UT4 - ADMINISTRACIÓN DE SERVIDORES DE APLICACIONES

Despliegue de Aplicaciones Web

2° DAW

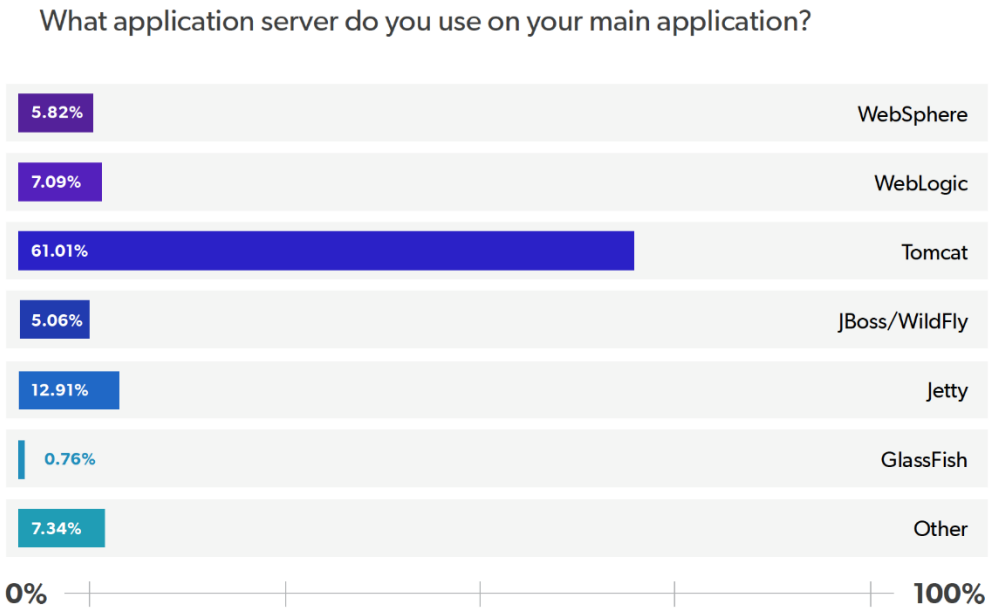
Servidores de aplicaciones web

Aplicaciones web vs páginas web

- A menudo el termino página web y aplicación web se confunden.
- Habitualmente se suelen diferenciar **en base a la tecnología** que utilizan. Es decir, si es HTML es un website, si usa Java en algunas de sus formas, es una aplicación web, por ejemplo.
- Esta definición es práctica pero **informal**. La diferencia fundamental es:
  - Una página web ofrece información.
  - Una aplicación web es interactiva.
- Por ejemplo, cuando entramos a la página de un restaurante que nos permite ver el menú, contactar mediante un formulario, ver la ubicación del restaurante, etc. estamos ante un **sitio web**.
- Sin embargo, cuando entramos a una página de un restaurante y, además de ver la información estática, podemos hacer una reserva online, hacer un presupuesto personalizado en base a la carta, hacer pedidos online o subir una foto de tu última visita, es una **aplicación web**.

Servidores de aplicaciones web

- Los servidores de aplicaciones web se suelen dividir en categorías en base a la tecnología para la que están diseñados.
- Así, por ejemplo los servidores **Java** más habituales son:

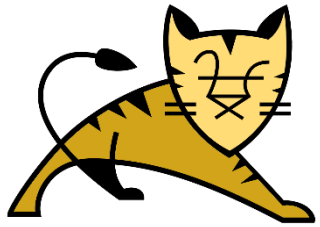


https://www.jrebel.com/blog/2020-java-technology-report

Servidores de aplicaciones web

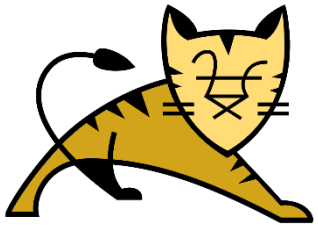
- Los servidores de aplicaciones son frameworks que permiten crear, mantener y servir aplicaciones web.
- Tienen una API que permite desarrollar páginas web dinámicas e interactivas.
- Se suelen dividir según las tecnologías soportadas. Estas tecnologías suelen estar agrupadas en bloques.
  - Tecnologías Java.
  - Tecnologías Microsoft (.NET + ASP).
  - Tecnologías PHP.
  - Tecnologías móviles.

Tomcat



- Tomcat es un servidor que implementa un contenedor web J2EE (Servlet/JSP).
- **Proporciona también un servidor HTTP** (escrito en Java).
- Servidor "**Open Source**". Tiene muchos colaboradores que contribuyen a su desarrollo en todo el mundo.
- **Multiplataforma:** Windows, Linux, Unix
- Desarrollado y mantenido por Apache Software Foundation.
- Descrita como la implementación de referencia de Java Servlet y JSP.
- Puede usarse con su propio servidor web o integrado con otros como Apache Web Server o Internet Information Services.

Tomcat. Instalación



- Deberemos comprobar la versión de Java con el comando **java –version**
  - En caso de que no esté instalado utilizar **sudo apt-get install default-jdk**
- Para instalar Tomcat 9: **sudo apt-get install tomcat9**
- Instalamos también herramientas que ofrecen aplicaciones web para documentar, probar y administrar Tomcat: **sudo apt-get install tomcat9-admin tomcat9-docs tomcat9-examples**
- Para iniciar, parar o reiniciar el servidor se utilizan los siguientes comandos:
  - **sudo systemctl start tomcat9**
  - **sudo systemctl stop tomcat9**
  - **sudo systemctl restart tomcat9**
- Para probar que se ha instalado correctamente se puede acceder a http://127.0.0.1:8080
- Los archivos de configuración estarán por defecto en la carpeta /etc/tomcat9
- Para usar la aplicación web tomcat9-admin hay que añadir las siguientes líneas en el archivo /etc/tomcat9/tomcat-users.xml (sustituyendo su nombre de usuario y contraseña):

```
<role rolename="manager-gui"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>
```

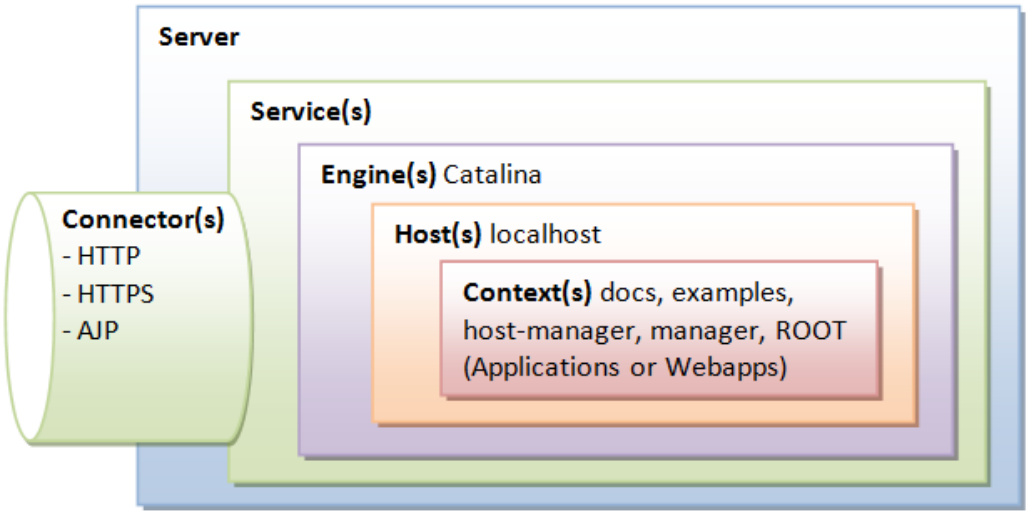
- Para usar la aplicación host-manager hay que incluir también el rol “admin-gui”

Tomcat. Instalación

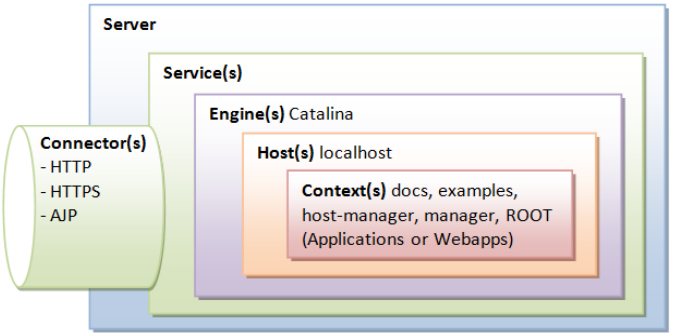
- En la instalación se han creado las siguientes ubicaciones:
  - /usr/share/tomcat9 (se denominará CATALINA_HOME)
  - /var/lib/tomcat9 (se denominará CATALINA_BASE)
- Edita el fichero /etc/environment y establece las variables CATALINA_HOME y CATALINA_BASE:
  - **CATALINA_HOME = /usr/share/tomcat9**
  - **CATALINA_BASE = /var/lib/tomcat9**
- Reiniciar el equipo para que las variables de entorno tomen valor
- Comprobamos que el servidor está iniciado y escuchando en el puerto 8080/TCP.
  - netstat –ltun
- Consulta el contenido del directorio $CATALINA_BASE/webapps/ROOT
- Consulta el contenido del fichero $CATALINA_BASE/webapps/ROOT/index.html

Tomcat. Arquitectura

- Tomcat esta formado por varios componentes que se organizan de forma **jerárquica**.
- Una instancia de Tomcat o servidor (Server) es el componente más alto de la jerarquía.
- Consiste en un grupo de aplicaciones contenedoras
- La configuración de los contenedores se realiza en el archivo principal de configuración **server.xml**.
- Veamos cómo es su arquitectura:



Tomcat. Arquitectura



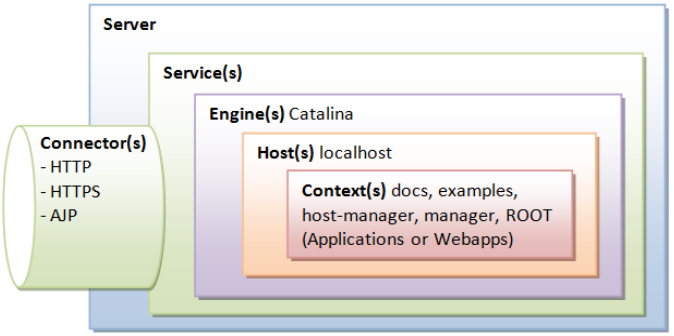
Server

- <Server>
- El primer elemento contenedor
- Puede contener uno o varios Services.

```
<Server port="-1" shutdown="SHUTDOWN">
  <Listener className="org.apache.catalina.startup.VersionLoggerListener" />
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
  <!--APR library loader. Documentation at /docs/apr.html -->
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />

  <!-- Global JNDI resources
       Documentation at /docs/jndi-resources-howto.html
  -->
  <GlobalNamingResources>
    <!-- Editable user database that can also be used by
         UserDatabaseRealm to authenticate users
    -->
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and saved"
```

Tomcat. Arquitectura



Service (s)

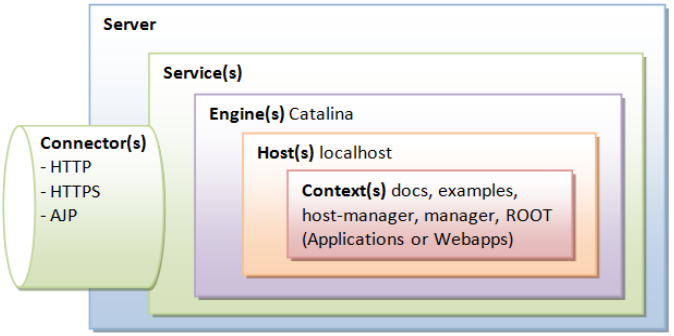
- <Service>
- Asocia uno o mas Connectors con un único Engine

```
<Service name="Catalina">

  <!--The connectors can use a shared executor, you can define one or more named thread pools-->
  <!--
  <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
      maxThreads="150" minSpareThreads="4"/>
  -->


  <!-- A "Connector" represents an endpoint by which requests are received
       and responses are returned. Documentation at :
       Java HTTP Connector: /docs/config/http.html
       Java AJP  Connector: /docs/config/ajp.html
       APR (HTTP/AJP) Connector: /docs/apr.html
       Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
  -->
  <Connector port="8080" protocol="HTTP/1.1"
             connectionTimeout="20000"
             redirectPort="8443" />
  <!-- A "Connector" using the shared thread pool-->
  <!--
  <Connector executor="tomcatThreadPool"
             port="8080" protocol="HTTP/1.1"
             connectionTimeout="20000"
             redirectPort="8443" />
  -->
```

Tomcat. Arquitectura

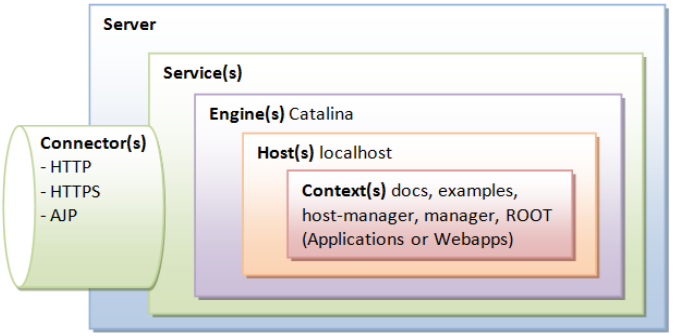


Connector (s)

- <Connector>
- Un conector es una asociación con puerto IP para manejar las peticiones y las respuestas con los clientes
- Por defecto hay creado un conector HTTP

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

Tomcat. Arquitectura



Engine

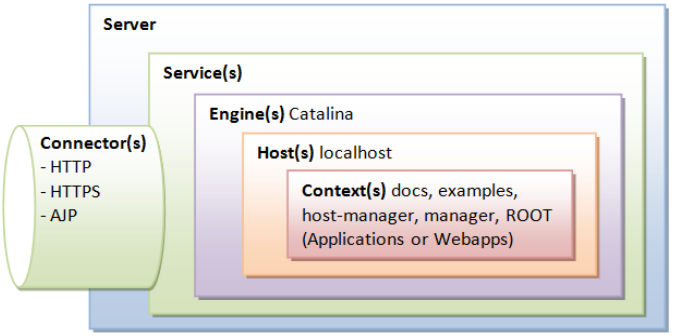
- <Engine>
- Contenedor de un o mas Hosts.
- Es posible configurar Virtual Hosts (como en Apache). Recibe las peticiones de los conectores y las traslada al Host correspondiente

```
<Engine name="Catalina" defaultHost="localhost">

  <!--For clustering, please take a look at documentation at:
      /docs/cluster-howto.html  (simple how to)
      /docs/config/cluster.html (reference documentation) -->
  <!--
  <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
  -->

  <!-- Use the LockOutRealm to prevent attempts to guess user passwords
       via a brute-force attack -->
  <Realm className="org.apache.catalina.realm.LockOutRealm">
    <!-- This Realm uses the UserDatabase configured in the global JNDI
         resources under the key "UserDatabase".  Any edits
         that are performed against this UserDatabase are immediately
         available for use by the Realm.  -->
    <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
           resourceName="UserDatabase"/>
  </Realm>
```

Tomcat. Arquitectura



Host (s)

- <Host>
- Define un servidor virtual (Virtual Host).
- Puede contener un o más aplicaciones web (webapp). Cada una de ellas se representa por un Context.
- Por defecto hay configurado un host llamado localhost.
- Las aplicaciones del host se colocan en $CATALINA_BASE\webapps

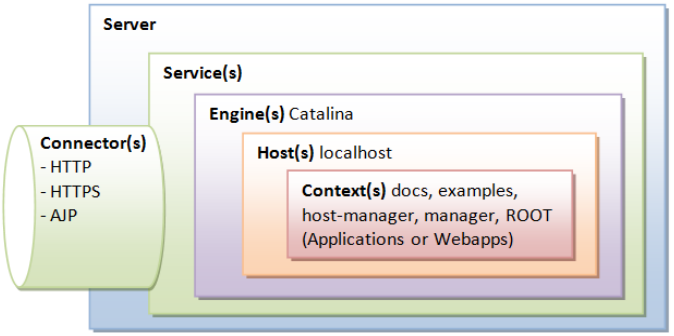
```
      <Host name="localhost"  appBase="webapps"
            unpackWARs="true" autoDeploy="true">

        <!-- SingleSignOn valve, share authentication between web applications
             Documentation at: /docs/config/valve.html -->
        <!--
        <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
        -->

        <!-- Access log processes all example.
             Documentation at: /docs/config/valve.html
             Note: The pattern used is equivalent to using pattern="common" -->
        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
               prefix="localhost_access_log" suffix=".txt"
               pattern="%h %l %u %t &quot;%r&quot; %s %b" />

      </Host>
    </Engine>
  </Service>
</Server>
```

Tomcat. Arquitectura



Context (s)

- <Context>
- Cada context representa una aplicación web (HTML, CSS, Servlets, JSP,...) ejecutándose dentro de un Host.

Tomcat. Estructura de directorios

CATALINA_BASE (/var/lib/tomcat9)

- /**conf**: ficheros de configuración (xml) y DTDs relacionados.
- /**logs**: ficheros de logs
- /**policy**: políticas de seguridad sobre la JVM
- /**webapps**: directorio con las aplicaciones desplegadas
- /**work**
  - Directorio de trabajo donde Tomcat almacena los Servlets generados a partir de JSPs.
  - Se organizan jerárquicamente en directorios con la siguiente estructura:
    - Engine(Catalina)/host(localhost)/webapp(examples)/estructura del paquete de clases.

```
daw@daw-ubuntu:/var/lib/tomcat9$ cd $CATALINA_BASE
daw@daw-ubuntu:/var/lib/tomcat9$ ls -l
total 12
lrwxrwxrwx 1 root   root     12 sep 11 21:47 conf -> /etc/tomcat9
drwxr-xr-x 2 tomcat tomcat 4096 sep 11 21:47 lib
lrwxrwxrwx 1 root   root     17 sep 11 21:47 logs -> ../../log/tomcat9
drwxr-xr-x 2 root   root   4096 nov  4 11:08 policy
drwxrwxr-x 3 tomcat tomcat 4096 nov  3 20:39 webapps
lrwxrwxrwx 1 root   root     19 sep 11 21:47 work -> ../../cache/tomcat9
```

Tomcat. Estructura de directorios

CATALINA_HOME (/usr/share/tomcat9)

- **/bin**
  - Scripts de inicio y parada.
  - Ficheros .jar requeridos para que Tomcat inicie.
- **/lib**
  - Ficheros .jar con librerías compartidas por todos los componentes de Tomcat.
  - Todas las aplicaciones puede acceder a estas librerías.
  - Incluye el API de Servlet y el API de JSP.

```
daw@daw-ubuntu:/var/lib/tomcat9$ cd $CATALINA_HOME
daw@daw-ubuntu:/usr/share/tomcat9$ ls -l
total 20
drwxr-xr-x 2 root root 4096 nov  3 20:39 bin
-rw-r--r-- 1 root root  944 feb  5  2019 default.template
drwxr-xr-x 2 root root 4096 nov  3 20:39 etc
drwxr-xr-x 2 root root 4096 nov  3 20:39 lib
-rw-r--r-- 1 root root  133 feb  5  2019 logrotate.template
```

Tomcat. Despliegue de aplicaciones

- Se trata de instalar una aplicación web en el servidor Tomcat (puede ser desarrollada por nosotros o un WAR cualquiera).
- Un **WAR** (*Web Application Resource*) es un archivo que aglutina los recursos de una web (archivos JAR, JSP, clases, Servlets, páginas estáticas, imágenes, etc.).
- El despliegue se puede realizar **antes** de arrancar el servidor o con el **servidor arrancado**, bien modificando los archivos, bien utilizando la aplicación web Tomcat Manager.
- Tomcat Manager ofrece dos interfaces, uno mediante una API de servicios web y otra gráficamente mediante HTML.
- Formas de realizar un despliegue en Tomcat:
  - Despliegue manual:
    - En estático (servidor parado)
    - En dinámico (servidor funcionando)
  - Despliegue asistido:
    - Usando Tomcat Manager
    - Integrándose con Ant o Maven (gestores de proyectos Java con diferentes características)

Tomcat. Despliegue de aplicaciones

- Despliegue en estático (antes de arrancar el servidor)
  - Permite desplegar aplicaciones sin usar Tomcat Manager
  - Se debe desplegar la aplicación en el **appBase** (una ubicación específica para cada host).
  - Se puede copiar la carpeta del proyecto sin comprimir dentro de esa ubicación o bien el .WAR comprimido.
  - Las aplicaciones en la ubicación **appBase** del host (el host por defecto es "localhost") se desplegarán en Tomcat al inicio del mismo si el **Host** tiene configurada la opción **unpackWARs** a "true".

```
<Host name="localhost"  appBase="webapps"
      unpackWARs="true" autoDeploy="true">
```

  - En tal caso, durante el arranque se desplegarán primero los "Context Descriptors". Tras ellos, las aplicaciones sin "Context Descriptor" y finalmente los archivos .WAR.

Tomcat. Despliegue de aplicaciones

- Despliegue en un servidor activo
  - Para poder hacer un despliegue mientras el servidor está funcionando debemos activar la opción en el host llamada **autoDeploy**. Cuando esta opción tiene valor true el Host tratará de cargar dinámicamente aplicaciones buscando, por ejemplo, archivos .WAR nuevos en su ubicación appBase.
  - Esta opción va a permitirnos:
    - Desplegar copiando archivos .WAR en el appBase del host.
    - Modificar el despliegue de una aplicación que fue desplegada con un .WAR copiando un nuevo archivo .WAR. En tal caso lo que sucederá es que el Host borrará la carpeta de la aplicación antigua y descomprimirá el contenido del archivo .WAR en una nueva carpeta con el mismo nombre. Esto solo sucederá si la opción del Host "unpackWARs" tiene valor true.

Tomcat. Configuración de Hosts y Contexts

- Configuración de Hosts
  - Fichero server.xml
    - name: nombre
    - appBase: directorio base de las aplicaciones del host. Relativo a CATALINA_BASE.
    - unpackWars
      - true → se descomprime el WAR.
      - false → no se descomprime el WAR, se sirve directamente (menos eficiente).
    - autoDeploy
      - Tomcat chequea periódicamente si hay nuevos archivos WAR y los despliega/redespliega si detecta cambios.

Tomcat. Configuración de Hosts y Contexts

- Configuración de Contexts
  - Cada Context se corresponde con una aplicación. En él se establece la ubicación y acceso a las aplicaciones.
  - Directivas de configuración:
    - path: define el path de acceso a la aplicación.
    - docBase: define donde se despliega la aplicación.
    - reloadable: si su valor es true Tomcat monitoriza la aplicación y la recarga si detecta cambios en las sus clases (WEB-INF/lib y WEB-INF/class).
  - Puede ser configurado en:
    - Fichero **server.xml** dentro de Host.
      - Hay que reiniciar el servidor para que los cambios tengan efecto.
    - En el fichero CATALINA_BASE/conf/ENGINE_NAME/HOST_NAME/**CONTEXT_NAME.xml**
    - Como parte de **aplicación** web en /METAINF/context.xml.
    - **Implícitamente** (no definir fichero de configuración), la aplicación se despliega automáticamente en webapps.

Hoja04_Tomcat_03

Tomcat. Tomcat Web Manager

- Aplicación web para administrar las aplicaciones en Tomcat.
- Facilita el despliegue de aplicaciones sin necesidad de acceso SSH, FTP, SFTP, … al servidor.
- Características:
  - Listar aplicaciones desplegadas.
  - Iniciar/detener aplicaciones.
  - Desplegar aplicaciones web.
  - Recargar aplicaciones que se han modificado.
  - Listar sesiones activas de las aplicaciones.
  - Mostrar estadísticas de sesiones.
  - Listar propiedades del SO y de la JVM.
  - En su descriptor de despliegue (web.xml) se pueden modificar su configuración de seguridad
    - Tipos de autenticación
    - Roles
    - …

Tomcat. Tomcat Web Manager

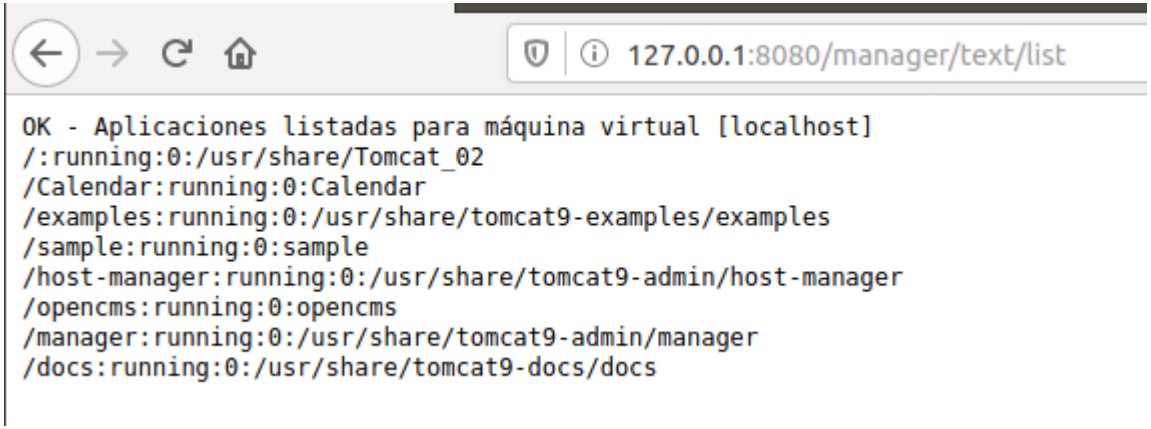
- Para instalarlo en Linux hacemos **sudo apt-get install tomcat9-admin**
- Permite los siguientes accesos:
  - Navegador Web.
    - /manager/html
  - Interfaz en modo texto (comandos, ANT, …).
    - /manager/text
  - Acceso vía JMX (Java Mangement Extension).
    - /manager/jmproxy
- Permite consultar el estado del servidor
  - /manager/status

Tomcat. Tomcat Web Manager

- **Interfaz basado en texto**
- Tomcat Web Manager ofrece una interfaz en modo texto que interpreta comandos.
- Para acceder hay que configurar un usuario con el rol **manager-script** en **tomcat-users.xml**

```
<role rolename="manager-script"/>
<user username="tomcatscript" password="tomcatscript" roles="manager-script"/>
```

- Los comandos se envían a través de HTTP.
- Las respuestas se envían en texto plano.



Tomcat. Tomcat Web Manager

- Formato general de peticiones:

```
http://{host}:{port}/manager/text/{command}?{parameters}
```

- Ejemplos de comandos
  - list, start, stop, install, deploy, sessions, …

Tomcat. Despliegue de aplicaciones con Ant

Apache Ant

- Herramienta de automatización de tareas que ayuda al desarrollador en la integración de trabajo involucrado en el proceso de desarrollo de software.
- http://ant.apache.org
- Es posible programar Ant para que interactúe con el interfaz en modo texto de Tomcat Web Manager para desplegar aplicaciones.

Configuración de Ant

- Instalación (en Eclipse está integrado).
- Instalar librerías de Tomcat para Ant.
  - Se distribuyen con Tomcat:
    - CATALINA_HOME/lib/catalina.jar
    - CATALINA_HOME/lib/catalina-ant.jar
    - CATALINA_HOME/lib/ tomcat-coyote.jar
    - CATALINA_HOME/lib/tomcat-util.jar
    - CATALINA_HOME/bin/ tomcat-juli.jar
  - Copiar en (o importar en el fichero buid.xml)
    - ANT_HOME/lib/
  - Crear el fichero build.xml
    - Estructura del proyecto.
    - Tareas (Tasks) a realizar.

Tomcat. Autenticación y autorización

Tipos de autenticación

- Basic

```
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Acceso al curso</realm-name>
</login-config>
```

- Digest
- Formularios

```
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>Compras</realm-name>
    <form-login-config>
        <form-login-page>/WEB-INF/seguro/login.jsp</form-login-page>
        <form-error-page>/WEB-INF/seguro/login-error.jsp</form-error-page>
    </form-login-config>
</login-config>
```

- Certificados digitales

Usuarios y roles

- Los usuarios de las aplicaciones web en Tomcat se identifican con **usuario/password**.
- Los usuarios son asignados a roles.
- Tomcat concede acceso a las aplicaciones a los roles en lugar de a usuarios individuales.

Tomcat. Autenticación y autorización

- Un **Realm** es un archivo, base de datos o servicio de directorio que contiene una colección de usuarios/passwords y roles.
- Elemento **<Realm>** para configurarlos:
  - Dentro de **<Engine>**
    - Será compartido por todas las aplicaciones de todos los virtual host a menos que sea sobrescrito en **<Host>** o **<Context>** subordinado.
  - Dentro de **<Host>**
    - Será compartido por todas las aplicaciones de ese virtual host a menos que sea sobrescrito en un **<Context>** subordinado
  - Dentro de un **<Context>**
    - Solo para esa aplicación.
- **Sólo puede existir un Realm activo para una aplicación en un momento dado**
- En la configuración inicia del Tomcat está definido un **Realm** a nivel de **<Engine>.**

Tomcat. Autenticación y autorización

- Tipos de Realms:
  - **MemoryRealm**
    - Acceso a información almacenada en un fichero (normalmente tomcat-users.xml).
  - **UserDatabaseRealm**
    - Acceso a información almacenada en un fichero Autenticación y autorización (normalmente tomcat-users.xml) vía JNDI.
  - **JDBCRealm**
    - Acceso a la información de autenticación almacenada en una base de datos relacional a través de un controlador JDBC.
  - **DataSourceRealm**
    - Acceso a la información de autenticación almacenada en una base de datos relacional a través de JNDI.
  - **JNDIRealm**
    - Acceso a la información almacenada en un servicio de directorio (LDAP) a través de JNDI.
  - **JaasRealm**
    - Acceso a un servidor JAAS (Security authentication using Java Authentication).
  - **CombinedRealm**
    - Permite el uso de múltiples Realms simultáneamente.
  - **LockOutRealm**
    - Extiende CombinedRealm para bloquear usuario con varios intentos de login fallidos.

Tomcat. MemoryRealm

- La información de usuarios y roles almacenada en un fichero que se carga en memoria al iniciar Tomcat.
- Por defecto el fichero es tomcat-users.xml.
- Configuración:
  1. Definir el fichero (por defecto tomcat-users.xml) con los usuario y roles.

```
<role rolename="appTomcat_02"/>
<user username="alumno" password="alumno" roles="appTomcat_02"/>
<user username="profesor" password="profesor" roles="appTomcat_02"/>
```

  2. Configurar el Realm en el ámbito que se considere más adecuado (<Engine>, <Host>, <Context>, …)

```
<Context>
        <Realm className="org.apache.catalina.realm.MemoryRealm" />
</Context>
```

Tomcat. MemoryRealm

3. Proteger el recurso (en el descriptor de despliegue web.xml de la aplicación).

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>MemoryRealm</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>appTomcat_02</role-name>
    </auth-constraint>
</security-constraint>
```

4. Configurar el tipo autenticación (en el descriptor de despliegue web.xml de la aplicación).

```
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Acceso a la app Tomcat_02</realm-name>
</login-config>
```

Tomcat. JDBCRealm

- La información de usuarios y roles es almacenada en una base de datos.
- El acceso a la base de datos se hace utilizando JDBC.
- Configuración:
  1. Crear la base datos para almacenar los usuarios y roles.
  2. Insertar los usuarios y roles.
  3. Configurar el Realm en el ámbito que se considere más adecuado (<Engine>, <Host>, <Context>, …)

```
<Context>
        <Realm  className="org.apache.catalina.realm.JDBCRealm"
                        driverName="com.mysql.jdbc.Driver"
                        connectionURL="jdbc:mysql://localhost/tomcatusers?user=tomcatusers&amp;password=tomcatusers"
                        userTable="usuarios"
                        userNameCol="nombre_usuario" userCredCol="password"
                        userRoleTable="roles"
                        roleNameCol="nombre_rol" />
</Context>
```

Tomcat. JDBCRealm

4. Proteger el recurso (en el descriptor de despliegue web.xml de la aplicación)

```
<security-constraint>
        <web-resource-collection>
                <web-resource-name>JDBCRealm</web-resource-name>
                <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
                <role-name>tienda</role-name>
        </auth-constraint>
</security-constraint>
```

5. Configurar el tipo autenticación (en el descriptor de despliegue **web.xml** de la aplicación)

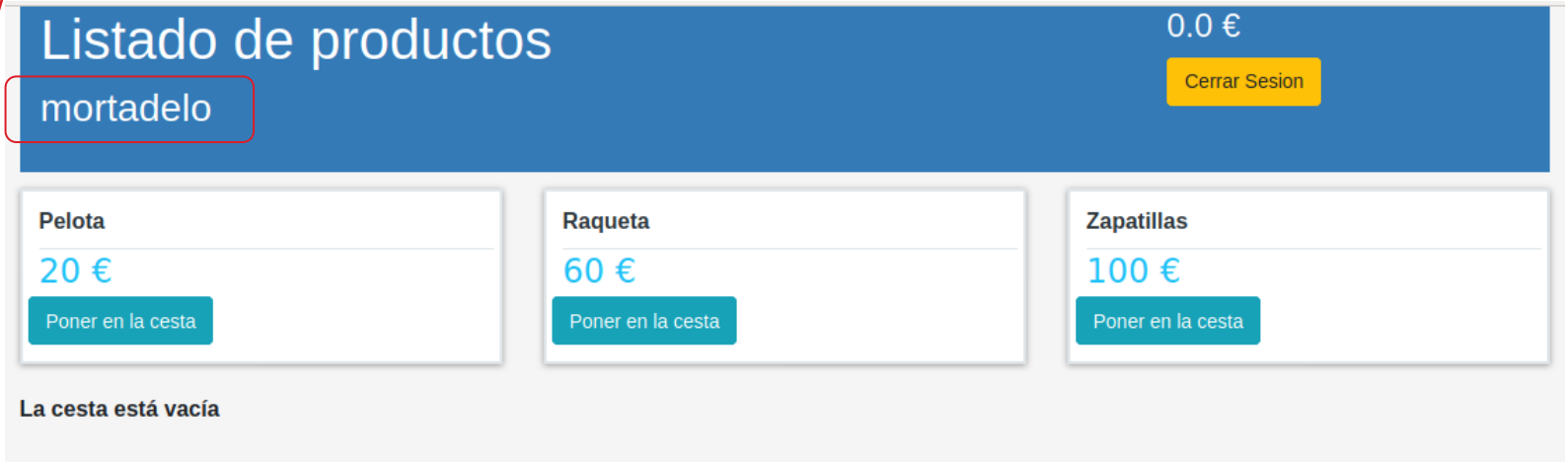
```
<login-config>
        <auth-method>FORM</auth-method>
        <realm-name>Tiendas</realm-name>
        <form-login-config>
                <form-login-page>/WEB-INF/seguro/login.jsp</form-login-page>
                <form-error-page>/WEB-INF/seguro/login-error.jsp</form-error-page>
        </form-login-config>
</login-config>
```

Tomcat. Acceso a usuarios autenticados

- Cuando un usuario se ha autentificado es posible acceder a él desde las aplicaciones porque su información se almacena en el objeto HttpServletRequest

```
<div class="col-md-9 col-sm-12 col-xs-12">
    <h1>Listado de productos</h1>
    <h3><%= request.getRemoteUser() %></h3>
</div>
```

- ¡Pruébalo!



Tomcat. Valves y filtros

- Tecnologías para interceptar y pre-procesar peticiones (request) y respuestas (response) HTTP con independencia de las aplicaciones.
  - **Valves**
    - Tecnología propietaria de Tomcat.
  - **Filtros**
    - Tecnología definida en el API de Servlets (desde la versión 2.3).
- Ejemplos de utilización:
  - Generar logs de las peticiones realizadas a una aplicación o Servlet.
  - Gestionar la seguridad permitiendo accesos sólo a determinadas IPs.
  - Comprimir datos antes de enviarlos al cliente.
  - Identificar la localización (país, lenguaje, …) de peticiones y responder en consecuencia

Tomcat. Valves

- Tecnología propietaria de Tomcat.
- Objetos que permite interceptar y pre-procesar peticiones y respuestas HTTP.
- Creación:
  - Interface
    - org.apache.catalina.Valve
  - Clase abstracta para implementar Valves.
    - org.apache.catalina.valves.ValveBase
  - Existen implementaciones predefinidas de Valves en Tomcat (RemotrAddrValve, RemoteHostValve, AccessLogValve, …)
- http://tomcat.apache.org/tomcat-9.0-doc/config/valve.html

Tomcat. Valves

- Elemento <Valve> para configurarlos.
  - Dentro de <Engine>
    - Procesará las peticiones de todas las aplicaciones de todos los virtual host.
  - Dentro de <Host>
    - Procesará las peticiones de todas aplicaciones de ese virtual host
  - Dentro de un <Context>
    - Procesará las peticiones de esa aplicación.
- Observa que se puede configurar un Valve para múltiples aplicaciones.

Tomcat. Filtros

- Definidos en el API de Servlets (desde la versión 2.3).
- Objetos que permite interceptar y pre-procesar peticiones y respuestas HTTP.
- **Creación**
  - Interface
    - javax.servlet.Filter
  - Existen implementaciones predefinidas de Filters en Tomcat (RemoteIPFilter, ExpiresFilter, RequestDumperFilter, …).
- http://tomcat.apache.org/tomcat-9.0-doc/config/filter.html
- Los filtros se configuran e inicializan **en el descriptor de despliegue** de cada aplicación.
- Es posible reutilizar filtros en varias aplicaciones pero cada filtro debe ser configurado en cada aplicación por separado.

Tomcat. Host virtuales

- Tomcat permite alojamiento virtual (virtual hosting).
- Recuerda
  - <Engine>
    - Contenedor de un o mas Hosts.
    - Es posible configurar Virtual Host (como en Apache).
    - Recibe las peticiones de los conectores y las traslada al Host correspondiente.
  - <Host>
    - Define un servidor virtual (Virtual Host).
    - Puede contener un o mas aplicaciones web (webapp).
    - Cada una de ellas se representa por un Context.
- Configuración
  - Crear el directorio donde se almacenarán las aplicaciones del Host.
  - Definir el <Host> dentro de <Engine> en el fichero server.xml.

```
<Host name="ivan.com"  appBase="webapps-ivan"
     unpackWARs="true" autoDeploy="true">

 <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
        prefix="ivan_access_log" suffix=".txt"
        pattern="%h %l %u %t &quot;%r&quot; %s %b" />

</Host>
```

Tomcat. Seguridad SSL

- Es posible configurar Tomcat para que sirva contenidos seguros usando el protocolo SSL.
- Existen dos posibilidades de configuración:
  - Utilizar la implementación SSL de Java (Java SSL).
  - Utilizar la implementación nativa ARP (OpenSSL).
- **Implementación SSL de Java**
  1. Crear un almacén de claves con un certificado SSL (usando la herramienta Keytool)
  2. Configurar un conector SSL en Tomcat.
- Para obligar a que una aplicación sólo responda a peticiones HTTPS hay que realizar la siguiente configuración en su descriptor de despliegue:

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>MemoryRealm</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>curso</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```