5.2.1. Instalación del servidor de aplicaciones Apache-Tomcat

El servidor de aplicaciones que se va a instalar y configurar es Apache-Tomcat, por ser uno de los más usados en el mundo empresarial, también por su facilidad de uso y por su aceptación en la comunidad de desarrolladores de software.

Apache-Tomcat es un contenedor de Servlets que se puede usar para compilar y ejecutar aplicaciones realizadas en Java. Tomcat puede funcionar de manera autónoma pero usualmente se puede utilizar con otros productos para permitir una mayor compatibilidad con las distintas tecnologías del mercado, y de esta forma incrementar su funcionalidad.

Es crucial conocer cómo funciona un contenedor de servlets, ya que es el funcionamiento de Apache-Tomcat. Para ello se explica en la siguiente lista y gráfico:

- El navegador web del cliente solicita una página al servidor HTTP.
- El contenedor de servlets procesa la petición y la asigna al servlet apropiado.
- El servlet elegido es el encargado de generar el texto de la página web y entregarla al contenedor de servlets.
- El contenedor devuelve la página web al navegador del cliente.

Por otro lado, además de servlets, se pueden ejecutar en la capa web más componentes, como, por ejemplo, Java Server Pages (JSP). Es un software que permite generar páginas web dinámicas principalmente (también genera otro tipo de documentos, como xml, etc.). Estos archivos .jsp se compilan y se transforman en un servlet.

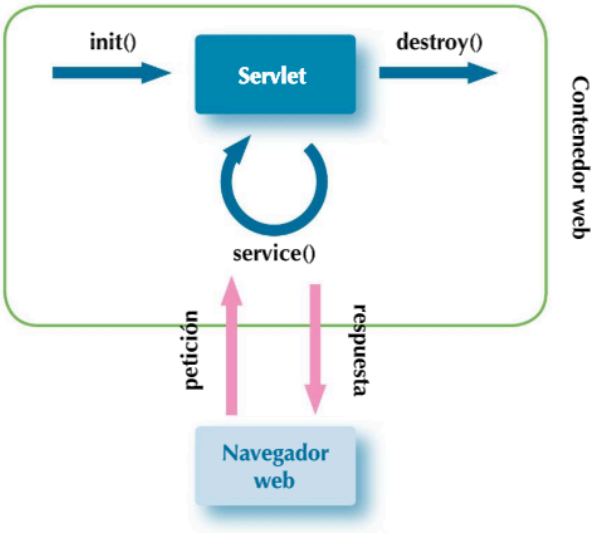


Figura 5.2
Funcionamiento del contenedor de servlets

5.2.2. Arquitectura de Apache-Tomcat y variables de entorno

Una vez instalado el servidor de aplicaciones Apache-Tomcat se va a analizar su estructura para que el programador o usuario de este servicio pueda comprender el funcionamiento a grandes rasgos de la aplicación.

Apache-Tomcat usa una arquitectura que permite de forma lógica conectar sus diferentes componentes y que cada uno realice una función determinada en la estructura arborescente. Por ello se van a listar los componentes que posee este servidor de aplicaciones:

a) *Server:* es el componente que representa el contenedor al completo y dentro de él se ejecutan los demás componentes. Posee una interfaz gráfica que prácticamente no se customiza.
b) *Service:* es un componente intermedio que se comunica y que está dentro del componente server, que posee varias conexiones.
c) *Engine:* es el servicio que permite procesar las peticiones que provienen desde el exterior y responder a tales solicitudes. Por ello, posee múltiples conectores que hacen posible resolver las peticiones.
d) *Host:* es un nombre de red que puede poseer varios hosts o alias, como pueden ser www.daw.com o www.daw.es.
e) *Connector:* es el encargado de controlar las comunicaciones con el cliente. Tomcat incluye varios conectores, como AJP Connector o HTTP Connector.
f) *Context:* representa una aplicación web. Un host puede poseer varios o múltiples contextos pero con una única ruta.

Para que se observe de forma gráfica, la arquitectura sería la siguiente:

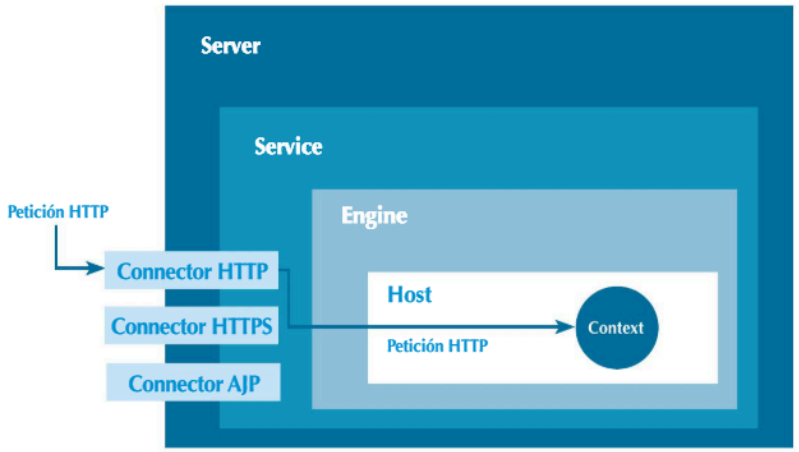


Figura 5.7
Arquitectura Apache-Tomcat 9

Anteriormente, se instaló Apache-Tomcat bajo la ruta /opt/tomcat9/apache-tomcat-9.0.37, que es el directorio que se ha descomprimido y descargado de Internet. A partir de esa localización se encuentran los siguientes directorios:

- ✓ *bin:* es el directorio que contiene los ficheros binarios y los scripts de inicio, tanto en sistema Windows como en Linux. Los dos ficheros importantes en Linux son startup.sh (inicio de Apache-Tomcat) y shutdown.sh (Parada de Apache-Tomcat).
- ✓ *conf:* este directorio es uno de los más importantes, ya que permite la configuración global del servidor de aplicaciones. A continuación, se listarán y explicarán cada uno de estos archivos, cruciales para el funcionamiento de Apache-Tomcat:

  - ➢ *catalina.policy:* la política de seguridad de Apache-Tomcat se realiza a partir de este fichero y está relacionada con Java. Una entrada en este fichero sería de la siguiente forma:

```
// Example policy file entry

grant [signedBy <signer>,] [codeBase <code source>] {
  permission  <class>  [<name> [, <action list>]];
};
```

  - ➢ *catalina.properties:* es el fichero en el que se relacionan los ficheros .jar de Java para la clase Catalina. La información está relacionada con los paquetes de seguridad y las rutas de las clases cargadas. También contiene algunas configuraciones de las cadenas más usadas en la caché.
  - ➢ *context.xml:* este fichero xml contiene la información de contexto común a todas las aplicaciones web que se ejecuten en Tomcat. También permite localizar el archivo web.xml de cada una de las aplicaciones.
  - ➢ *jaspic-propierties.xml:* es un fichero que permite definir un módulo de autenticación diferente a JAAS, pero con sus peculiaridades.
  - ➢ *jaspic-propierties.xsd:* es el esquema XSD que define si es válido el fichero anterior xml, esto es, su estructura, sus componentes, qué tipos de datos, etc.
  - ➢ *logging.propierties:* este fichero permite configurar las funciones de logging de todas las aplicaciones y de la actividad del servidor. Es crucial para poder solucionar cualquier error en caso de fallo del servidor o de una aplicación.
  - ➢ *server.xml:* es el fichero principal de configuración de Tomcat, que contiene un gran número de parámetros, de ahí su complejidad. Por ello se van a numerar los parámetros que usan dentro de él:

Parámetros del server.xml

Marca	Descripción	Ejemplo
<server> </server>	Es la etiqueta principal que engloba a toda la configuración del fichero. Engloba a uno o más servicios, y contiene al atributo port, que permite definir el puerto por el que escucha.	<Server port="8005" shutdown="SHUTDOWN">
<Listener ….>	Para definir las extensiones JMX ("Java Management Extensions") que serán usadas por Apache-Tomcat. Tienen un atributo principal llamado className.	<Listener className="org. apache.catalina.startup. VersionLoggerListener"/>
<GlobalNamingResources> …… </GlobalNamingResources>	Esta marca define los recursos tipo JNDI para usarlos globalmente en el servidor de aplicaciones. Usa la etiqueta Resource para especificar la localización.	<Resource name="UserDatabase" auth="Container" type="org. apache.catalina.UserDatabase" description="User database that can be updated and saved" factory="org. apache.catalina.users. MemoryUserDatabaseFactory" pathname="conf/ tomcat-users.xml" />

[.../...]

<Service> ….. </Service>	Esta marca permite agrupar uno o más conectores. Si se teclea Catalina se ejecutará el servidor como independiente.	<Service name="Catalina">
<Connector />	Representa las conexiones TCP desde el exterior que serán abiertas cuando arranca el servidor. Uno de los principales es el HTTPConnector.	<Connector port="8090" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8444" />
<Engine> …… </Engine>	Se usan dentro de la marca Service o de Host. Se procesan las peticiones que llegan a la marca Connector y que la cabecera disponga el Host por defecto.	<Engine name="Catalina" defaultHost="localhost">
<Logger/>	Esta marca indicará dónde serán envíados los registros de logs.	<Logger className="org.apache. catalina.logger.FileLogger" directory="logs" prefix="localhost_log." suffix=".txt" timestamp="true"/>
<Host> …. </Host>	A partir de esta etiqueta se pueden definir varios hosts virtuales para atender las peticiones.	<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
<Context> …. <Context>	Indicará la ruta a partir de la cual se encontrarán las aplicaciones ejecutadas en Tomcat. Normalmente se encuentran debajo del directorio webapps.	

- *tomcat-users.xml:* este fichero contiene los usuarios, contraseñas y roles que serán usados para acceder al servidor Apache-Tomcat. Existen algunas líneas con comentarios que pueden ser descomentadas para ponerlas en funcionamiento.
- *tomcat-users.xsd:* es el esquema XSD que define si es válido el fichero anterior xml, esto es, su estructura, sus componentes, qué tipos de datos, etc.
- *web.xml:* es un fichero estándar para las aplicaciones web, común a todas las aplicaciones web, ya que posee la configuración global a todas ellas.

- ✓ *lib:* este directorio contiene todos los ficheros .jar usados en el servidor que corresponden a Tomcat, a JSP, etc.
- ✓ *logs:* el directorio donde se almacenarán los logs de Catalina y de las aplicaciones.
- ✓ *temp:* es el directorio donde se almacenan los temporales del servidor Tomcat.

- ✓ *webapps:* directorio que contiene todas las aplicaciones web. Solamente existe una que se denomina ROOT.
- ✓ *work:* directorio de almacenamiento temporal de ficheros, por ejemplo, de compilación.

Con relación a las variables de entorno que usa Apache-Tomcat en el arranque para que pueda funcionar este servidor, se pueden comentar las siguientes:

- CATALINA_BASE: es el directorio que representa la configuración de una instancia de Tomcat, normalmente coincide con el valor de la variable CATALINA_HOME. Solamente se diferencia en el caso de que existan varias instancias en la misma máquina.
- CATALINA_HOME: indica el directorio raíz de la instalación del servidor. En nuestro caso es /opt/tomcat9/apache-tomcat-9.0.37, en Windows podría ser algo parecido a C:\Program Files\ tomcat9\apache-tomcat-9.0.37.
- CATALINA_TMPDIR: es el directorio temporal de Apache-Tomcat donde se pueden almacenar ficheros de compilación, intermedios, etc.
- JRE_HOME: almacena la ruta donde se encuentran los ejecutables de Java que usa Apache-Tomcat.
- CLASSPATH: es el conjunto de rutas que usa Apache-Tomcat, sobre todo los ficheros .jar.

5.3. Administrar aplicaciones web

Una vez explicada la arquitectura de Apache-Tomcat es el momento de administrar las aplicaciones web que soporta el servidor. Para ello, una vez iniciado el servicio de Tomcat se debe teclear en el navegador la siguiente URL: http://localhost:8080/manager/html o http://IP:8080/manager/html (la IP que aparezca al ejecutar el comando ifconfig). También, se puede acceder mediante la URL http://localhost:8080 y después pulsando en el botón *Manager App*. De las dos formas se observará una pantalla como la de la figura 5.8.

Según se puede ver en la figura 5.8, está solicitando un usuario y una contraseña que es configurable dentro del fichero tomcat-users.xml. Por defecto, el acceso al administrador de aplicaciones está deshabilitado. Para entrar es necesario autenticar a un usuario. Para habilitar un usuario es necesario definirlo mediante el atributo username, password y roles. El rol que permite manejar las aplicaciones es manager-gui. Por lo tanto, si se añaden las siguientes líneas en el fichero de usuarios de Tomcat, esto permitiría administrar las aplicaciones del servidor:

```
<user username="tomcat" password="cat.123" roles="manager-gui"/>
```

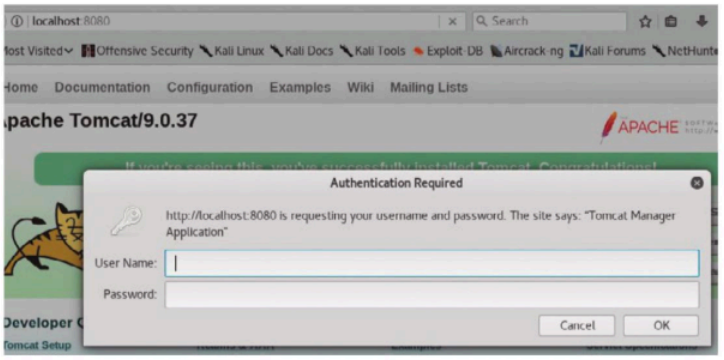


Figura 5.8
Aplicaciones

Una vez introducidas estas líneas en el fichero $CATALINA_HOME/conf/tomcat-users.xml, ya se puede introducir el usuario tomcat y la contraseña cat.123 que permitirá visualizar una pantalla como la de la figura 5.9.

Como se puede observar en la siguiente pantalla, se tiene una fila por cada aplicación desplegada con una serie de opciones (figura 5.10).

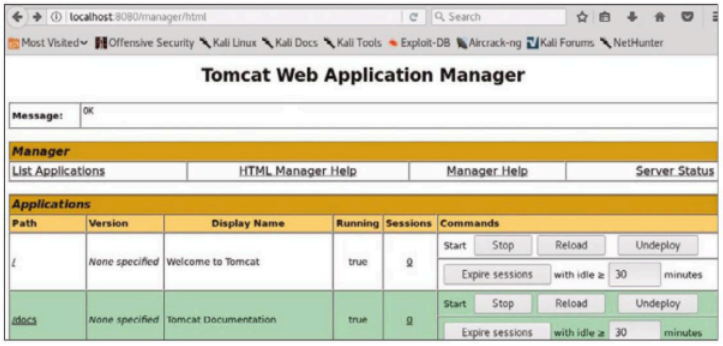


Figura 5.9
Administrar aplicaciones

- ✓ *Sessions:* si existe alguna sesión abierta, indica el número con la aplicación.
- ✓ *Commands:* son los comandos que nos permiten controlar la aplicación. Existen cuatro comandos en la parte de arriba de la columna. En nuestro caso, la aplicación está iniciada, y permite pararla con el botón de *Stop,* recargarla mediante *Reload* y eliminar el despliegue de la aplicación con *Undeploy.* Por último, permite controlar el tiempo que una sesión puede estar en espera, y a partir del cual se elimina la sesión, en nuestro caso es de 30 minutos.

En el siguiente apartado se puede observar el despliegue de una aplicación de forma automática y a partir de un fichero .war (figura 5.11).

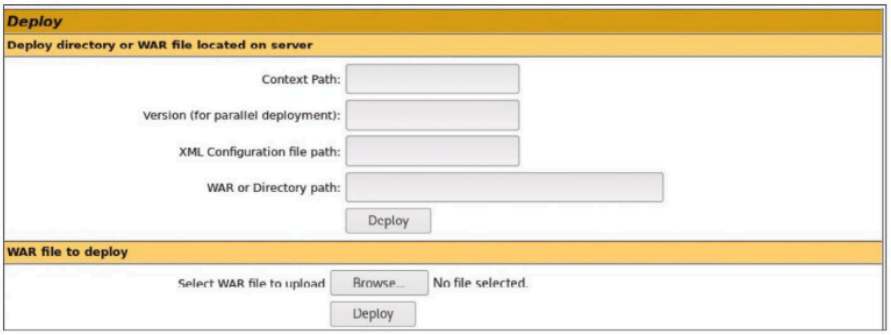


Figura 5.11
Despliegue de una aplicación

Si se observa, existen dos opciones para desplegar una aplicación, la primera forma indicando la ruta del contexto de la aplicación, la versión de la misma, la ruta del fichero .xml de la configuración y la ruta del directorio de la aplicación o el fichero .war.

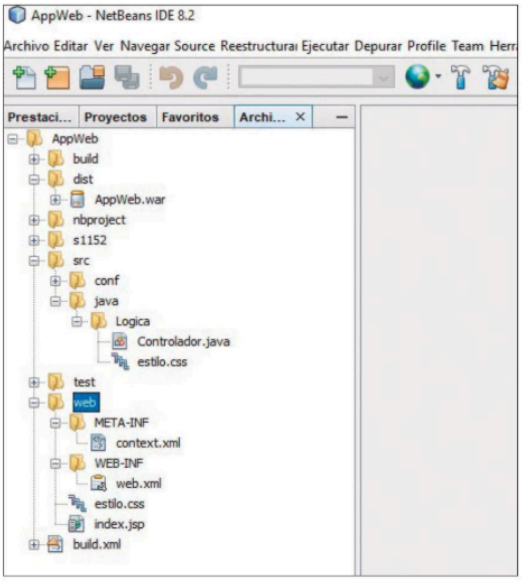
La segunda opción sería subiendo al administrador de aplicaciones el fichero .war. El fichero .war se puede generar de varias formas. La primera de ellas sería ubicarse en el directorio de la aplicación y posteriormente ejecutar el siguiente comando:

```
#jar –cvf app.war *
```

Las opciones del comando .jar son las siguientes

*c: crea un fichero sobre la salida estándar.
*d: visualiza la información sobre el fichero creado, tamaño, modificación, etc.
*f: este argumento permite especificar el fichero .jar que se va a procesar.

Otra forma sería a partir de un entorno de desarrollo de programación de Java, por ejemplo Netbeans IDE 8.2. Una vez que se tiene la aplicación web funcionando y probada, se ejecuta la opción *Ejecutar* y dentro de esta la opción *Limpiar y generar Proyecto*. Se generaría un fichero llamado como la aplicación terminado en .war dentro de la estructura de directorios de la aplicación en el directorio dist. Se puede observar en la figura 5.12.



Por último, se va a generar un fichero .war de la aplicación anterior y, a continuación, se va a desplegar para que se observe el funcionamiento. Se genera el fichero Appweb.war y se carga en la opción que existe en el administrador de aplicaciones en el apartado *WAR file to deploy*. Además, es necesario el permiso de admin-gui para poder desplegar aplicaciones.

```
<user username="tomcat" password="cat.123" roles="manager-gui, admin-gui"/>
```

Tal como se observa en la figura 5.13, y se pulsa el botón Deploy. Al desplegarla, se visualizará en el administrador de aplicaciones. Ahora, si se quiere comprobar que funciona, es necesario teclear la URL http://localhost:8080/AppWeb/ o http://IP_host(192.168.1.20):8080/AppWeb/ y se observará una imagen como la de la figura 5.14:

Figura 5.13
Despliegue

Figura 5.14
URL aplicación