

---

# **Tomcat y Maven**

***Versión 1.0.0***

**Fernando Raya**

**06 de enero de 2025**

---

## Contents

---

<b>1</b>	<b>Introducción</b>	<b>2</b>
<b>2</b>	<b>Instalación de Tomcat</b>	<b>3</b>
2.1	OpenJDK . . . . .	3
2.2	Tomcat . . . . .	3
<b>3</b>	<b>Configuración de la administración</b>	<b>6</b>
3.1	Usuarios y permisos . . . . .	6
3.2	Instalación del administrador web . . . . .	7
<b>4</b>	<b>Despliegue manual mediante GUI</b>	<b>11</b>
<b>5</b>	<b>Despliegue con Maven</b>	<b>12</b>
5.1	Instalación de Maven . . . . .	12
5.2	Configuración de Maven . . . . .	12
<b>6</b>	<b>Tarea</b>	<b>17</b>
<b>7</b>	<b>Referencias</b>	<b>18</b>

En esta práctica realizaremos un despliegue de una aplicación Java usando Tomcat y Maven.



Figure1: De los creadores del *Apache webserver* llega ..



Figure2: Apache Tomcat, el servidor de aplicaciones web para Java, dibujado con el pie por [RRZEicons](#) CC BY-SA 3.0, via Wikimedia Commons



Figure3: Maven, la herramienta para automatizar la construcción de proyectos principalmente utilizada para Java, que mira que bonitas quedan las plumas después de masacrar a la nación Apache. Apache Software Foundation, Public domain, via Wikimedia Commons

# CHAPTER 1

---

## Introducción

---

Si consultamos el apartado de [versiones de Tomcat](#) en su página oficial, nos daremos cuenta de que no vamos a usar la última versión, para esta práctica, sino la 9. La pregunta es casi inevitable: ¿Por qué?

En el enlace anterior vemos como desde su versión 9, Tomcat da soporte a Java 8 y superiores. Sin embargo, a partir de la versión 10.1.x, da soporte a Java 11 y superiores. ¿Qué significa esto?

En Java 9 se introdujeron novedades como un nuevo sistema de módulos (Jigsaw), [entre otras](#). En Java 11 se dio un paso más al haber renombrado completamente las rutas de paquetes `javax.` a `jakarta.`. Oracle, a pesar de haber hecho público el desarrollo de Java, no hizo lo mismo con el nombre de Java.

Así las cosas, resulta que Java 8 puede que a día de hoy aún sea la más usada en proyectos reales. Dicho esto, podría realizarse un proceso de migración de un proyecto de Java 8 a Java 11 y utilizarlo en Tomcat 10.

No obstante, para Java 8 su soporte para uso comercial (pagando) acabó en [Marzo de 2022](#), pero para uso no comercial sigue hasta 2030.

En conclusión, no es raro encontrarse en el mundo real un proyecto a desplegar realizado en Java 8.

Podría realizarse una migración y los conceptos de despliegue que veremos seguirían aplicando. Así las cosas, por facilidad en la realización de las prácticas utilizaremos Tomcat 9 y el plugin oficial de Maven para Tomcat 7 para el despliegue (luego veremos el motivo).

---

### Instalación de Tomcat

---

Esta práctica es muy sencilla y va a consistir en realizar la instalación del servidor de aplicaciones Tomcat 9, en una máquina virtual corriendo Debian 11 Bullseye.

Para ello realizaremos:

- Instalación de OpenJDK.
- Instalación de Tomcat 9

## 2.1 OpenJDK

Instalaremos el *kit* de desarrollo de java:

```
sudo apt install -y openjdk-11-jdk
```

## 2.2 Tomcat

Instalaremos el servidor de aplicaciones Tomcat y realizaremos las siguientes tareas:

- Creación de un grupo de usuarios específico para Tomcat.
- Creación de un usuario dedicado para el servicio.
- Comprobación del estado del servicio para verificar que está activo.

### 2.2.1 Instalación del paquete

Listing 1: Instalación del servidor de aplicaciones

```
sudo apt install -y tomcat9
```

### 2.2.2 Creación del grupo

Crearemos un grupo de usuarios para *tomcat9*

```
sudo groupadd tomcat9
```

### 2.2.3 Creación del usuario

Listing 2: Crearemos un usuario para el servicio

```
sudo useradd -s /bin/false -g tomcat9 -d /etc/tomcat9 tomcat9
```

Esta orden crea un nuevo usuario llamado *tomcat9* con características específicas para su uso como cuenta de servicio en el sistema. Aquí se desglosa lo que hace cada opción:

- `-s /bin/false`: Establece el intérprete de comandos (*shell*) como `/bin/false`, lo que impide que este usuario inicie sesión en el sistema. Esto se hace por seguridad, ya que el usuario solo será utilizado para gestionar el servicio Tomcat y no necesita acceso interactivo.
- `-g tomcat9`: Especifica que el grupo principal del usuario será *tomcat9*. Este grupo debe haberse creado previamente con el comando `groupadd tomcat9`.
- `-d /etc/tomcat9`: Define el directorio de inicio del usuario como `/etc/tomcat9`. Aunque este usuario no tendrá un entorno de inicio interactivo, esta carpeta puede utilizarse para almacenar configuraciones específicas relacionadas con Tomcat.
- *tomcat9*: Es el nombre del usuario que se crea.

### 2.2.4 Arranque y comprobación del servicio

Listing 3: Arrancaremos el servicio *tomcat9*

```
sudo systemctl start tomcat9
```

Listing 4: Comprobamos que el servicio funciona correctamente

```
sudo systemctl status tomcat9
```

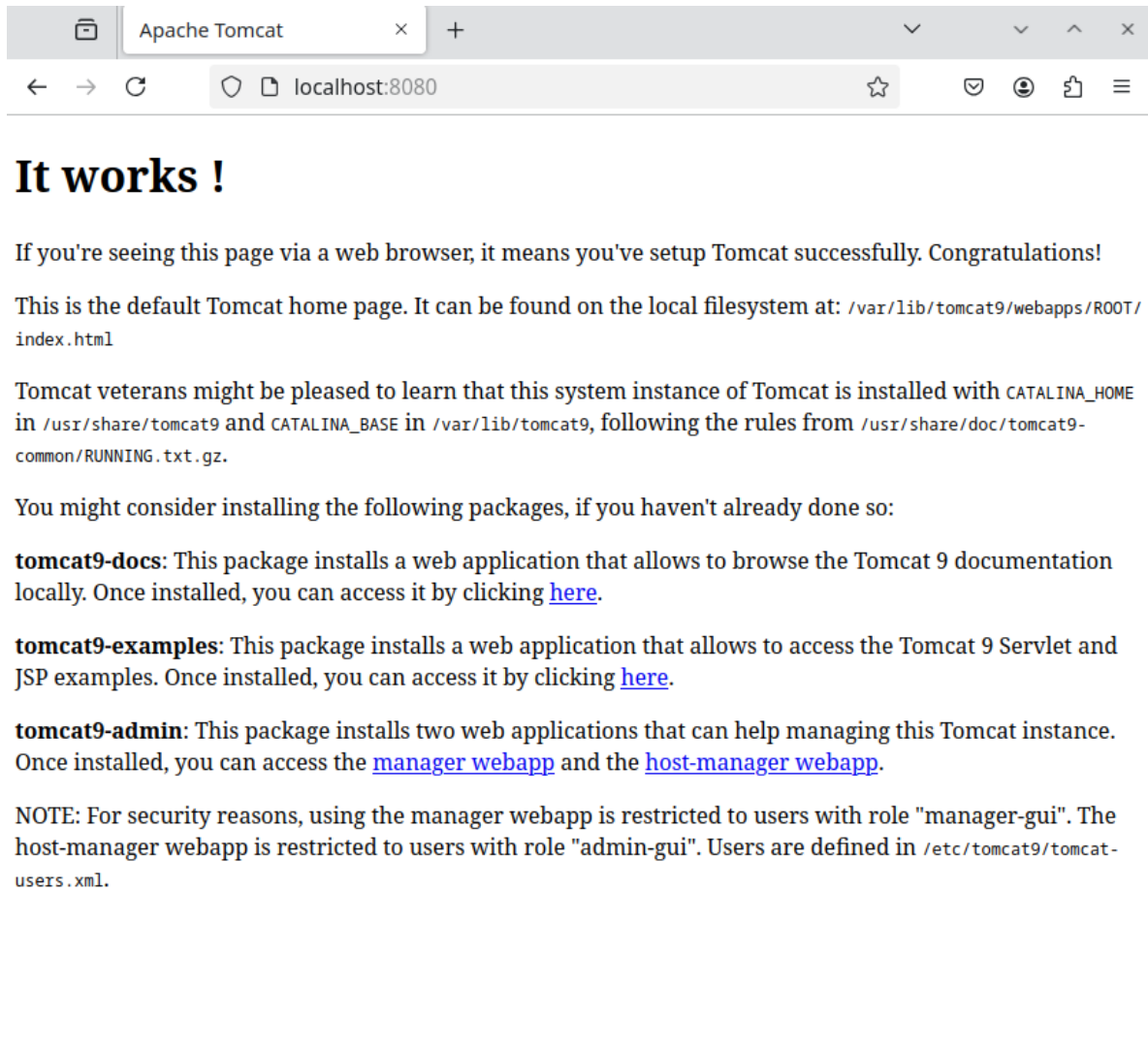
Listing 5: Status del servicio *tomcat9*

```
* tomcat9.service - Apache Tomcat 9 Web Application Server
   Loaded: loaded (/lib/systemd/system/tomcat9.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2025-01-03 08:31:10 UTC; 4s ago
     Docs: https://tomcat.apache.org/tomcat-9.0-doc/index.html
    Process: 6716 ExecStartPre=/usr/libexec/tomcat9/tomcat-update-policy.sh (code=exited,
↳ status=0/SUCCESS)
   Main PID: 6720 (java)
    Tasks: 29 (limit: 1114)
   Memory: 71.6M
      CPU: 4.876s

...

Jan 03 08:31:11 bullseye tomcat9[6720]: OpenSSL successfully initialized [OpenSSL 1.1.1w 11
↳ Sep 2023]
```

Accederemos al servicio en <http://localhost:8080>



---

## Configuración de la administración

---

### 3.1 Usuarios y permisos

Ahora es el momento de definir el usuario con acceso a Tomcat. Para ello, es necesario editar el archivo de configuración `/etc/tomcat9/tomcat-users.xml`.

```
sudo nano /etc/tomcat9/tomcat-users.xml
```

Este archivo configura un usuario llamado `alumno` con acceso completo a las funcionalidades administrativas y de gestión del servidor Tomcat, tanto en la interfaz gráfica como en las operaciones relacionadas con aplicaciones. Es una configuración típica para entornos de prueba o desarrollo.

Listing 1: Fichero `/etc/tomcat9/tomcat-users.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users xmlns="http://tomcat.apache.org/xml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
  version="1.0">
  <role rolename="admin"/>
  <role rolename="admin-gui"/>
  <role rolename="manager"/>
  <role rolename="manager-gui"/>
  <user username="alumno"
    password="1234"
    roles="admin,admin-gui,manager,manager-gui"/>
</tomcat-users>
```

Los elementos principales de la configuración son:

1. **Roles:**

- Se definen roles específicos con el elemento `<role>` y su atributo `rolename`.
- Roles creados:



- admin: Permite realizar tareas administrativas.
- admin-gui: Habilita el acceso a la interfaz gráfica de administración.
- manager: Da permisos para gestionar aplicaciones.
- manager-gui: Permite utilizar la interfaz gráfica de gestión de aplicaciones.

## 2. Usuarios:

- El elemento <user> define un usuario con los siguientes atributos:
- roles="admin,admin-gui,manager,manager-gui": **Asigna los roles** definidos anteriormente al usuario.

## 3.2 Instalación del administrador web

Ahora, instalemos el administrador web y el administrador de host de Tomcat ejecutando el siguiente comando.

```
sudo apt install -y tomcat9-admin
```

El paquete tomcat9-admin proporciona herramientas adicionales para la administración del servidor Tomcat 9, específicamente las aplicaciones de gestión basadas en web. Estas herramientas incluyen las interfaces de administración y gestión que permiten realizar tareas administrativas de forma más visual e interactiva.

### 3.2.1 Funciones del paquete tomcat9-admin

#### Aplicación de Administración (Admin GUI)

Permite gestionar configuraciones del servidor, como recursos, conexiones, y otros parámetros avanzados. Es útil para administradores que necesitan realizar cambios en la configuración de Tomcat sin editar directamente los archivos XML.

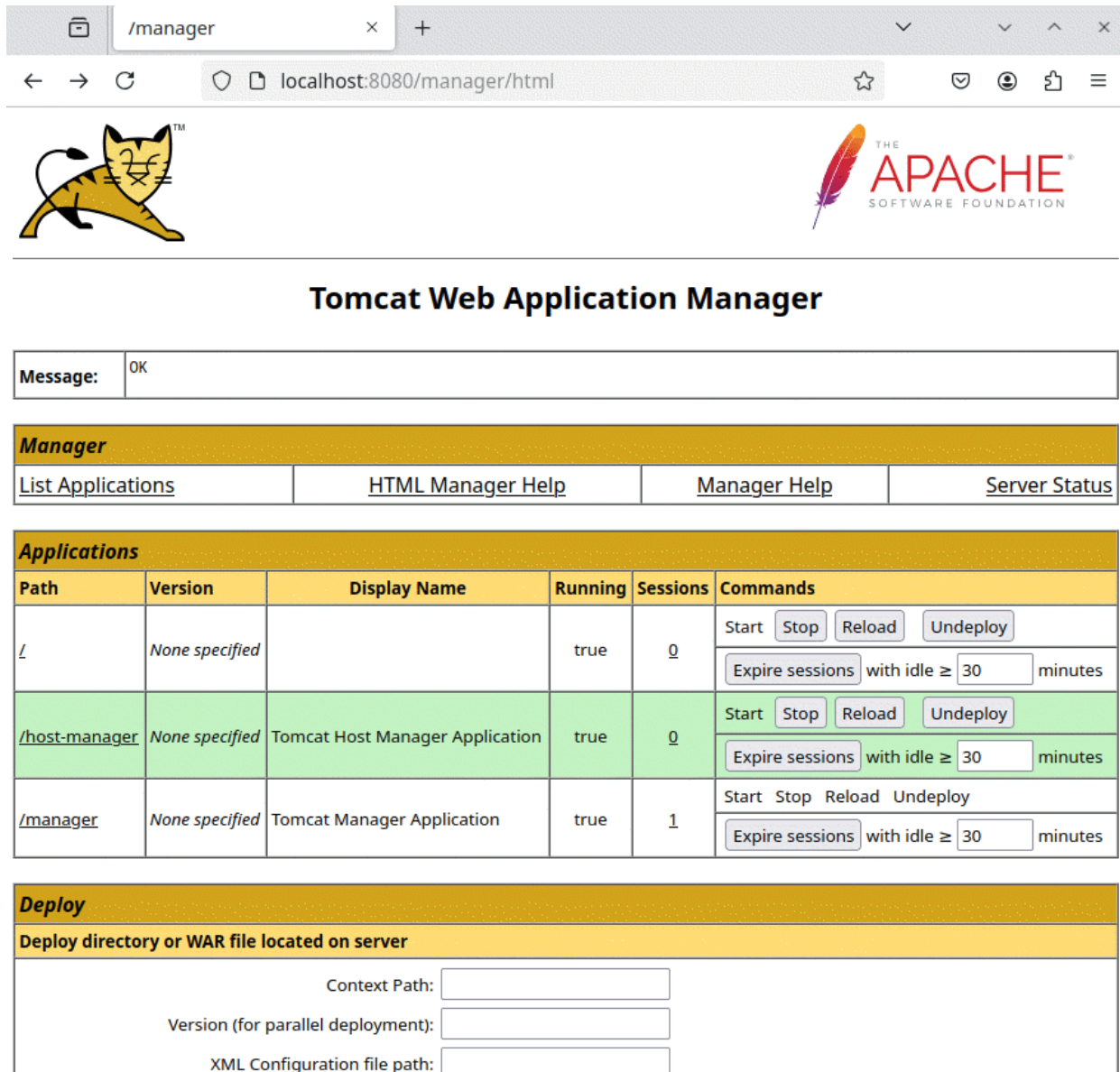
#### Aplicación de Gestión (Manager GUI)

Permite realizar tareas relacionadas con las aplicaciones web desplegadas en Tomcat, como: Desplegar nuevas aplicaciones. Detener o reiniciar aplicaciones existentes. Consultar información sobre el estado de las aplicaciones (carga de memoria, uso de hilos, etc.).

Proporciona una interfaz gráfica para operaciones que podrían realizarse mediante scripts o herramientas CLI.

### 3.2.2 Acceso a los paneles de administración

Accede a <http://localhost:8080/manager/html> e introduce el nombre de usuario y la contraseña que escribió al editar el archivo /etc/tomcat9/tomcat-users.xml.



**Tomcat Web Application Manager**

Message: OK

**Manager**

[List Applications](#) [HTML Manager Help](#) [Manager Help](#) [Server Status](#)

**Applications**

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

**Deploy**

Deploy directory or WAR file located on server

Context Path:

Version (for parallel deployment):

XML Configuration file path:

Ahora accedamos a <http://localhost:8080/host-manager/html> para verificar el Administrador de host virtual Tomcat; escribe el nombre de usuario y la contraseña cuando te lo soliciten.



## Tomcat Virtual Host Manager

Message:	OK
----------	----

Host Manager			
<a href="#">List Virtual Hosts</a>	<a href="#">HTML Host Manager Help</a>	<a href="#">Host Manager Help</a>	<a href="#">Server Status</a>

Host name		
Host name	Host aliases	Commands
<a href="#">localhost</a>		Host Manager installed - commands disabled

Add Virtual Host	
Host	
Name:	<input type="text"/>
Aliases:	<input type="text"/>
App base:	<input type="text"/>
AutoDeploy	<input checked="" type="checkbox"/>
DeployOnStartup	<input checked="" type="checkbox"/>
DeployXML	<input checked="" type="checkbox"/>
UnpackWARs	<input checked="" type="checkbox"/>
Manager App	<input checked="" type="checkbox"/>
CopyXML	<input type="checkbox"/>
<input type="button" value="Add"/>	

Persist configuration	
<input type="button" value="All"/>	Save current configuration (including virtual hosts) to server.xml and per web application context.xml files

Server Information					
Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/9.0.43 (Debian)	11.0.25+9-post-Debian-1deb11u1	Debian	Linux	5.10.0-32-amd64	amd64

Copyright © 1999-2021, Apache Software Foundation

### 3.2.3 Acceso remoto

Tomcat no permite por defecto que se acceda a algunos paneles de control desde un ordenador distinto al localhost. Para solucionarlo sustituiremos el fichero context.xml del directorio /usr/share/tomcat9-admin/host-manager/META-INF/ por el siguiente:

Listing 2: Fichero /usr/share/tomcat9-admin/host-manager/META-INF/context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
    sameSiteCookies="strict" />
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="\d+\.\d+\.\d+\.\d+" />
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?
↪:Boolean|Integer|Long|Number|string)|org\.apache\.catalina\.filters\.CsrfPreventionFilter\
↪$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
</Context>
```

La línea `<Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="\d+\.\d+\.\d+\.\d+" />` dentro del código XML tiene el propósito de restringir el acceso a una aplicación o recurso de Tomcat basado en la dirección IP del cliente. En este caso, permite el acceso desde todas las direcciones IP, ya que no hay restricciones específicas configuradas.

Recargaremos el servidor:

```
sudo systemctl restart tomcat9
```

## Despliegue manual mediante GUI

Realizaremos el despliegue manual de una aplicación ya previamente empaquetada en formato WAR.

### **Nota**

Un archivo `.war` (*Web Application Archive*) es un paquete comprimido que contiene todos los recursos necesarios para desplegar una aplicación web Java en un servidor de aplicaciones, como Tomcat. Incluye código, configuraciones (como *web.xml*), librerías y archivos públicos. Facilita el despliegue y asegura compatibilidad con servidores que siguen las especificaciones de Java EE.

Para ello:

1. Nos logueamos con el usuario previamente creado.
2. Descargamos el archivo `tomcat1.war` que acompaña a las prácticas.
3. Buscamos la sección que nos permite desplegar un WAR manualmente.

WAR file to deploy

Select WAR file to upload  No file selected.

4. Pulsamos en *Examinar/Browse*, y buscamos el fichero `tomcat1.war`.
5. Pulsamos *Desplegar/Deploy*. Se nos listará la aplicación ya desplegada `/tomcat1` como un directorio más y podremos acceder a ella.

<a href="#">/tomcat1</a>	None specified	true	0	<input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/>
				<input type="button" value="Expire sessions"/> with idle <input type="text" value="30"/> minutes

---

## Despliegue con Maven

---

### 5.1 Instalación de Maven

Para instalar Maven en nuestro Debian tenemos, de nuevo, dos opciones:

- Instalación mediante gestor de paquetes APT
- Instalación manual

La primera, recomendada, es mucho más sencilla y automatizada (establece todos los paths y variables de entorno), aunque con la segunda se podría conseguir un paquete más actualizado. Ambos métodos vienen explicados [aquí](#).

Si decidimos seguir el primer método, el más sencillo, vemos que es tan simple como actualizar los repositorios e instala Maven:

```
sudo apt-get update && sudo apt-get -y install maven
```

Para comprobar que todo ha ido correctamente, podemos ver la versión instalada de Maven:

```
mvn --v
```

### 5.2 Configuración de Maven

Para poder realizar despliegues en nuestro Tomcat previamente instalado, necesitamos realizar la configuración adecuada para Maven. Ya sabemos que esto en Linux significa editar los archivos de configuración adecuados. Vamos a ello.

#### 5.2.1 Roles

En primer lugar necesitamos asegurarnos de que en el apartado anterior de la práctica hemos añadido todos los usuarios necesarios, así como sus respectivos roles. Debemos añadir el rol de `manager-script` para permitir que Maven se autentique contra Tomcat y pueda realizar el despliegue. Los roles utilizados por Tomcat vienen detallados en su documentación, que merece ser consultada:

You can find the role names in the web.xml file of the Manager web application. The available roles are:

**manager-gui**

Access to the HTML interface.

**manager-status**

Access to the «Server Status» page only.

**manager-script**

Access to the tools-friendly plain text interface that is described in this document, and to the «Server Status» page.

**manager-jmx**

Access to JMX proxy interface and to the «Server Status» page.

En dicha documentación se nos indica que, por temas de seguridad, es recomendable no otorgar los roles de manager-script o manager-jmx al mismo usuario que tenga el rol de manager-gui.

**i Nota**

Tendremos dos usuarios, uno para la GUI y otro exclusivamente para hacer los despliegues de Maven.

Así las cosas, modificamos el archivo /etc/tomcat9/tomcat-users.xml acorde a nuestras necesidades (los nombres de usuario y contraseña deberán ser los que elijáis para vosotros):

Listing 1: Fichero /etc/tomcat9/tomcat-users.xml

```
...
<role rolename="admin"/>
<role rolename="admin-gui"/>
<role rolename="manager"/>
<role rolename="manager-gui"/>
<role rolename="manager-status"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<user username="alumno"
      password="1234"
      roles="admin, admin-gui, manager, manager-gui"
/>
<user username="deploy" password="1234" roles="manager-script"/>
...
```

## 5.2.2 settings.xml

Edita el archivo /etc/maven/settings.xml para indicarle a Maven, un identificador para el servidor sobre el que vamos a desplegar (no es más que un nombre, ponédle el nombre que consideréis), así como las credenciales. Todo esto se hará dentro del bloque <servers> del XML:

Listing 2: Extracto del fichero /etc/maven/settings.xml

```
...
<servers>
  <server>
    <id>Tomcat</id>
```

(continúe en la próxima página)

(proviene de la página anterior)

```
<username>deploy</username>
<password>1234</password>
</server>
</servers>
...
```

### 5.2.3 Generar una aplicación

Generaremos una aplicación de ejemplo en nuestro directorio personal:

```
cd
```

Listing 3: Generamos una aplicación de prueba

```
$ mvn archetype:generate -DgroupId=org.zaidinvergeles \
-DartifactId=tomcat-war \
-Ddeployment \
-DarchetypeArtifactId=maven-archetype-webapp \
-DinteractiveMode=fa
```

Podéis sustituir los valores de groupId y artifactId (este será el nombre de la aplicación) por lo que queráis.

Tras muchas líneas veremos:

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
```

Se habrá creado un directorio llamado tomcat-war-deployment. Entraremos en él:

```
$ cd tomcat-war-deployment
```

### 5.2.4 POM

Ahora debemos modificar el POM del proyecto para que haga referencia a que el despliegue se realice con el plugin de Maven para Tomcat.

#### **i** Nota

No existen plugins oficiales para Tomcat más allá de la versión 7 del servidor. No obstante, el plugin para Tomcat 7 sigue funcionando correctamente con Tomcat 9. Otra opción sería utilizar el plugin **Cargo**

Listing 4: Fichero pom.xml donde modificaremos la parte enfatizada

```
<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/maven-
↪v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.zaidinvergeles</groupId>
```

(continúe en la próxima página)



(proviene de la página anterior)

```

<artifactId>tomcat-war-deployment</artifactId>
<packaging>war</packaging>
<version>1.0-SNAPSHOT</version>
<name>tomcat-war-deployment Maven Webapp</name>
<url>http://maven.apache.org</url>
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <finalName>tomcat-war-deployment</finalName>
</build>
</project>

```

Donde lo que añadimos es el bloque:

Listing 5: Bloque a insertar en <build></build>

```

1 <build>
2   <finalName>tomcat-war-deployment</finalName>
3   <plugins>
4     <plugin>
5       <groupId>org.apache.tomcat.maven</groupId>
6       <artifactId>tomcat7-maven-plugin</artifactId>
7       <version>2.2</version>
8       <configuration>
9         <url>http://localhost:8080/manager/text</url>
10        <server>Tomcat</server>
11        <path>/despliegue</path>
12      </configuration>
13    </plugin>
14  </plugins>
15 </build>

```

#### Línea 2

pondremos el nombre final del ejecutable .jar que se va a generar.

#### Línea 9

URL del servidor Tomcat donde se hará el despliegue. Como en nuestro caso Maven y Tomcat están en el mismo servidor, la URL corresponde a localhost. Esta URL debe ir seguida por /manager/text, tal y como leemos en la [documentación del plugin](#)

#### Línea 10

Nombre del servidor donde se va a desplegar la aplicación. El nombre debe ser consistente con lo que hayamos puesto en el settings.xml del paso anterior.

#### Línea 11

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

### 5.2.5 Despliegue

Tras generar esta aplicación, los comandos finales que se utilizan en Maven para desplegar son:

Listing 6: Desplegar una aplicación

```
mvn tomcat7:deploy
```

Listing 7: Volver a desplegar una aplicación

```
mvn tomcat7:redploy
```

Listing 8: Retirar una aplicación desplegada

```
mvn tomcat7:undeploy
```

Así pues, tras el despliegue con Maven nos indicará que todo ha ido correctamente con un mensaje de BUILD SUCCESS, tal que así:

```
default: [INFO] Deploying war to http://localhost:8080/despliegue  
default: Uploading: http://localhost:8080/manager/text/deploy?path=%2Fmyapp  
Uploaded: http://localhost:8080/manager/text/deploy?path=%2Fmyapp (3 KB at 334.1 KB/sec)
```

Y, accediendo a través de la GUI, debemos ver que la aplicación está desplegado y que podemos acceder a ella perfectamente:

<http://servidor:8080/despliegue>

## CHAPTER 6

---

### Tarea

---

Realizar el despliegue con la aplicación de prueba.

Repetir el despliegue pero esta vez con otra aplicación que no es la de prueba. Usaremos los comandos que veremos a continuación.

Listing 1: Clonamos el repositorio

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

Listing 2: Nos situamos dentro de el directorio clonado

```
cd rock-paper-scissors
```

Listing 3: Cambiamos de rama

```
git checkout patch-1
```

Tras esto debemos proceder exactamente igual que en el caso anterior, con la ventaja de que ya tenemos configurados los usuarios de Tomcat y los parámetros de Maven.

Así pues, sólo habría que añadir el bloque `<plugin> . . </plugin>` adecuado para poder hacer nuestro despliegue.

Documenta, incluyendo capturas de pantallas, el proceso que has seguido para realizar el despliegue de esta nueva aplicación, así como el resultado final.

## CHAPTER 7

---

### Referencias

---

1. Tutorial Tomcat I
2. Tutorial Tomcat II
3. Tutorial Tomcat para Ubuntu
4. Instalación Maven
5. **`JSF 3.0 en Tomcat 10 con Java 11 <<https://www.nestoralmeida.com/jsf-3-0-en-tomcat-10-con-java-11/#9-renombrar-javax-a-jakarta>`\_**
6. Migración de Java 8 a Java 11