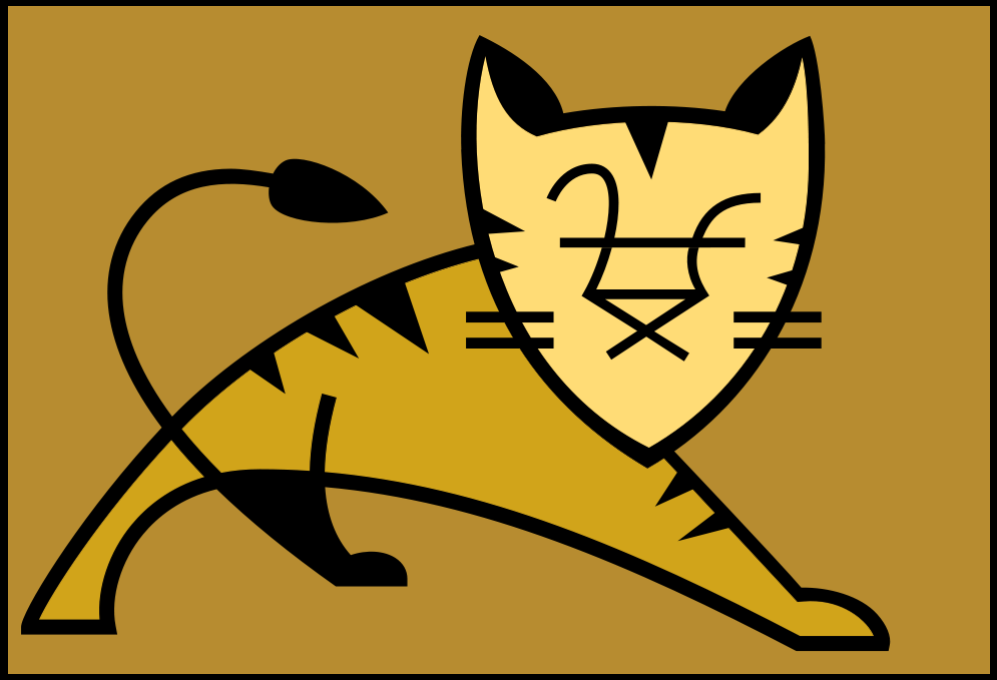
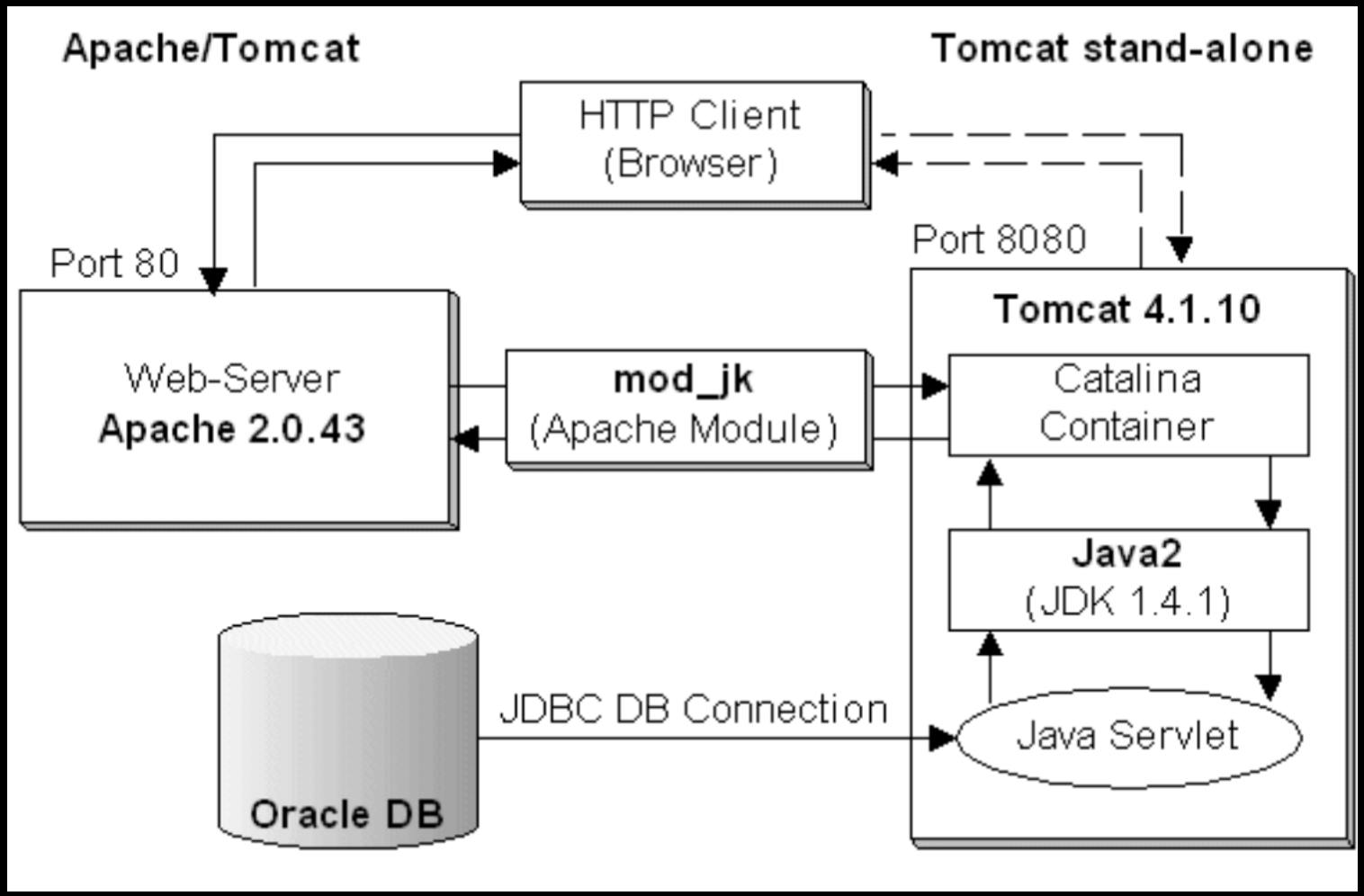
I

ARQUITECTURA E INSTALACIÓN DE APACHE TOMCAT



Iñaki Respaldiza

Arquitectura de Tomcat



Arquitectura de Tomcat

A partir de la versión 4.x, Tomcat fue lanzado con el contenedor de servlets “Catalina”, con el conector HTTP “Coyote” y un motor para JSP llamdo “Jasper”.

Arquitectura de Tomcat

Catalina

Es el contenedor de Servlet. Implementa las especificaciones para servlet de Sun Microsystems y JavaServer Pages (JSP). Contituye la base de datos de usuarios, contraseñas y roles.

Arquitectura de Tomcat

Coyote

Es el conector de componentes de Tomcat que soporta el protocolo HTTP como servidor web. Escucha en un puerto TCP y envía la solicitud al motor Tomcat para que éste procese la solicitud y envie una respuesta al cliente. Esto permite que Catalina, nominalmente un Servlet Java o contenedor JSP, también actúe como un servidor web sencillo que sirve archivos locales como documentos HTTP.

Arquitectura de Tomcat

Jasper

Jasper es el motor JSP de Tomcat. Jasper analiza archivos JSP para compilarlos en código Java como servlets (que pueden ser manejados por Catalina). En tiempo de ejecución, Jasper detecta cambios en archivos JSP y los recompila.

Requerimientos e instalación

Para instalar Tomcat, como cualquier otra aplicación basada en Java, es indispensable que tengamos instalada la JVM (Java Virtual Machine). Por tanto, antes de instalar Tomcat, es necesario comprobar si tenemos lo instalado y si la versión es compatible con la versión de Tomcat que queremos instalar.

Requerimientos e instalación

8.5.15

Please see the README file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
  - zip (pgp, md5, sha1)
  - tar.gz (pgp, md5, sha1)
  - 32-bit Windows zip (pgp, md5, sha1)
  - 64-bit Windows zip (pgp, md5, sha1)
  - 32-bit/64-bit Windows Service Installer (pgp, md5, sha1)
- Full documentation:
  - tar.gz (pgp, md5, sha1)
- Deployer:
  - zip (pgp, md5, sha1)
  - tar.gz (pgp, md5, sha1)
- Extras:
  - JMX Remote jar (pgp, md5, sha1)
  - Web services jar (pgp, md5, sha1)
- Embedded:
  - tar.gz (pgp, md5, sha1)
  - zip (pgp, md5, sha1)

Source Code Distributions

- tar.gz (pgp, md5, sha1)
- zip (pgp, md5, sha1)

Pasos de preinstalación y pos instalación

Antes de instalar Tomcat debemos:
Comprobar nuestra versión de Java
Localizar Java_Home

Despues de instalar:
Definir usuario en tomcat-users.xml
Definir las variables de entorno

Conociendo la estructura interna

- /bin - Startup, shutdown, and other scripts. The *.sh files (for Unix systems) are functional duplicates of the *.bat files (for Windows systems). Since the Win32 command-line lacks certain functionality, there are some additional files in here.
- /conf - Configuration files and related DTDs. The most important file in here is server.xml. It is the main configuration file for the container.
- /logs - Log files are here by default.
- /webapps - This is where your webapps go.

Relación de archivos de script y su configuración

- catalina.sh: script con multiples funciones.
- startup.sh: Inicia el servidor Tomcat (catalina.sh start)
- shutdown.sh: Detiene el servidor Tomcat (catalina.sh stop)
- configtest.sh: Chequea la configuración del servidor (catalina.sh configtest)
- version.sh: Muestra la versión (catalina.sh version)
- digest.sh: making password hash and encrypting password.
- Otros scripts de uso interno setclasspath.sh, classpath append, tool-wrapper.sh, daemon.sh.

Relación de archivos de script y su configuración

- server.xml: que contiene la definición estructural del servidor: Nombre del host, servicios, conectores, etc.
- web.xml: recoge los valores por defecto a utilizar por todas las aplicaciones web cargadas en la instancia de Tomcat, como pudiera ser, por ejemplo, la página a cargar por defecto.
- context.xml: contiene las etiquetas que marcan configuraciones particulares de una aplicación.
- tomcat-users.xml: Base de datos de usuarios, contraseñas y perfiles para la autentificación y el control de acceso.

Opciones de arranque y variables de entorno

La variable JAVA_OPTS permite especificar en una instancia de Tomcat distinas opciones de configuración y variables de entorno, que se aplican a la hora de iniciar o parar el servidor de aplicaciones. Existe controversia en si determinadas variables y parámetros deben ser aplicados dentro de la variable JAVA_OPTS o en CATALINA_ OPTS, por ejemplo para límites de memoria. En principio, el contenido de ambas variables se envía a Tomcat durante el arranque del servicio, pero sólo las aplicadas en JAVA_OPTS son enviadas durante la parada del servicio. La otra y diferencia más importante es que la variable de entorno “JAVA_OPTS” puede ser utilizada por más servicios dentro del mismo sistema mientras que “CATALINA_OPTS” sólo es utilizada por Tomcat.

Configuración inicial de la JVM

server:
Permite cambiar el complilador con nivel de optimización bajo por un compilador que pueda optimizarse. Este cambio aumenta el rendimiento del servidor pero este tardará mas en tiempo en cargarse cuando se utilice el compilador de optimización.

Configuración inicial de la JVM

Parámetros de Establecimiento del tamaño de almacenamiento dinámico

-Xms

Es el tamaño inicial de la pila de JAVA, en resumen, es el parámetro que controla el tamaño inicial del almacenamiento dinámico Java. Ajustandolo correctamente este parámetro reduce la actividad general del garbage collector mejorando asi la productividad y el tiempo de respuesta del servidor.

-Xmx

Es el tamaño maximo de la pila de JAVA, en resumen, es el parámetro que controla el tamaño máximo del almacenamiento dinámico Java. Ajustandolo correctamente este parámetro reduce la actividad general del garbage collecto, mejorando asi la productividad y el tiempo de respuesta del servidor.

Configuración inicial de la JVM

Parámetro de configuración del Garbage Collector

-XX:PermSize

Es el tamaño inicial de la memoria permanente, en resumen, es el parámetro que controla el almacenamiento dinámico reservado para la generación permanente contiene todos los datos reflectivos de la JVM. Este tamaño debe aumentarse para optimizar el rendimiento de las aplicaciones que carga y descarga dinámicamente muchas clases.

-XX:MaxPermSize

Es el tamaño máximo de la memoria permanente, en resumen, es el parámetro que controla el almacenamiento dinámico reservado para la generación permanente contiene todos los datos reflectivos de la JVM. Este tamaño debe aumentarse para optimizar el rendimiento de las aplicaciones que carga y descarga dinámicamente muchas clases.

-XX:UseParNewGC

Este parámetro es un flag que activa el garbaga collector en paralelo.

Configuración inicial de la JVM

Parámetros de Configuración de Archivos de registro para el análisis y optimización del rendimiento del GC

-verbose:gc
Parámetro que prepara la JVM para el registro de información de GC.

-XX:+PrintGCTimeStamps
Parámetro que captura la Fecha y Hora exacta en la que el Garbage Collector.

-XX:+PrintGCDetails
Parámetro que obtiene los detalles acerca de la GC, como el tamaño de la memoria usada o liberada antes y después del GC.

-X:loggc
Parámetro que se usa para especificar el nombre del archivo de registro o log del Garbage Collector, el contenido del mismo puede ser la información registrada en lugar de la salida estándar.

Estructura de una aplicación web

Toda aplicación en Tomcat se encuentra agrupada en WARS (“Web-Archives”), la estructura de un WAR es definida por Sun la cual debe ser implementada en cualquier producto de “Servlet Engine”(Web-Container).

Estructura de una aplicación web

Toda aplicación en Tomcat se encuentra agrupada en WARS (“Web-Archives”), la estructura de un WAR es definida por Sun la cual debe ser implementada en cualquier producto de “Servlet Engine”(Web-Container).

Estructura de una aplicación web

La estructura de un Archivo WAR es la siguiente:
/ *.html *.jsp *.css : Este directorio base contiene los elementos que comúnmente son utilizados en un sitio, Documentos en HTML , JSP's , CSS("Cascading Style Sheets") y otros elementos.

/WEB-INF/web.xml : Contiene elementos de seguridad de la aplicación así como detalles sobre los Servlets que serán utilizados dentro de la misma.

Estructura de una aplicación web

/WEB-INF/classes/ : Contiene las clases Java adicionales a las del JDK que serán empleadas en los JSP's y Servlets

/WEB-INF/lib/ : Contiene los JAR's que serán utilizados por su aplicación.