Apache Tomcat

Introduction

Introduction

For administrators and web developers alike, there are some important bits of information you should familiarize yourself with before starting out. This document serves as a brief introduction to some of the concepts and terminology behind the Tomcat container. As well, where to go when you need help.

Terminology

In the course of reading these documents, you will run across a number of terms; some specific to Tomcat, and others defined by the Servlet and JSP specifications.

- **Context** - In a nutshell, a Context is a web application.

That is it. If you find any more terms we need to add to this section, please do let us know.

Directories and Files

These are some of the key tomcat directories:

- **/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.

CATALINA_HOME and CATALINA_BASE

Throughout the documentation, there are references to the two following properties:

- **CATALINA_HOME**: Represents the root of your Tomcat installation, for example /home/tomcat/apache-tomcat-11.0.0 or C:\Program Files\apache-tomcat-11.0.0.
- **CATALINA_BASE**: Represents the root of a runtime configuration of a specific Tomcat instance. If you want to have multiple Tomcat instances on one machine, use the CATALINA_BASE property.

If you set the properties to different locations, the CATALINA_HOME location contains static sources, such as .jar files, or binary files. The CATALINA_BASE location contains configuration files, log files, deployed applications, and other runtime requirements.

Why Use CATALINA_BASE

By default, CATALINA_HOME and CATALINA_BASE point to the same directory. Set CATALINA_BASE manually when you require running multiple Tomcat instances on one machine. Doing so provides the following benefits:

- Easier management of upgrading to a newer version of Tomcat. Because all instances with single CATALINA_HOME location share one set of .jar files and binary files, you can easily upgrade the files to newer version and have the change propagated to all Tomcat instances using the same CATALIA_HOME directory.
- Avoiding duplication of the same static .jar files.
- The possibility to share certain settings, for example the setenv shell or bat script file (depending on your operating system).

Contents of CATALINA_BASE

Before you start using CATALINA_BASE, first consider and create the directory tree used by CATALINA_BASE. Note that if you do not create all the recommended directories, Tomcat creates the directories automatically. If it fails to create the necessary directory, for example due to permission issues, Tomcat

will either fail to start, or may not function correctly.

Consider the following list of directories:

- The bin directory with the setenv.sh, setenv.bat, and tomcat-juli.jar files.

Recommended: No.

Order of lookup: CATALINA_BASE is checked first; fallback is provided to CATALINA_HOME.

- The lib directory with further resources to be added on classpath.

Recommended: Yes, if your application depends on external libraries.

Order of lookup: CATALINA_BASE is checked first; CATALINA_HOME is loaded second.

- The logs directory for instance-specific log files.

Recommended: Yes.

- The webapps directory for automatically loaded web applications.

Recommended: Yes, if you want to deploy applications.

Order of lookup: CATALINA_BASE only.

- The work directory that contains temporary working directories for the deployed web applications.

Recommended: Yes.

- The temp directory used by the JVM for temporary files.

Recommended: Yes.

We recommend you not to change the tomcat-juli.jar file. However, in case you require your own logging implementation, you can replace the tomcat-juli.jar file in a CATALINA_BASE location for the specific Tomcat instance.

We also recommend you copy all configuration files from the CATALINA_HOME/conf directory into the CATALINA_BASE/conf/ directory. In case a configuration file is missing in CATALINA_BASE, there is no fallback to

CATALINA_HOME. Consequently, this may cause failure.

At minimum, CATALINA_BASE must contain:

- conf/server.xml
- conf/web.xml

That includes the conf directory. Otherwise, Tomcat fails to start, or fails to function properly.

For advanced configuration information, see the RUNNING.txt file.

How to Use CATALINA_BASE

The CATALINA_BASE property is an environment variable. You can set it before you execute the Tomcat start script, for example:

- On Unix: CATALINA_BASE=/tmp/tomcat_base1 bin/catalina.sh start
- On Windows: CATALINA_BASE=C:\tomcat_base1 bin/catalina.bat start

Tomcat Setup

Introduction

There are several ways to set up Tomcat for running on different platforms. The main documentation for this is a file called RUNNING.txt. We encourage you to refer to that file if the information below does not answer some of your questions.

Windows

Installing Tomcat on Windows can be done easily using the Windows installer. Its interface and functionality is similar to other wizard based installers, with only a few items of interest.

- **Installation as a service**: Tomcat will be installed as a Windows service no matter what setting is selected. Using the checkbox on the component page sets the service as "auto" startup, so that Tomcat is automatically started when Windows starts. For optimal security, the service should be run as a separate user, with reduced permissions (see the Windows

Services administration tool and its documentation).

- **Java location**: The installer will provide a default JRE to use to run the service. The installer uses the registry to determine the base path of a Java 17 or later JRE, including the JRE installed as part of the full JDK. The installer will first look for a JRE and only look for a JDK if a JRE is not found. Finally, if a JRE or JDK has not been found, the installer will try to use the JAVA_HOME environment variable. It is not mandatory to use the default JRE detected by the installer. Any installed Java 17 or later JRE may be used.
- **Tray icon**: When Tomcat is run as a service, there will not be any tray icon present when Tomcat is running. Note that when choosing to run Tomcat at the end of installation, the tray icon will be used even if Tomcat was installed as a service.
- **Defaults**: The defaults used by the installer may be overridden by use of the /C=<config file> command line argument. The configuration file uses the format name=value with each pair on a separate line. The names of the available configuration options are:
    - JavaHome
    - TomcatPortShutdown
    - TomcatPortHttp
    - TomcatMenuEntriesEnable
    - TomcatShortcutAllUsers
    - TomcatServiceDefaultName
    - TomcatServiceName
    - TomcatServiceFileName
    - TomcatServiceManagerFileName
    - TomcatAdminEnable
    - TomcatAdminUsername

    - TomcatAdminPassword
    - TomcatAdminRoles

By using /C=... along with /S and /D= it is possible to perform fully configured unattended installs of Apache Tomcat.

- Refer to the Windows Service How-To for information on how to manage Tomcat as a Windows service.

The installer will create shortcuts allowing starting and configuring Tomcat. It is important to note that the Tomcat administration web application can only be used when Tomcat is running.

Unix daemon

Tomcat can be run as a daemon using the jsvc tool from the commons-daemon project. Source tarballs for jsvc are included with the Tomcat binaries, and need to be compiled. Building jsvc requires a C ANSI compiler (such as GCC), GNU Autoconf, and a JDK.

Before running the script, the JAVA_HOME environment variable should be set to the base path of the JDK. Alternately, when calling the ./configure script, the path of the JDK may be specified using the --with-java parameter, such as ./configure --with-java=/usr/java.

Using the following commands should result in a compiled jsvc binary, located in the $CATALINA_HOME/bin folder. This assumes that GNU TAR is used, and that CATALINA_HOME is an environment variable pointing to the base path of the Tomcat installation.

Please note that you should use the GNU make (gmake) instead of the native BSD make on FreeBSD systems.

cd $CATALINA_HOME/bin

tar xvfz commons-daemon-native.tar.gz

cd commons-daemon-1.1.x-native-src/unix

./configure

make

cp jsvc ../..

cd ../..

Tomcat can then be run as a daemon using the following commands.

CATALINA_BASE=$CATALINA_HOME

cd $CATALINA_HOME

./bin/jsvc \

    -classpath $CATALINA_HOME/bin/bootstrap.jar:$CATALINA_HOME/bin/tomcat-juli.jar \

    -outfile $CATALINA_BASE/logs/catalina.out \

    -errfile $CATALINA_BASE/logs/catalina.err \

    --add-opens=java.base/java.lang=ALL-UNNAMED \

    --add-opens=java.base/java.io=ALL-UNNAMED \

    --add-opens=java.base/java.util=ALL-UNNAMED \

    --add-opens=java.base/java.util.concurrent=ALL-UNNAMED \

    --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED \

    -Dcatalina.home=$CATALINA_HOME \

    -Dcatalina.base=$CATALINA_BASE \

    -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager \

    -Djava.util.logging.config.file=$CATALINA_BASE/conf/logging.properties \

    org.apache.catalina.startup.Bootstrap

You may also need to specify -jvm server if the JVM defaults to using a server VM rather than a client VM. This has been observed on OSX.

jsvc has other useful parameters, such as -user which causes it to switch to another user after the daemon initialization is complete. This allows, for

example, running Tomcat as a non privileged user while still being able to use privileged ports. Note that if you use this option and start Tomcat as root, you'll need to disable the org.apache.catalina.security.SecurityListener check that prevents Tomcat starting when running as root.

jsvc --help will return the full jsvc usage information. In particular, the -debug option is useful to debug issues running jsvc.

The file $CATALINA_HOME/bin/daemon.sh can be used as a template for starting Tomcat automatically at boot time from /etc/init.d with jsvc.

Note that the Commons-Daemon JAR file must be on your runtime classpath to run Tomcat in this manner. The Commons-Daemon JAR file is in the Class-Path entry of the bootstrap.jar manifest, but if you get a ClassNotFoundException or a NoClassDefFoundError for a Commons-Daemon class, add the Commons-Daemon JAR to the -cp argument when launching jsvc.

Tomcat Web Application Deployment

Introduction

Deployment is the term used for the process of installing a web application (either a 3rd party WAR or your own custom web application) into the Tomcat server.

Web application deployment may be accomplished in a number of ways within the Tomcat server.

- Statically; the web application is setup before Tomcat is started
- Dynamically; by directly manipulating already deployed web applications (relying on *auto-deployment* feature) or remotely by using the Tomcat Manager web application

The Tomcat Manager is a web application that can be used interactively (via HTML GUI) or programmatically (via URL-based API) to deploy and manage web applications.

There are a number of ways to perform deployment that rely on the Manager web

application. Apache Tomcat provides tasks for Apache Ant build tool. Apache Tomcat Maven Plugin project provides integration with Apache Maven. There is also a tool called the Client Deployer, which can be used from a command line and provides additional functionality such as compiling and validating web applications as well as packaging web application into web application resource (WAR) files.

Installation

There is no installation required for static deployment of web applications as this is provided out of the box by Tomcat. Nor is any installation required for deployment functions with the Tomcat Manager, although some configuration is required as detailed in the Tomcat Manager manual. An installation is however required if you wish to use the Tomcat Client Deployer (TCD).

The TCD is not packaged with the Tomcat core distribution, and must therefore be downloaded separately from the Downloads area. The download is usually labelled *apache-tomcat-11.0.x-deployer*.

TCD has prerequisites of Apache Ant 1.6.2+ and a Java installation. Your environment should define an ANT_HOME environment value pointing to the root of your Ant installation, and a JAVA_HOME value pointing to your Java installation. Additionally, you should ensure Ant's ant command, and the Java javac compiler command run from the command shell that your operating system provides.

1. Download the TCD distribution
2. The TCD package need not be extracted into any existing Tomcat installation, it can be extracted to any location.
3. Read Using the Tomcat Client Deployer

A word on Contexts

In talking about deployment of web applications, the concept of a *Context* is required to be understood. A Context is what Tomcat calls a web application.

In order to configure a Context within Tomcat a *Context Descriptor* is required. A

Context Descriptor is simply an XML file that contains Tomcat related configuration for a Context, e.g naming resources or session manager configuration. In earlier versions of Tomcat the content of a Context Descriptor configuration was often stored within Tomcat's primary configuration file *server.xml* but this is now discouraged (although it currently still works).

Context Descriptors not only help Tomcat to know how to configure Contexts but other tools such as the Tomcat Manager and TCD often use these Context Descriptors to perform their roles properly.

The locations for Context Descriptors are:

1. $CATALINA_BASE/conf/[enginename]/[hostname]/[webappname].xml
2. $CATALINA_BASE/webapps/[webappname]/META-INF/context.xml

Files in (1) are named [webappname].xml but files in (2) are named context.xml. If a Context Descriptor is not provided for a Context, Tomcat configures the Context using default values.

Deployment on Tomcat startup

If you are not interested in using the Tomcat Manager, or TCD, then you'll need to deploy your web applications statically to Tomcat, followed by a Tomcat startup. The location you deploy web applications to for this type of deployment is called the appBase which is specified per Host. You either copy a so-called *exploded web application*, i.e non-compressed, to this location, or a compressed web application resource .WAR file.

The web applications present in the location specified by the Host's (default Host is "localhost") appBase attribute (default appBase is "$CATALINA_BASE/webapps") will be deployed on Tomcat startup only if the Host's deployOnStartup attribute is "true".

The following deployment sequence will occur on Tomcat startup in that case:

1. Any Context Descriptors will be deployed first.
2. Exploded web applications not referenced by any Context Descriptor will then be deployed. If they have an associated .WAR file in the appBase and

it is newer than the exploded web application, the exploded directory will be removed and the webapp will be redeployed from the .WAR

3. .WAR files will be deployed

Deploying on a running Tomcat server

It is possible to deploy web applications to a running Tomcat server.

If the Host autoDeploy attribute is "true", the Host will attempt to deploy and update web applications dynamically, as needed, for example if a new .WAR is dropped into the appBase. For this to work, the Host needs to have background processing enabled which is the default configuration.

autoDeploy set to "true" and a running Tomcat allows for:

- Deployment of .WAR files copied into the Host appBase.
- Deployment of exploded web applications which are copied into the Host appBase.
- Re-deployment of a web application which has already been deployed from a .WAR when the new .WAR is provided. In this case the exploded web application is removed, and the .WAR is expanded again. Note that the explosion will not occur if the Host is configured so that .WARs are not exploded with a unpackWARs attribute set to "false", in which case the web application will be simply redeployed as a compressed archive.
- Re-loading of a web application if the /WEB-INF/web.xml file (or any other resource defined as a WatchedResource) is updated.
- Re-deployment of a web application if the Context Descriptor file from which the web application has been deployed is updated.
- Re-deployment of dependent web applications if the global or per-host Context Descriptor file used by the web application is updated.
- Re-deployment of a web application if a Context Descriptor file (with a filename corresponding to the Context path of the previously deployed web application) is added to the $CATALINA_BASE/conf/[enginename]/[hostname]/ directory.

- Undeployment of a web application if its document base (docBase) is deleted. Note that on Windows, this assumes that anti-locking features (see Context configuration) are enabled, otherwise it is not possible to delete the resources of a running web application.

Note that web application reloading can also be configured in the loader, in which case loaded classes will be tracked for changes.

Deploying using the Tomcat Manager

The Tomcat Manager is covered in its own manual page.

Deploying using the Client Deployer Package

Finally, deployment of web application may be achieved using the Tomcat Client Deployer. This is a package which can be used to validate, compile, compress to .WAR, and deploy web applications to production or development Tomcat servers. It should be noted that this feature uses the Tomcat Manager and as such the target Tomcat server should be running.

It is assumed the user will be familiar with Apache Ant for using the TCD. Apache Ant is a scripted build tool. The TCD comes pre-packaged with a build script to use. Only a modest understanding of Apache Ant is required (installation as listed earlier in this page, and familiarity with using the operating system command shell and configuring environment variables).

The TCD includes Ant tasks, the Jasper page compiler for JSP compilation before deployment, as well as a task which validates the web application Context Descriptor. The validator task (class org.apache.catalina.ant.ValidatorTask) allows only one parameter: the base path of an exploded web application.

The TCD uses an exploded web application as input (see the list of the properties used below). A web application that is programmatically deployed with the deployer may include a Context Descriptor in /META-INF/context.xml.

The TCD includes a ready-to-use Ant script, with the following targets:

- compile (default): Compile and validate the web application. This can be used standalone, and does not need a running Tomcat server. The

compiled application will only run on the associated *Tomcat X.Y.Z* server release, and is not guaranteed to work on another Tomcat release, as the code generated by Jasper depends on its runtime component. It should also be noted that this target will also compile automatically any Java source file located in the /WEB-INF/classes folder of the web application.

- deploy: Deploy a web application (compiled or not) to a Tomcat server.
- undeploy: Undeploy a web application
- start: Start web application
- reload: Reload web application
- stop: Stop web application

In order for the deployment to be configured, create a file called deployer.properties in the TCD installation directory root. In this file, add the following name=value pairs per line:

Additionally, you will need to ensure that a user has been setup for the target Tomcat Manager (which TCD uses) otherwise the TCD will not authenticate with the Tomcat Manager and the deployment will fail. To do this, see the Tomcat Manager page.

- build: The build folder used will be, by default, ${build}/webapp/${path} (${build}, by default, points to ${basedir}/build). After the end of the execution of the compile target, the web application .WAR will be located at ${build}/webapp/${path}.war.
- webapp: The directory containing the exploded web application which will be compiled and validated. By default, the folder is myapp.
- path: Deployed context path of the web application, by default /myapp.
- url: Absolute URL to the Tomcat Manager web application of a running Tomcat server, which will be used to deploy and undeploy the web application. By default, the deployer will attempt to access a Tomcat instance running on localhost, at http://localhost:8080/manager/text.
- username: Tomcat Manager username (user should have a role of

manager-script)

- password: Tomcat Manager password.