en petición a www.google.es](http://www.google.es)

Servidores Web VS Servidor Aplicaciones

La principal diferencia entre un servidor web y un servidor de aplicaciones es que el servidor web se utiliza principalmente para alojar y entregar páginas web estáticas, mientras que el servidor de aplicaciones se utiliza para alojar y entregar aplicaciones web dinámicas y escalables.

Otra diferencia importante entre un servidor web y un servidor de aplicaciones es el nivel de complejidad de la aplicación que pueden alojar. Los servidores web son adecuados para sitios web estáticos y aplicaciones simples que no requieren interacciones complejas con bases de datos o servicios de backend. En contraste, los servidores de aplicaciones son más adecuados para aplicaciones empresariales y de misión crítica que requieren una alta disponibilidad, escalabilidad y un alto nivel de procesamiento.

Servidores de Aplicaciones JAVA

- **Apache Tomcat:** es un servidor de aplicaciones web de código abierto que soporta el Servlet API y el JavaServer Pages (JSP) API.
- **JBoss Application Server:** es un servidor de aplicaciones empresariales de código abierto que soporta una amplia gama de tecnologías Java EE, como Servlets, JSP, EJB, CDI, JPA, entre otros.
- **WebSphere Application Server:** es un servidor de aplicaciones empresariales de IBM. Proporciona gran cantidad de herramientas y características de gestión y administración.
- **Oracle Weblogic:** servidor de aplicaciones de Oracle. Proporciona gran cantidad de herramientas y características de gestión y administración.

Apache Tomcat

¿Qué es y para qué sirve?

<https://tomcat.apache.org/>

- Es el servidor Web para aplicaciones Java de Apache
- Ejecución de JSPs y Servlets en el lado del servidor (Servidor Web dinámico)
- Puede funcionar como servidor Web estático también
- Fácil instalación : podemos descargar la versión comprimida del servidor y ejecutar los Scripts que hay en ella para ponerlo en funcionamiento
- Por defecto responde a los puertos (8005,8009,8080)
(<http://localhost:8080>)

Apache Tomcat

Configuración desde Eclipse

Necesitamos tener instalados los siguientes plugins:

Eclipse Marketplace


Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.



Search Recent Popular Favorites Installed Giving IoT an Edge

Find: All Markets All Categories Go


Eclipse Tomcat Plugin 9.1.4

 The Eclipse Tomcat Plugin provides simple integration of a tomcat servlet container for the development of java web applications. This project is the successor of... [more info](#)

by [Markus Keunecke](#), Apache 2.0
[tomcatplugin](#) [tomcat](#) [java](#) [JDT](#) [plugin](#)

★ 203 Installs: 380K (2,944 last month) Installed

Eclipse JST Server Adapters (Apache Tomcat, JOnAS, J2EE) Luna

 This is quick way to install Eclipse JST Server Adapters and JST Server Adapters Extentions (Apache Tomcat, JOnAS, J2EE) This entry was created when Eclipse IDE... [more info](#)

by [Nodeclipse/Enide](#), EPL
[wtp](#) [JST](#) [Server](#) [Adapters](#) [tomcat](#) [Apache](#) [Tomcat](#)

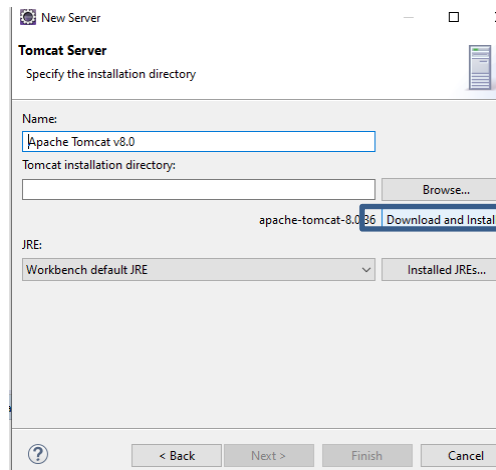
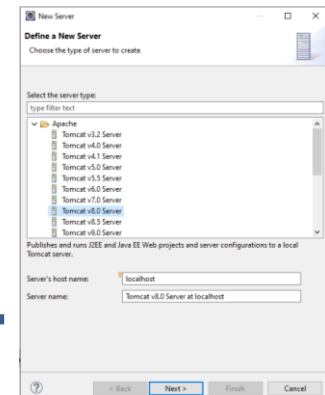
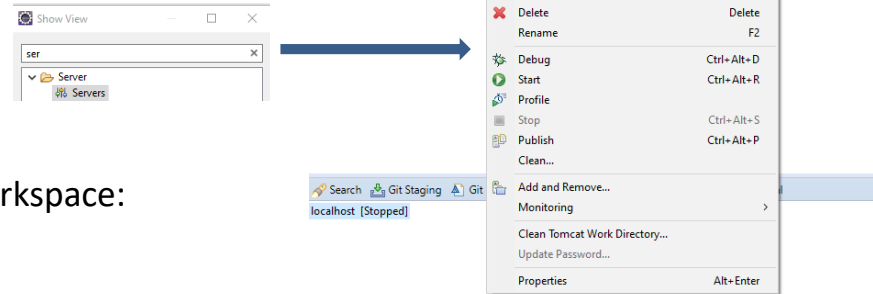
★ 211 Installs: 133K (1,436 last month) Installed

Apache Tomcat

Configuración desde Eclipse

Añadimos ahora un nuevo server a nuestro workspace:

1. Buscamos la vista de servers:
2. Botón derecho sobre la vista que se abre en la pestaña servers y new
3. Seleccionamos el server que queremos instalar
4. Le decimos a eclipse que lo descargue e instale.

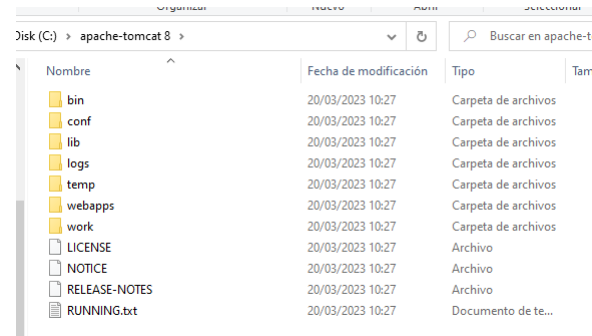


Apache Tomcat

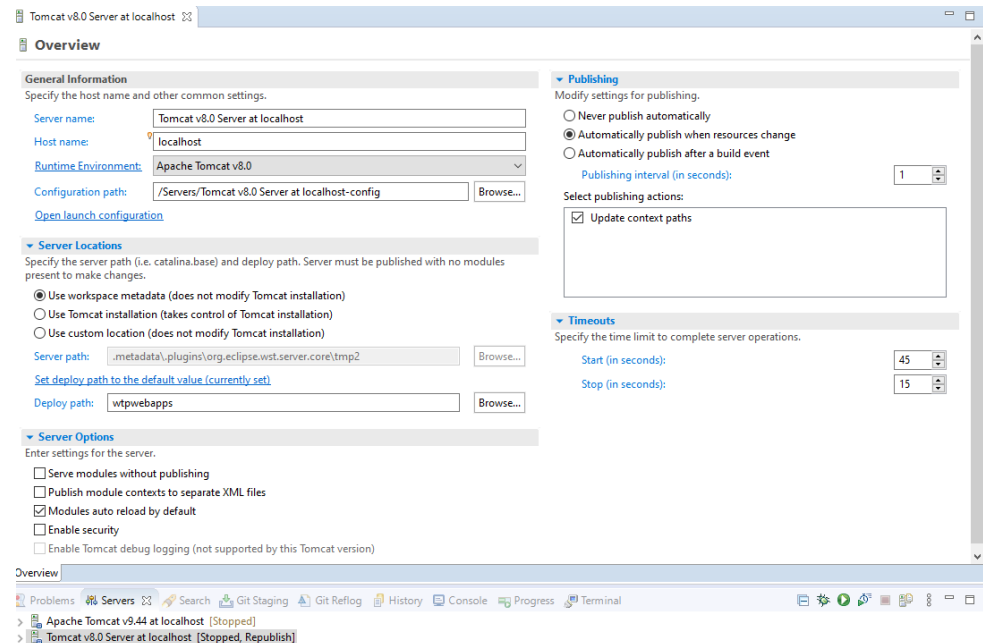
Configuración desde Eclipse

Ya tendremos descargado el tomcat en la carpeta de destino indicada

Ahora vamos a ver con un doble click sobre el server **APAGADO** las opciones de configuración que se nos permiten modificar:



Nombre	Fecha de modificación	Tipo
bin	20/03/2023 10:27	Carpeta de archivos
conf	20/03/2023 10:27	Carpeta de archivos
lib	20/03/2023 10:27	Carpeta de archivos
logs	20/03/2023 10:27	Carpeta de archivos
temp	20/03/2023 10:27	Carpeta de archivos
webapps	20/03/2023 10:27	Carpeta de archivos
work	20/03/2023 10:27	Carpeta de archivos
LICENSE	20/03/2023 10:27	Archivo
NOTICE	20/03/2023 10:27	Archivo
RELEASE-NOTES	20/03/2023 10:27	Archivo
RUNNING.txt	20/03/2023 10:27	Documento de te...



Apache Tomcat

Configuración desde Eclipse

Propiedades importantes:

Si decidimos no utilizar la instalación por defecto de tomcat como directorio, eclipse generara en tiempo de ejecución una carpeta interna donde “clonara” la instalación del tomcat y desplegará.

- **Ventaja:** Esto es muy útil para desarrollar y no preocuparnos por dejar proyectos viejos en la carpeta de despliegues.
- **Problema:** Localizar los proyectos desplegados puede resultar complejo.

▼ **Server Locations**

Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

☒ Use workspace metadata (does not modify Tomcat installation)
☐ Use Tomcat installation (takes control of Tomcat installation)
☐ Use custom location (does not modify Tomcat installation)

Server path:

[Set deploy path to the default value \(currently set\)](#)

Deploy path:

Apache Tomcat

Configuración desde Eclipse

Propiedades importantes:

Si utilizamos la instalación de tomcat:

- **Ventaja:** Puedo arrancar el servidor de manera externa y sin necesitar Eclipse, y localizo fácilmente los despliegues.
- **Problema:** Acumulación de diversos proyectos y por tanto lentitud en el despliegue,.

▼ **Server Locations**

Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

☐ Use workspace metadata (does not modify Tomcat installation)

☒ Use Tomcat installation (takes control of Tomcat installation)

☐ Use custom location (does not modify Tomcat installation)

Server path:

[Set deploy path to the default value \(currently set\)](#)

Deploy path:

Apache Tomcat

Mi primer proyecto WEB

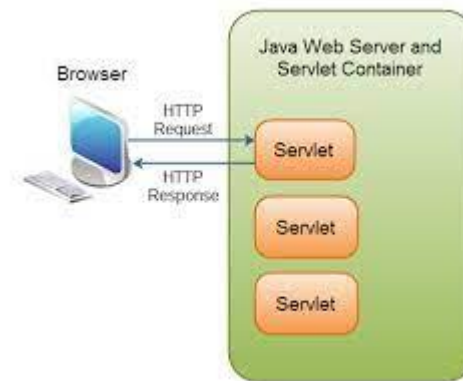
Ejercicio Guiado: Generar mi primera aplicación WEB y aprender a utilizar un tomcat como servidor de contenido estático.



Apache Tomcat - Servlets

Definición:

- Código Java que se ejecuta en un servidor web y que es responsable de recibir y responder a solicitudes de los clientes, como un navegador web.
- Los Servlets son una tecnología clave para el desarrollo de aplicaciones web en Java. Son capaces de manejar **solicitudes HTTP**.



Apache Tomcat - Servlets

Mi primer Servlet

- Vamos a crear un Servlet en Java a través del eclipse y vamos a hacer una llamada por GET desde el navegador y ver que efectivamente podemos comunicarnos vía HTTP.

```
public class EjemploServlet extends HttpServlet {  
  
    public void doGet(HttpServletRequest request, HttpServletResponse  
response)  
        throws ServletException, IOException {  
  
        // Establecer el tipo de contenido de la respuesta  
        response.setContentType("text/html");  
  
        // Obtener el objeto PrintWriter para escribir la respuesta  
        PrintWriter out = response.getWriter();  
  
        // Escribir la respuesta  
        out.println("<html>");  
        out.println("<head>");  
        out.println("<title>Ejemplo de Servlet</title>");  
        out.println("</head>");  
        out.println("<body>");  
        out.println("<h1>Hola Mundo desde un Servlet!</h1>");  
        out.println("</body>");  
        out.println("</html>");  
    }  
}
```


Servicios Web

¿Por que aparecen?

Los servicios web nacen debido a la necesidad de compartir información y funcionalidades entre aplicaciones y sistemas distribuidos en diferentes plataformas, lenguajes de programación y sistemas operativos. Antes de la aparición de los servicios web, las aplicaciones estaban diseñadas para funcionar de manera aislada, sin poder compartir información o funcionalidades con otras aplicaciones.



Servicios Web

Definición:

Los servicios web son una tecnología que permite la comunicación y la interoperabilidad entre diferentes aplicaciones y sistemas informáticos a través de Internet. Básicamente, un servicio web es un conjunto de protocolos y estándares que definen un método para que dos aplicaciones se comuniquen entre sí a través de la red, independientemente de la plataforma, el lenguaje de programación o el sistema operativo que utilicen.

Estos servicios se pueden utilizar para realizar una amplia variedad de tareas, como la integración de diferentes sistemas, la automatización de procesos de negocio y la creación de aplicaciones web dinámicas y personalizadas.



Servicios Web – Tipología

1.SOAP (Simple Object Access Protocol): Es uno de los primeros protocolos utilizados en servicios web y es un estándar de comunicación basado en XML. SOAP se utiliza para intercambiar información entre sistemas distribuidos.

2.REST (Representational State Transfer): Es un estilo de arquitectura web que utiliza protocolos y estándares HTTP para permitir la comunicación entre sistemas. REST se basa en la idea de que todo es un recurso (resource) que puede ser accedido mediante una URL.

3.JSON-RPC: Es un protocolo de comunicación ligero y de alto rendimiento que se utiliza para enviar y recibir datos en formato JSON (JavaScript Object Notation). Se utiliza principalmente en aplicaciones web y móviles.

4.XML-RPC: Es un protocolo de comunicación basado en XML que se utiliza para enviar y recibir datos entre diferentes sistemas. XML-RPC se utiliza principalmente en sistemas de gestión de contenidos y en blogs.

5.GraphQL: Es un lenguaje de consulta de datos que se utiliza para acceder a APIs y servicios web. GraphQL permite que las aplicaciones obtengan solo la información que necesitan, lo que mejora la eficiencia y el rendimiento de la aplicación.