

Trabajo Práctico curso de Linux.

Instalación y configuración de Tomcat en entorno Linux.

Alumno: Pablo Emanuel Goette.

Copyright (c) 2009, Goette, Pablo Emanuel

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Índice de contenido

¿ Como instalar Tomcat 6 ?.....	4
¿ Como descomprimir la distribución binaria ?.....	4
Instalar jdk:.....	4
¿ Como ejecutar Tomcat ?	5
¿ Como es la estructura de directorios de Tomcat ?.....	6
¿ Como configurar Tomcat ?.....	7
<Server> - </Server>	7
<Listener> - </Listener>.....	7
<GlobalNamingResources> , <Resource> y <ResourceParams>	7
<Connector> - </Connector>	8
<Engine> - </Engine>	9
<Logger> - </Logger>.....	9
<Host> - </Host>	9
<Context> - </Context>	10
¿ Como desplegar aplicaciones web en Tomcat ?.....	10
¿ Como Configurar virtual host con Tomcat ?.....	11
¿ Como Configuración de Realms ?.....	12

¿ Como instalar Tomcat 6 ?

En distribución basada en debían:

```
apt-get install tomcat6
```

Instalar en distribuciones basadas en herramienta de administración de paquete rpm

```
rpm -ql tomcat
```

¿ Como descomprimir la distribución binaria ?

Instalar jdk:

Bajar de java.sun.com la versión de jdk superior a 1.5 para la distribución de linux que corresponda.

```
wget http://cds.sun.com/is-  
bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-  
Site/en_US/-/USD/VerifyItem-Start/jdk-6u2-linux-i586.bin?  
BundledLineItemUUID=LlxIBe.oig0AAAEiNBFGFS8k&OrderID=vDpIBe.  
oEckAAAEiIRFGFS8k&ProductID=KRDACUFBSAQAAAEYifo5AXuS&FileName  
=/jdk-6u2-linux-i586.bin
```

Cambiar el permiso del archivo descargado:

```
chmod 770 jdk-6u2-linux-i586.bin
```

Ejecutar binario:

```
./jdk-6u2-linux-i586.bin
```

Aceptar el termino de licencia.

Mover la jdk y crear link para que linux tome la nueva jdk:

```
mv jdk-1_6_0_02 /usr/local/java  
cd /usr/local/java  
ln -s jdk-1_6_0_02 java
```

Obtener Tomcat, desempaquetarlo y mover lo al directorio donde se quiera ubicar la instalación, en el ejemplo se mueve a /usr/local/:

```
wget http://apache.patan.com.ar/tomcat/tomcat-  
6/v6.0.20/bin/apache-tomcat-6.0.20.tar.gz  
tar xvfz jakarta-tomcat-6.0.20.tar.gz  
mv jakarta-tomcat-6.0.20 /usr/local/  
cd /usr/local  
ln -s jakarta-tomcat-4.1.29 tomcat
```

Configurar las siguientes variables de entorno:

```
export JAVA_HOME=/usr/local/java/java
export CATALINA_HOME=/usr/local/tomcat
export PATH=$JAVA_HOME/bin:$PATH:$HOME/bin:/usr/sbin
export CLASSPATH=$CATALINA_HOME/bin/bootstrap.jar:
$JAVA_HOME/lib/tools.jar:
$CATALINA_HOME/common/lib/servlet.jar:/usr/local/pgsql/share
/java/postgresql.jar:../lib/struts.jar
```

¿ Como ejecutar Tomcat ?

Para ejecutar tomcat debe ejecutar el siguiente comando:

```
./$CATALINA_HOME/bin/start.sh
```

Donde \$CATALINA_HOME es el directorio donde esta instalado tomcat. Si tomcat se instalo por medio de apt-get o rpm tambien se puede ejecutar de la siguiente forma:

```
./etc/init.d/tomcat6 start
```

Y en debian también:

```
invoke-rc.d tomcat6 start
```

Comprobar instalación:

Ingresar a algún browser (firefox por ejemplo) y ingresar a la url: <http://localhost:8080/> y si la instalación fue exitosa se vera la siguiente pagina:

Apache Tomcat

The Apache Software Foundation
<http://www.apache.org/>

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.html
```

where "\$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the [Tomcat Documentation](#) for more detailed setup and administration information than is found in the INSTALL file.

NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in \$CATALINA_HOME/conf/tomcat-users.xml.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- users@tomcat.apache.org for general questions related to configuring and using Tomcat
- dev@tomcat.apache.org for developers working on Tomcat

Thanks for using Tomcat!

Powered by
TOMCAT
Copyright © 1999-2007 Apache Software Foundation
All Rights Reserved

Terminado

¿ Como es la estructura de directorios de Tomcat ?

Asumiendo que hemos descomprimido la distribución binaria de Tomcat deberíamos tener la siguiente estructura de directorios:

Nombre de Directorio	Descripción
bin	Contiene los scripts de arrancar/parar
conf	Contiene varios ficheros de configuración incluyendo <code>server.xml</code> (el fichero de configuración principal de Tomcat) y <code>web.xml</code> que configura los valores por defecto para las distintas aplicaciones desplegadas en Tomcat. Si el tomcat se instalo con <code>apt-get</code> o <code>rpm</code> esta configuración estará en <code>etc/tomcat6</code> y sera referenciada de un link <code>\$CATALINA_HOME/conf</code> (donde <code>\$CATALINA_HOME</code> contiene la ruta donde se instalo tomcat)
lib	Contiene varios ficheros jar que son utilizados por Tomcat. Sobre UNIX/Linux, cualquier fichero de este directorio se añade al classpath de Tomcat. Si se desea agregar un jar para que lo utilice tomcat se debe agregar a este directorio por ejemplo jars <code>jdbc</code> , que utilizara tomcat para conectarse a la base de datos.
logs	Aquí es donde Tomcat sitúa los ficheros de log. Si el tomcat se instalo con <code>apt-get</code> o <code>rpm</code> es un link a <code>/var/log/tomcat6</code>
webapps	Contiene aplicaciones Web de Ejemplo. Y es donde vamos a desplegar nuestras aplicaciones.

Adicionalmente podemos, o Tomcat creará, los siguientes directorios:

Nombre de Directorio	Descripción
work	Generado automáticamente por Tomcat, este es el sitio donde Tomcat sitúa los ficheros intermedios (como las páginas JSP compiladas) durante su trabajo. Si borramos este directorio mientras se está ejecutando Tomcat no podremos ejecutar páginas JSP. Es prudente borrarlo antes de hacer re-deployear una aplicación.
classes	Podemos crear este directorio para añadir clases adicionales al classpath. Cualquier clase que añadamos a este directorio encontrará un lugar en el classpath de Tomcat.

¿ Como configurar Tomcat ?

`server.xml` es el archivo principal de configuración para Tomcat, al igual que otros archivos de configuración para productos empleados en servidor puede contener una gran variedad de parámetros, sin embargo, esta guía se concentrará en los parámetros principales.

El archivo `server.xml` es un archivo en XML, el cual de no contener una estructura conforme a XML, se indicará al arranque de Tomcat; dicho archivo se encuentra bajo el directorio `/etc/tomcat6`, si se instalo con `apt-get` o `rpm`; si se descomprimió la distribución binaria estará en

\$CATALINA_HOME/conf.

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.

<Server> - </Server>

`<Server>` es el elemento principal del archivo `server.xml` y todas las demás secciones deben encontrarse entre estos nodos; el atributo `port` indica el puerto TCP donde se espera la señal de cierre (shutdown) de Tomcat, el cual rara vez es modificado.

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

<Listener> - </Listener>

A través de los elementos `<Listener>` se configuran las extensiones JMX ("Java Management Extensions") que serán utilizadas por Tomcat, dichos elementos toman dos atributos: `className` que indica la Clase diseñada para escuchar sobre eventos JMX y `debug` para especificar el nivel de "debug" generado al tiempo de ejecución. Si entrar en detalle de JMX esto permite monitorizar Tomcat con cualquier aplicación.

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

<GlobalNamingResources> , <Resource> y <ResourceParams>

Anidado dentro de los elementos `<GlobalNamingResources>` es posible definir recursos JNDI para ser utilizados globalmente en Tomcat. Lo anterior evita que estos recursos tengan que ser declarados a nivel de WAR de manera individual.

A través de `<Resource>` es como define el tipo de recurso JNDI que será utilizado y mediante `<ResourceParams>` se especifican los parámetros específicos que tomará el recurso en dicha instancia de Tomcat.

<Connector> - </Connector>

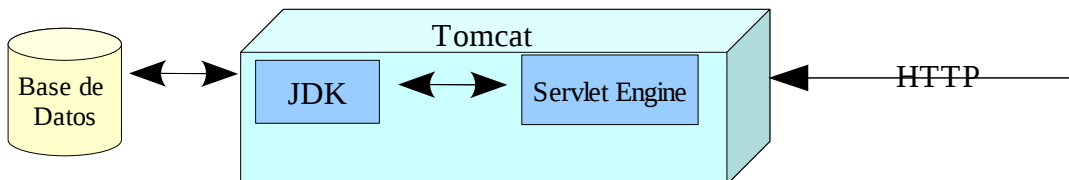
El elemento `Connector` representa las conexiones (Puertos TCP) que serán abiertas por Tomcat al arranque, a su vez dentro de cada elemento `Connector` se definen diversos atributos los cuales dan más detalles acerca de la conexión.

El elemento `Connector` más importante es aquel que define la clase: `HttpConnector`.

```
<Connector port="8080"
    maxThreads="150" minSpareThreads="25"
    maxSpareThreads="75"
    enableLookups="false"
    redirectPort="8443" acceptCount="100"
    debug="0" connectionTimeout="20000"
    disableUploadTimeout="true" />
```

La declaración anterior indica que Tomcat está dispuesto a dar respuesta a requisiciones que arriben en el puerto 8080 del "Host" donde está instalado Tomcat; si recuerda la ejecución y pruebas en Tomcat a esto se debió el agregar el fragmento :8080.

Para ambientes de producción en los cuales toda requisición será atendida por Tomcat el parámetro port de este elemento debe ser modificado al valor de 80, este puerto es el ampliamente conocido puerto TCP 80 con el que intenta comunicarse cualquier Navegador ("Netscape", "Explorer", "Opera" u otro) en Internet.



Otras declaraciones para Connectors son aquellas para utilizar encriptación (HTTPS) y los conectores AJP; la configuración de HTTPS es un tanto extensa y no se describirán los detalles, sin embargo, los conectores AJP son aquellos utilizados para la comunicación entre Apache y Tomcat y son los siguientes:

```
<Connector port="8009"
           enableLookups="false" redirectPort="8443"
           debug="0"
           protocol="AJP/1.3" />
```

Esta declaración indica que el puerto 8009 estará recibiendo conexiones bajo ajp13, lo anterior es solo de interés si utiliza Apache en conjunción con Tomcat.

<Engine> - </Engine>

Los elementos <Engine> los cuales deben encontrarse anidados dentro de <Service> representan el mecanismo que atenderá toda solicitud arribada Tomcat, esto es, toda solicitud recibida por las definiciones Connectors será procesada por <Engine>, los atributos de este elemento son los siguientes:

```
<Engine name="Catalina" defaultHost="localhost"
        debug="0">
```

DefaultHost representa el nombre del servidor Tomcat mientras debug indica el nivel de "debug" generado al tiempo de ejecución.

<Logger> - </Logger>

Los elementos Logger le indican a Tomcat hacia donde deben ser enviados los registros "Logs":

```
<Logger className="org.apache.catalina.logger.FileLogger"
        directory="logs"
        prefix="localhost_log."
        suffix=".txt"
        timestamp="true" />
```

Lo anterior indica que los registros de Tomcat deben ser enviados al archivo `localhost_log.txt`; la ubicación de este registro ("log") puede ser modificada al nivel de los elementos Host los cuales permiten definir *Virtual Hosts*.

<Host> - </Host>

Los elementos Host permiten definir varios Hosts "Virtuales" para Tomcat, esto es, a través del elemento <Engine> se define un *sitio*(localhost) para atender solicitudes, a través de Host es posible definir diversos sitios "Virtuales", su configuración es la siguiente:

```
<Host name="desarrollo.servidorprueba.com"
      debug="0"
      appBase="webapps"
      unpackWARs="true">
```

Lo anterior indica que el sitio `desarrollo.servidorprueba.com` contiene sus aplicaciones (Archivos WAR) bajo el directorio `$/CATALINA_HOME/webapps`; el atributo `unpackWARs` le indica a Tomcat que debe descomprimir los archivos WAR's al arranque. En java las aplicaciones web se empaquetan con la extensión `war`.

Como ya fue mencionado, dentro de estos elementos <Host> es posible utilizar <Logger> para generar registros ("logs") por cada sitio virtual.

<Context> - </Context>

Context es un elemento utilizado para indicar la ubicación de las aplicaciones ejecutadas en Tomcat, en su configuración "Default" estas aplicaciones se encuentran dentro del directorio `webapps` bajo el directorio raíz de Tomcat (`/usr/local/tomcat`), o en entornos que se descomprimió la versión binaria `$/CATALINA_HOME/webapps`.

Una aplicación en Tomcat o cualquier Servlet Engine(Web-Container) es un conjunto de "JSP's (*Java Server Pages*)" y/o "Servlets" agrupados con ciertos parámetros de arranque y seguridad, este conjunto de archivos / aplicación en *todo Servlet Engine* es conocido como un *WAR (Web-Archive)*. En conclusión todos los wars (aplicaciones web empaquetadas) o las aplicaciones web sin empaquetar van en donde indica la propiedad `path` del tag `context`.

¿ Como desplegar aplicaciones web en Tomcat ?

Para desplegar aplicación web en Tomcat solo se debe parar el servidor:
Para distribuciones debian en las que se instalo tomcat:

```
invoke-rc.d tomcat6 stop
```

Para cualquier distribución que se allá instalado tomcat:

```
./etc/init.d/tomcat6 stop
```

Para versiones descomprimidas:

```
./$/CATALINA_HOME/bin/shutdown.sh
```

Mover la aplicación al directorio configurado en context que se usara para el despliegue. Por ejemplo se desplegara una aplicación llamada myaplicacion al directorio webapps

```
mv myaplicacion.war $CATALINA_HOME/webapps
```

Donde \$CATALINA_HOME contiene el path donde se instalo el tomcat.

Levantar tomcat:

Para distribuciones debian en las que se instalo tomcat:

```
invoke-rc.d tomcat6 start
```

Para cualquier distribución que se allá instalado tomcat:

```
./etc/init.d/tomcat6 start
```

Para versiones descomprimidas:

```
./$CATALINA_HOME/bin/start.sh
```

De esta forma la url donde se desplegara la aplicación sera

<http://localhost:8080/myaplicacion/>

Muchas veces deseamos desplegar aplicaciones de forma que la URL para acceder sea <http://host> para esto lo primero que hay que hacer es configurar el tomcat para que escuche el puerto 80. Cambiando el atributo port a 80 del tag Connector del archivo server.xml.

```
<Connector port="80" ...
```

Luego desplegar la aplicación pero cambiando el nombre del war a ROOT.war

```
mv myaplicacion.war $CATALINA_HOME/webapps/ROOT.war
```

Si se encuentra la carpeta ROOT cambiarle el nombre o borrarlo. En el ejemplo cambiamos el nombre.

```
mv ROOT ROOT_
```

Levantar el tomcat y listo!

¿ Como Configurar virtual host con Tomcat ?

Para configurar virtual host en Tomcat es muy fácil tenemos que seguir los siguientes pasos:

Ir a el archivo server.xml (\$CATALINA_HOME/conf/) y agregar los host (en el ejemplo vamos a configurar prueba1.com y prueba2.com)

server.xml original	Nuevo server.xml
<pre><Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true"</pre>	<pre><Host name="prueba1.com" appBase="webapps" unpackWARs="true" autoDeploy="true"</pre>

<pre> xmlValidation="false" xmlNamespaceAware="false"> </Host> </pre>	<pre> xmlValidation="false" xmlNamespaceAware="false"> </Host> <Host name="prueba2.com" appBase="otroWebapps" unpackWARs="true" autoDeploy="true" xmlValidation="false" xmlNamespaceAware="false"> </Host> </pre>
---	---

Podemos ver que en prueba1.com se van a encontrar las aplicaciones desplegadas en webapps y en prueba2.com las de otroWebapps.

Para poder probar su correcto funcionamiento agregamos la siguiente entrada a /etc/host

```
127.0.0.1 localhost prueba1.com prueba2.com
```

Para que tome las url, luego reinicie el servidor.

```
./etc/init.d/tomcat6 restart
```

Al ingresar a <http://prueba1.com:8080> vera las aplicaciones desplegadas en webapps y al ir <http://prueba2.com:8080> las de otroWebapps.

¿ Como Configuración de Realms ?

En Apache Tomcat no existe el concepto htaccess como se entiende en Apache. Para obtener este comportamiento se deben configurar realms.

Un realm es una "base de datos" de usuarios y passwords que nos sirve para implementar un mecanismo de autenticación para aplicaciones Web. Se puede pensar en roles similar a grupos de unix/linux, y se puede dar permisos de acceso a un recurso a un grupo.

En muchos casos es deseable que se autentique con mecanismos ya implementados dentro de una aplicación. Para esto Apache tomcat define una serie interfaces java que se encuentran en el paquete org.apache.catalina.Realm; se puede implementar "plug in" para establecer esta conexión. Tomcat provee 5 formas estándares para autenticación:

JDBCRealm: Autenticación contra una base de datos relacional usando driver jdbc.

```
<Realm className="org.apache.catalina.realm.JDBCRealm"
        debug="99"
        driverName="org.gjt.mm.mysql.Driver"
        connectionURL="jdbc:mysql://localhost/authority?
user=dbuser&password=dbpass"
        userTable="users"
        userNameCol="user_name"
        userCredCol="user_pass"
```

```
userRoleTable="user_roles" roleNameCol="role_name"/>
```

Para el ejemplo se debe tener las siguientes tablas en la base de datos:

```
create table users (  
  user_name      varchar(15) not null primary key,  
  user_pass      varchar(15) not null  
);  
  
create table user_roles (  
  user_name      varchar(15) not null,  
  role_name      varchar(15) not null,  
  primary key (user_name, role_name)  
);
```

DataSourceRealm: Autenticación contra una base de datos relacional, accedida vía un nombre JNDI JDBC DataSource.

```
<Realm className="org.apache.catalina.realm.DataSourceRealm"  
  debug="99"  
  dataSourceName="jdbc/authority"  
  userTable="users"  
  userNameCol="user_name"  
  userCredCol="user_pass"  
  userRoleTable="user_roles" roleNameCol="role_name"/>
```

Para el ejemplo se debe tener las siguientes tablas en la base de datos:

```
create table users (  
  user_name      varchar(15) not null primary key,  
  user_pass      varchar(15) not null  
);  
  
create table user_roles (  
  user_name      varchar(15) not null,  
  role_name      varchar(15) not null,  
  primary key (user_name, role_name)  
);
```

JNDIRealm: Autenticación contra una base de datos LDAP, accedida vía un nombre JNDI provider.

```
<Realm className="org.apache.catalina.realm.JNDIRealm" debug="99"  
  connectionURL="ldap://localhost:389"  
  userPattern="uid={0},ou=people,dc=mycompany,dc=com"  
  roleBase="ou=groups,dc=mycompany,dc=com"  
  roleName="cn"  
  roleSearch="(uniqueMember={0})"
```

```
/>
```

Para el ejemplo se debe tener las siguientes estructura en el ldap:

```
# Define top-level entry
dn: dc=mycompany,dc=com
objectClass: dcObject
dc:mycompany

# Define an entry to contain people
# searches for users are based on this entry
dn: ou=people,dc=mycompany,dc=com
objectClass: organizationalUnit
ou: people

# Define a user entry for Janet Jones
dn: uid=jjones,ou=people,dc=mycompany,dc=com
objectClass: inetOrgPerson
uid: jjones
sn: jones
cn: janet jones
mail: j.jones@mycompany.com
userPassword: janet

# Define a user entry for Fred Bloggs
dn: uid=fbloggs,ou=people,dc=mycompany,dc=com
objectClass: inetOrgPerson
uid: fbloggs
sn: bloggs
cn: fred bloggs
mail: f.bloggs@mycompany.com
userPassword: fred

# Define an entry to contain LDAP groups
# searches for roles are based on this entry
dn: ou=groups,dc=mycompany,dc=com
objectClass: organizationalUnit
ou: groups

# Define an entry for the "tomcat" role
dn: cn=tomcat,ou=groups,dc=mycompany,dc=com
objectClass: groupOfUniqueNames
cn: tomcat
uniqueMember: uid=jjones,ou=people,dc=mycompany,dc=com
uniqueMember: uid=fbloggs,ou=people,dc=mycompany,dc=com

# Define an entry for the "role1" role
dn: cn=role1,ou=groups,dc=mycompany,dc=com
objectClass: groupOfUniqueNames
cn: role1
uniqueMember: uid=fbloggs,ou=people,dc=mycompany,dc=com
```

MemoryRealm: Autenticación contra una colección de objetos user que se encuentran en memoria estos usuarios se cargan al iniciar el servidor del archivo conf/tomcat-users.xml

```
<Realm className="org.apache.catalina.realm.MemoryRealm"
        pathname="$CATALINA_HOME/conf/tomcat-users.xml"
/>
```

Para el ejemplo se debe agregar los usuarios a
\$CATALINA_HOME/conf/tomcat-users.xml

```
<tomcat-users>

  <user name="tomcat" password="tomcat" roles="tomcat" />

  <user name="role1" password="tomcat" roles="role1" />

  <user name="both" password="tomcat" roles="tomcat,role1" />

</tomcat-users>
```

JAASRealm: Autenticación usando Java Authentication & Authorization Service (JAAS) framework.

```
<Realm className="org.apache.catalina.realm.JAASRealm"
        appName="MyFooRealm"
        userClassNames="org.foobar.realm.FooUser"
        roleClassNames="org.foobar.realm.FooRole"
        debug="99" />
```

Pero también es posible escribir nuestras propias implementaciones y integrarlas con tomcat, para esto necesitaremos:

- Implementar org.apache.catalina.Realm.
- Agregar el jar en \$CATALINA_HOME/lib
- Declarar el realm.
- Declarar el realm en el Mbeans Descriptor.

Luego de entender como el funcionamiento de los Reamls vamos a explicar como se configuran:

Se debe agregar el tag Realm

```
<Realm className="... clase que implementa
org.apache.catalina.Realm"
        ... otros atributos.../>
```

El elemento Realm puede estar contenido en los siguientes elementos:

- Dentro de un elemento <Engine>: Este elemento es compartido por todos los virtual host y todas las aplicaciones web. Se puede pisar con los realms configurados en <Host> o <Context>.
- Dentro de un elemento <Host>: Este elemento es por virtual host y todas las aplicaciones web que se encuentran en el mismo. Se puede

pisar con los realms configurados en <Context>.

- Dentro de un elemento <Context>: Este elemento es por aplicación web.