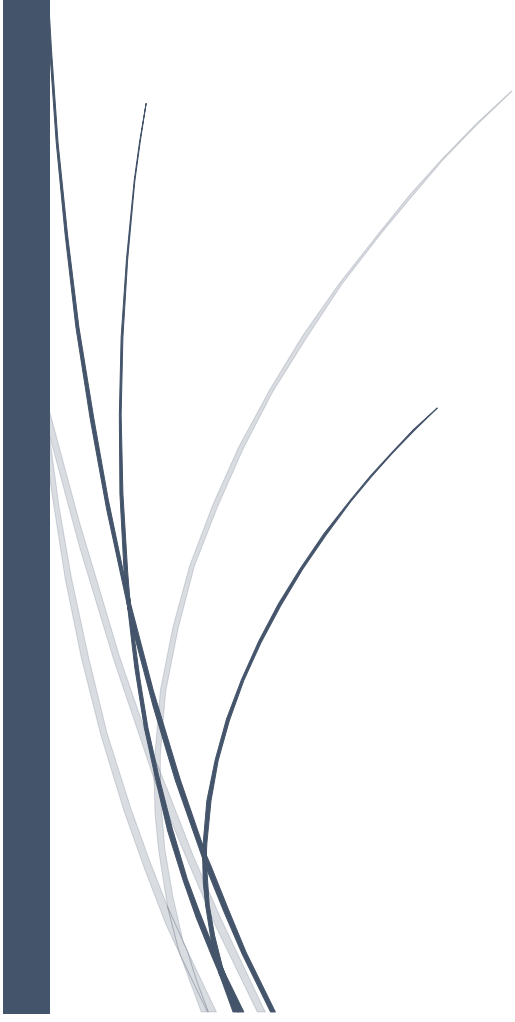
DESPLIEGUE DE APLICACIONES WEB

INSTALACIÓN DE APACHE TOMCAT Y DE MAVEN



ÍNDICE

INTRODUCCIÓN

En esta práctica vamos a realizar la instalación del servidor de aplicaciones Apache Tomcat.

Se va a instalar la versión 9 del servidor de aplicaciones (en lugar de la versión más actual, la 11) porque de esta forma vamos a poder utilizar aplicaciones Java escritas en Java 8 (mientras que Tomcat 9 da soporte a Java 8 en adelante, Tomcat 11 da soporte de Java 21 en adelante).

De esta forma aprenderemos a utilizar el servidor de aplicaciones con la versión Java que hoy en día sigue siendo la más usada en proyectos reales. Esta versión de Java 8 perdió su soporte para uso comercial en marzo de 2020 pero el soporte para uso no comercial continuará hasta 2030.

A. ¿QUÉ ES TOMCAT?

Hasta ahora en las prácticas hemos tenido un sistema cliente-servidor donde este último sólo se encargaba de tener:

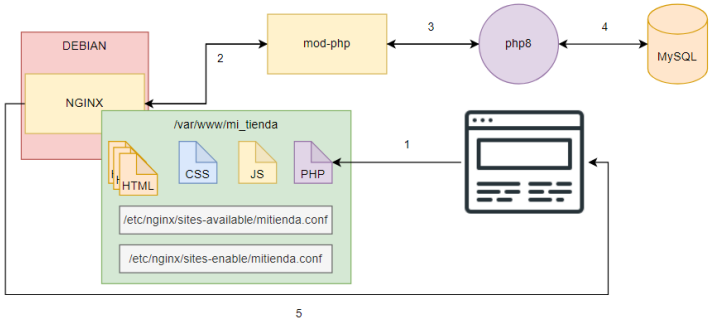
- Un servidor web o servidor HTTP: Nginx, Apache, etc.
- Un conjunto de archivos destinados a que el cliente HTTP (Firefox, Edge, Brave…) los interprete y ejecute: HTML, CSS, JavaScript…

Ahora además de esto vamos a tener programas que van a estar escritos en un lenguaje como Java, PHP, Python… y que se van a conectar a una base de datos como MySQL, PostgreSQL, SQLite…

Apache Tomcat es un servidor HTTP compatible con Java capaz de ejecutar programas especiales de Java llamados Servlets y Java Server Pages (JSP).

B. ¿CÓMO FUNCIONA UN SERVIDOR DE APLICACIONES?

Vamos a pensar en PHP y a ver el siguiente gráfico:



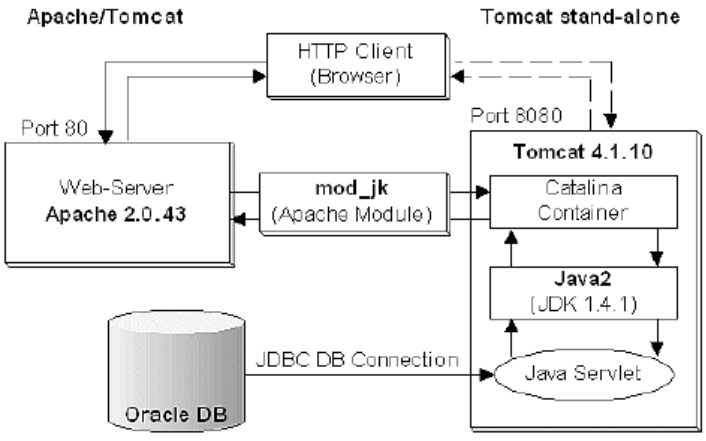
Tenemos un servidor Debian con un servidor web NGINX instalado en su interior, dentro de este servidor tenemos un directorio /var/www/mi_tienda con varios archivos HTML, CSS, JS y un archivo PHP así como los archivos de configuración en los directorios correspondientes.

Un cliente web (1) realiza una petición GET sobre el archivo .php (por ejemplo, articulos.php) y esta petición llega al servidor web pero NGINX es incapaz de interpretar este archivo ya que está escrito en un lenguaje que no conoce, lo que ocurre a continuación (2) es que se pasa este archivo al mod-php que es una extensión que permite comunicar (3) el servidor web con el intérprete php (el archivo php.exe), este intérprete se va a encarga de interpretar el archivo .php y generar el archivo correspondiente (por ejemplo, un documento .html) así como conectarse a la base de datos (4) en caso de ser necesario. Una vez que el archivo ya interpretado ha sido devuelto al servidor web, este se lo sirve al cliente web (5).

Por tanto, de lo que se va a encargar un servidor de aplicaciones es de interpretar los archivos que estén en lenguajes de programación y que generan elementos dinámicos, es decir, archivos escritos en un lenguaje de programación como Java, PHP, Python, etc.

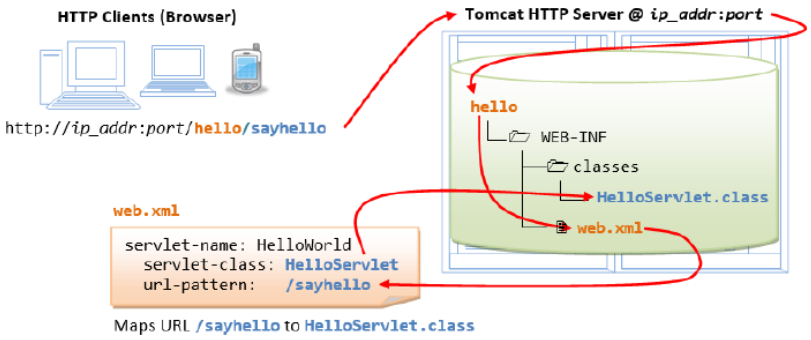
C. ¿CÓMO FUNCIONA TOMCAT?

Tomcat va a funcionar de una manera similar:

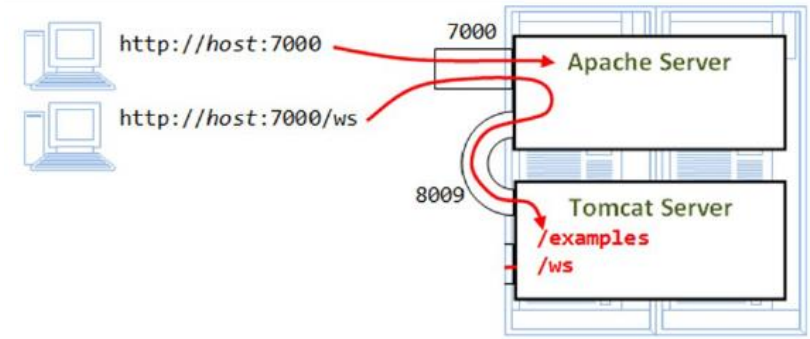


El cliente web realiza una petición al servidor sobre una aplicación web escrita en Java, esto hará que el servidor se lo pasé al mod_jk y este se lo pasará a Tomcat el cual a través de Catalania interpretará los servlets de la aplicación web, una vez hecho esto, el resultado será devuelto al servidor y este se lo enviará al cliente.

Aun así, Tomcat no requiere de un servidor web para funcionar y, como vemos en la siguiente imagen, podremos tener un sistema donde el cliente se conecta directamente a Tomcat funcionando como servidor:



También tenemos la opción de configurar un trabajo conjunto de Tomcat y el servidor web, por ejemplo, Apache:



En este caso el servidor web necesita de un módulo para poder conectarse con Tomcat.

Por último, tenemos la opción de NGINX en la que este va a trabajar como proxy inverso:

Una vez instalado, crea un nuevo archivo de configuración virtual host para Tomcat.

```
# nano /etc/nginx/sites-available/tomcat.conf
```

Añade las siguientes líneas:

```
upstream tomcat {
  server 127.0.0.1:8080 weight=100 max_fails=5 fail_timeout=5;
}

server {
  listen          80;
  server_name     ejemplo.com;

   location / {
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://tomcat/;
  }
}
```

Guarda y cierra el archivo. Después, habilita el archivo virtual host del servidor web de Nginx para aplicar la configuración:

```
# ln -s /etc/nginx/sites-available/tomcat.conf /etc/nginx/sites-enabled/
systemctl restart nginx
```

¡Enhorabuena! Ya puedes acceder al servidor Tomcat a través de la URL http://ejemplo.com sin especificar el puerto Tomcat 8080.

D. ¿QUÉ ES CATALINA?

Tomcat es conocido como un contenedor de servlets y se compone de una serie de elementos Jasper para JSP, Coyote que funciona como conector HTTP y otros módulos para poder trabajar con las tecnologías de la Enterprise Edition, entre ellos se encuentra Catalina.

Catalina es el verdadero contenedor de servlets de Apache Tomcat ejecutando aplicaciones web basadas en Java.

Es responsable de gestionar el ciclo de vida de los servlets y JSP, manejar solicitudes HTTP y gestionar la comunicación entre la aplicación web y el servidor web.

Es decir, como se ha visto en los gráficos de arriba, Catalina va a ser el módulo de Tomcat que se va a encargar de interpretar los servlets contenidos en el directorio WEB-INF de nuestro proyecto web.

E. MAVEN

Maven es una herramienta de código abierto que simplifica los procesos de compresión y creación de ejecutables a partir del código fuente. Antes había que conocer qué librerías se compilaban y cuáles no, qué dependencias existían, etc. Ahora escribiendo "mvn install" se realiza todo el proceso. Además de compilar, Maven como se puede ver en el siguiente esquema con las distintas fases de un proyecto Java, también gestiona un proyecto completo desde la comprobación de código hasta el despliegue:

Para ello, en Maven se definen tres ciclos de build del software con una serie de etapas diferenciadas. Por ejemplo, el ciclo por defecto tiene las etapas de:

- Validación (validate): Validar que el proyecto es correcto.
- Compilación (compile).
- Test (test): Probar el código fuente usando un framework de pruebas unitarias.
- Empaquetar (package): Empaquetar el código compilado y transformarlo en algún formato tipo .jar o .war.

- Pruebas de integración (integration-test): Procesar y desplegar el código en algún entorno donde se puedan ejecutar las pruebas de integración.
- Verificar que el código empaquetado es válido y cumple los criterios de calidad (verify).
- Instalar el código empaquetado en el repositorio local de Maven, para usarlo como dependencia de otros proyectos (install).
- Desplegar el código a un entorno (deploy).

Para poder llevar a cabo alguna de estas fases en nuestro código, tan solo tendremos que ejecutar mvn y el parámetro indicado entre paréntesis en la lista anterior.

Además, van en cadena, es decir, si empaquetamos el código (package), Maven ejecutará desde la fase de validación (validate) a empaquetación (package). Así de simple

En el fichero POM de Maven es donde se indican entre otra información las librerías necesarias. A continuación, un ejemplo del contenido en un fichero POM:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.urjc.libromaven.multiproyecto</groupId>
        <artifactId>multiproyecto</artifactId>
        <version>1.0</version>
    </parent>

    <artifactId>simple-weather</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>simple-weather</name>
    <url>http://maven.apache.org</url>

    <licenses>
        <license>
            <name>Apache License, Version 2.0</name>
            <url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>
            <distribution>repo</distribution>
            <comments>A business-friendly OSS license</comments>
        </license>
    </licenses>

    <dependencies>
        <dependency>
            <groupId>dom4J</groupId>
            <artifactId>dom4J</artifactId>
            <version>1.6.1</version>
        </dependency>
        <dependency>
```

Como puedes ver, en el mismo archivo también se indican aspectos como el nombre del proyecto, licencias, quién desarrolla el código, el sitio web, el repositorio de control de versiones, un identificador único del proyecto etc. Digamos que Maven aporta una semántica común al proceso de build y desarrollo del software. Incluso, establece una estructura común de directorios para todos los proyectos. Por ejemplo:

- El código estará en ${raíz del proyecto}/src/main/java.
- Los recursos en ${raíz del proyecto}/src/main/resources.
- Los tests están en ${raíz del proyecto}/src/test etc.

INSTALACIÓN DE TOMCAT

La instalación del software de aplicaciones se va a llevar a cabo sobre una máquina virtual similar a las que hemos usado en prácticas anteriores, en concreto una que tenga instalado el sistema operativo Debian 12 Bookworm.

Esta instalación se puede realizar tanto por el administrador de paquetes apt como de forma manual, en esta práctica vamos a aprender a hacerlo de forma manual debido a que las versiones de JDK y Tomcat que utilizamos ya no se encuentran en los repositorios de Linux.

A. <u>CONFIGURACIONES PREVIAS</u>

Estos pasos previos son necesarios siempre, se haga una instalación manual o por apt.

Lo primero que necesitamos es abrir el puerto 8080 que es el puerto por defecto de Tomcat, para ello podemos usar el servicio ufw. En caso de no tenerlo instalado podemos hacerlo con el siguiente comando:

sudo apt-get install ufw

```
albertom-servidor@debian-deaw:~$ sudo apt-get install ufw
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Paquetes sugeridos:
  rsyslog
Se instalarán los siguientes paquetes NUEVOS:
  ufw
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 168 kB de archivos.
Se utilizarán 878 kB de espacio de disco adicional después de esta operación.
Des:1 http://deb.debian.org/debian bookworm/main amd64 ufw all 0.36.2-1 [168 kB]
Descargados 168 kB en 0s (952 kB/s)
Preconfigurando paquetes ...
Seleccionando el paquete ufw previamente no seleccionado.
```

Podemos comprobar si se ha instalado de forma correcta haciendo un start y un status del servicio:

```
albertom-servidor@debian-deaw:~$ sudo systemctl start ufw.service
albertom-servidor@debian-deaw:~$ sudo systemctl status ufw.service
● ufw.service - Uncomplicated firewall
     Loaded: loaded (/lib/systemd/system/ufw.service; enabled; preset: enabled)
     Active: active (exited) since Fri 2023-11-17 15:17:12 CET; 10s ago
       Docs: man:ufw(8)
    Process: 3710 ExecStart=/lib/ufw/ufw-init start quiet (code=exited, status=0/SUCCESS)
   Main PID: 3710 (code=exited, status=0/SUCCESS)
        CPU: 2ms

nov 17 15:17:12 debian-deaw systemd[1]: Starting ufw.service - Uncomplicated firewall...
nov 17 15:17:12 debian-deaw systemd[1]: Finished ufw.service - Uncomplicated firewall.
```

Vamos a abrir el puerto 8080 para ello usamos el comando:

sudo ufw allow 8080

```
albertom-servidor@debian-deaw:~$ sudo ufw allow 8080
Rules updated
Rules updated (v6)
```

En este caso como vamos a instalar una versión de la JDK que ya se encuentra obsoleta en la lista de repositorios, debemos descargar los siguientes paquetes desde la web oficial de Debian:

1. **java-common_0.75_all.deb**: Este archivo de instalación contiene un conjunto de archivos (scripts, configuraciones, archivos compartidos…) comunes relacionados con Java.
2. **openjdk-11-jre-headless_11.0.21+9-1_amd64.deb**: Contiene la version 11 del entorno de ejecución de Java (JRE). El término "headless" significa que esta instalación no incluye las bibliotecas gráficas ni las dependencias necesarias para aplicaciones con interfaz gráfica.
3. **openjdk-11-jre_11.0.21+9-1_amd64.deb**: Contiene la version 11 del entorno de ejecución de Java (JRE) con la parte gráfica.
4. **openjdk-11-jdk-headless_11.0.21+9-1_amd64.deb**: En este archive tenemos el instalador del kit de desarrollo de Java (JDK) en su versión 11 sin las bibliotecas gráficas.
5. **openjdk-11-jdk_11.0.21+9-1_amd64.deb**: Ídem al anterior, pero con las bibliotecas gráficas.

En nuestro caso tenemos todos los paquetes comprimidos en un mismo archivo .zip por lo que primero debemos descomprimirlo con el comando unzip:

sudo unzip /home/albertom-servidor/Descargas/paquetes_jdk_tomcat.zip -d /home/albertom-servidor/Descargas/instaladores

```
albertom-servidor@debian-deaw:~$ sudo unzip /home/albertom-servidor/Descargas/paquetes_jdk_tomcat.zip -d /home/albertom-servidor/Descargas/instaladores
Archive:  /home/albertom-servidor/Descargas/paquetes_jdk_tomcat.zip
  inflating: /home/albertom-servidor/Descargas/instaladores/1_java-common_0.75_all.deb
  inflating: /home/albertom-servidor/Descargas/instaladores/2_openjdk-11-jre-headless_11.0.21+9-1_amd64.deb
  inflating: /home/albertom-servidor/Descargas/instaladores/3_openjdk-11-jre_11.0.21+9-1_amd64.deb
  inflating: /home/albertom-servidor/Descargas/instaladores/4_openjdk-11-jdk-headless_11.0.21+9-1_amd64.deb
  inflating: /home/albertom-servidor/Descargas/instaladores/5_openjdk-11-jdk_11.0.21+9-1_amd64.deb
  inflating: /home/albertom-servidor/Descargas/instaladores/6_apache-tomcat-9.0.83.tar.gz
```

Y ahora debemos instalar paquete a paquete siguiendo el siguiente orden:

sudo dpkg -i /home/albertom-servidor/Descargas/instaladores/1_java-common_0.75_all.deb

```
albertom-servidor@debian-deaw:~$ sudo dpkg -i /home/albertom-servidor/Descargas/instaladores/1_java-common_0.75_all.deb
(Leyendo la base de datos ... 216156 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../1_java-common_0.75_all.deb ...
Desempaquetando java-common (0.75) sobre (0.74) ...
Configurando java-common (0.75) ...
Procesando disparadores para man-db (2.11.2-2) ...
```

A la hora de instalar el 2º paquete de dependencias tenemos que tener instalado el paquete ca-certificates-java que actúa como puerta de enlace entre el sistema operativo y las implementaciones de Java para gestionar los certificados de seguridad.

sudo apt-get install ca-certificates-java

```
albertom-servidor@debian-deaw:~$ sudo apt-get install ca-certificates-java
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  ca-certificates-java
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
```

Una vez hecho esto podemos instalar el siguiente paquete de dependencias:

sudo dpkg -i /home/albertom-servidor/Descargas/instaladores/2_openjdk-11-jre-headless_11.0.21+9-1_amd64.deb

```
albertom-servidor@debian-deaw:~$ sudo dpkg -i /home/albertom-servidor/Descargas/instaladores/2_openjdk-11-jre-headless_11.0.21+9-1_amd64.deb
Seleccionando el paquete openjdk-11-jre-headless:amd64 previamente no seleccionado.
(Leyendo la base de datos ... 216156 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../2_openjdk-11-jre-headless_11.0.21+9-1_amd64.deb ...
Desempaquetando openjdk-11-jre-headless:amd64 (11.0.21+9-1) ...
Configurando openjdk-11-jre-headless:amd64 (11.0.21+9-1) ...
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/jjs para proveer /usr/bin/jjs (jjs) en modo automático
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/rmid para proveer /usr/bin/rmid (rmid) en modo automático
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/pack200 para proveer /usr/bin/pack200 (pack200) en modo automático
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/unpack200 para proveer /usr/bin/unpack200 (unpack200) en modo automático
Procesando disparadores para ca-certificates-java (20230620~deb12u1) ...
done.
```

Para el siguiente paquete también necesitamos tener otro paquete instalado antes, el mailcap:

sudo apt-get install mailcap

```
albertom-servidor@debian-deaw:~$ sudo apt-get install mailcap
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
mailcap ya está en su versión más reciente (3.70+nmu1).
fijado mailcap como instalado manualmente.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
1 no instalados del todo o eliminados.
Se utilizarán 0 B de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Configurando mailcap (3.70+nmu1) ...
```

Una vez instalado dicho paquete podemos seguir adelante con la instalación del resto de dependencias:

sudo dpkg -i /home/albertom-servidor/Descargas/instaladores/3_openjdk-11-jre_11.0.21+9-1_amd64.deb

```
albertom-servidor@debian-deaw:~$ sudo dpkg -i /home/albertom-servidor/Descargas/instaladores/3_openjdk-11-jre_11.0.21+9-1_amd64.deb
Seleccionando el paquete openjdk-11-jre:amd64 previamente no seleccionado.
(Leyendo la base de datos ... 216489 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../3_openjdk-11-jre_11.0.21+9-1_amd64.deb ...
Desempaquetando openjdk-11-jre:amd64 (11.0.21+9-1) ...
Configurando openjdk-11-jre:amd64 (11.0.21+9-1) ...
Procesando disparadores para ca-certificates-java (20230620~deb12u1) ...
done.
Procesando disparadores para gnome-menus (3.36.0-1.1) ...
Procesando disparadores para desktop-file-utils (0.26-1) ...
Procesando disparadores para mailcap (3.70+nmu1) ...
Procesando disparadores para hicolor-icon-theme (0.17-2) ...
```

sudo dpkg -i /home/albertom-servidor/Descargas/instaladores/4_openjdk-11-jdk-headless_11.0.21+9-1_amd64.deb

```
albertom-servidor@debian-deaw:~$ sudo dpkg -i /home/albertom-servidor/Descargas/instaladores/4_openjdk-11-jdk-headless_11.0.21+9-1_amd64.deb
Seleccionando el paquete openjdk-11-jdk-headless:amd64 previamente no seleccionado.
(Leyendo la base de datos ... 216505 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../4_openjdk-11-jdk-headless_11.0.21+9-1_amd64.deb ...
Desempaquetando openjdk-11-jdk-headless:amd64 (11.0.21+9-1) ...
Configurando openjdk-11-jdk-headless:amd64 (11.0.21+9-1) ...
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/jar para proveer /usr/bin/jar (jar) en modo automático
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/jarsigner para proveer /usr/bin/jarsigner (jarsigner) en modo automático
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/javac para proveer /usr/bin/javac (javac) en modo automático
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/javadoc para proveer /usr/bin/javadoc (javadoc) en modo automático
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/javap para proveer /usr/bin/javap (javap) en modo automático
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/jcmd para proveer /usr/bin/jcmd (jcmd) en modo automático
```

sudo dpkg -i /home/albertom-servidor/Descargas/instaladores/5_openjdk-11-jdk_11.0.21+9-1_amd64.deb

```
albertom-servidor@debian-deaw:~$ sudo dpkg -i /home/albertom-servidor/Descargas/instaladores/5_openjdk-11-jdk_11.0.21+9-1_amd64.deb
Seleccionando el paquete openjdk-11-jdk:amd64 previamente no seleccionado.
(Leyendo la base de datos ... 216646 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../5_openjdk-11-jdk_11.0.21+9-1_amd64.deb ...
Desempaquetando openjdk-11-jdk:amd64 (11.0.21+9-1) ...
Configurando openjdk-11-jdk:amd64 (11.0.21+9-1) ...
update-alternatives: utilizando /usr/lib/jvm/java-11-openjdk-amd64/bin/jconsole para proveer /usr/bin/jconsole (jconsole) en modo automático
Procesando disparadores para ca-certificates-java (20230620~deb12u1) ...
done.
```

Cuando hayamos completado todas las instalaciones debemos comprobar la versión de Java que tenemos instalada en nuestro sistema:

java -version

```
albertom-servidor@debian-deaw:~$ java -version
openjdk version "11.0.21" 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Debian-1)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Debian-1, mixed mode, sharing)
```

B. INSTALACIÓN DE TOMCAT 9

En nuestro paquete de instalación ya teníamos el instalador de tomcat comprimido por lo que lo que tenemos que hacer ahora es descomprimirlo con el comando:

sudo tar -zxvf /home/albertom-servidor/Descargas/instaladores/6_apache-tomcat-9.0.83.tar.gz -C /home/albertom-servidor/Descargas/instaladores/

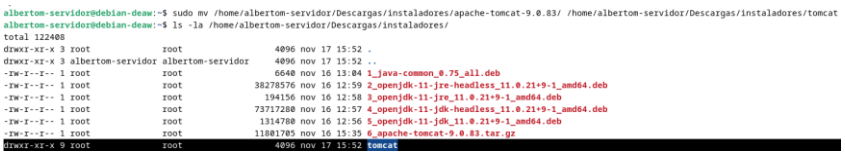
```
albertom-servidor@debian-deaw:~$ sudo tar -zxvf /home/albertom-servidor/Descargas/instaladores/6_apache-tomcat-9.0.83.tar.gz -C /home/albertom-servidor/Descargas/instaladores/
apache-tomcat-9.0.83/conf/
apache-tomcat-9.0.83/conf/catalina.policy
apache-tomcat-9.0.83/conf/catalina.properties
apache-tomcat-9.0.83/conf/context.xml
apache-tomcat-9.0.83/conf/jaspic-providers.xml
apache-tomcat-9.0.83/conf/jaspic-providers.xsd
apache-tomcat-9.0.83/conf/logging.properties
apache-tomcat-9.0.83/conf/server.xml
apache-tomcat-9.0.83/conf/tomcat-users.xml
apache-tomcat-9.0.83/conf/tomcat-users.xsd
apache-tomcat-9.0.83/conf/web.xml
apache-tomcat-9.0.83/bin/
```

Esto nos habrá creado una carpeta en dicho directorio donde se encuentran todos estos archivos:

```
albertom-servidor@debian-deaw:~$ sudo tree /home/albertom-servidor/Descargas/instaladores/apache-tomcat-9.0.83/
/home/albertom-servidor/Descargas/instaladores/apache-tomcat-9.0.83/
├── bin
│   ├── bootstrap.jar
│   ├── catalina.bat
│   ├── catalina.sh
│   ├── catalina-tasks.xml
│   ├── ciphers.bat
│   ├── ciphers.sh
│   ├── commons-daemon.jar
│   ├── commons-daemon-native.tar.gz
│   ├── configtest.bat
│   ├── configtest.sh
│   ├── daemon.sh
│   ├── digest.bat
│   ├── digest.sh
│   ├── makebase.bat
```

Para facilita el mencionar la carpeta en próximos comandos vamos a cambiarle el nombre a tomcat:

sudo mv /home/albertom-servidor/Descargas/instaladores/apache-tomcat-9.0.83/ /home/albertom-servidor/Descargas/instaladores/tomcat



Ahora debemos mover esta carpeta al directorio /opt

sudo mv /home/albertom-servidor/Descargas/instaladores/tomcat/ /opt

```
albertom-servidor@debian-deaw:~$ sudo mv /home/albertom-servidor/Descargas/instaladores/tomcat/ /opt
albertom-servidor@debian-deaw:~$ ls -la /opt/
total 16
drwxr-xr-x  4 root root 4096 nov 17 15:54 .
drwxr-xr-x 19 root root 4096 nov  1 11:12 ..
drwxr-xr-x  9 root root 4096 nov 17 15:52 tomcat
drwxr-xr-x  8 root root 4096 nov  1 11:29 VBoxGuestAdditions-7.0.10
```

Debemos crear un grupo tomcat y un usuario tomcat:

sudo groupadd tomcat

sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat

```
albertom-servidor@debian-deaw:~$ sudo groupadd tomcat
albertom-servidor@debian-deaw:~$ sudo useradd -s /bin/false -g tomcat -d /opt/tomcat/ tomcat
```

Este usuario debe ser el nuevo propietario del directorio que hemos creado en /opt con los documentos de tomcat:

sudo chown -R tomcat:tomcat /opt/tomcat

```
albertom-servidor@debian-deaw:~$ sudo chown -R tomcat:tomcat /opt/tomcat/
albertom-servidor@debian-deaw:~$ ls -la /opt/
total 16
drwxr-xr-x  4 root   root   4096 nov 17 15:54 .
drwxr-xr-x 19 root   root   4096 nov  1 11:12 ..
drwxr-xr-x  9 tomcat tomcat 4096 nov 17 15:52 tomcat
drwxr-xr-x  8 root   root   4096 nov  1 11:29 VBoxGuestAdditions-7.0.10
```

Ahora debemos modificar los permisos del directorio /opt/tomcat/bin para que todos los archivos .sh tengan el permiso de ejecución:

sudo sh -c 'chmod +x /opt/tomcat/bin/*.sh'

```
albertom-servidor@debian-deaw:~$ sudo sh -c 'chmod +x /opt/tomcat/bin/*.sh'
```

Lo siguiente que tenemos que hacer es crear un archivo de servicio para tomcat en el directorio /etc/systemd/system con el nombre de tomcat.service:

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

```
albertom-servidor@debian-deaw:~$ sudo nano /etc/systemd/system/tomcat.service
```

Y en el fichero debemos tener el siguiente contenido:

```
  GNU nano 7.2                    /etc/systemd/system/tomcat.service *
[Unit]
Description=Tomcat webs servlet container
After=network.target

[Service]
Type=forking

User=tomcat
Group=tomcat
RestartSec=10
Restart=always
Environment="JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64"
Environment="JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom"

Environment="CATALINA_BASE=/opt/tomcat"
Environment="CATALINA_HOME=/opt/tomcat"
Environment="CATALINA_PID=/opt/tomcat/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"

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

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

Salimos y guardamos los cambios.

Vamos a lanzar el servicio, pero antes debemos listar las diferentes versiones de Java instaladas en tu sistema y sus respectivos nombres para poder seleccionar la versión que deseas utilizar como predeterminada.

sudo update-java-alternatives -l

```
albertom-servidor@debian-deaw:~$ sudo update-java-alternatives -l
java-1.11.0-openjdk-amd64      1111       /usr/lib/jvm/java-1.11.0-openjdk-amd64
```

También debemos recargar los archivos de configuración del sistema del gestor de servicios systemd:

sudo systemctl daemon-reload

```
albertom-servidor@debian-deaw:~$ sudo systemctl daemon-reload
```

Una vez hecho esto debemos iniciar tomcat, establecer que se inicie con la carga del sistema y comprobar el estado:

sudo systemctl start tomcat.service

sudo systemctl enable tomcat.service

sudo systemctl status tomcat.service

```
albertom-servidor@debian-deaw:~$ sudo systemctl start tomcat.service
albertom-servidor@debian-deaw:~$ sudo systemctl enable tomcat.service
Created symlink /etc/systemd/system/multi-user.target.wants/tomcat.service → /etc/systemd/system/tomcat.service.
albertom-servidor@debian-deaw:~$ sudo systemctl status tomcat.service
● tomcat.service - Tomcat webs servlet container
     Loaded: loaded (/etc/systemd/system/tomcat.service; enabled; preset: enabled)
     Active: active (running) since Fri 2023-11-17 16:11:19 CET; 18s ago
   Main PID: 4849 (java)
      Tasks: 29 (limit: 2284)
     Memory: 145.9M
        CPU: 3.791s
     CGroup: /system.slice/tomcat.service
             └─4849 /usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -Djava.util.logging.config.file=/op
```

Ahora debemos crear a nuestro usuario de Tomcat, para ello debemos modificar el archivo /tomcat/conf/tomcat-users.xml:

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

```
albertom-servidor@debian-deaw:~$ sudo nano /opt/tomcat/conf/tomcat-users.xml
```

Al final del fichero y antes del cierre </tomcat-users> debemos incluir una serie de roles y un usuario con nombre, contraseña y los roles que tiene asignados:

```
-->
<role rolename="admin"/>
<role rolename="admin-gui"/>
<role rolename="manager"/>
<role rolename="manager-gui"/>

<user username="h2s" password="pwd" roles="admin,admin-gui,manager,manager-gui"/>
```

Ahora debemos editar el archivo de configuración /tomcat/webapps/manager/META-INF/context.xml:

sudo nano /opt/tomcat/webapps/manager/META-INF/context.xml

```
albertom-servidor@debian-deaw:~$ sudo nano /opt/tomcat/webapps/manager/META-INF/context.xml
```

Debemos comentar las siguientes líneas de código:

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
       allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
```

Para que queden así:

```
<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"
       allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
```

Lo que hemos hecho en este paso es comentar un elemento valve que es un componente que se inserta en el ciclo de procesamientos de la petición, de esta forma se pueden filtrar las peticiones sospechosas de ser ataques o filtrar que no se acepten peticiones salvo que vengan de un determinado host.

Esto lo podemos hacer tanto a nivel global (dentro del engine) como a un host en concreto (dentro de host) o para una única aplicación (dentro de context).

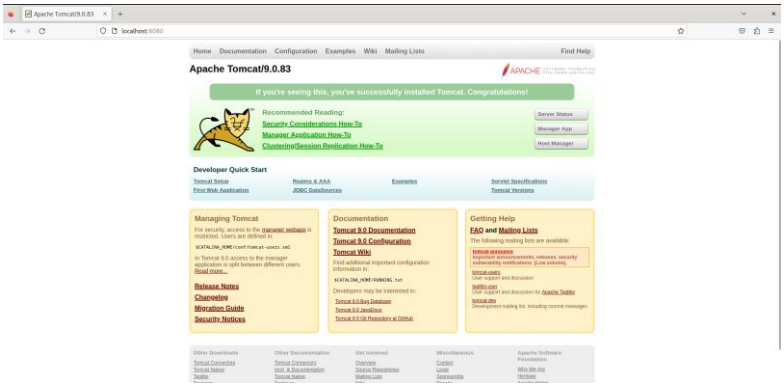
Tomcat tiene varios valves predefinidos (que en el fondo son clases Java) aunque cada usuario puede escribir los suyos propios.

Una vez hecho esto, reiniciamos el servicio:

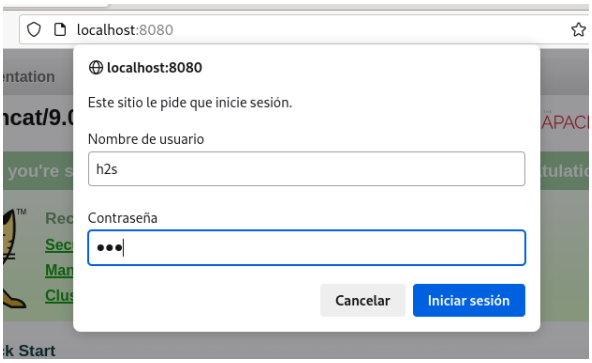
sudo systemctl restart tomcat.service

```
albertom-servidor@debian-deaw:~$ sudo systemctl restart tomcat.service
albertom-servidor@debian-deaw:~$ sudo systemctl status tomcat.service
● tomcat.service - Tomcat webs servlet container
     Loaded: loaded (/etc/systemd/system/tomcat.service; enabled; preset: enabled)
     Active: active (running) since Fri 2023-11-17 16:17:59 CET; 10s ago
    Process: 5007 ExecStart=/opt/tomcat/bin/startup.sh (code=exited, status=0/SUCCESS)
   Main PID: 5015 (java)
      Tasks: 29 (limit: 2284)
     Memory: 148.2M
        CPU: 3.134s
     CGroup: /system.slice/tomcat.service
             └─5015 /usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -Djava.util.logging
```

Para comprobar que todo ha sido configurado de forma correcta accedemos a la web localhost:8080 y deberíamos ver una página similar a esta:



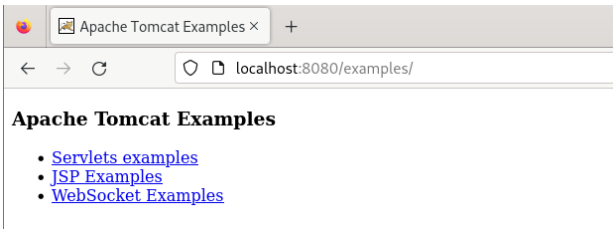
Podemos acceder al panel de control debemos acceder a “Manager App”, lo cual hará que se nos solicite un usuario y contraseña. Estos datos son los que utilizamos al crear el usuario en /opt/tomcat/conf/tomcat-users.xml



Dentro de este panel encontramos en la parte superior una tabla con las aplicaciones que están actualmente desplegadas:

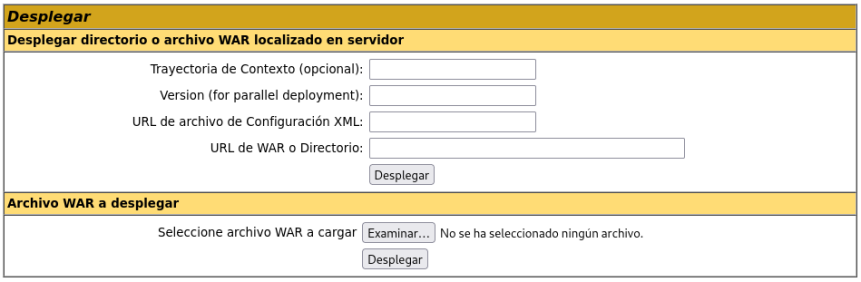
Aplicaciones					
Ruta	Versión	Nombre a Mostrar	Ejecutándose	Sesiones	Comandos
/	Ninguno especificado	Welcome to Tomcat	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/docs	Ninguno especificado	Tomcat Documentation	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/examples	Ninguno especificado	Servlet and JSP Examples	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/host-manager	Ninguno especificado	Tomcat Host Manager Application	true	1	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos
/manager	Ninguno especificado	Tomcat Manager Application	true	1	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos

Si hacemos clic en los enlaces de la izquierda o ponemos el nombre del directorio en la URL (por ejemplo, localhost:8080/examples) accederemos a la aplicación web:

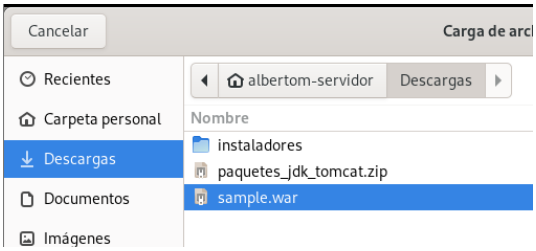


DESPLIEGUE DE UNA APLICACIÓN USANDO TOMCAT

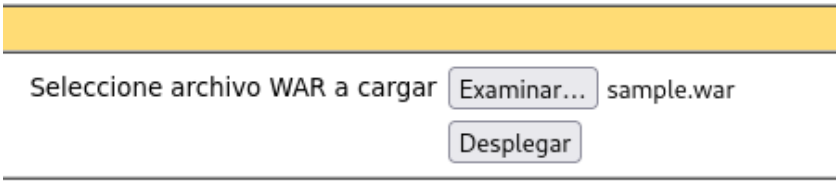
Si queremos añadir una nueva, debemos entrar en el "Manager app" y bajar al menú "Desplegar" y en concreto a la sección "archivo WAR a desplegar".



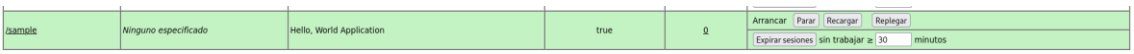
Hacemos clic en "Examinar" y buscamos el archivo .war que queramos usar:



Una vez seleccionado hacemos clic en "Desplegar":

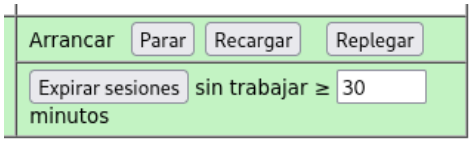


Y nos aparecerá en el listado de aplicaciones web desplegadas:



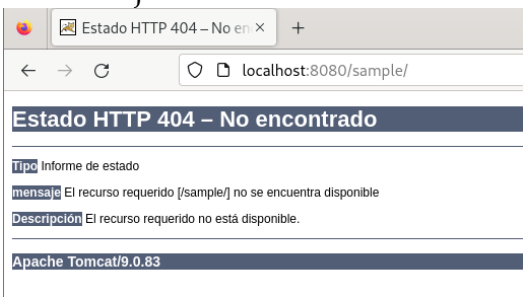
Si ahora accedemos a la URL http://localhost:8080/sample (nombre que tiene la aplicación en el cuadro) accederemos al sitio web que contiene la aplicación web:

Tenemos una serie de botones en la parte derecha del menú de control:



Estos nos van a servir para:

- **Arrancar**: Inicializa la aplicación web.
- **Parar**: Para la aplicación web. De forma que si intentamos acceder a la web veremos el siguiente mensaje:



- **Recargar**: Para la aplicación web y a continuación la vuelve a arrancar.
- **Replegar**: Elimina la aplicación web del listado de aplicaciones web desplegadas. Es decir, lo borra de /opt/tomcat/webapps.
- **Expirar sesiones**: Cierra sesiones tras X tiempo de inactividad para liberar recursos.

DESPLIEGUE CON MAVEN

A. INSTALACIÓN DE MAVEN

La instalación de Maven la vamos a realizar a través del comando apt:

sudo apt-get install maven

```
albertom-servidor@debian-deaw:~$ sudo apt-get install maven
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  ca-certificates-java default-jre-headless java-common libaopalliance-java libapache-pom-java
  libatinject-jsr330-api-java libcdi-api-java libcommons-cli-java libcommons-io-java
  libcommons-lang3-java libcommons-parent-java liberror-prone-java
  libgeronimo-annotation-1.3-spec-java libgeronimo-interceptor-3.0-spec-java libguava-java
  libguice-java libjansi-java libjsr305-java libmaven-parent-java libmaven-resolver-java
  libmaven-shared-utils-java libmaven3-core-java libplexus-cipher-java libplexus-classworlds-java
  libplexus-component-annotations-java libplexus-interpolation-java libplexus-sec-dispatcher-java
  libplexus-utils2-java libsisu-inject-java libsisu-plexus-java libslf4j-java libwagon-file-java
  libwagon-http-shaded-java libwagon-provider-api-java openjdk-17-jre-headless
Paquetes sugeridos:
```

Si queremos comprobar la versión de Maven que hemos instalado podemos utilizar el siguiente comando:

sudo mvn -v

```
albertom-servidor@debian-deaw:~$ sudo mvn -v
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 11.0.21, vendor: Debian, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: es_ES, platform encoding: UTF-8
OS name: "linux", version: "6.1.0-13-amd64", arch: "amd64", family: "unix"
```

B. CONFIGURACIÓN DE MAVEN

Para poder realizar despliegues en nuestro Tomcat debemos hacer ciertos cambios en la configuración de los ficheros:

1. Debemos añadir un nuevo rol, el rol manager-script en el archivo de usuarios de Tomcat: /opt/tomcat/conf/tomcat-users.xml

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

```
albertom-servidor@debian-deaw:~$ sudo nano /opt/tomcat/conf/tomcat-users.xml
```

```
<role rolename="admin"/>
<role rolename="admin-gui"/>
<role rolename="manager"/>
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
```

   Además, debemos crear un segundo usuario porque la documentación recomienda que no se otorgue los roles de manager-script y manager-gui al mismo usuario:

```
<user username="h2s" password="pwd" roles="admin,admin-gui,manager,manager-gui"/>
<user username="h2s-script" password="pwd" roles="manager-script"/>
```

2. Se debe editar el archivo de opciones de Maven (/etc/maven/settings.xml) para indicarle a Maven el servidor sobre el que se va a desplegar y las credenciales:

   sudo nano /etc/maven/settings.xml

```
albertom-servidor@debian-deaw:~$ sudo nano /etc/maven/settings.xml
```

   Y añadimos el siguiente bloque de código en el bloque <servers>:

```
<server>
        <id>Tomcat.P.3.1</id>
        <username>h2s-script</username>
        <password>pwd</password>
</server>
```

C. DESPLIEGUE DE UNA APLICACIÓN CREADA DE CERO

Vamos a crear una aplicación web nueva, para ello usamos el siguiente comando:

mvn archetype:generate -DgroupId=albertom-servidor -DartifactId=mi-aplicacion-war -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

```
albertom-servidor@debian-deaw:~$ mvn archetype:generate -DgroupId=albertom-servidor -DartifactId=mi-aplicacion-war -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.pom (3.9 kB at 4.5 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/22/maven-plugins-22.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/22/maven-plugins-22.pom (13 kB at 83 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/21/maven-parent-21.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/21/maven-parent-21.pom (26 kB at 235 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/apache/10/apache-10.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/10/apache-10.pom (15 kB at 361 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.jar (25 kB at 770 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/2.4/maven-install-plugin-2.4.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/2.4/maven-install-plugin-2.4.pom (6.4 kB at 164 kB/s)
```

Si todo es correcto al final de la ejecución deberemos ver algo similar a esto:

```
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-webapp:1.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: /home/albertom-servidor
[INFO] Parameter: package, Value: albertom-servidor
[INFO] Parameter: groupId, Value: albertom-servidor
[INFO] Parameter: artifactId, Value: mi-aplicacion-war
[INFO] Parameter: packageName, Value: albertom-servidor
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: /home/albertom-servidor/mi-aplicacion-war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  17.370 s
[INFO] Finished at: 2023-11-24T17:05:40+01:00
[INFO] ------------------------------------------------------------------------
```

Esto nos ha debido crear un nuevo directorio en el directorio en el que hayamos lanzado el comando, este directorio tendrá el mismo nombre que el artifactID del comando:

```
albertom-servidor@debian-deaw:~$ ls -la
total 136
drwx------ 19 albertom-servidor albertom-servidor 4096 nov 24 17:05 .
drwxr-xr-x  3 root              root              4096 nov  1 11:15 ..
-rw-------  1 albertom-servidor albertom-servidor 3045 nov 17 16:36 .bash_history
-rw-r--r--  1 albertom-servidor albertom-servidor  220 nov  1 11:15 .bash_logout
-rw-r--r--  1 albertom-servidor albertom-servidor 3526 nov  1 11:15 .bashrc
drwx------ 12 albertom-servidor albertom-servidor 4096 nov 17 15:35 .cache
drwxr-xr-x 12 albertom-servidor albertom-servidor 4096 nov  1 11:30 .config
drwxr-xr-x  3 albertom-servidor albertom-servidor 4096 nov 17 16:24 Descargas
drwxr-xr-x  2 albertom-servidor albertom-servidor 4096 nov  1 11:19 Documentos
drwxr-xr-x  2 albertom-servidor albertom-servidor 4096 nov  1 11:19 Escritorio
-rw-r--r--  1 albertom-servidor albertom-servidor 5290 nov  1 11:15 .face
lrwxrwxrwx  1 albertom-servidor albertom-servidor    5 nov  1 11:15 .face.icon -> .face
drwx------  2 albertom-servidor albertom-servidor 4096 nov  1 11:27 .gnupg
drwxr-xr-x  2 albertom-servidor albertom-servidor 4096 nov  1 11:19 Imágenes
drwx------  4 albertom-servidor albertom-servidor 4096 nov  1 11:19 .local
drwxr-xr-x  3 albertom-servidor albertom-servidor 4096 nov 24 17:05 .m2
drwxr-xr-x  3 albertom-servidor albertom-servidor 4096 nov 24 17:05 mi-aplicacion-war
drwxr-xr-x  4 albertom-servidor albertom-servidor 4096 nov 21 16:43 MiProyecto
drwx------  4 albertom-servidor albertom-servidor 4096 nov 17 15:35 .mozilla
drwxr-xr-x  2 albertom-servidor albertom-servidor 4096 nov  1 11:19 Música
drwxr-xr-x  2 albertom-servidor albertom-servidor 4096 nov  1 11:19 Plantillas
-rw-r--r--  1 albertom-servidor albertom-servidor  807 nov  1 11:15 .profile
drwxr-xr-x  2 albertom-servidor albertom-servidor 4096 nov  1 11:19 Público
drwx------  2 albertom-servidor albertom-servidor 4096 nov  1 11:27 .ssh
```

Ahora vamos a modificar el archivo POM para que nuestra aplicación pueda funcionar y que se haga referencia a que el despliegue se realice con el plugin de Maven para Tomcat.

sudo nano ./mi-aplicacion-war/pom.xml

```
albertom-servidor@debian-deaw:~$ sudo nano ./mi-aplicacion-war/pom.xml
```

```
<build>
  <finalName>mi-aplicacion-war</finalName>
  <plugins>
      <plugin>
      <groupId>org.apache.tomcat.maven</groupId>
      <artifactId>tomcat7-maven-plugin</artifactId>
      <version>2.2</version>
      <configuration>
          <url>http://localhost:8080/manager/text</url>
          <server>Tomcat.P.3.1</server>
          <path>/miaplicacion</path>
      </configuration>
      </plugin>
      </plugins>
</build>
```

URL → La URL del servidor Tomcat donde se realizará el despliegue (localhost:8080) debe ir seguida de /manager/text.

Server → Nombre del servidor donde se va a desplegar la aplicación, debe ser el mismo que el <id> del server que hayamos puesto en /etc/maven/settings.xml.

Path → Nombre que utilizará la aplicación en el path de la URL.

Guardamos cambios y tras esto debemos lanzar el siguiente comando en el directorio de la aplicación para desplegar la aplicación:

sudo mvn tomcat7:deploy

```
albertom-servidor@debian-deaw:~/mi-aplicacion-war$ sudo mvn tomcat7:deploy
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/tom
gin/2.2/tomcat7-maven-plugin-2.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/tomc
in/2.2/tomcat7-maven-plugin-2.2.pom (12 kB at 48 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/tom
in/2.2/tomcat-maven-plugin-2.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/tomc
```

Si todo ha salido de forma correcta veremos el siguiente mensaje en pantalla:

```
[INFO] tomcatManager status code:200, ReasonPhrase:
[INFO] OK - Desplegada aplicación en trayectoria de contexto [/miaplicacion]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  20.829 s
[INFO] Finished at: 2023-11-24T17:19:44+01:00
[INFO] ------------------------------------------------------------------------
```

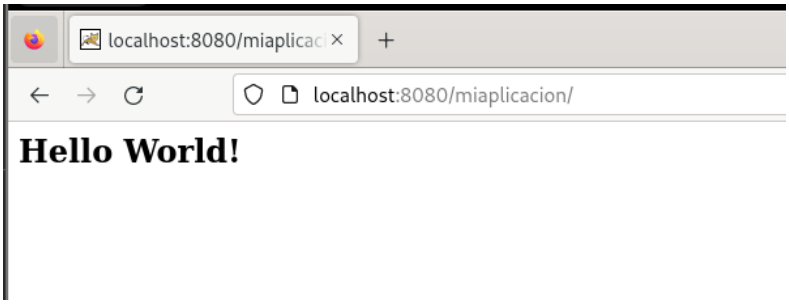
Esto habrá dos archivos en /opt/tomcat/webapps/, es decir, el directorio donde se guardan las aplicaciones que tenemos actualmente desplegadas:

```
albertom-servidor@debian-deaw:~/mi-aplicacion-war$ sudo ls -la /opt/tomcat/webapps/
total 128
drwxr-x--- 10 tomcat tomcat  4096 nov 24 17:19 .
drwxr-xr-x  9 tomcat tomcat  4096 nov 17 15:52 ..
drwxr-x--- 16 tomcat tomcat  4096 nov 17 15:52 docs
drwxr-x---  7 tomcat tomcat  4096 nov 17 15:52 examples
drwxr-x---  6 tomcat tomcat  4096 nov 17 15:52 host-manager
drwxr-x---  6 tomcat tomcat  4096 nov 17 15:52 manager
drwxr-x---  4 tomcat tomcat  4096 nov 24 17:19 miaplicacion
-rw-r-----  1 tomcat tomcat  2359 nov 24 17:19 miaplicacion.war
drwxr-x---  5 tomcat tomcat  4096 nov 21 16:43 MiProyecto
-rwxr-xr-x  1 tomcat tomcat 74446 nov 21 16:41 MiProyecto.war
drwxr-x---  3 tomcat tomcat  4096 nov 17 15:52 ROOT
drwxr-x---  5 tomcat tomcat  4096 nov 17 16:26 sample
-rw-r-----  1 tomcat tomcat  4604 nov 17 16:26 sample.war
```

Si ahora vamos a la aplicación gráfica de Tomcat veremos la aplicación en el selector de aplicaciones:

/miaplicacion	*Ninguno especificado*	Archetype Created Web Application	true	0	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos

Y si accedemos a la web, bien haciendo clic en /miaplicacion bien a través de la URL localhost:8080/miaplicacion/ veremos esta página:



D. DESPLIEGUE DE UNA APLICACIÓN WEB YA CREADA PREVIAMENTE

Vamos a descargarnos una aplicación desde un repositorio de GitHub:

sudo git clone https://github.com/cameronmcnz/rock-paper-scissors.git

```
albertom-servidor@debian-deaw:~/Descargas$ sudo git clone https://github.com/cameronmcnz/rock-paper-scissors.git
Clonando en 'rock-paper-scissors'...
remote: Enumerating objects: 999, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 999 (delta 11), reused 22 (delta 7), pack-reused 971
Recibiendo objetos: 100% (999/999), 29.14 MiB | 12.91 MiB/s, listo.
Resolviendo deltas: 100% (219/219), listo.
```

Nos situamos dentro de él y hacemos un cambio de rama:

cd ./rock-paper-scissors

sudo git checkout patch-1

```
albertom-servidor@debian-deaw:~/Descargas$ cd ./rock-paper-scissors/
albertom-servidor@debian-deaw:~/Descargas/rock-paper-scissors$ sudo git checkout patch-1
rama 'patch-1' configurada para rastrear 'origin/patch-1'.
Cambiado a nueva rama 'patch-1'
```

Debemos modificar el archivo pom.xml del proyecto:

```
albertom-servidor@debian-deaw:~/rock-paper-scissors$ ls -la
total 2560
drwxr-xr-x  7 root              root                4096 nov 24 17:24 .
drwx------ 20 albertom-servidor albertom-servidor   4096 nov 24 17:24 ..
-rw-r--r--  1 root              root                  44 nov 24 17:24 desktop.ini
-rw-r--r--  1 root              root                  76 nov 24 17:24 Dockerfile.dockerfile
drwxr-xr-x  8 root              root                4096 nov 24 17:24 .git
drwxr-xr-x  3 root              root                4096 nov 24 17:24 ibrahim
-rw-r--r--  1 root              root                 158 nov 24 17:24 index.html
-rw-r--r--  1 root              root                1237 nov 24 17:24 install-all.sh
-rw-r--r--  1 root              root             1744513 nov 24 17:24 LabGuide-Protected.pdf
-rw-r--r--  1 root              root                   0 nov 24 17:24 new
-rw-r--r--  1 root              root                2141 nov 24 17:24 pom.xml
-rw-r--r--  1 root              root                 856 nov 24 17:24 portal-9-ubuntu-install.sh
-rw-r--r--  1 root              root                2410 nov 24 17:24 remote-vm.rdp
drwxr-xr-x  2 root              root                4096 nov 24 17:24 rock-paper-scissors
-rw-r--r--  1 root              root                1007 nov 24 17:24 scripted-pipeline
-rw-r--r--  1 root              root              801624 nov 24 17:24 Slides-Protected.pdf
drwxr-xr-x  4 root              root                4096 nov 24 17:24 src
drwxr-xr-x  3 root              root                4096 nov 24 17:24 sulthan
-rw-r--r--  1 root              root                   0 nov 24 17:24 sweet-day.html
-rw-r--r--  1 root              root                 876 nov 24 17:24 TanslationService.java
-rw-r--r--  1 root              root                  24 nov 24 17:24 test_file.txt
-rw-r--r--  1 root              root                3288 nov 24 17:24 tidbits.sh
-rw-r--r--  1 root              root                   0 nov 24 17:24 training-day.html
```

Y añadir el siguiente bloque <plugin> para que se haga referencia a que el despliegue se realice con el plugin de Maven para Tomcat.

```
albertom-servidor@debian-deaw:~/Descargas/rock-paper-scissors$ sudo nano ./pom.xml
```

```
<plugin>
  <groupId>org.apache.tomcat.maven</groupId>
  <artifactId>tomcat7-maven-plugin</artifactId>
  <version>2.2</version>
  <configuration>
    <url>http://localhost:8080/manager/text</url>
    <server>Tomcat.P.3.1</server>
    <path>/rockpaperscissors</path>
  </configuration>
</plugin>
```

sudo mvn tomcat7:deploy

```
albertom-servidor@debian-deaw:~/Descargas/rock-paper-scissors$ sudo mvn tomcat7:deploy
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.mcnz.rps.web:roshambo >----------------------
[INFO] Building roshambo web application 1.0
[INFO] --------------------------------[ war ]---------------------------------
```

Una vez finalizado el despliegue obtendremos el siguiente resultado:

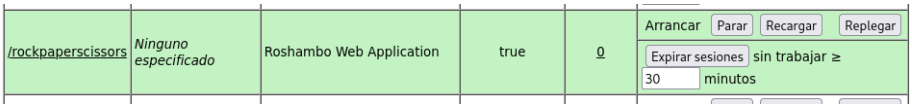
```
[INFO] Deploying war to http://localhost:8080/myapp
Uploading: http://localhost:8080/manager/text/deploy?path=%2Fmyapp
Uploaded: http://localhost:8080/manager/text/deploy?path=%2Fmyapp (11 KB)

[INFO] tomcatManager status code:200, ReasonPhrase:
[INFO] OK - Desplegada aplicación en trayectoria de contexto [/myapp]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  17.077 s
[INFO] Finished at: 2023-11-13T20:43:36+01:00
[INFO] ------------------------------------------------------------------------
```

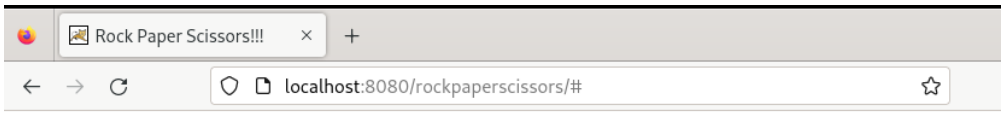
Al igual que en el apartado anterior esto nos ha tenido que crear dos ficheros en /opt/tomcat/webapps/:

```
albertom-servidor@debian-deaw:~/rock-paper-scissors$ sudo ls -la /opt/tomcat/webapps/
total 144
drwxr-x--- 11 tomcat tomcat  4096 nov 24 17:29 .
drwxr-xr-x  9 tomcat tomcat  4096 nov 17 15:52 ..
drwxr-x--- 16 tomcat tomcat  4096 nov 17 15:52 docs
drwxr-x---  7 tomcat tomcat  4096 nov 17 15:52 examples
drwxr-x---  6 tomcat tomcat  4096 nov 17 15:52 host-manager
drwxr-x---  6 tomcat tomcat  4096 nov 17 15:52 manager
drwxr-x---  4 tomcat tomcat  4096 nov 24 17:19 miaplicacion
-rw-r-----  1 tomcat tomcat  2359 nov 24 17:19 miaplicacion.war
drwxr-x---  5 tomcat tomcat  4096 nov 21 16:43 MiProyecto
-rwxr-xr-x  1 tomcat tomcat 74446 nov 21 16:41 MiProyecto.war
drwxr-x---  4 tomcat tomcat  4096 nov 24 17:29 rockpaperscissors
-rw-r-----  1 tomcat tomcat 10534 nov 24 17:29 rockpaperscissors.war
drwxr-x---  3 tomcat tomcat  4096 nov 17 15:52 ROOT
drwxr-x---  5 tomcat tomcat  4096 nov 17 16:26 sample
-rw-r-----  1 tomcat tomcat  4604 nov 17 16:26 sample.war
```

Además, ahora aparecerá en el Manager App:



Si ahora accedemos a Tomcat a través del Manager App o desde la URL http://localhost:8080/rockpaperscissors podremos ver la nueva aplicación en el registro de aplicaciones web desplegadas:



Which one will it be for you today?
rock paper scissors
lose
1
1
1

Client	Server	Result	Date
scissors	rock	lose	Fri Nov 24 2023 17:31:10 GMT+0100 (hora estándar de Europa central)
paper	rock	win	Fri Nov 24 2023 17:31:09 GMT+0100 (hora estándar de Europa central)
rock	rock	tie	Fri Nov 24 2023 17:31:08 GMT+0100 (hora estándar de Europa central)

CUESTIONES FINALES

a. *Habéis visto que los archivos de configuración que hemos tocado contienen contraseñas en texto plano, por lo que cualquiera con acceso a ellos obtendría las credenciales de nuestras herramientas.*

   En principio esto representa un gran riesgo de seguridad, ¿sabrías razonar o averiguar por qué esto está diseñado de esta forma?

   Las razones pueden ser varias:

   1. Simplifica la configuración inicial: Almacenar contraseñas en texto plano facilita el poder realizar cambios directamente sobre el archivo de configuración sin necesidad preocuparse por cifrados o procesos adicionales.
   2. Uso por parte de la aplicación: Las contraseñas deben ser utilizadas por la aplicación, si estuvieran cifradas requeriría de un proceso adicional para descifrarlas antes de ser utilizadas, lo que complicaría la implementación de las aplicaciones.