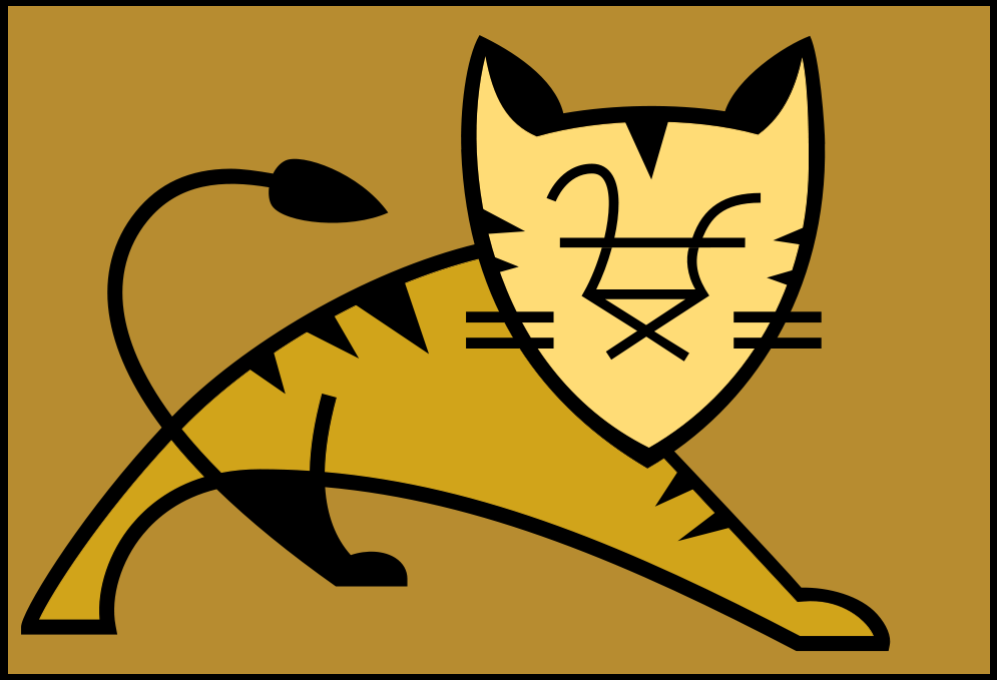
I

CONFIGURACIÓN



Iñaki
Respaldiza

Configuración

- Archivos de configuración y parámetros
- EstructurabásicadeJNDI
- Conectores de Tomcat
- Fundamentos sobre el Realm
- El contexto y su configuración
- Definición de un pool de conexiones C3PO y DBCP
- ElelementoengineenTomcat

Archivos de configuración y parámetros

<server> </server>	(Único y engloba toda la configuración) Define el elemento de configuración básico del fichero server.xml. Es único y contiene uno o más servicios ("Service"). El atributo "port" indica el puerto destinado a la escucha del comando de cierre, indicado por "shutdown" o cierre.	<Server port="8005" shutdown="SHUTDOWN">
<Listener/>	(Único) Permiten definir las clases JMX que permitirá escuchar Tomcat.	<Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" />
<GlobalNamingResources> </GlobalNamingResources>	(Único) Permite definir elementos JNDI para ser utilizados globalmente.	
<Service> </Service>	Estas etiquetas permiten agrupar uno o más conectores de forma que compartan un único contenedor de aplicaciones. Poseen un único atributo, "name", que fija los identificadores individuales. Si "name" se fija como "Catalina" o "Tomcat-Standalone", se habilitará a Tomcat como servidor web independiente.	<Service name="Tomcat-Standalone"> </Service>

Archivos de configuración y parámetros

<Connector /> (dentro de "Service")	Conecta un contenedor de datos con el exterior, definiendo el elemento final a través del cual se realizarán las peticiones de usuario y se enviarán las respuestas. Entre sus parámetros de configuración están el puerto de escucha, "port", la clase encargada de su definición, "className" y el número máximo de conexiones simultáneas permitidas, "acceptCount".	<Connector className="org.apache.coyote.tomcat4.CoyoteConnector" port="8080" minProcessors="5" maxProcessors="75" enableLookups="true" acceptCount="100" connectionTimeout="20000" useURIValidationHack="false" disableUploadTimeout="true" />
<Engine> </Engine>	(dentro de "Service") Punto donde se procesan las peticiones que llegan a los "Connector" que posean en la cabecera el valor de "defaultHost" como destino.	<Engine name="Standalone" defaultHost="localhost">
<Logger/>	(dentro de "Service" o de "Host") Permite establecer el nombre del fichero de logs. Como parámetros tiene la clase encargada de su definición, "className", el formato nombre del archivo, como la unión de un prefijo, "preffix", y un sufijo, "suffix".	<Logger className="org.apache.catalina.logger .FileLogger" prefix="catalina_log." suffix=".txt" timestamp="true"/>

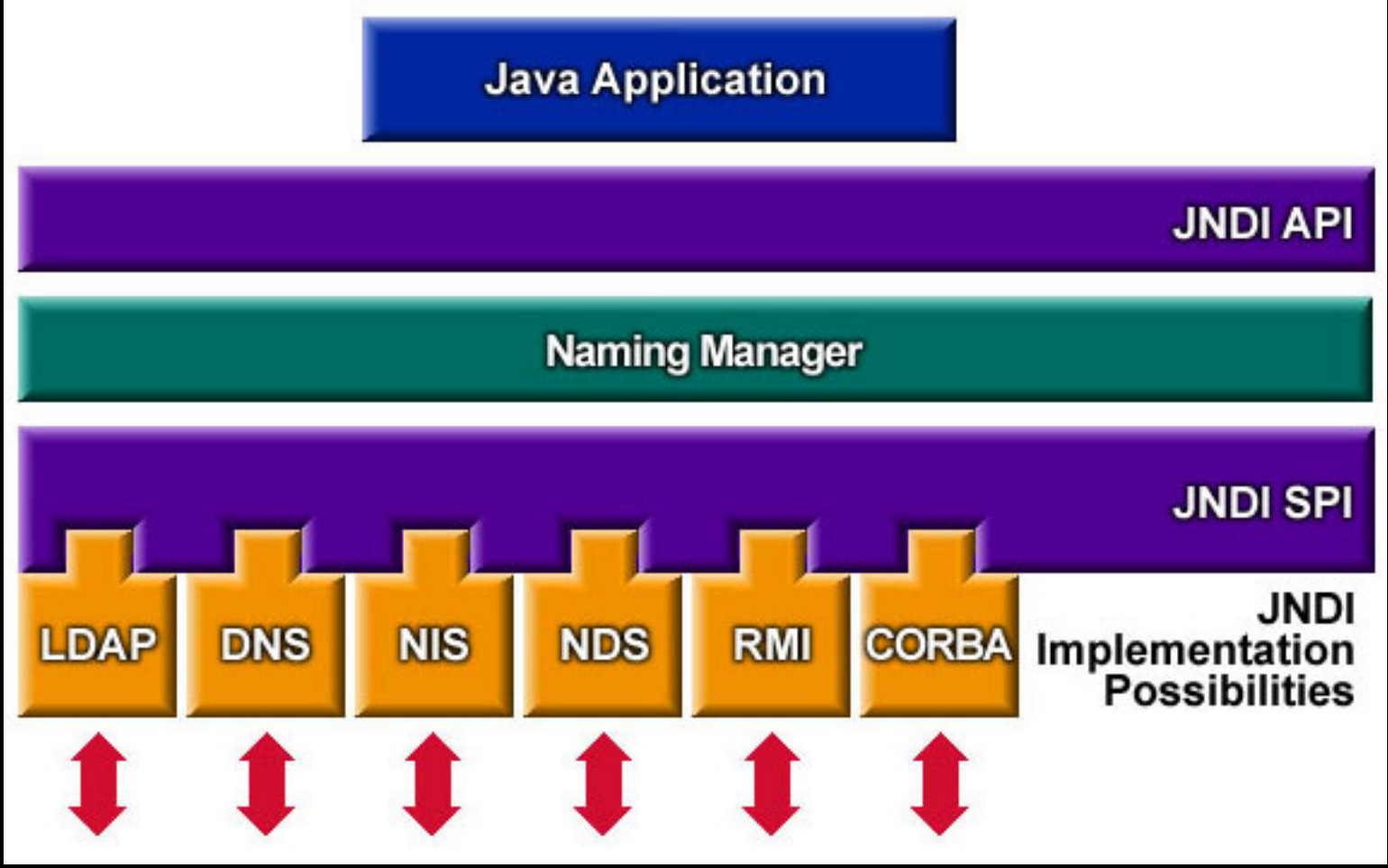
Archivos de configuración y parámetros

<Host> </Host>	Con estas etiquetas podemos definir uno o más elementos Host virtuales para atender a las peticiones.	<Host name=”localhost” debug=”0” appBase=”webapps” unpackWARs=”-true” autoDeploy=”true”>
<Context> </Context>	(dentro de “Host”) Se utiliza para indicar la ruta (“docBase”) a partir de la cual se encuentran las aplicaciones a ser ejecutadas en Tomcat (a partir de “%CATALINA_HOME%webapps” y el path url (“path”) a partir del cual acceder a los servicios.	

Archivos de configuración y parámetros

<Host> </Host>	Con estas etiquetas podemos definir uno o más elementos Host virtuales para atender a las peticiones.	<Host name=”localhost” debug=”0” appBase=”webapps” unpackWARs=”-true” autoDeploy=”true”>
<Context> </Context>	(dentro de “Host”) Se utiliza para indicar la ruta (“docBase”) a partir de la cual se encuentran las aplicaciones a ser ejecutadas en Tomcat (a partir de “%CATALINA_HOME%webapps” y el path url (“path”) a partir del cual acceder a los servicios.	
<Logger/>	(dentro de “Service” o de “Host”) Permite establecer el nombre del fichero de logs. Como parámetros tiene la clase encargada de su definición, “className”, el formato nombre del archivo, como la unión de un prefijo, “preffix”, y un sufijo, “suffix”.	<Logger className=”org.apache.catalina.logger .FileLogger” prefix=”catalina_log.” suffix=”.txt” timestamp=”true”/>

Estructura básica de JNDI



Estructura básica de JNDI

El Java Naming and Directory Interface (JNDI) es un interface de programación (API) que proporciona funcionalidades de nombrado y directorio a las aplicaciones escritas usando Java. Está definido para ser independiente de cualquier implementación de servicio de directorio. Así se puede acceder a una gran variedad de directorios,-- nuevos, emergentes, y ya desarrollados -- de una forma común.

Estructura básica de JNDI

Tomcat permite que las distintas aplicaciones desplegadas (ficheros .war o similar) recoger o compartir información, conexiones a base de datos, etc. De hecho, es práctica habitual que el servidor web ponga DataSource ya preparados a los distintos war, evitando que todos ellos abran sus propias conexiones a la base de datos.

Estructura básica de JNDI

La configuración de recursos específicos de Tomcat se realiza utilizando los siguientes elementos:

- <Environment> - Configuración de nombres y valores para las entradas del entorno escalar que se exponen a la aplicación Web a través del JNDI InitialContext (equivalente a la inclusión de un elemento <env-entry> en el descriptor de implementación de la aplicación web).
- <Resource> - Configuración del nombre y tipo de datos de un recurso disponible para la aplicación (equivalente a la inclusión de un elemento <resource-ref> en el descriptor de implementación de la aplicación web).
- <ResourceLink> - Agrega un enlace a un recurso definido en el contexto JNDI global. Utilice los enlaces de recursos para proporcionar a una aplicación web acceso a un recurso definido en el elemento secundario <GlobalNamingResources> del elemento <Server>.
- <Transaction>: agrega una fábrica de recursos para instanciar el objeto UserTransaction disponible en java: comp / UserTransaction.

Conectores de Tomcat

HTTP

El conector HTTP se configura de forma predeterminada con Tomcat y está listo para su uso. Este conector tiene la menor latencia y el mejor rendimiento general.

Para un cluster, se debe instalar un balanceador de carga HTTP con soporte para sesiones web para dirigir el tráfico a los servidores Tomcat. Tomcat soporta mod_proxy (en Apache HTTP Server 2.x, e incluido de forma predeterminada en Apache HTTP Server 2.2) como balanceador de carga.

Conectores de Tomcat

AJP

El conector AJP es funcionalmente equivalente al agrupamiento HTTP. Si bien cuando lo utilizamos en un solo servidor observamos un rendimiento menor hay ocasiones en las que alguna circunstancia recomienda su uso.
AJP clustering es el más eficiente desde la perspectiva de Tomcat.

Los conectores nativos compatibles con esta versión de Tomcat son:
JK 1.2.x con cualquiera de los servidores compatibles
Mod_proxy en Apache HTTP Server 2.x (incluido de forma predeterminada en Apache HTTP Server 2.2), con AJP habilitado

Fundamentos sobre el Realm

Realm es una “base de datos” de nombres de usuario y contraseñas que identifican usuarios válidos de una aplicación web (o conjunto de ellas), además de un listado de funciones asociadas a cada usuario. Se puede considerar que los roles son similares a los de los sistemas operativos de tipo Unix, ya que el acceso a recursos específicos de la aplicación web se concede a todos los usuarios que poseen una función específica. Un usuario en particular puede tener cualquier número de roles asociados con su nombre de usuario.

Fundamentos sobre el Realm

- JDBCRealm - Accede a la información de autenticación en una base de datos relacional, a la que se accede a través de un controlador JDBC.
- DataSourceRealm - Accede a la información de autenticación almacenada en una base de datos relacional, a la que se accede a través de un JNDI JDBC DataSource con nombre.
- JNDIRealm - Accede a la información de autenticación almacenada en un servidor de directorio basado en LDAP, accesado a través de un proveedor JNDI.
- UserDatabaseRealm - Accede a la información de autenticación almacenada en un recurso JNDI de UserDatabase, que suele estar respaldado por un documento XML (conf / tomcat-users.xml).
- MemoryRealm: Accede a la información de autenticación almacenada en una colección de objetos en memoria, que se inicializa desde un documento XML (conf / tomcat-users.xml).
- JAASRealm - Accede a la información de autenticación a través del marco de Java Authentication & Authorization Service (JAAS).

El contexto y su configuración

Crear contexto de aplicación web.
Crear un archivo XML, para el nuevo contexto llamado “dev” (ejemplo anterior). El archivo debe quedar en:
CATALINA_HOME / conf / Catalina / localhost / dev.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>

<Context path="/dev" docBase="C:/dir/dev/web"
 reloadable="true" crossContext="true" debug="3">
        <Logger className="org.apache.catalina.logger.FileLogger"
         prefix="localhost_dev_log." suffix=".txt" timestamp="true"
         verbosity="4" />
</Context>
```

Definición de un pool de conexiones C3PO y DBCP

Un pool de conexiones a base de datos es un mecanismo para optimizar el desempeño de una aplicación así como la utilización de recursos, teniendo varias conexiones ya establecidas al RDBMS, las cuales pueden ser utilizadas por cualquier proceso que las necesite.
Esto significa que en vez de que un componente establezca su propia conexión, la toma del pool y al final cuando ya no la necesita, la devuelve al pool. Hay incluso maneras transparentes de lograr esto, utilizando una implementación especial de DataSource, la cual cuando se le pide una conexión, tome una del pool, y dicha conexión se devuelva al pool cuando se invoque close() (en vez de cerrarse físicamente).

Definición de un pool de conexiones C3PO y DBCP

DBCP

La manera de utilizarlo, es crear una instancia de org.apache.commons.dbcp.BasicDataSource a la cual debemos configurarle varios parámetros, dependiendo de las necesidades de nuestra aplicación. Las propiedades más esenciales son driverClassName, url, username y password para que el DataSource se pueda conectar a una base de datos.

Definición de un pool de conexiones C3PO y DBCP

Algunas propiedades interesantes:

- maxActive: El límite de conexiones que puede haber activas (es decir fuera del pool).
- maxIdle. El límite de conexiones que debe haber disponibles en el pool.
- minIdle: El mínimo de conexiones que debe haber disponibles en el pool en todo momento.
- initialSize: El número de conexiones que se deben crear cuando se inicializa el pool.
- maxWait: El límite de tiempo, en milisegundos, que el pool debe esperar para que regrese una conexión al pool.
- validationQuery: Esta propiedad opcional puede especificar un query que se ejecuta en una conexión antes de prestarla a un objeto en la aplicación.

Definición de un pool de conexiones C3PO y DBCP

C3P0

C3P0 por su parte nos ofrece la clase com.mchange.v2.c3p0.ComboPooledDataSource, que es un tanto similar a BasicDataSource de DBCP, excepto que cambian algunos nombres y algunos mecanismos de validación. En esta clase, las propiedades esenciales para la conexión se llaman driverClass, jdbcUrl, user y password.

Definición de un pool de conexiones C3PO y DBCP

Algunas propiedades interesantes son:
minPoolSize: El número mínimo de conexiones que debe tener el pool.
maxPoolSize: El número máximo de conexiones que debe manejar el pool.
acquireIncrement: Esta propiedad indica el número de conexiones a crear cuando ya no hay disponibles.
automaticTestTable: C3P0 valida las conexiones que maneja antes de prestarlas, haciendo un query a una tabla de prueba.
checkoutTimeout: En esta propiedad se define el tiempo máximo de espera para que regrese una conexión al pool.
numHelperThreads: C3P0 realiza ciertas operaciones, como la creación de nuevas conexiones, validaciones, etc, en hilos separados. En apicaciones que hacen uso intensivo del pool se puede incrementar este parámetro si se detecta que el pool está siendo un cuello de botella.

Definición de un pool de conexiones C3PO y DBCP

El pool de conexiones JDBC org.apache.tomcat.jdbc.pool es una alternatica al Apache Commons DBCP connection pool.

El elemento engine en Tomcat

El elemento Engine representa toda la maquinaria de procesamiento de solicitud asociada con un Servicio Catalina en particular. Recibe y procesa todas las solicitudes de uno o más conectores y devuelve la respuesta al conector para la transmisión final al cliente.

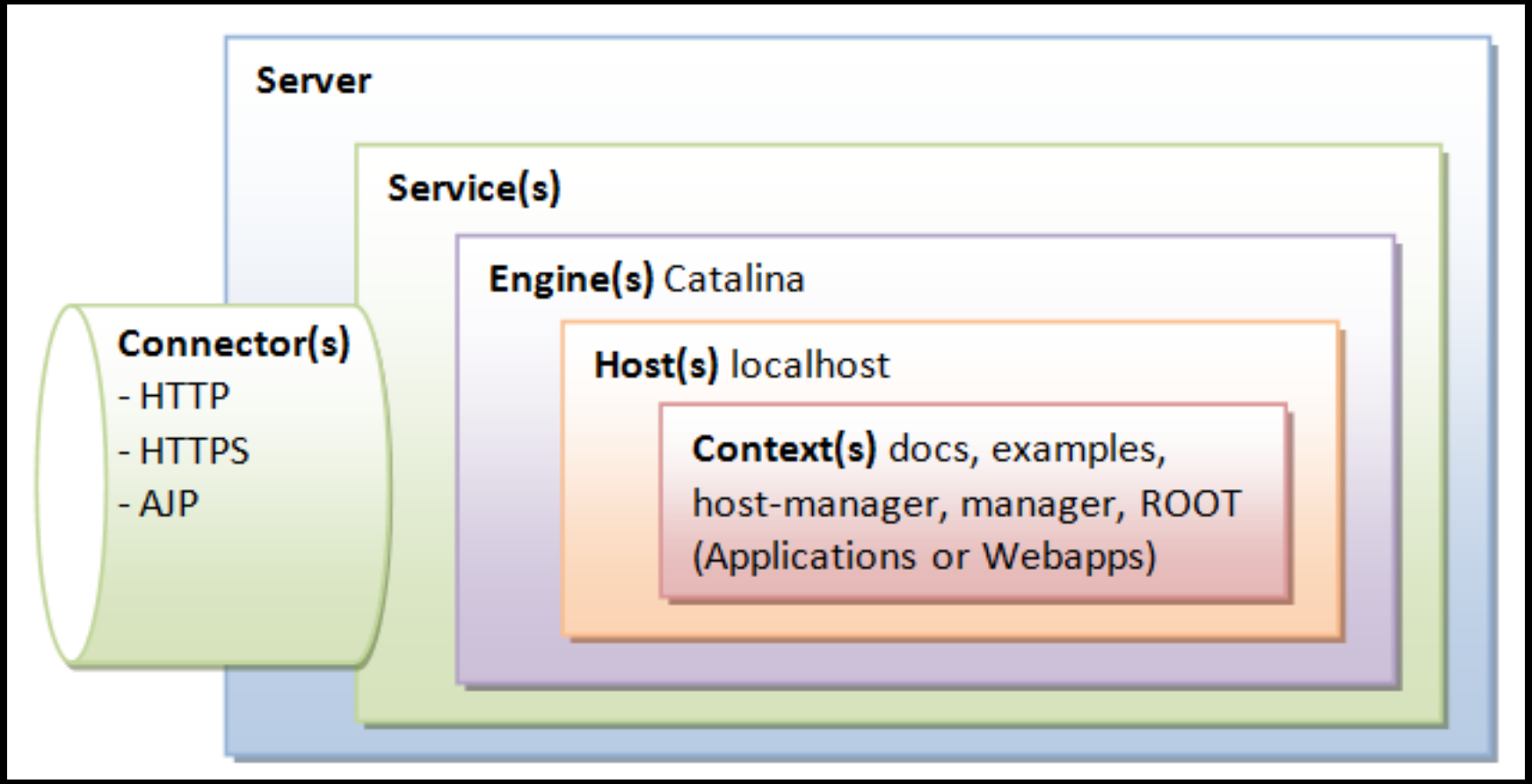
Un elemento Engine ha de ser anidado dentro de un elemento de servicio, siguiendo todos los elementos de conector asociados con a dicho servicio.

El elemento engine en Tomcat

El elemento Engine representa toda la maquinaria de procesamiento de solicitud asociada con un Servicio Catalina en particular. Recibe y procesa todas las solicitudes de uno o más conectores y devuelve la respuesta al conector para la transmisión final al cliente.

Un elemento Engine ha de ser anidado dentro de un elemento de servicio, siguiendo todos los elementos de conector asociados con a dicho servicio.

El elemento engine en Tomcat



El elemento engine en Tomcat

Algunos atributos de Engine:

BackgroundProcessorDelay
Este valor representa el retraso en segundos entre la invocación del método backgroundProcess en este motor y sus contenedores secundarios, incluidos todos los hosts y contextos. Después de esperar la cantidad de tiempo especificada, el subproceso invocará el método backgroundProcess en este motor y todos sus contenedores secundarios. Si no se especifica, el valor predeterminado para este atributo es 10.

El elemento engine en Tomcat

Algunos atributos de Engine:

className
Nombre de clase Java de la implementación que se va a usar. Esta clase debe implementar la interfaz org.apache.catalina.Engine. Si no se especifica, se utilizará el valor estándar (definido a continuación).

El elemento engine en Tomcat

Algunos atributos de Engine:

DefaultHost
El nombre de host predeterminado, que identifica al host que procesará las solicitudes dirigidas a host en este servidor, pero que no están configuradas en este archivo de configuración.

El elemento engine en Tomcat

Algunos atributos de Engine:

Name

Nombre lógico de este Engine, utilizado en mensajes de registro y de error. Cuando se utilizan varios elementos de servicio en el mismo servidor, cada motor debe ser asignado un nombre único.

El elemento engine en Tomcat

Algunos atributos de Engine:

StartStopThreads
El número de subprocesos que este motor utilizará para iniciar elementos secundarios de host en paralelo.

El elemento engine en Tomcat

Algunos atributos de Engine:

undeployOldVersions
Comprobará las versiones antiguas y no utilizadas de las aplicaciones web implementadas mediante despliegue automatico y, si las hay, las eliminará. Este indicador sólo se aplica si autoDeploy es true. Si no se especifica, se utilizará el valor por defecto de false.