INAP
INSTITUTO NACIONAL DE ADMINISTRACIÓN PÚBLICA

Curso online: **Instalación, Configuración y Administración de Apache + Tomcat**

Módulo 3. Apache Tomcat

Capítulo 4. Configuración de Apache Tomcat

Autores

Janine García Morera

Alexandra López de la Oliva Portugués

Julio Villena Román

Octubre de 2014

Índice de contenidos

CAPÍTULO 4 CONFIGURACIÓN DE APACHE TOMCAT

4.1. Introducción

La configuración de Apache Tomcat se realiza principalmente modificando el fichero `server.xml`, de manera parecida al fichero `httpd.conf` de Apache.

La estructura del fichero `server.xml` es, sin embargo, mucho más rígida que la del fichero `httpd.conf`: sigue una estructura de componentes anidados, siguiendo una relación de pertenencia.

Algunos de estos elementos son de definición obligatoria, aunque en una instalación por defecto no es necesario modificarlos.

En versiones anteriores de Tomcat, el fichero `server.xml` contenía toda la información sobre los contextos de las aplicaciones desplegadas. A partir de la versión 5.5 esta información se extrae de este fichero (aunque se permite también que se incluya) y se distribuye a lo largo de la estructura de directorios de Tomcat.

Toda la información de este capítulo se encuentra disponible en la documentación de Tomcat, en:

<tomcat.apache.org/tomcat-7.0-doc/config/index.html>

4.2. Objetivos

- Conocer la estructura del fichero básico de configuración de Tomcat.
- Conocer la configuración de los componentes de Tomcat.

4.3. Opciones de Configuración

Todos los ficheros de configuración de Tomcat son ficheros XML sin esquema. Los elementos y atributos son sensibles a mayúsculas y minúsculas. Tomcat se configura mediante el fichero `conf/server.xml`, que se considera el "corazón" de Tomcat.

La estructura lógica del fichero `server.xml` se muestra en la figura siguiente.

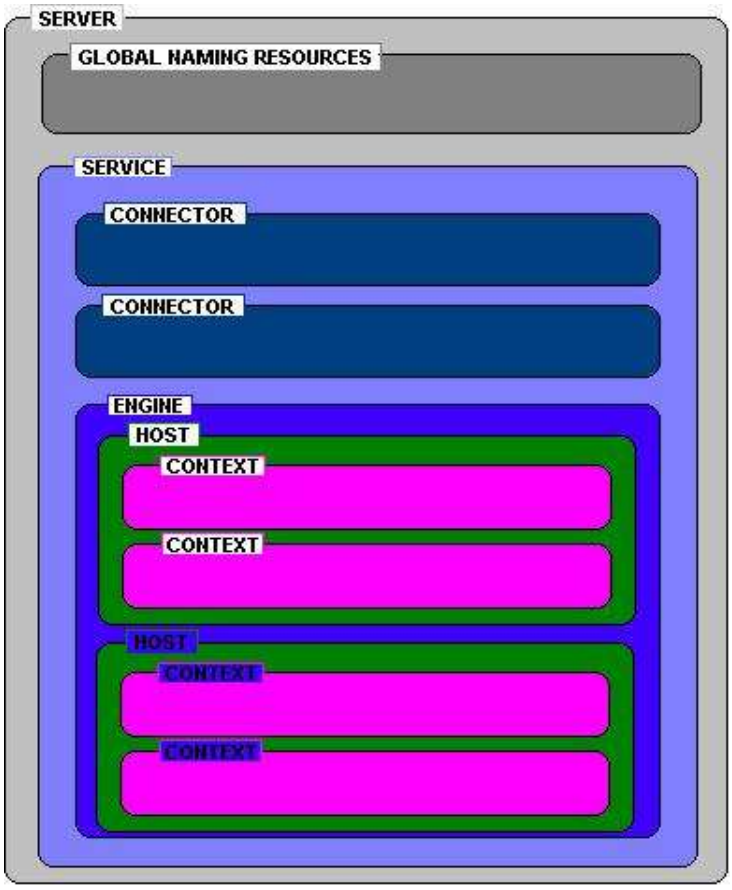

Figura 3.4.1: Estructura del fichero server.xml

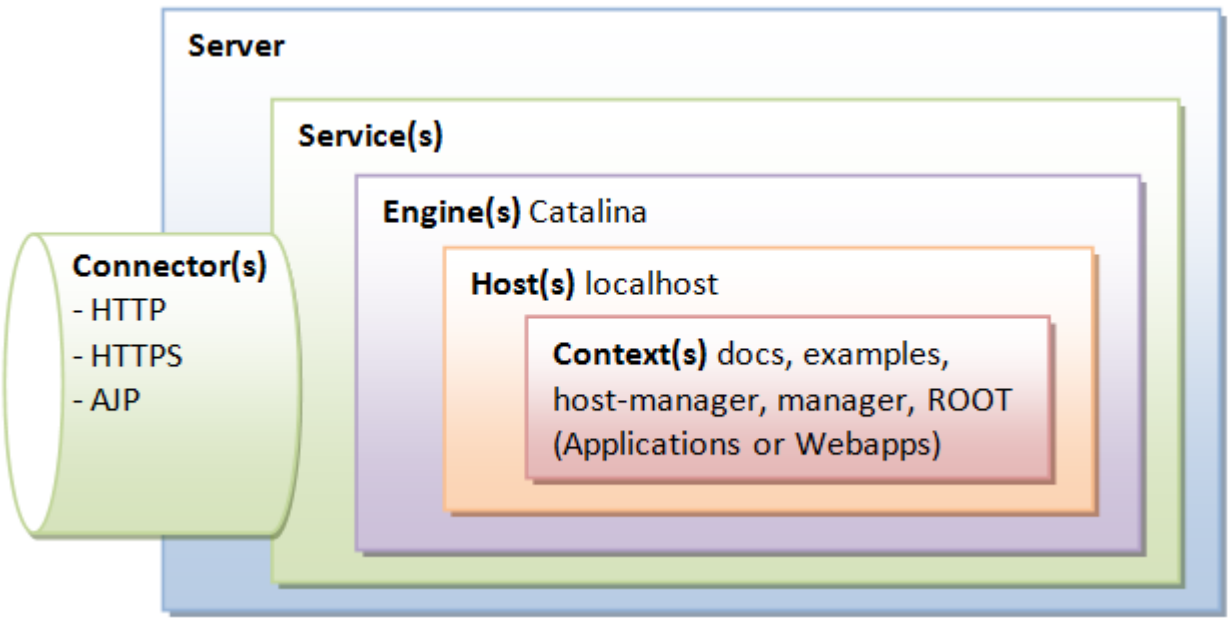

Figura 3.4.2: Estructura del fichero server.xml

El “esqueleto” XML de este fichero sería el siguiente:

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

Como cualquier otro documento en XML todo contenido entre `<!-- -->` es considerado un comentario, y por lo tanto cualquier parámetro que se encuentre entre estos caracteres no es utilizado por "Tomcat"; aquellos parámetros que no sean definidos dentro de `server.xml` son asignados un valor "Default" por Tomcat.

La descripción de los elementos de configuración se organiza en las siguientes categorías:

- **Elementos de primer nivel**. El elemento raíz del fichero de configuración es el elemento **<Server>**, mientras que **<Service>** representa un grupo de conectores asociados a un motor.
- **Conectores** (**Connector**). Representan la interface entre los clientes externos que envían las peticiones a (y reciben las respuestas de) un servicio en particular.
- **Contenedores** (**containers**). Representan componentes cuya función es procesar las peticiones entrantes y generar las correspondientes respuestas. Un motor (**Engine**) gestiona todas las peticiones para un servicio (Service), un **Host** gestiona todas las peticiones para un host virtual y un **contexto** gestiona todas las peticiones para una aplicación web específica.
- **Componentes anidados**. Representan elementos que pueden ser anidados dentro de otro elemento. Algunos elementos pueden ser anidados dentro de un contenedor mientras que otros sólo pueden ser anidados dentro de un contexto.

Aunque esta guía presentará un resumen de cada elemento, en la documentación de Tomcat, para cada elemento se presenta la siguiente información:

- **Introducción**. Breve descripción del componente en particular. Cada componente tiene su correspondiente interfaz Java en el paquete `org.apache.catalina`, que es implementada por una o más implementaciones estándar.
- **Atributos**. Conjunto de atributos (o parte de ellos) de este elemento. Puede haber atributos comunes y atributos específicos de una determinada implementación estándar (clase Java particular).

- **Componentes anidados**. Enumeración de qué componentes pueden ser anidados en ese elemento.
- **Características especiales**. Descripción de características especiales soportadas específicas de cada elemento.

4.4. Elementos de primer nivel

4.4.1. El elemento <Server>

Es el elemento de mayor nivel para una única instancia Tomcat. Representa el contenedor de servlets Catalina al completo. El resto de elementos deben encontrarse entre este elemento. Puede contener una o más instancias <Service>.

El elemento <Server> está definido por la interface `org.apache.catalina.Server`. Los atributos más importantes son los mostrados en la siguiente tabla.

Atributo	Descripción
address	Dirección IP donde el servidor espera el comando `shutdown`. Si no se especifica, se utiliza localhost por defecto
port	El puerto TCP donde el servidor espera el comando `shutdown`. Se pone a -1 si se quiere deshabilitar el comando. Esto funciona correctamente cuando Tomcat se arranca como servicio, pero no puede usarse cuando se arranca como aplicación porque deshabilitar el puerto impediría que `shutdown.bat|.sh` y `catalina.bat|.sh` detuvieran Tomcat correctamente.
shutdown	La cadena que debe recibir en el puerto indicado para cerrar Tomcat.

El siguiente código define los atributos del elemento <Server> en un fichero `server.xml` típico:

```
<Server port="8005" shutdown="SHUTDOWN" debug=”0”>
```

El atributo **debug** se puede utilizar para cualquier elemento de Tomcat. Fija el nivel de debug de los mensajes de traza a un fichero de log. Cuanto mayor sea el nivel de traza, mayor será el detalle de ésta.

El atributo **shutdown** es un string arbitrario que será enviado a la instancia de Tomcat en ejecución cuando se invoque el script `catalina` con el parámetro stop. Puesto que el fichero `server.xml` no debería ser visible fuera del servidor, se puede cambiar este string por otro, para mayor seguridad.

El elemento <**Server**> no puede ser configurado como hijo de ningún otro elemento, pero sí como padre de uno o más elementos <Service> y <GlobalNamingResources> (representa los recursos globales JNDI -Java Naming and Directory Interface- que se definen para el <Server>).

4.4.2. El elemento <Service>

Es el siguiente elemento del fichero `server.xml`. Contiene una colección de uno o más conectores que comparten un único elemento <**Engine**> al que pasan las peticiones entrantes en un determinado puerto y protocolo y reciben las respuestas de éste. Se pueden anidar varios elementos <Service> en un mismo elemento <Server>.

El elemento <Service> se define por la interface `org.apache.catalina.Service`, y los atributos más importantes son los mostrados en la siguiente tabla.

Atributo	Descripción
name	El nombre de visualización del servicio definido. Es el valor que se utiliza en los mensajes de log de Tomcat. El nombre de cada servicio asociado a un Server particular debe ser único.

La definición de un servicio Tomcat independiente (standalone) que maneja todas las peticiones directas recibidas por Tomcat podría ser:

```
<Service name="Catalina">
```

El elemento <Service> puede ser configurado como hijo del elemento <Server> y como padre de los elementos <Connector> (varios) y <Engine> (sólo uno).

4.5. Contenedores

4.5.1. El elemento <Engine>

Es el primer elemento contenedor en el fichero `server.xml`. Representa el mecanismo de procesamiento de las peticiones para un servicio (<Service>) dado. Cada elemento <Service> puede tener un único elemento <Engine>, y este único elemento <Engine> recibe y procesa todas las peticiones recibidas por todos los componentes <Connector> definidos. El elemento <Engine> debe incluirse inmediatamente después de los elementos <Connector>, en el elemento <Service> al que pertenece.

El elemento <Engine> se define por la interface `org.apache.catalina.Engine` y sus atributos más importantes se muestran a continuación.

Atributo	Descripción
defaultHost	El nombre del host al que se asignan todas las peticiones que no tienen una cabecera de host HTTP1.1 o que están dirigidas a un sitio web no configurado en este `Engine`. El nombre que se asigne debe ser uno de los elementos Host definidos para ese `Engine`.
name	Nombre de visualización del `Engine`, usado en mensajes de log y errores. Si se utilizando varios elementos `Service` en el mismo Server, es necesario asignar un nombre único.
startStopThreads	Número de hilos que se pueden arrancar en paralelo para cada elemento anidado de este `Engine`. Por defecto toma el valor de 1. Si se utiliza 0, se usará el valor `Runtime.getRuntime().availableProcessors()`.

La siguiente entrada del fichero `server.xml` define un engine llamado Catalina con un default host localhost:

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

El elemento <Engine> puede ser configurado como un hijo del elemento <Service> y como un padre de los siguientes elementos:

- <Host>. Se necesita definir al menos un host.
- <Realm>. Configura un realm (base de datos de usuarios, claves y roles) que podrá ser compartido entre todos los Hosts y Contextos subordinados, a no ser que se configure un `Realm` a un nivel inferior.

Adicionalmente, al ejecutar un servidor web, se generan logs de acceso, con una línea de información por cada petición atendida por el servidor, en un formato estándar. El elemento `Valve` permite especificar, entre otras cosas, la configuración de los log de acceso generados, por ejemplo:

```
<Engine name="Catalina" ...>
  ...
  <Valve className="org.apache.catalina.valves.AccessLogValve"
         prefix="catalina_access_log." suffix=".txt"
         pattern="common"/>
  ...
</Engine>
```

Ver "Access Logging Valves" (<tomcat.apache.org/tomcat-7.0-doc/config/valve.html#Access_Logging>) para más información sobre los parámetros soportados.

Además, el elemento `Valve` se puede utilizar también para filtrar las peticiones entrantes a un `Engine`, `Host` o `Context` por IP o nombre de host, utilizando filtros "accept" (aceptar) o "deny" (rechazar) basados en expresiones regulares estándar. Las peticiones rechazadas recibirán un error HTTP "Forbidden". Por ejemplo:

```
<Engine name="Catalina" ...>
  ...
  <Valve className="org.apache.catalina.valves.RemoteHostValve"
         allow=".*\.mycompany\.com|www\.yourcompany\.com"/>
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         deny="192\.168\.1\.\d+"/>
  ...
</Engine>
```

Para más información accede a *"Remote Address Filter"* (<tomcat.apache.org/tomcat-7.0-doc/config/valve.html#Remote_Address_Filter>) y *"Remote Host Filter"* (<tomcat.apache.org/tomcat-7.0-doc/config/valve.html#Remote_Address_Filter>).

4.5.2. El elemento <Host>

Cada elemento <`Host`> representa un host virtual que está contenido en cada instancia de un <`Engine`> de Tomcat. Un host virtual es una asociación de un nombre de red para un servidor (como puede ser `www.ejemplo.com`) con el servidor específico en el que se está ejecutando Tomcat.

En muchos casos se desea configurar más de un nombre para un mismo servidor (por ejemplo `www.mycompany.com` y `company.com`), para lo que se utiliza la característica `Host Name Aliases` (<tomcat.apache.org/tomcat-7.0-doc/config/host.html#Host_Name_Aliases>) descrita a continuación.

Cada <`Host`> puede ser padre de una o más aplicaciones Web, cada una de ellas representada por un componente <`Context`>. Debe definirse al menos un <`Host`> por cada elemento <`Engine`>. Este <`Host`> es normalmente llamado **localhost**.

La implementación estándar es `org.apache.catalina.core.StandardHost` y sus posibles atributos son:

Atributo	Descripción
appBase	El directorio base de las aplicaciones web desplegadas en este host. Se puede usar una ruta absoluta o relativa a `$CATALINA_BASE`. Si no se especifica, se utiliza `webapps`.
xmlBase	El directorio base de los ficheros XML de configuración de este servidor virtual. Si no se especifica, se toma `conf/<engine_name>/<host_name>`.
createDirs	Si se configura a true, Tomcat intentará crear los directorios anteriores si no existen al arrancar el servidor.
autoDeploy	Este flag indica a Tomcat que compruebe periódicamente si alguna aplicación web desplegada ha sido añadida o actualizada, y, en caso afirmativo, que la despliegue (re-compile) automáticamente en el servidor.
deployIgnore	Rutas a ignorar en el proceso de autodespliegue de aplicaciones web.
deployOnStartup	Indica si las aplicaciones en este host deben ser desplegadas automáticamente al arrancar (true por defecto).
name	Nombre del servidor virtual. Este atributo es obligatorio y debe ser único entre todos los hosts virtuales que se ejecutan en ese contenedor.
startStopThreads	Número de hilos que se pueden arrancar en paralelo para cada elemento anidado de este `Host`. Por defecto toma el valor de 1. Si se utiliza 0, se usará el valor `Runtime.getRuntime().availableProcessors()`.

Además existen atributos adicionales, mostrados en la siguiente tabla.

Atributo	Descripción
copyXML	Poner a true si se desea que los ficheros de definición de contexto de las aplicaciones (en los directorios `META-INF/context.xml`) se copien a `xmlBase` automáticamente al desplegar la aplicación. Por defecto, el valor es false y no tiene sentido si `deployXML` es false.
deployXML	Poner a false si se quiere que Tomcat ignore los ficheros de definición de contexto de las aplicaciones (en los directorios `META-INF/context.xml`). En entornos seguros, debería ponerse a false para evitar posibles problemas con las aplicaciones desplegadas.
errorReportValveClass	Definición de la clase va a emplearse para mostrar los errores de las aplicaciones (que debe implementar la interfaz `org.apache.catalina.Valve`).
unpackWARs	Poner a true si se desea que las aplicaciones web que se copien en el directorio `appBase` como ficheros `WAR` se descompriman y desplieguen automáticamente.

workDir	Ruta de un directorio de trabajo que pueden utilizar las aplicaciones para sus ficheros temporales. El directorio debe tener permisos de lectura/escritura. Por defecto, un directorio específico bajo `$CATALINA_BASE/work`

Por ejemplo, una definición del elemento <Host> puede ser:

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

El elemento <Host> se configura como un hijo del elemento <Engine> y como un padre de los siguientes elementos:

- <Context>. Define las diferentes aplicaciones web desplegadas en el host.
- <Realm>. Configura un realm (base de datos de usuarios, claves y roles) que podrá ser compartido entre todos los Contextos subordinados, a no ser que se configure un `Realm` a un nivel inferior.

El proceso de autodespliegue de las aplicaciones se puede controlar en toda su extensión con una serie de parámetros combinados que definen diferentes modos de funcionamiento. La casuística completa queda fuera del ámbito de este curso, pero se puede obtener más información en la sección "Automatic Application Deployment" (<tomcat.apache.org/tomcat-7.0-doc/config/host.html#Automatic_Application_Deployment>) o también en la página "Automatic Deployment Use Cases" (<tomcat.apache.org/tomcat-7.0-doc/config/automatic-deployment.html>)

De forma similar al elemento `Engine`, se pueden configurar los logs de acceso con un elemento `Valve`:

```
<Host name="localhost" ...>
  ...
  <Valve className="org.apache.catalina.valves.AccessLogValve"
         prefix="localhost_access_log." suffix=".txt"
         pattern="common"/>
  ...
</Host>
```

Ver "Access Logging Valves" (<tomcat.apache.org/tomcat-7.0-doc/config/valve.html#Access_Logging>) para más información sobre los parámetros soportados.

Para dar más de un nombre a un host virtual se puede utilizar la directiva `Alias`:

```
<Host name="www.mycompany.com" ...>
  ...
  <Alias>mycompany.com</Alias>
  ...
</Host>
```

Como en `Engine`, el elemento `Valve` se puede utilizar también para filtrar las peticiones entrantes:

```
<Host name="localhost" ...>
  ...
  <Valve className="org.apache.catalina.valves.RemoteHostValve"
         allow=".*\.mycompany\.com|www\.yourcompany\.com"/>
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         deny="192\.168\.1\.\d+"/>
  ...
</Host>
```

Para más información accede a "Remote Address Filter" (<tomcat.apache.org/tomcat-7.0-doc/config/valve.html#Remote_Address_Filter>) y "Remote Host Filter" (<tomcat.apache.org/tomcat-7.0-doc/config/valve.html#Remote_Address_Filter>).

Adicionalmente se puede configurar un mecanismo de Single Sign On a nivel del host virtual, de tal forma que los usuarios sólo tengan que autenticarse una sola vez en una de las aplicaciones. En la práctica, todas las aplicaciones comparten el mismo Real (a nivel de Host) y los tokens de autenticación adquiridos se mantienen y propagan a lo largo de todas las aplicaciones. Para configurarlo:

```
<Host name="localhost" ...>
  ...
  <Valve
className="org.apache.catalina.authenticator.SingleSignOn"/>
  ...
</Host>
```

Por último, Tomcat también soporta directorios de usuario, los accedidos con URL que empieza por ~ seguido del nombre de usuario. Para configurarlo (<tomcat.apache.org/tomcat-7.0-doc/config/host.html#User_Web_Applications>):

```
<Host name="localhost" ...>
  ...
  <Listener className="org.apache.catalina.startup.UserConfig"
            directoryName="public_html"
            homeBase=c:\Homes"

userClass="org.apache.catalina.startup.HomesUserDatabase"/>
  ...
</Host>
```

4.5.3. El elemento <Context>

El elemento <Context> es el contenedor más comúnmente usado en el fichero server.xml. Representa una aplicación web individual que se ejecuta en un <Host> definido. Se puede definir cualquier número de contextos en un <Host>, pero cada definición de <Context> debe tener un único path de contexto, que se define en el atributo path. Además se debe definir un Contexto con un context path vacío, que se entenderá como la aplicación web por defecto para ese host virtual.

Cada aplicación Web se basa en un fichero WAR (Web Application Archive), o en el correspondiente directorio que contiene los contenidos convenientemente desempaquetados, según se describe en la especificación Servlet 2.2 y superiores.

La configuración de los contextos tiene mucho que ver con el proceso de desarrollo y configuración específica de las aplicaciones web, que queda fuera del alcance de este curso. Por ello, sólo se presentan los atributos más generales para el elemento <Context>, en la tabla siguiente. La implementación estándar es org.apache.catalina.core.StandardContext.

Atributo	Descripción
Cookies	Poner a true si se desea que se utilicen cookies para identificar la sesión.
docBase	El documento base (o contexto base) es el directorio de la aplicación, o la ruta al ficheroWAR si se despliega de esta forma. Es una ruta absoluta o bien una ruta relativa a appBase.
override	Poner a true para indicar que se ignoren las configuraciones en el contexto global o de Host y sólo se utilicen las del contexto específico.
path	La ruta de contexto de la aplicación, que se compara con el inicio de la URL recibida para seleccionar la aplicación web que se está solicitando. Todos los paths de un host deben ser únicos, ya que identifican de forma unívoca una aplicación web determinada.

reloadable	Si es true, solicita a Tomcat que revise si hay cambios en las clases ubicadas en `/WEB-INF/classes` y `/WEB-INF/lib` y las recargue automáticamente en caso afirmativo. Este parámetro es muy útil en servidores de desarrollo, pero puede afectar seriamente a los servidores en producción.

Existen además otros atributos adicionales, bastante parecidos en funcionalidad a los del elemento Host, que permiten configurar alias de aplicaciones, especificar si se siguen enlaces simbólicos o no en las rutas de aplicaciones, valores de caché, así como especificaciones sobre el autodespliegue y autorecarga de las aplicaciones web.

Por ejemplo, el elemento <Context> que define la aplicación `/examples` se especifica cómo sigue en el fichero `server.xml`:

```
<Context path="/examples" docBase="examples" debug="0"
reloadable="true">
```

La definición de contexto define una aplicación web llamada `/examples` que tiene todos sus recursos almacenados en el directorio relativo `examples` y que se recarga cuando los ficheros de clases cambian.

El elemento <Context> se configura como un hijo del elemento <Host> y como un padre de los siguientes elementos (se puede configurar como mucho una instancia de cada elemento):

- <Loader>.- Configura el cargador de clases de la aplicación web que se usará para cargar los servlets. Normalmente, servirá con el de defecto. Más información en <tomcat.apache.org/tomcat-7.0-doc/config/loader.html>
- <Realm>.- Configura un realm para esta aplicación web específicamente. Si no se especifica se utilizará el definido para el Host o el Engine al que pertenece. Más información en <tomcat.apache.org/tomcat-7.0-doc/config/realm.html>.
- <Manager>.- Configura el manager de sesión que se usará para crear, destruir y mantener las sesiones HTTP para esta aplicación Web. Más información en <tomcat.apache.org/tomcat-7.0-doc/config/manager.html>.
- <Resource>.- Configura el manager de recursos que se usará para acceder a los recursos estáticos asociados con esta aplicación web. Normalmente servirá con el defecto. Más información en <tomcat.apache.org/tomcat-7.0-doc/config/manager.html>.

Se pueden definir tantos elementos <Context> como se quiera, cada uno con un único context path que es definido por el atributo path. Además, se debe definir un Context con un context path de longitud cero. Este contexto pasa a ser la aplicación web por defecto de ese host virtual y se usa para procesar todas las requests que no casan con el context path de cualquier otro contexto.

Además de anidar elementos `Context` en un elemento Host, se pueden almacenar en:

- Un único fichero `$CATALINA_HOME/conf/context.xml`. el elemento de contexto será cargado por todas las aplicaciones web.
- En un fichero dependiente del contexto, del motor y del host, llamado `$CATALINA_HOME/conf/[enginename]/[hostname]/context.xml .default` la información del elemento de contexto será cargada por todas las aplicaciones web del host.
- En ficheros, con la extensión `.xml`, en el directorio `$CATALINA_HOME/conf/[enginename]/ [hostname]`
- Si en los ficheros anteriores no se encuentra el contexto para la aplicación, en ficheros individuales en `/META-INF/context.xml`, dentro de los ficheros de aplicación.

Adicionalmente se pueden añadir parámetros de contexto que se harán visibles a la aplicación web como parámetros de inicialización del contexto del servlet. Para ello se utilizan elementos `<Parameter>` por ejemplo:

```
<Context>
  ...
  <Parameter name="companyName" value="My Company, Incorporated"
         override="false"/>
  ...
</Context>
```

Esto es totalmente equivalente a incluir la información en el fichero de descripción de la aplicación web (`/WEB-INF/web.xml`), pero la ventaja es que no requiere la modificación de la aplicación:

```
<context-param>
  <param-name>companyName</param-name>
  <param-value>My Company, Incorporated</param-value>
</context-param>
```

4.6. Conectores

El elemento `<Connector>` define la clase que gestiona las peticiones y respuestas a y desde una aplicación cliente. Está definido por la interface `org.apache.catalina.Connector`.

El elemento <Connector> se configura en el fichero server.xml como un hijo del elemento <Service> y no puede configurarse como padre de ningún elemento.

Hay dos definiciones importantes de conectores en el fichero `server.xml` por defecto: conector HTTP/1.1 y conector AJP.

4.6.1. Conector HTTP/1.1

El conector HTTP/1.1 representa un componente conector que soporta el protocolo HTTP/1.1 (sustituye al antiguo conector HTTP), habilitando a Tomcat para funcionar como un servidor web standalone, además de su capacidad para ejecutar servlets y páginas JSPs. Al levantar el servidor, este conector creará un número de hilos de procesamiento de peticiones igual al valor del atributo minProcessors.

Internamente puede usar diferentes implementaciones de protocolo: APR nativo (el más eficiente, usando por defecto en la instalación estándar habitual), conector Java asíncrono y conector Java síncrono (el menos eficiente), cada uno con sus peculiaridades. Normalmente esto no hace falta configurarlo. Más información en <tomcat.apache.org/tomcat-7.0-doc/config/http.html#Connector_Comparison>.

La configuración de este conector tiene un gran número de atributos específicos, de los cuales los más importantes son (entre comunes y adicionales):

Atributo	Descripción
enableLookups	Activa la realización de llamadas DNS para obtener el nombre de la máquina que hace la petición, a partir de su IP.
maxParameterCount	Número máximo de parámetros GET más POST que se admiten. Por defecto, 10000.
maxPostSize	Tamaño máximo de datos en una petición POST.
port	Puerto de escucha de las peticiones entrantes.
maxConnections	Número máximo de conexiones concurrentes soportadas.
server	Especificación de la cabecera Server enviada en las respuestas. Por defecto, Apache-Coyote/1.1
SSLEnabled	Habilitar tráfico SSL en un conector (HTTPS). Más información en <tomcat.apache.org/tomcat-7.0-doc/config/http.html#SSL_Support>

El conector HTTP/1.1 tiene la siguiente funcionalidad:

- Soporte HTTP/1.0 y HTTP/1.1.
- Soporte para proxy (cuando Tomcat se ejecuta por detrás de un proxy). Más información en "Proxy Support HOWTO" (<tomcat.apache.org/tomcat-7.0-doc/proxy-howto.html>)
- Soporte SSL para conexiones cifradas HTTPS. Más información en "SSL Configuration HOWTO" (<tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html>)

Por ejemplo:

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

Otro ejemplo de conector SSL:

```
<Connector port="8443"
protocol="org.apache.coyote.http11.Http11Protocol"
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" />
```

4.6.2. Conector AJP

El conector AJP representa un componente conector que comunica con un conector web a través del protocolo AJP. Se usa en los casos en que se quiere integrar de forma invisible Tomcat en una instalación Apache, de manera que Apache gestione el contenido estático en la aplicación web o utilizar su gestión SSL.

Este conector soporta balanceo de carga cuando se usa en conjunción con el atributo jvmRoute de `Engine`.

Los atributos específicos de esta implementación son similares al del conector HTTP/1.1, en referencia al puerto, tamaño de datos soportado y número de servidores e hilos.

Por ejemplo:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Los conectores nativos soportados en Tomcat 7 son:

- JK 1.2.x: más información en "JK doc" (<tomcat.apache.org/connectors-doc/>)
- mod_proxy en Apache httpd 2.x (incluido por defecto en Apache HTTP Server 2.2), con AJP (ver <httpd.apache.org/docs/2.2/mod/mod_proxy_ajp.html>).

En versiones anteriores de Tomcat existieron otros conectores nativos con soporte AJP, pero ya no están soportados.

4.7. Otros elementos de configuración

Las diferentes aplicaciones pueden incluir una autenticación mediante usuarios/roles. Esta definición de los usuarios y roles para la autenticación se escriben en el elemento `<tomcat-users>`, que puede anidado en diferentes elementos.

La configuración general de Tomcat está en el fichero `$CATALINA_HOME/conf/tomcat-users.xml`, y su aspecto es similar al siguiente:

```
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
</tomcat-users>
```

Este fichero no surge de forma mágica sino que está enlazado en realidad desde `server.xml`, en la sección `GlobalNamingResources`:

```
<GlobalNamingResources>
    <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" />
  </GlobalNamingResources>
```

4.8. Análisis de la configuración

Como ya sabes, el fichero principal de configuración de Tomcat es `server.xml`, en el directorio `<CATALINA_HOME>\conf`. A continuación vamos a analizar el fichero por defecto:

```
1  <?xml version='1.0' encoding='utf-8'?>
2  <Server port="8005" shutdown="SHUTDOWN">
3    <Listener className="org.apache.catalina.core.JasperListener" />
4    <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
5    <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
6    <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
7    <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />
8
9    <GlobalNamingResources>
10     <Resource name="UserDatabase" auth="Container"
11               type="org.apache.catalina.UserDatabase"
12               description="User database that can be updated and saved"
13               factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
14               pathname="conf/tomcat-users.xml" />
15   </GlobalNamingResources>
16
17   <Service name="Catalina">
18     <Connector port="8080" protocol="HTTP/1.1"
19                connectionTimeout="20000"
20                redirectPort="8443" />
21     <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
22
23     <Engine name="Catalina" defaultHost="localhost">
24
25       <Realm className="org.apache.catalina.realm.LockOutRealm">
26         <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
27                resourceName="UserDatabase"/>
28       </Realm>
29
30       <Host name="localhost"  appBase="webapps"
31             unpackWARs="true" autoDeploy="true">
32         <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
33                prefix="localhost_access_log." suffix=".txt"
34                pattern="%h %l %u %t &quot;%r&quot; %s %b" />
35       </Host>
36     </Engine>
37   </Service>
38 </Server>
```

Figure 3.4.3: Fichero server.xml

- `Server` (línea 2) es el componente principal, que representa la instancia complete de Tomcat. Puede contener uno o más `Services`, cada uno con sus `Engines` y `Connectors`.

```
<Server port="8005" shutdown="SHUTDOWN"> ..... </Server>
```

- El `Server` contiene varios `Listeners` (líneas 3-7). Un `Listener` recibe y responde a eventos específicos.
  - El `JasperListener` active el motor de JSP `Jasper`, responsable de recompilar los JSP según cambien.

```
<Listener
className="org.apache.catalina.core.JasperListener
" />
```

- El `GlobalResourcesLifecycleListener` activa los recursos globales y hace posible el uso de JNDI (Java Naming and Directory Interface) para acceder a recursos con nombres comprensibles.

```
<Listener
className="org.apache.catalina.mbeans.GlobalResourcesLi
fecycleListener" />
```

- El elemento `<GlobalNamingResources>` (líneas 9-15) define los recursos JNDI que permiten a los clientes Java descubrir y buscar datos y objetos con un nombre. La configuración por defecto incluye un nombre JNDI llamado `UserDatabase` mediante el elemento `<Resource>` (líneas 10-14), que es una base de datos en memoria para autenticación de usuarios, cargada del fichero en disco `conf/tomcat-users.xml`.

```
<GlobalNamingResources>
  <Resource name="UserDatabase" auth="Container"
            type="org.apache.catalina.UserDatabase"
            description="User database that can be
updated and saved"

factory="org.apache.catalina.users.MemoryUserDatabaseFa
ctory"
            pathname="conf/tomcat-users.xml" />
</GlobalNamingResources>
```

Se pueden definir otros recursos JNDI como conexiones a bases de datos externas (MySQL, Oracle…).

- Un `Service` asocia uno o más `Connectors` a un `Engine`. La configuración por defecto define un servicio `Catalina` y asocial dos `Connectors`: HTTP y AJP al `Engine`.

```
<Service name="Catalina"> ..... </Service>
```

- Un `Connector` se asocia a un Puerto TCP para gestionar la comunicación entre el `Service` y los clientes. La configuración por defecto define dos conectores, HTTP/1.1 y AJP:

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

El Puerto de escucha es 8080. El atributo `connectionTimeout` define el número de milisegundos de espera máxima tras una conexión (por defecto, 20 segundos). El atributo redirect redirige las peticiones SSL al puerto 8443.

El puerto por defecto es 8080, pero puedes escoger cualquier puerto libre entre 1024 y 65535. Para sistemas en producción donde Tomcat sea el servidor por defecto, sería el puerto HTTP por defecto, o sea, el 80.

```
<Connector port="8009" protocol="AJP/1.3"
redirectPort="8443" />
```

- Tomcat llama ***container*** a `Engine`, `Host`, `Context`, y `Cluster`. El nivel superior es `Engine`; el más bajo es `Context`. Algunos componentes, como `Realm` y `Valve`, se ubican dentro de contenedores.
- Un `Engine` es el contenedor de mayor nivel. Contiene uno o más `Hosts`. Se puede configurar un servidor Tomcat para ejecutar varios hostnames, llamados hosts ***virtuales***.

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

- Un `Realm` es una base de datos de usuarios, claves y roles para autenticación. Se pueden definir `Realm` para cualquier contenedor, como `Engine`, `Host`, `Context`, y `Cluster`.

```
<Realm
className="org.apache.catalina.realm.LockOutRealm">
 <Realm
className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase"/>
</Realm>
```

Por defecto está el `UserDatabaseRealm` para el motor `Catalina`, que utilice el nombre JNDI `UserDatabase` definido en `GlobalNamingResources`. Aparte del `UserDatabaseRealm`, existen: `JDBCRealm` (autenticación basada en base de datos relacionalvía un driver JDBC); `DataSourceRealm` (conectar con un `DataSource` vía JNDI); `JNDIRealm` (conexión con LDAP); y `MemoryRealm` (carga un fichero XML en memoria).

- Un `Host` define un servidor virtual bajo el `Engine`, que a su vez puede soportar muchos `Contexts` (webapps).

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

La configuración por defecto define un host llamado `localhost`. El atributo `appBase` define el directory base de todas las aplicaciones web contenidas, en este caso, `<CATALINA_HOME>\webapps`. Por defecto, la URL de cada webapp es igual que su nombre de directorio (por ejemplo la instalación por defecto proporciona 4 webapps: `docs`, `examples`, `host-manager` and `manager` bajo el directorio `webapps`) . La única excepción es ROOT, que se identifica por una cadena vacía (esto es, la URL es `http://localhost:8080/`).

El parámetro `unpackWARs` indica si los ficheros WAR en el directorio se descomprimen automáticamente. El parámetro `autoDeploy` especifica si las aplicaciones se despliegan automáticamente.

- Tomcat soporta clustering de servidores. Puede replicar sesiones y atributos de contexto entre nodos y desplegar WARs en todo el cluster.
- Un `Valve` puede interceptor peticiones HTTP antes de enviarlas a las aplicaciones, para preprocesar dichas peticiones.
- En la configuración por defecto, el `AccessLogValve` intercepta las peticiones HTTP y crea una entrada en el fichero de log:

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

Otros Valves incluyen:

- `RemoteAddrValve`: que bloquea peticiones de ciertas direcciones IP.
- `RemoteHostValve`: que bloquea peticiones basándose en hostnames.
- `RequestDumperValve`: que registra detalles de las peticiones.
- `SingleSignOn` Valve: que permite configurar single sign-on para el acceso de todas las aplicaciones web bajo el host configurado.

Otros parámetros de configuración útiles serían:

1. `conf\web.xml`: activar el listado de directorios.

   Al activar el listado de directories, si accedemos a una URL y no contiene un fichero índice (`index.htm`, `index.jsp`…), se mostrará el listado de directorio con todos los archivos contenidos. Para ello hay que cambiar el parámetro `listings` a `true` para el servlet por defecto ("default"). Esto es útil para sistemas de test, pero no para producción, por razones de seguridad.

```
<servlet>
  <servlet-name>default</servlet-name>
  <servlet-
class>org.apache.catalina.servlets.DefaultServlet</serv
let-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <init-param>
    <param-name>listings</param-name>
    <param-value>true</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

2. `conf\context.xml`: habilitar la recarga automática.

   Si se añade el atributo `reloadable="true"` al contexto, Tomcat recargará cualquier cambio en el código de forma automática, que resulta muy útil en sistemas de test.

```
<Context reloadable="true">
   ......
</Context>
```

3. `conf\tomcat-users.xml`: añadir/modificar usuarios

   Editar la información de los usuarios en los elementos `<tomcat-users>`.

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

4.9. Conclusiones

En este capítulo se ha presentado la configuración de Apache Tomcat.

El fichero `server.xml` es el fichero principal de configuración de una instalación Tomcat. Debe haber uno por instancia Tomcat.

Los componentes de Tomcat se representan como elementos anidados dentro del fichero de configuración de Tomcat.

El elemento de primer nivel `Server`, representa la instancia Tomcat. Por tanto sólo puede haber un elemento `Server`.

Puede haber más de un elemento `Service` en un `Server`. Cada `Service` representa uno o más conectores que pasan las peticiones a un único motor (`Engine`). Cada motor puede contener uno o más elementos hosts.