

Aplicaciones web

Se define una aplicación web como una aplicación informática que se ejecuta en un entorno web, de forma que se trata de una aplicación cliente-servidor junto con un protocolo de comunicación previamente establecido:

- **Cliente:** navegador.
- **Servidor:** servidor web.
- **Comunicación:** protocolo HTTP.

Un **servidor de aplicaciones** es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general, a través de Internet y utilizando el protocolo HTTP. Los servidores de aplicación se distinguen de los servidores web en el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos.

Un servidor de aplicaciones también es una máquina en una red de computadores que ejecuta determinadas aplicaciones, gestionando la mayor parte de las funciones de acceso a los datos de la aplicación.

Las principales ventajas de la tecnología de los servidores de aplicaciones es la **centralización** y **disminución** de la complejidad en el desarrollo de las aplicaciones, ya que no necesitan ser programadas, sino que son ensambladas desde bloques provistos por el servidor de aplicación.

Otra de las ventajas es la **integridad de datos y código** ya que, al estar centralizada en una o un pequeño número de máquinas servidoras, las actualizaciones están garantizadas para todos los usuarios.

El término servidor de aplicaciones se aplica a todas las plataformas. Dicho término se utiliza para referirse a los servidores de aplicaciones basadas en web, como el control de las plataformas de comercio electrónico integrado, sistemas de gestión de contenido de sitios web y asistentes o constructores de sitios de Internet.

Uno de los ejemplos destacados es el de Sun Microsystems, la plataforma J2EE. Los servidores de aplicaciones Java se basan en la Plataforma Java™ 2 Enterprise Edition (J2EE™). J2EE utiliza un modelo de este tipo y en general, incluye un nivel Cliente, un nivel Medio, y un EIS. El servidor de tipo Cliente puede contener una o más aplicaciones o navegadores. La Plataforma J2EE es del Nivel Medio y consiste en un servidor web y un servidor EJB. (Estos servidores son también llamados "contenedores".) También podría haber subniveles adicionales en el nivel intermedio. El nivel del Sistema Enterprise Information System (EIS, o "Sistema de Información Empresarial") contiene las aplicaciones existentes, archivos y bases de datos.

Estructura y despliegue

Una aplicación web está compuesta de una serie de servlets, páginas JSP, ficheros HTML, ficheros de imágenes, ficheros de sonidos, texto, clases, etc.; de forma que todos estos recursos se pueden empaquetar y ejecutar en varios contenedores distintos.

Un servlet es una aplicación java encargada de realizar un servicio específico dentro de un servidor web. La especificación Servlet 2.2 define la estructura de directorios para los ficheros de una aplicación web. El directorio raíz debería tener el nombre de la aplicación y define la raíz de documentos para la aplicación web. Todos los ficheros debajo de esta raíz pueden servirse al cliente excepto aquellos ficheros que están bajo los directorios especiales **META-INF** y **WEB-INF** en el directorio raíz. Todos los ficheros privados, al igual que los ficheros **.class** de los servlets, deberían almacenarse bajo el directorio **WEB-INF**.

Durante la etapa de desarrollo de una aplicación web se emplea la estructura de directorios, a pesar de que luego en la etapa de producción, toda la estructura de la aplicación se empaqueta en un archivo **.war**.

El código necesario para ejecutar correctamente una aplicación web se encuentra distribuido en una estructura de directorios, agrupándose ficheros según su funcionalidad. Un ejemplo de la estructura de carpetas de una aplicación web puede ser el siguiente:

```
/index.jsp
/WebContent/jsp/welcome.jsp
/WebContent/css/estilo.css
/WebContent/js/utils.js
/WebContent/img/welcome.jpg
/WEB-INF/web.xml
/WEB-INF/struts-config.xml
/WEB-INF/lib/struts.jar
/WEB-INF/src/com/empresa/proyecto/action/welcomeAction.java
/WEB-INF/classes/com/empresa/proyecto/action/welcomeAction.class
/WEB-INF/classes/com/empresa/proyecto/action/welcomeAction.class
```

De forma genérica podríamos decir que una aplicación web se estructura en tres capas:

1. Navegador web.
2. Tecnología web dinámica (PHP, Java Servlets, ASP, etc.)
3. Base de datos encargada de almacenar de forma permanente y actualizada la información que la aplicación web necesita.

Archivos WAR

Su nombre procede de Web Application Archive (Archivo de Aplicación Web); permiten empaquetar en una sola unidad aplicaciones web de Java completas, es decir que su contenido sea:

- Servlets y JSP.
- Contenido estático: HTML, imágenes, etc.
- Otros recursos web.

Aportan como ventaja, la simplificación del despliegue de aplicaciones web, debido a que su instalación es sencilla y solamente es necesario un fichero para cada servidor en un cluster, además de incrementar la seguridad ya que no permite el acceso entre aplicaciones web distintas.

Su estructura es la siguiente:

- **/:** En la carpeta raíz del proyecto se almacenan elementos empleados en los sitios web: documentos html, hojas de estilo y los elementos JSP (*.html, *.jsp, *.css).
- **/WEB-INF/:** Aquí se encuentran los elementos de configuración del archivo **.war** como pueden ser: la página de inicio, la ubicación de los servlets, parámetros adicionales para otros componentes. El más importante de éstos es el archivo **web.xml**.
- **/WEB-INF/classes/:** Contiene las clases Java empleadas en el archivo **.war** y, normalmente, en esta carpeta se encuentran los servlets.
- **/WEB-INF/lib/:** Contiene los archivos JAR utilizados por la aplicación y que normalmente son las clases empleadas para conectarse con la base de datos o las empleadas por librerías de JSP.

Para generar archivos **.war** se pueden emplear diversas herramientas desde entorno IDE. Por ejemplo, encontramos: **NetBeans** y **Eclipse**, ambos Open-Source y también **Jbuilder** de Borland, **Jdeveloper** de Oracle; otro modo de construir archivos **.war** es mediante **Ant**. Se trata de una herramienta Open-Source que facilita la construcción de aplicaciones en Java. No es considerado un IDE pero para los que conocen el entorno Linux, es considerado el **make** de Java.

Servidor de aplicaciones Tomcat

Tomcat es el servidor web (incluye el servidor Apache) y de aplicaciones del proyecto Yakarta, con lo cual, gestiona las solicitudes y respuestas HTTP y, además, es servidor de aplicaciones o contenedor de Servlets y JSP.

Incluye el compilador Jasper, que compila JSP convirtiéndolas en servlets.

Tomcat es un contenedor de servlets con un entorno JSP. Un contenedor de servlets es un shell de ejecución que maneja e invoca servlets por cuenta del usuario. Podemos dividir los contenedores de servlets en:

1. Contenedores de servlets **stand-alone** (independientes): Estos son una parte integral del servidor web. Este es el caso en el que se usa un servidor web basado en Java, por ejemplo, el contenedor de servlets es parte de JavaWebServer (actualmente sustituido por **iPlanet**). Por defecto Tomcat trabaja en este modo, sin embargo, la mayoría de los servidores no están basados en Java.
2. Contenedores de servlets **dentro-de-proceso**: El contenedor servlets es una combinación de un plugin para el servidor web y una implementación de contenedor Java. El plugin del servidor web abre una JVM (Máquina Virtual Java) dentro del espacio de direcciones del servidor web y permite que el contenedor Java se ejecute en él. En el caso de que una petición debiera ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando JNI). Un contenedor de este tipo es adecuado para servidores multi-thread de un sólo proceso y proporciona un buen rendimiento pero está limitado en escalabilidad.
3. Contenedores de servlets **fuera-de-proceso**: El contenedor servlets es una combinación de un plugin para el servidor web y una implementación de contenedor Java que se ejecuta en una JVM fuera del servidor web. El plugin del servidor web y el JVM del contenedor Java se comunican usando algún mecanismo IPC (normalmente sockets TCP/IP). Si una cierta petición tuviese que ejecutar un servlets, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando IPCs). El tiempo de respuesta en este tipo de contenedores no es tan bueno como el anterior, pero obtiene mejores rendimientos en otras cosas (escalabilidad, estabilidad, etc.).

Tomcat puede utilizarse como un contenedor solitario (principalmente para desarrollo y depuración) o como plugin para un servidor web existente (actualmente soporta los servidores Apache, IIS). Esto significa que siempre que desplaguemos Tomcat tendremos que decidir cómo usarlo y, si seleccionamos las opciones 2 o 3, también necesitaremos instalar un adaptador de servidor web.

Instalación y configuración

En primer lugar, destacar que para instalar cualquier versión de Tomcat es necesario tener instalado JDK (Kit de desarrollo de Java), ya que el objetivo es que las peticiones a Apache se redirijan a Tomcat empleando un conector proporcionado por Java en este caso.

Empezamos actualizando el sistema:

```
$ sudo apt update
```

Instalamos el siguiente paquete por ser el que más se adapta a nuestras necesidades:

```
$ sudo apt install default-jdk
```

Crearemos un grupo **tomcat**, y un usuario **tomcat** miembro del grupo anterior, con un directorio de inicio **/opt/tomcat** (donde instalaremos Tomcat) y un shell de **/bin/false** (de modo que nadie pueda iniciar sesión en la cuenta):

```
$ sudo groupadd tomcat
$ sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

Ahora descargaremos e instalaremos Tomcat. Para descargar la última versión de Tomcat 9.0.x, iremos a su [página de descargas](https://tomcat.apache.org/download-90.cgi). En este ejemplo vamos a usarla última versión **9.0.35**, copiamos el enlace del archivo **.tar.gz**.

Nos movemos al directorio **/tmp** y descargamos el archivo: (si no disponemos de **curl**, lo instalamos con **sudo apt install curl**)

```
$ cd /tmp
$ curl -O https://apache.brunneis.com/tomcat/tomcat-9/v9.0.35/bin/apache-tomcat-9.0.35.tar.gz
```

Una vez que se complete la descarga, extraeremos el archivo comprimido en el directorio **/opt/tomcat**:

```
$ sudo mkdir /opt/tomcat
$ sudo tar xzvf apache-tomcat-*tar.gz -C /opt/tomcat --strip-components=1
```

Nos cambiamos al directorio en el que desempaquetamos la instalación de Tomcat y le damos propiedad sobre todo el directorio de instalación al grupo **tomcat**:

```
$ cd /opt/tomcat
$ sudo chgrp -R tomcat /opt/tomcat
```

A continuación, le proporcionamos al grupo **tomcat** acceso de lectura al directorio **conf** y a todos sus contenidos, y acceso de ejecución al directorio, además nos aseguramos de que el usuario **tomcat** sea el propietario de los directorios **webapps**, **work**, **temp** y **logs**:

```
$ sudo chmod -R g+r conf
$ sudo chmod g+x conf
$ sudo chown -R tomcat webapps/ work/ temp/ logs/
```

Ahora tenemos que crear el archivo **tomcat.service** en el directorio **/etc/systemd/system**:

```
$ sudo nano /etc/systemd/system/tomcat.service
```

Y escribiremos lo siguiente:

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:
+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -
Djava.security.egd=file:/dev/./urandom'
```

```
ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

En la variable de entorno **JAVA_HOME** tendremos que poner la ruta que tengamos en nuestro equipo.

Por último, recargamos el demon **systemd** para que reciba información sobre nuestro archivo de servicio e iniciamos el servicio Tomcat:

```
$ sudo systemctl daemon-reload
$ sudo systemctl start tomcat
```

Para comprobar que el servicio está iniciado sin errores podemos escribir:

```
$ sudo systemctl status tomcat
```

Y nos mostrará si el servidor está arrancado, desde cuando, la versión, etc.

En Tomcat, la gestión del servicio se realiza a través del script incluido llamado **CATALINA**, al que le podemos proporcionar los parámetros **start** y **stop**, con lo que arrancaríamos o pararíamos el servicio manualmente.

Para comprobar que nuestro servidor está ya escuchando, introducimos en un navegador la URL: **http://localhost:8080** o **http://127.0.0.1:8080**, y éste debería mostrar la página de inicio de Tomcat.

Inicio

Tomcat va a estar escuchando en el puerto **8080** y va a tener su propio directorio de trabajo. La misión de **apache2** va a ser interceptar todas las peticiones en el puerto **80** y derivar las que considere necesarias a Tomcat; de este modo observamos la ventaja de la escalabilidad, ya que apache, al funcionar como proxy, puede tener una batería de tomcats a los que balancear las conexiones, haciendo que, si nuestras necesidades crecen, nuestras máquinas puedan ampliarse en número siendo completamente transparente para los usuarios.

Apache por defecto busca los ficheros en **/var/www/html**, Tomcat trabaja sobre la carpeta **/opt/tomcat/webapps**. La petición de una URL se puede gestionar, parte por apache y parte por Tomcat, por lo que vamos a cambiar la carpeta por defecto de trabajo para unificarlo. Para ello editamos el fichero **/opt/tomcat/conf/conf/server.xml**.

```
$ sudo nano /opt/tomcat/conf/server.xml
```

en donde encontraremos una línea con **"Host name="** y debería tener:

```
Host name="localhost" appBase="webapps"
```

Para usar la aplicación de administración web que viene con Tomcat, debemos añadir un inicio de sesión a nuestro servidor de Tomcat. Para ello editaremos el archivo **tomcat-users.xml**:

```
$ sudo nano /opt/tomcat/conf/tomcat-users.xml
```

Deberíamos añadir un usuario que pueda acceder a **manager-gui** y **admin-gui** (aplicaciones web que vienen con Tomcat). Podemos definir un usuario, por ejemplo como el que se muestra a continuación, entre las etiquetas **tomcat-users**, y poniendo nombres de usuarios y contraseñas seguras:

```
<tomcat-users . . .>
  <user username="admin" password="password" roles="manager-gui,admin-gui"/>
</tomcat-users>
```

Si en cualquier momento deseásemos parar o arrancar el servidor, podríamos emplear los siguientes comandos respectivamente:

```
$ sudo systemctl stop tomcat
$ sudo systemctl start tomcat
```

O bien:

```
$ sudo service tomcat stop
$ sudo service tomcat start
```


Despliegue de aplicaciones en Tomcat

Desplegar un servlet consiste en situar una serie de archivos en un contenedor web para que los clientes puedan acceder a su funcionalidad; una aplicación web es un conjunto de servlets, páginas HTML, JSP, clases y otros recursos que se pueden empaquetar de una forma determinada.

Una aplicación web puede ser desplegada en diferentes servidores web manteniendo su funcionalidad y sin ningún tipo de modificación en su código debido a la especificación servlet 2.2.

Las aplicaciones web deben organizarse según la siguiente estructura de directorios:

- Directorio principal (raíz): Contendrá los ficheros estáticos (HTML, imágenes, etc...) y JSPs.
 - Carpeta **WEB-INF**: contiene el fichero "*web.xml*" (descriptor de la aplicación), encargado de configurar la aplicación.
 - Subcarpeta **classes**: contiene los ficheros compilados (servlets, beans).
 - Subcarpeta **lib**: librerías adicionales.
 - Resto de carpetas para ficheros estáticos.

Una aplicación web puede ser desplegada empleando uno de los siguientes métodos:

- Por medio de archivos WAR.
- Editando los archivos *web.xml* y *server.xml*, este método es el que se pasa a tratar a continuación.

Los directorios que forman una aplicación compilada suelen ser : *www*, *bin*, *src*, *tomcat*, *gwt-cache*.

La carpeta **www** contiene a su vez una carpeta, con el nombre y ruta del proyecto, que contiene los ficheros que forman la interfaz (HTML, js, css...). La carpeta **bin** contiene las clases de java de la aplicación.

Para desplegar la aplicación en Tomcat se deben realizar los siguientes pasos:

1. Copiar la carpeta contenida en *www* (con el nombre del proyecto) en el directorio *webapps* de Tomcat.
2. Renombrar la nueva carpeta así creada en Tomcat con un nombre más sencillo. Esa será la carpeta de la aplicación en Tomcat.
3. Crear, dentro de dicha carpeta, otra nueva, y darle el nombre **WEB-INF** (respetando las mayúsculas).
4. Crear, dentro de **WEB-INF**, otros dos subdirectorios, llamados **lib** y **classes**.
5. Copiar en *lib* todas las librerías (.jar) que necesite la aplicación para su funcionamiento.

6. Copiar el contenido de la carpeta bin de la aplicación en el subdirectorio **WEB-INF/classes** del Tomcat.
7. Crear en WEB-INF un fichero de texto llamado **web.xml**, con las rutas de los servlets utilizados en la aplicación.
8. Ya puede accederse a la aplicación en el servidor, el modo de hacerlo es poniendo en el navegador la ruta del fichero HTML de entrada, que estará ubicado en la carpeta de la aplicación en Tomcat.

Vamos a partir de una máquina con el sistema operativo Ubuntu 20.04 la cual tenemos el servidor Tomcat corriendo para mostrar el proceso creación y despliegue de aplicaciones. Debido a que pretendemos montar una plataforma LAMP, por sus ventajas derivadas de las características del software libre, instalaremos también los siguientes componentes: MySQL y PHP.

Destacar que, para instalar cualquier versión de Tomcat es necesario tener instalado JDK (Kit de desarrollo de Java), ya que el objetivo es que las peticiones a Apache se redirijan a Tomcat empleando un conector proporcionado por Java en este caso.

Creación de una aplicación web

El servidor de aplicaciones Tomcat cuenta con una serie de ejemplos, tanto de servlets como de JSP, que sirven de ayuda para aprender a realizar las tareas creación y despliegue de aplicaciones web.

Es muy interesante crear dos variables de entorno: **JAVA_HOME** que indique la ubicación de los archivos binarios de Java y **CATALINA_HOME** que apunta a la ubicación de los scripts de Tomcat, para ello podemos añadir el siguiente código al archivo **/etc/profile**.

```
CATALINA_HOME=/opt/tomcat
JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
PATH=$PATH:$JAVA_HOME/bin:$CATALINA_HOME
export PATH JAVA_HOME CATALINA_HOME
```

Actualizamos las variables de entorno mediante el comando:

```
source /etc/profile
```

El lenguaje Javascript se ejecuta del lado del cliente, es un lenguaje interpretado de scripting que no permite acceder a información local del cliente ni puede conectarse a otros equipos de red.

En primer lugar crearemos una carpeta con el nombre que nos interese para identificar la aplicación, en este ejemplo hemos optado por **Aplic_Web** una estructura como la de la siguiente imagen:

La aplicación que pretendemos desarrollar contiene un archivo al que llamaremos **index.jsp** muy sencillo con el siguiente contenido:

```
<html>

<head>
  <title>C.F. DESARROLLO DE APLICACIONES WEB</title>
  <script language="Javascript">
    function popup() { alert("U.T. 3: CONFIGURACION Y ADMINISTRACION DE
SERVIDORES DE APLICACIONES"); }
  </script>
</head>

<body>
  <h1 align=center>DESPLIEGUE DE APLICACIONES WEB</h1>
  <div align=center>
    <form><input type="button" value="UNIDAD 3" onclick="popup()"></form>
  </div>
</body>

</html>
```

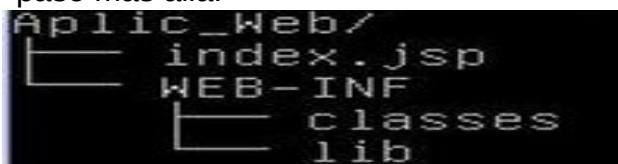
para acabar, solamente nos quedaría hacer una copia de la carpeta de nuestra aplicación en **\$CATALINA_HOME/webapps** y si, posteriormente desde un navegador, accedemos en local a **http://127.0.0.1:8080/Aplic_Web** tendríamos la aplicación funcionando.

Despliegue de una aplicación web

Uno de los objetivos que se persigue en el momento de desarrollar aplicaciones web, es que éstas puedan ser desplegadas en diferentes servidores web, manteniendo su funcionalidad y sin ninguna modificación de código.

Los WARs simplemente son archivos Java de una aplicación web con una extensión diferente para diferenciarlos de los comúnmente usados JARs.

Antes de la especificación Servlet 2.2, era bastante diferente desplegar servlets entre diferentes contenedores de servlets, anteriormente también llamados motores servlet. La especificación 2.2 estandarizó el despliegue entre contenedores, llevando así la portabilidad del código Java un paso más allá.



El método más sencillo para desplegar una aplicación, que sobre todo se utiliza durante la

etapa de desarrollo de la misma, es el realizado en el punto anterior, es decir, copiar la carpeta correspondiente a nuestra aplicación en la carpeta \$CATALINA_HOME/webapps, teniendo en cuenta que la variable \$CATALINA_HOME es la ruta de los scripts que emplea Tomcat.

Siguiendo con la aplicación desarrollada en el punto anterior (Aplic_Web), vamos a crear un fichero descriptor del despliegue web.xml que es el encargado de describir las características de despliegue de la aplicación.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  version="2.5">
  <display-name>Descriptor Aplicacion Aplic_Web</display-name>
  <description>
    Mi primer descriptor web.xml.
  </description>
</web-app>
```

Este archivo lo situaremos en la carpeta WEB-INF perteneciente a la aplicación en desarrollo, de forma que la estructura de la carpeta resultante sería el mostrado en esta imagen:



```
Aplic_Web/
├── index.jsp
└── WEB-INF
    ├── classes
    ├── lib
    └── web.xml

3 directories, 2 files
```

Una vez consideramos terminada nuestra aplicación web podremos generar el archivo .WAR perteneciente a la aplicación, para ello podemos aplicar los siguientes comandos:

javac -d WEB-INF/classes *.java este comando tiene como finalidad la compilación de las clases Java de nuestra aplicación.

jar cvf Aplic_Web.war * para crear el archivo .WAR.

Una vez hecho lo anterior podríamos acceder vía web a: <http://127.0.0.1:8080> y, en el apartado "Administration", accedemos a la opción "Tomcat Manager" y desde la ventana resultante tenemos las opciones que aparecen en la siguiente imagen para desplegar el archivo .WAR:

Desplegar

Desplegar directorio o archivo WAR localizado en servidor

Trayectoria de Contexto (opcional):

URL de archivo de Configuración XML:

URL de WAR o Directorio:

Desplegar

Archivo WAR a desplegar

Seleccione archivo WAR a cargar Examinar...

Desplegar