## Documentation Index

### Introduction

This is the top-level entry point of the documentation bundle for the **Apache Tomcat** Servlet/JSP container（tomcat就是servlet/jsp容器）. Apache Tomcat version 8.5 implements the Servlet 3.1 and JavaServer Pages 2.3 [specifications](http://wiki.apache.org/tomcat/Specifications)（遵从了servlet3.1版本和jsp 2.3的约定） from the [Java Community Process](http://www.jcp.org/), and includes many additional features（添加了一些特点） that make it a useful platform for developing and deploying web applications and web services（开发和部署web应用和服务，应用就是为了最终用户服务，而服务可能是为了中间件/服务提供服务）.

Select one of the links from the navigation menu (to the left) to drill down to the more detailed documentation that is available. Each available manual is described in more detail below.

### Apache Tomcat User Guide

The following documents will assist you in downloading and installing Apache Tomcat, and using many of the Apache Tomcat features.

1. [**Introduction**](https://tomcat.apache.org/tomcat-8.5-doc/introduction.html) - A brief, high level, overview of Apache Tomcat.
2. [**Setup**](https://tomcat.apache.org/tomcat-8.5-doc/setup.html) - How to install and run Apache Tomcat on a variety of platforms.
3. [**First web application**](https://tomcat.apache.org/tomcat-8.5-doc/appdev/index.html) - An introduction to the concepts of a web application as defined in the Servlet Specification. Covers basic organization of your web application source tree, the structure of a web application archive, and an introduction to the web application deployment descriptor (/WEB-INF/web.xml).
4. [**Deployer**](https://tomcat.apache.org/tomcat-8.5-doc/deployer-howto.html) - Operating the Apache Tomcat Deployer to deploy, precompile, and validate web applications.
5. [**Manager**](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html) - Operating the **Manager** web app to deploy, undeploy, and redeploy applications while Apache Tomcat is running.
6. [**Realms and Access Control**](https://tomcat.apache.org/tomcat-8.5-doc/realm-howto.html) - Description of how to configure Realms (databases of users, passwords, and their associated roles) for use in web applications that utilize Container Managed Security.
7. [**Security Manager**](https://tomcat.apache.org/tomcat-8.5-doc/security-manager-howto.html) - Configuring and using a Java Security Manager to support fine-grained control over the behavior of your web applications.
8. [**JNDI Resources**](https://tomcat.apache.org/tomcat-8.5-doc/jndi-resources-howto.html) - Configuring standard and custom resources in the JNDI naming context that is provided to each web application.
9. [**JDBC DataSource**](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html) - Configuring a JNDI DataSource with a DB connection pool. Examples for many popular databases.
10. [**Classloading**](https://tomcat.apache.org/tomcat-8.5-doc/class-loader-howto.html) - Information about class loading in Apache Tomcat, including where to place your application classes so that they are visible.
11. [**JSPs**](https://tomcat.apache.org/tomcat-8.5-doc/jasper-howto.html) - Information about Jasper configuration, as well as the JSP compiler usage.
12. [**SSL/TLS**](https://tomcat.apache.org/tomcat-8.5-doc/ssl-howto.html) - Installing and configuring SSL/TLS support so that your Apache Tomcat will serve requests using the https protocol.
13. [**SSI**](https://tomcat.apache.org/tomcat-8.5-doc/ssi-howto.html) - Using Server Side Includes in Apache Tomcat.
14. [**CGI**](https://tomcat.apache.org/tomcat-8.5-doc/cgi-howto.html) - Using CGIs with Apache Tomcat.
15. [**Proxy Support**](https://tomcat.apache.org/tomcat-8.5-doc/proxy-howto.html) - Configuring Apache Tomcat to run behind a proxy server (or a web server functioning as a proxy server).
16. [**MBeans Descriptors**](https://tomcat.apache.org/tomcat-8.5-doc/mbeans-descriptors-howto.html) - Configuring MBean descriptors files for custom components.
17. [**Default Servlet**](https://tomcat.apache.org/tomcat-8.5-doc/default-servlet.html) - Configuring the default servlet and customizing directory listings.
18. [**Apache Tomcat Clustering**](https://tomcat.apache.org/tomcat-8.5-doc/cluster-howto.html) - Enable session replication in a Apache Tomcat environment.
19. [**Balancer**](https://tomcat.apache.org/tomcat-8.5-doc/balancer-howto.html) - Configuring, using, and extending the load balancer application.
20. [**Connectors**](https://tomcat.apache.org/tomcat-8.5-doc/connectors.html) - Connectors available in Apache Tomcat, and native web server integration.
21. [**Monitoring and Management**](https://tomcat.apache.org/tomcat-8.5-doc/monitoring.html) - Enabling JMX Remote support, and using tools to monitor and manage Apache Tomcat.
22. [**Logging**](https://tomcat.apache.org/tomcat-8.5-doc/logging.html) - Configuring logging in Apache Tomcat.
23. [**Apache Portable Runtime**](https://tomcat.apache.org/tomcat-8.5-doc/apr.html) - Using APR to provide superior performance, scalability and better integration with native server technologies.
24. [**Virtual Hosting**](https://tomcat.apache.org/tomcat-8.5-doc/virtual-hosting-howto.html) - Configuring virtual hosting in Apache Tomcat.
25. [**Advanced IO**](https://tomcat.apache.org/tomcat-8.5-doc/aio.html) - Extensions available over regular, blocking IO.
26. [**Additional Components**](https://tomcat.apache.org/tomcat-8.5-doc/extras.html) - Obtaining additional, optional components.
27. [**Using Tomcat libraries with Maven**](https://tomcat.apache.org/tomcat-8.5-doc/maven-jars.html) - Obtaining Tomcat jars through Maven.
28. [**Security Considerations**](https://tomcat.apache.org/tomcat-8.5-doc/security-howto.html) - Options to consider when securing an Apache Tomcat installation.
29. [**Windows Service**](https://tomcat.apache.org/tomcat-8.5-doc/windows-service-howto.html) - Running Tomcat as a service on Microsoft Windows.
30. [**Windows Authentication**](https://tomcat.apache.org/tomcat-8.5-doc/windows-auth-howto.html) - Configuring Tomcat to use integrated Windows authentication.
31. [**High Concurrency JDBC Pool**](https://tomcat.apache.org/tomcat-8.5-doc/jdbc-pool.html) - Configuring Tomcat to use an alternative JDBC pool.
32. [**WebSocket support**](https://tomcat.apache.org/tomcat-8.5-doc/web-socket-howto.html) - Developing WebSocket applications for Apache Tomcat.
33. [**URL rewrite**](https://tomcat.apache.org/tomcat-8.5-doc/rewrite.html) - Using the regexp based rewrite valve for conditional URL and host rewrite.

### Reference

The following documents are aimed at System Administrators who are responsible for installing, configuring, and operating an Apache Tomcat server.

* [**Release notes**](https://tomcat.apache.org/tomcat-8.5-doc/RELEASE-NOTES.txt) - Known issues in this Apache Tomcat release.
* [**Apache Tomcat Server Configuration Reference**](https://tomcat.apache.org/tomcat-8.5-doc/config/index.html) - Reference manual that documents all available elements and attributes that may be placed into the Apache Tomcat conf/server.xml file.
* [**JK Documentation**](http://tomcat.apache.org/connectors-doc/index.html) - Complete documentation and HOWTOs on the JK native webserver connector, used to interface Apache Tomcat with servers like Apache HTTPd, IIS and others.
* Servlet 3.1 [**Specification**](http://jcp.org/aboutJava/communityprocess/final/jsr340/index.html) and **[Javadoc](http://docs.oracle.com/javaee/7/api/javax/servlet/package-summary.html)**
* JSP 2.3 [**Specification**](http://jcp.org/aboutJava/communityprocess/mrel/jsr245/index2.html) and **[Javadoc](http://docs.oracle.com/javaee/7/api/javax/servlet/jsp/package-summary.html)**
* EL 3.0 [**Specification**](http://jcp.org/aboutJava/communityprocess/final/jsr341/index.html) and **[Javadoc](http://docs.oracle.com/javaee/7/api/javax/el/package-summary.html)**
* WebSocket 1.1 [**Specification**](https://jcp.org/aboutJava/communityprocess/mrel/jsr356/index.html) and **[Javadoc](http://docs.oracle.com/javaee/7/api/javax/websocket/package-summary.html)**
* JASPIC 1.1 [**Specification**](https://jcp.org/aboutJava/communityprocess/mrel/jsr196/index.html) and **[Javadoc](http://docs.oracle.com/javaee/7/api/javax/security/auth/message/package-summary.html)**

### Apache Tomcat Developers

The following documents are for Java developers who wish to contribute to the development of the Apache Tomcat project.

* [**Building from Source**](https://tomcat.apache.org/tomcat-8.5-doc/building.html) - Details the steps necessary to download Apache Tomcat source code (and the other packages that it depends on), and build a binary distribution from those sources.
* [**Changelog**](https://tomcat.apache.org/tomcat-8.5-doc/changelog.html) - Details the changes made to Apache Tomcat.
* [**Status**](http://wiki.apache.org/tomcat/TomcatVersions) - Apache Tomcat development status.
* [**Developers**](https://tomcat.apache.org/tomcat-8.5-doc/developers.html) - List of active Apache Tomcat contributors.
* [**Functional Specifications**](https://tomcat.apache.org/tomcat-8.5-doc/funcspecs/index.html) - Requirements specifications for features of the Catalina servlet container portion of Apache Tomcat.
* [**Javadocs**](https://tomcat.apache.org/tomcat-8.5-doc/api/index.html) - Javadoc API documentation for Apache Tomcat's internals.
* [**Apache Tomcat Architecture**](https://tomcat.apache.org/tomcat-8.5-doc/architecture/index.html) - Documentation of the Apache Tomcat Server Architecture.

### Comments

**Notice:**This comments section collects your suggestions on improving documentation for Apache Tomcat.  
  
If you have trouble and need help, read [Find Help](http://tomcat.apache.org/findhelp.html) page and ask your question on the tomcat-users [mailing list](http://tomcat.apache.org/lists.html). Do not ask such questions here. This is not a Q&A section.  
  
The Apache Comments System is explained [here](https://tomcat.apache.org/tomcat-8.5-doc/comments.html). Comments may be removed by our moderators if they are either implemented or considered invalid/off-topic.

#### [RSS](https://comments.apache.org/rss.lua?site=tomcat&page=http://tomcat.apache.org/tomcat-8.5-doc/index.html)   [Log in / register](https://comments.apache.org/portal.lua?site=tomcat)

No comments have been made so far.

### Apache Tomcat User Guide

## Introduction

### Table of Contents

* [Introduction](https://tomcat.apache.org/tomcat-8.5-doc/introduction.html#Introduction)
* [Terminology](https://tomcat.apache.org/tomcat-8.5-doc/introduction.html#Terminology)
* [Directories and Files](https://tomcat.apache.org/tomcat-8.5-doc/introduction.html#Directories_and_Files)
* [Configuring Tomcat](https://tomcat.apache.org/tomcat-8.5-doc/introduction.html#Configuring_Tomcat)
* [Where to Go for Help](https://tomcat.apache.org/tomcat-8.5-doc/introduction.html#Where_to_Go_for_Help)

### 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（术语） 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（术语）; some specific to Tomcat, and others defined by the [Servlet and JSP specifications](http://wiki.apache.org/tomcat/Specifications).

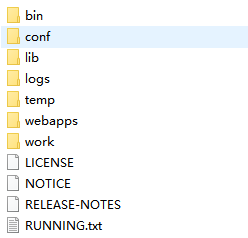
* **Context** - In a nutshell, a Context is a web application（一个context就是一个web应用）.
* **Term2** - This is it.
* **Term3** - This is it!

### Directories and Files

Throughout the docs, you'll notice there are numerous references to **$CATALINA\_HOME**. This represents the root of your Tomcat installation（安装tomcat的根目录）. When we say, "This information can be found in your $CATALINA\_HOME/README.txt file" we mean to look at the README.txt file at the root of your Tomcat install. Optionally, Tomcat may be configured for multiple instances（tomcat可以配置多实例web应用） by defining **$CATALINA\_BASE** for each instance. If multiple instances are not configured, **$CATALINA\_BASE** is the same as **$CATALINA\_HOME**.

These are some of the key tomcat directories:

* **/bin** - （脚本目录）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** - （配置文件和DTD，server.xml）Configuration files and related DTDs. The most important file in here is server.xml. It is the main configuration file for the container（server是容器的配置）.
* **/logs** - Log files are here by default.
* **/webapps** -（应用部署目录） This is where your webapps go.



### Configuring Tomcat

This section will acquaint you with the basic information used during the configuration of the container.

All of the information in the configuration files is read at startup（容器/web应用启动时会读取配置，所以修改后要重新启动）, meaning that any change to the files necessitates a restart of the container.

### Where to Go for Help

While we've done our best to ensure that these documents are clearly written and easy to understand, we may have missed something. Provided below are various web sites and mailing lists in case you get stuck.

Keep in mind that some of the issues and solutions vary between the major versions of Tomcat. As you search around the web, there will be some documentation that is not relevant to Tomcat 8, but only to earlier versions.

* Current document - most documents will list potential hangups. Be sure to fully read the relevant documentation as it will save you much time and effort. There's nothing like scouring the web only to find out that the answer was right in front of you all along!
* [Tomcat FAQ](http://wiki.apache.org/tomcat/FAQ)
* [Tomcat WIKI](http://wiki.apache.org/tomcat/)
* Tomcat FAQ at [jGuru](http://www.jguru.com/faq/home.jsp?topic=Tomcat)
* Tomcat mailing list archives - numerous sites archive the Tomcat mailing lists. Since the links change over time, clicking here will search [Google](http://www.google.com/search?q=tomcat+mailing+list+archives).
* The TOMCAT-USER mailing list, which you can subscribe to [here](http://tomcat.apache.org/lists.html). If you don't get a reply, then there's a good chance that your question was probably answered in the list archives or one of the FAQs. Although questions about web application development in general are sometimes asked and answered, please focus your questions on Tomcat-specific issues.
* The TOMCAT-DEV mailing list, which you can subscribe to [here](http://tomcat.apache.org/lists.html). This list is **reserved** for discussions about the development of Tomcat itself. Questions about Tomcat configuration, and the problems you run into while developing and running applications, will normally be more appropriate on the TOMCAT-USER list instead.

And, if you think something should be in the docs, by all means let us know on the TOMCAT-DEV list.

### Comments

**Notice:**This comments section collects your suggestions on improving documentation for Apache Tomcat.  
  
If you have trouble and need help, read [Find Help](http://tomcat.apache.org/findhelp.html) page and ask your question on the tomcat-users [mailing list](http://tomcat.apache.org/lists.html). Do not ask such questions here. This is not a Q&A section.  
  
The Apache Comments System is explained [here](https://tomcat.apache.org/tomcat-8.5-doc/comments.html). Comments may be removed by our moderators if they are either implemented or considered invalid/off-topic.

#### [RSS](https://comments.apache.org/rss.lua?site=tomcat&page=http://tomcat.apache.org/tomcat-8.5-doc/introduction.html)   [Log in / register](https://comments.apache.org/portal.lua?site=tomcat)

**Uwe Freese** *177 days ago* Rating: 0 (register an account in order to rate comments)

When reading as new user, it is a little disturbing and looks like old documentation when already the 3rd section ("Terminology") contains placeholders...

**Tim** *204 days ago* Rating: 0 (register an account in order to rate comments)

Please explain what a "container" is, perhaps Term 2. Is it a java process running tomcat or some tomcat controlled division?

**Konstantin Kolinko** *204 days ago* Rating: 0 (register an account in order to rate comments)

On these pages "container" is a term from Java EE specifications. E.g. see Servlet 3.1 specification - chapter 1.2, JSP 2.3 specification - chapter JSP.1.1.1, Java API for WebSocket spec. 1.1 - chapter 2.1.7. Tomcat is a "Servlet / JSP / WebSocket container". It can be run standalone (and be an "application server") or can be a component of a larger product - e.g. of Apache TomEE.

## Tomcat Setup

### Table of Contents

* [Introduction](https://tomcat.apache.org/tomcat-8.5-doc/setup.html#Introduction)
* [Windows](https://tomcat.apache.org/tomcat-8.5-doc/setup.html#Windows)
* [Unix daemon](https://tomcat.apache.org/tomcat-8.5-doc/setup.html#Unix_daemon)

### Introduction

There are several ways to set up Tomcat for running on different platforms（不同平台有多种方式启动tomcat）. The main documentation for this is a file called [RUNNING.txt](https://tomcat.apache.org/tomcat-8.5-doc/RUNNING.txt)(根目录下的RUNNKING.txt文件，具体本节下). 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（单独的用户来运行以便控制用户的操作权限）, 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 7 or later JRE, including the JRE installed as part of the full JDK. When running on a 64-bit operating system, the installer will first look for a 64-bit JRE and only look for a 32-bit JRE if a 64-bit JRE is not found. It is not mandatory to use the default JRE detected by the installer（不强制必须使用tomcat指定的jre，可以使用任何java 7及之后的版本）. Any installed Java 7 or later JRE (32-bit or 64-bit) may be used.
* **Tray icon**: When Tomcat is run as a service, there will not be any tray icon（托盘图标） 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 formatname=value with each pair on a separate line（安装时可以使用/c=<config file>来制定配置文件，这个配置文件中类似app.properites，k=v的格式来配置相关选项）. The names of the available configuration options are:
  + JavaHome
  + TomcatPortShutdown
  + TomcatPortHttp
  + TomcatPortAjp
  + 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 Tomact.

* Refer to the [Windows Service HOW-TO](https://tomcat.apache.org/tomcat-8.5-doc/windows-service-howto.html) 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（只有启动了tomcat才能使用管理功能） can only be used when Tomcat is running.

### Unix daemon

Tomcat can be run as a daemon using the jsvc tool（使用jsvc工具启动tomcat为服务进程） 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（也可以在安装时直接指定java home 目录） 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（CATALINA\_HOME为tomcat的根目录环境变量）.

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-1.0.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 \ 启动

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

-outfile $CATALINA\_BASE/logs/catalina.out \ 指定输出日志

-errfile $CATALINA\_BASE/logs/catalina.err \ 指定错误日志

-Dcatalina.home=$CATALINA\_HOME \ 指定home

-Dcatalina.base=$CATALINA\_BASE \ 指定base，多实例

-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager \ 指定日志依赖

-Djava.util.logging.config.file=$CATALINA\_BASE/conf/logging.properties \ 指定日志配置

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（以root用户启动需要禁止SecurityListener，因为它会检查并阻止以root用户启动tomcat）, you'll need to disable theorg.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(启动脚本模板) 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.

### Comments

**Notice:**This comments section collects your suggestions on improving documentation for Apache Tomcat.  
  
If you have trouble and need help, read [Find Help](http://tomcat.apache.org/findhelp.html) page and ask your question on the tomcat-users [mailing list](http://tomcat.apache.org/lists.html). Do not ask such questions here. This is not a Q&A section.  
  
The Apache Comments System is explained [here](https://tomcat.apache.org/tomcat-8.5-doc/comments.html). Comments may be removed by our moderators if they are either implemented or considered invalid/off-topic.

#### [RSS](https://comments.apache.org/rss.lua?site=tomcat&page=http://tomcat.apache.org/tomcat-8.5-doc/setup.html)   [Log in / register](https://comments.apache.org/portal.lua?site=tomcat)

**Wouter** *201 days ago* Rating: 0 (register an account in order to rate comments)

The difference between the different Binary Distribution downloads on the download page is unclear to me (<http://tomcat.apache.org/download-80.cgi)>. I was hoping to find an explanation of the difference between the different Core download options, specifically the difference between "Windows zip" and "Windows Service Installer". But the information on this page seems to suggest Tomcat always installs as a service. I might have overlooked something...

**Tim** *204 days ago* Rating: 0 (register an account in order to rate comments)

Surely you would only have to use "-jvm server" if the JVM normally defaults to a "client" VM when starting the Tomcat daemon?

## Table of Contents

### Preface

This manual includes contributions from many members of the Tomcat Project developer community. The following authors have provided significant content:

* Craig R. McClanahan ([craigmcc@apache.org](mailto:craigmcc@apache.org))

### Table of Contents

The information presented is divided into the following sections:

* [**Introduction**](https://tomcat.apache.org/tomcat-8.5-doc/appdev/introduction.html) - Briefly describes the information covered here, with links and references to other sources of information.
* [**Installation**](https://tomcat.apache.org/tomcat-8.5-doc/appdev/installation.html) - Covers acquiring and installing the required software components to use Tomcat for web application development.
* [**Deployment Organization**](https://tomcat.apache.org/tomcat-8.5-doc/appdev/deployment.html) - Discusses the standard directory layout for a web application (defined in the Servlet API Specification), the Web Application Deployment Descriptor, and options for integration with Tomcat in your development environment.
* [**Source Organization**](https://tomcat.apache.org/tomcat-8.5-doc/appdev/source.html) - Describes a useful approach to organizing the source code directories for your project, and introduces the build.xml used by Ant to manage compilation.
* [**Development Processes**](https://tomcat.apache.org/tomcat-8.5-doc/appdev/processes.html) - Provides brief descriptions of typical development processes utilizing the recommended deployment and source organizations.
* [**Example Application**](https://tomcat.apache.org/tomcat-8.5-doc/appdev/sample/) - This directory contains a very simple, but functionally complete, "Hello, World" application built according to the principles described in this manual. You can use this application to practice using the described techniques.

### Comments

**Notice:**This comments section collects your suggestions on improving documentation for Apache Tomcat.  
  
If you have trouble and need help, read [Find Help](http://tomcat.apache.org/findhelp.html) page and ask your question on the tomcat-users [mailing list](http://tomcat.apache.org/lists.html). Do not ask such questions here. This is not a Q&A section.  
  
The Apache Comments System is explained [here](https://tomcat.apache.org/tomcat-8.5-doc/comments.html). Comments may be removed by our moderators if they are either implemented or considered invalid/off-topic.

#### [RSS](https://comments.apache.org/rss.lua?site=tomcat&page=http://tomcat.apache.org/tomcat-8.5-doc/appdev/index.html)   [Log in / register](https://comments.apache.org/portal.lua?site=tomcat)

No comments have been made so far.

## Tomcat Web Application Deployment

### Table of Contents

* [Introduction](https://tomcat.apache.org/tomcat-8.5-doc/deployer-howto.html#Introduction)
* [Installation](https://tomcat.apache.org/tomcat-8.5-doc/deployer-howto.html#Installation)
* [A word on Contexts](https://tomcat.apache.org/tomcat-8.5-doc/deployer-howto.html#A_word_on_Contexts)
* [Deployment on Tomcat startup](https://tomcat.apache.org/tomcat-8.5-doc/deployer-howto.html#Deployment_on_Tomcat_startup)
* [Deploying on a running Tomcat server](https://tomcat.apache.org/tomcat-8.5-doc/deployer-howto.html#Deploying_on_a_running_Tomcat_server)
* [Deploying using the Tomcat Manager](https://tomcat.apache.org/tomcat-8.5-doc/deployer-howto.html#Deploying_using_the_Tomcat_Manager)
* [Deploying using the Client Deployer Package](https://tomcat.apache.org/tomcat-8.5-doc/deployer-howto.html#Deploying_using_the_Client_Deployer_Package)

### 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（部署web应用的几种方式）.

* 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 手动部署web或者远程通过tomcat管理web应用

The [Tomcat Manager](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html) is a web application that can be used interactively (via HTML GUI) or programmatically (via URL-based API) to deploy and manage web applications.类似resin，tomcat提供了一个web交互界面来部署和管理web应用

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](http://tomcat.apache.org/maven-plugin.html) 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.部署web应用或者服务时一般打成war包

### Installation

There is no installation required for static deployment of web applications as this is provided out of the box by Tomcat（web应用的静态部署不需要安装）. Nor is any installation required for deployment functions with the Tomcat Manager（也不需要安装tomcat manager的不是功能）, although some configuration is required as detailed in the [Tomcat Manager manual](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html). An installation is however required if you wish to use the Tomcat Client Deployer (TCD)（如果使用TCD部署的话需要安装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-8.5.x-deployer*.

TCD has prerequisites of Apache Ant 1.6.2+ and a Java installation. Your environment should define an ANT\_HOME environment value（TCD的安装需要指定配置ant环境） 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](https://tomcat.apache.org/tomcat-8.5-doc/deployer-howto.html#Deploying_using_the_Client_Deployer_Package)

### 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在tomcat中，context被称作web应用.

In order to configure a Context within Tomcat a *Context Descriptor* is required. A Context Descriptor（context描述器，其实即使配置文件） 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（*Context*以前配置在server.xml中，但是现在不提倡这么做，尽管配置在server.xml中仍然能够起作用，其实就是将web.xml从serve.xml中分类出来）* 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:即web.xml

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（如果没有配置则使用默认值）.

### 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(即需要先停止tomcat，部署后再启动), 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使用压缩或者未压缩的war包都行.

The web applications present in the location specified by the Host's (default Host is "localhost") appBase attribute (default appBase is "$CATALINA\_BASE/webapps"server.xml中可以配置具体的路径) will be deployed on Tomcat startup only if the Host's deployOnStartup attribute is "true"（只有在host的deployOnStartup值设置为true才会在tomcat启动时部署）.

The following deployment sequence（部署流程） will occur on Tomcat startup in that case:

1. Any Context Descriptors will be deployed first.先加载部署context的配置
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部署.war包

Context配置-🡪.war包

### 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.热部署需要autoDeploy属性设置为true;当部署文件改变时会自动加载部署，此host需要后台进程，默认会提供

autoDeploy set to "true" and a running Tomcat allows for:一个正在运行的tomcat允许如下操作

* 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 unpackWARsattribute 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.web.xml改变
* 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](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html).

### 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这些特性需要tomcat manager的功能提供，并且运行tomcat 服务器.

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.

### Comments

**Notice:**This comments section collects your suggestions on improving documentation for Apache Tomcat.  
  
If you have trouble and need help, read [Find Help](http://tomcat.apache.org/findhelp.html) page and ask your question on the tomcat-users [mailing list](http://tomcat.apache.org/lists.html). Do not ask such questions here. This is not a Q&A section.  
  
The Apache Comments System is explained [here](https://tomcat.apache.org/tomcat-8.5-doc/comments.html). Comments may be removed by our moderators if they are either implemented or considered invalid/off-topic.

#### [RSS](https://comments.apache.org/rss.lua?site=tomcat&page=http://tomcat.apache.org/tomcat-8.5-doc/deployer-howto.html)   [Log in / register](https://comments.apache.org/portal.lua?site=tomcat)

No comments have been made so far.

## Manager App HOW-TO

### Table of Contents

* [Introduction](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Introduction)
* [Configuring Manager Application Access](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Configuring_Manager_Application_Access)
* [HTML User-friendly Interface](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#HTML_User-friendly_Interface)
* [Supported Manager Commands](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Supported_Manager_Commands)
  1. [Common Parameters](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Common_Parameters)
  2. [Deploy A New Application Archive (WAR) Remotely](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Deploy_A_New_Application_Archive_(WAR)_Remotely)
  3. [Deploy A New Application from a Local Path](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Deploy_A_New_Application_from_a_Local_Path)
     1. [Deploy a previously deployed webapp](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Deploy_a_previously_deployed_webapp)
     2. [Deploy a Directory or WAR by URL](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Deploy_a_Directory_or_WAR_by_URL)
     3. [Deploy a Directory or War from the Host appBase](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Deploy_a_Directory_or_War_from_the_Host_appBase)
     4. [Deploy using a Context configuration ".xml" file](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Deploy_using_a_Context_configuration_%22.xml%22_file)
     5. [Deployment Notes](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Deployment_Notes)
     6. [Deploy Response](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Deploy_Response)
  4. [List Currently Deployed Applications](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#List_Currently_Deployed_Applications)
  5. [Reload An Existing Application](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Reload_An_Existing_Application)
  6. [List OS and JVM Properties](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#List_OS_and_JVM_Properties)
  7. [List Available Global JNDI Resources](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#List_Available_Global_JNDI_Resources)
  8. [Session Statistics](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Session_Statistics)
  9. [Expire Sessions](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Expire_Sessions)
  10. [Start an Existing Application](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Start_an_Existing_Application)
  11. [Stop an Existing Application](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Stop_an_Existing_Application)
  12. [Undeploy an Existing Application](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Undeploy_an_Existing_Application)
  13. [Finding memory leaks](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Finding_memory_leaks)
  14. [Connector SSL/TLS diagnostics](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Connector_SSL/TLS_diagnostics)
  15. [Thread Dump](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Thread_Dump)
  16. [VM Info](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#VM_Info)
  17. [Save Configuration](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Save_Configuration)
* [Server Status](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Server_Status)
* [Using the JMX Proxy Servlet](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Using_the_JMX_Proxy_Servlet)
  1. [What is JMX Proxy Servlet](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#What_is_JMX_Proxy_Servlet)
  2. [JMX Query command](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#JMX_Query_command)
  3. [JMX Get command](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#JMX_Get_command)
  4. [JMX Set command](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#JMX_Set_command)
  5. [JMX Invoke command](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#JMX_Invoke_command)
* [Executing Manager Commands With Ant](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Executing_Manager_Commands_With_Ant)
  1. [Tasks output capture](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Tasks_output_capture)

### Introduction

In many production environments, it is very useful to have the capability to deploy a new web application, or undeploy an existing one, without having to shut down and restart the entire container. In addition, you can request an existing application to reload itself, even if you have not declared it to be reloadable in the Tomcat server configuration file.

To support these capabilities, Tomcat includes a web application (installed by default on context path /manager) that supports the following functions:

* Deploy a new web application from the uploaded contents of a WAR file.
* Deploy a new web application, on a specified context path, from the server file system.
* List the currently deployed web applications, as well as the sessions that are currently active for those web apps.
* Reload an existing web application, to reflect changes in the contents of /WEB-INF/classes or /WEB-INF/lib.
* List the OS and JVM property values.
* List the available global JNDI resources, for use in deployment tools that are preparing <ResourceLink> elements nested in a <Context> deployment description.
* Start a stopped application (thus making it available again).
* Stop an existing application (so that it becomes unavailable), but do not undeploy it.
* Undeploy a deployed web application and delete its document base directory (unless it was deployed from file system).

A default Tomcat installation includes the Manager. To add an instance of the Manager web application Context to a new host install the manager.xml context configuration file in the $CATALINA\_BASE/conf/[enginename]/[hostname] folder. Here is an example:

<Context privileged="true" antiResourceLocking="false"

docBase="${catalina.home}/webapps/manager">

<Valve className="org.apache.catalina.valves.RemoteAddrValve"

allow="127\.0\.0\.1" />

</Context>

If you have Tomcat configured to support multiple virtual hosts (websites) you would need to configure a Manager for each.

There are three ways to use the **Manager** web application.

* As an application with a user interface you use in your browser. Here is an example URL where you can replace localhost with your website host name:http://localhost:8080/manager/html .
* A minimal version using HTTP requests only which is suitable for use by scripts setup by system administrators. Commands are given as part of the request URI, and responses are in the form of simple text that can be easily parsed and processed. See [Supported Manager Commands](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Supported_Manager_Commands) for more information.
* A convenient set of task definitions for the Ant (version 1.4 or later) build tool. See [Executing Manager Commands With Ant](https://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html#Executing_Manager_Commands_With_Ant) for more information.

### Configuring Manager Application Access

The description below uses the variable name $CATALINA\_BASE to refer the base directory against which most relative paths are resolved. If you have not configured Tomcat for multiple instances by setting a CATALINA\_BASE directory, then $CATALINA\_BASE will be set to the value of $CATALINA\_HOME, the directory into which you have installed Tomcat.

It would be quite unsafe to ship Tomcat with default settings that allowed anyone on the Internet to execute the Manager application on your server. Therefore, the Manager application is shipped with the requirement that anyone who attempts to use it must authenticate themselves, using a username and password that have one of **manager-xxx** roles associated with them (the role name depends on what functionality is required). Further, there is no username in the default users file ($CATALINA\_BASE/conf/tomcat-users.xml) that is assigned to those roles. Therefore, access to the Manager application is completely disabled by default.

You can find the role names in the web.xml file of the Manager web application. The available roles are:

* **manager-gui** — Access to the HTML interface.
* **manager-status** — Access to the "Server Status" page only.
* **manager-script** — Access to the tools-friendly plain text interface that is described in this document, and to the "Server Status" page.
* **manager-jmx** — Access to JMX proxy interface and to the "Server Status" page.

The HTML interface is protected against CSRF (Cross-Site Request Forgery) attacks, but the text and JMX interfaces cannot be protected. It means that users who are allowed access to the text and JMX interfaces have to be cautious when accessing the Manager application with a web browser. To maintain the CSRF protection:

* If you use web browser to access the Manager application using a user that has either **manager-script** or **manager-jmx** roles (for example for testing the plain text or JMX interfaces), you MUST close all windows of the browser afterwards to terminate the session. If you do not close the browser and visit other sites, you may become victim of a CSRF attack.
* It is recommended to never grant the **manager-script** or **manager-jmx** roles to users that have the **manager-gui** role.

**Note** that JMX proxy interface is effectively low-level root-like administrative interface of Tomcat. One can do a lot, if he knows what commands to call. You should be cautious when enabling the **manager-jmx** role.

To enable access to the Manager web application, you must either create a new username/password combination and associate one of the **manager-xxx** roles with it, or add a **manager-xxx** role to some existing username/password combination. As the majority of this document describes the using the text interface, this example will use the role name **manager-script**. Exactly how the usernames/passwords are configured depends on which [Realm implementation](https://tomcat.apache.org/tomcat-8.5-doc/config/realm.html) you are using:

* UserDatabaseRealm plus MemoryUserDatabase, or MemoryRealm — The UserDatabaseRealm and MemoryUserDatabase are configured in the default $CATALINA\_BASE/conf/server.xml. Both MemoryUserDatabase and MemoryRealm read an XML-format file by default stored at $CATALINA\_BASE/conf/tomcat-users.xml, which can be edited with any text editor. This file contains an XML <user> for each individual user, which might look something like this:

<user username="craigmcc" password="secret" roles="standard,manager-script" />

which defines the username and password used by this individual to log on, and the role names he or she is associated with. You can add the **manager-script** role to the comma-delimited roles attribute for one or more existing users, and/or create new users with that assigned role.

* DataSourceRealm or JDBCRealm — Your user and role information is stored in a database accessed via JDBC. Add the **manager-script** role to one or more existing users, and/or create one or more new users with this role assigned, following the standard procedures for your environment.
* JNDIRealm — Your user and role information is stored in a directory server accessed via LDAP. Add the **manager-script** role to one or more existing users, and/or create one or more new users with this role assigned, following the standard procedures for your environment.

The first time you attempt to issue one of the Manager commands described in the next section, you will be challenged to log on using BASIC authentication. The username and password you enter do not matter, as long as they identify a valid user in the users database who possesses the role **manager-script**.

In addition to the password restrictions, access to the Manager web application can be restricted by the **remote IP address** or host by adding a RemoteAddrValve or RemoteHostValve. See [valves documentation](https://tomcat.apache.org/tomcat-8.5-doc/config/valve.html#Remote_Address_Filter) for details. Here is an example of restricting access to the localhost by IP address:

<Context privileged="true">

<Valve className="org.apache.catalina.valves.RemoteAddrValve"

allow="127\.0\.0\.1"/>

</Context>

### HTML User-friendly Interface

The user-friendly HTML interface of Manager web application is located at

http://{host}:{port}/manager/html

As has already been mentioned above, you need **manager-gui** role to be allowed to access it. There is a separate document that provides help on this interface. See:

* [HTML Manager documentation](https://tomcat.apache.org/tomcat-8.5-doc/html-manager-howto.html)

The HTML interface is protected against CSRF (Cross-Site Request Forgery) attacks. Each access to the HTML pages generates a random token, which is stored in your session and is included in all links on the page. If your next action does not have correct value of the token, the action will be denied. If the token has expired you can start again from the main page or List Applications page of Manager.

### Supported Manager Commands

All commands that the Manager application knows how to process are specified in a single request URI like this:

http://{host}:{port}/manager/text/{command}?{parameters}

where {host} and {port} represent the hostname and port number on which Tomcat is running, {command} represents the Manager command you wish to execute, and{parameters} represents the query parameters that are specific to that command. In the illustrations below, customize the host and port appropriately for your installation.

The commands are usually executed by HTTP GET requests. The /deploy command has a form that is executed by an HTTP PUT request.

#### Common Parameters

Most commands accept one or more of the following query parameters:

* **path** - The context path (including the leading slash) of the web application you are dealing with. To select the ROOT web application, specify "/". **NOTE**: It is not possible to perform administrative commands on the Manager application itself.
* **version** - The version of this web application as used by the [parallel deployment](https://tomcat.apache.org/tomcat-8.5-doc/config/context.html) feature. If you use parallel deployment wherever a path is required you must specify a version in addition to the path and it is the combination of path and version that must be unique rather than just the path.
* **war** - URL of a web application archive (WAR) file, or pathname of a directory which contains the web application, or a Context configuration ".xml" file. You can use URLs in any of the following formats:
  + **file:/absolute/path/to/a/directory** - The absolute path of a directory that contains the unpacked version of a web application. This directory will be attached to the context path you specify without any changes.
  + **file:/absolute/path/to/a/webapp.war** - The absolute path of a web application archive (WAR) file. This is valid **only** for the /deploy command, and is the only acceptable format to that command.
  + **file:/absolute/path/to/a/context.xml** - The absolute path of a web application Context configuration ".xml" file which contains the Context configuration element.
  + **directory** - The directory name for the web application context in the Host's application base directory.
  + **webapp.war** - The name of a web application war file located in the Host's application base directory.

Each command will return a response in text/plain format (i.e. plain ASCII with no HTML markup), making it easy for both humans and programs to read). The first line of the response will begin with either OK or FAIL, indicating whether the requested command was successful or not. In the case of failure, the rest of the first line will contain a description of the problem that was encountered. Some commands include additional lines of information as described below.

Internationalization Note - The Manager application looks up its message strings in resource bundles, so it is possible that the strings have been translated for your platform. The examples below show the English version of the messages.

#### Deploy A New Application Archive (WAR) Remotely

http://localhost:8080/manager/text/deploy?path=/foo

Upload the web application archive (WAR) file that is specified as the request data in this HTTP PUT request, install it into the appBase directory of our corresponding virtual host, and start, deriving the name for the WAR file added to the appBase from the specified path. The application can later be undeployed (and the corresponding WAR file removed) by use of the /undeploy command.

This command is executed by an HTTP PUT request.

The .WAR file may include Tomcat specific deployment configuration, by including a Context configuration XML file in /META-INF/context.xml.

URL parameters include:

* update: When set to true, any existing update will be undeployed first. The default value is set to false.
* tag: Specifying a tag name, this allows associating the deployed webapp with a tag or label. If the web application is undeployed, it can be later redeployed when needed using only the tag.

**NOTE** - This command is the logical opposite of the /undeploy command.

If installation and startup is successful, you will receive a response like this:

OK - Deployed application at context path /foo

Otherwise, the response will start with FAIL and include an error message. Possible causes for problems include:

* Application already exists at path /foo

The context paths for all currently running web applications must be unique. Therefore, you must undeploy the existing web application using this context path, or choose a different context path for the new one. The update parameter may be specified as a parameter on the URL, with a value of true to avoid this error. In that case, an undeploy will be performed on an existing application before performing the deployment.

* Encountered exception

An exception was encountered trying to start the new web application. Check the Tomcat logs for the details, but likely explanations include problems parsing your /WEB-INF/web.xml file, or missing classes encountered when initializing application event listeners and filters.

#### Deploy A New Application from a Local Path

Deploy and start a new web application, attached to the specified context path (which must not be in use by any other web application). This command is the logical opposite of the /undeploy command.

This command is executed by an HTTP GET request. There are a number of different ways the deploy command can be used.

#### Deploy a previously deployed webapp

http://localhost:8080/manager/text/deploy?path=/footoo&tag=footag

This can be used to deploy a previously deployed web application, which has been deployed using the tag attribute. Note that the work directory of the Manager webapp will contain the previously deployed WARs; removing it would make the deployment fail.

#### Deploy a Directory or WAR by URL

Deploy a web application directory or ".war" file located on the Tomcat server. If no path is specified, the path and version are derived from the directory name or the war file name. The war parameter specifies a URL (including the file: scheme) for either a directory or a web application archive (WAR) file. The supported syntax for a URL referring to a WAR file is described on the Javadocs page for the java.net.JarURLConnection class. Use only URLs that refer to the entire WAR file.

In this example the web application located in the directory /path/to/foo on the Tomcat server is deployed as the web application context named /footoo.

http://localhost:8080/manager/text/deploy?path=/footoo&war=file:/path/to/foo

In this example the ".war" file /path/to/bar.war on the Tomcat server is deployed as the web application context named /bar. Notice that there is no pathparameter so the context path defaults to the name of the web application archive file without the ".war" extension.

http://localhost:8080/manager/text/deploy?war=file:/path/to/bar.war

#### Deploy a Directory or War from the Host appBase

Deploy a web application directory or ".war" file located in your Host appBase directory. The path and optional version are derived from the directory or war file name.

In this example the web application located in a sub directory named foo in the Host appBase directory of the Tomcat server is deployed as the web application context named /foo. Notice that the context path used is the name of the web application directory.

http://localhost:8080/manager/text/deploy?war=foo

In this example the ".war" file bar.war located in your Host appBase directory on the Tomcat server is deployed as the web application context named /bar.

http://localhost:8080/manager/text/deploy?war=bar.war

#### Deploy using a Context configuration ".xml" file

If the Host deployXML flag is set to true you can deploy a web application using a Context configuration ".xml" file and an optional ".war" file or web application directory. The context path is not used when deploying a web application using a context ".xml" configuration file.

A Context configuration ".xml" file can contain valid XML for a web application Context just as if it were configured in your Tomcat server.xml configuration file. Here is an example:

<Context path="/foobar" docBase="/path/to/application/foobar">

</Context>

When the optional war parameter is set to the URL for a web application ".war" file or directory it overrides any docBase configured in the context configuration ".xml" file.

Here is an example of deploying an application using a Context configuration ".xml" file.

http://localhost:8080/manager/text/deploy?config=file:/path/context.xml

Here is an example of deploying an application using a Context configuration ".xml" file and a web application ".war" file located on the server.

http://localhost:8080/manager/text/deploy

?config=file:/path/context.xml&war=file:/path/bar.war

#### Deployment Notes

If the Host is configured with unpackWARs=true and you deploy a war file, the war will be unpacked into a directory in your Host appBase directory.

If the application war or directory is installed in your Host appBase directory and either the Host is configured with autoDeploy=true or the Context path must match the directory name or war file name without the ".war" extension.

For security when untrusted users can manage web applications, the Host deployXML flag can be set to false. This prevents untrusted users from deploying web applications using a configuration XML file and also prevents them from deploying application directories or ".war" files located outside of their Host appBase.

#### Deploy Response

If installation and startup is successful, you will receive a response like this:

OK - Deployed application at context path /foo

Otherwise, the response will start with FAIL and include an error message. Possible causes for problems include:

* Application already exists at path /foo

The context paths for all currently running web applications must be unique. Therefore, you must undeploy the existing web application using this context path, or choose a different context path for the new one. The update parameter may be specified as a parameter on the URL, with a value of true to avoid this error. In that case, an undeploy will be performed on an existing application before performing the deployment.

* Document base does not exist or is not a readable directory

The URL specified by the war parameter must identify a directory on this server that contains the "unpacked" version of a web application, or the absolute URL of a web application archive (WAR) file that contains this application. Correct the value specified by the war parameter.

* Encountered exception

An exception was encountered trying to start the new web application. Check the Tomcat logs for the details, but likely explanations include problems parsing your /WEB-INF/web.xml file, or missing classes encountered when initializing application event listeners and filters.

* Invalid application URL was specified

The URL for the directory or web application that you specified was not valid. Such URLs must start with file:, and URLs for a WAR file must end in ".war".

* Invalid context path was specified

The context path must start with a slash character. To reference the ROOT web application use "/".

* Context path must match the directory or WAR file name:

If the application war or directory is installed in your Host appBase directory and either the Host is configured with autoDeploy=true the Context path must match the directory name or war file name without the ".war" extension.

* Only web applications in the Host web application directory can be installed

If the Host deployXML flag is set to false this error will happen if an attempt is made to deploy a web application directory or ".war" file outside of the Host appBase directory.

#### List Currently Deployed Applications

http://localhost:8080/manager/text/list

List the context paths, current status (running or stopped), and number of active sessions for all currently deployed web applications. A typical response immediately after starting Tomcat might look like this:

OK - Listed applications for virtual host localhost

/webdav:running:0:webdav

/examples:running:0:examples

/manager:running:0:manager

/:running:0:ROOT

/test:running:0:test##2

/test:running:0:test##1

#### Reload An Existing Application

http://localhost:8080/manager/text/reload?path=/examples

Signal an existing application to shut itself down and reload. This can be useful when the web application context is not reloadable and you have updated classes or property files in the /WEB-INF/classes directory or when you have added or updated jar files in the /WEB-INF/lib directory.

If this command succeeds, you will see a response like this:

OK - Reloaded application at context path /examples

Otherwise, the response will start with FAIL and include an error message. Possible causes for problems include:

* Encountered exception

An exception was encountered trying to restart the web application. Check the Tomcat logs for the details.

* Invalid context path was specified

The context path must start with a slash character. To reference the ROOT web application use "/".

* No context exists for path /foo

There is no deployed application on the context path that you specified.

* No context path was specified

The path parameter is required.

* Reload not supported on WAR deployed at path /foo

Currently, application reloading (to pick up changes to the classes or web.xml file) is not supported when a web application is deployed directly from a WAR file. It only works when the web application is deployed from an unpacked directory. If you are using a WAR file, you should undeploy and then deploy or deploy with the update parameter the application again to pick up your changes.

#### List OS and JVM Properties

http://localhost:8080/manager/text/serverinfo

Lists information about the Tomcat version, OS, and JVM properties.

If an error occurs, the response will start with FAIL and include an error message. Possible causes for problems include:

* Encountered exception

An exception was encountered trying to enumerate the system properties. Check the Tomcat logs for the details.

#### List Available Global JNDI Resources

http://localhost:8080/manager/text/resources[?type=xxxxx]

List the global JNDI resources that are available for use in resource links for context configuration files. If you specify the type request parameter, the value must be the fully qualified Java class name of the resource type you are interested in (for example, you would specify javax.sql.DataSource to acquire the names of all available JDBC data sources). If you do not specify the type request parameter, resources of all types will be returned.

Depending on whether the type request parameter is specified or not, the first line of a normal response will be:

OK - Listed global resources of all types

or

OK - Listed global resources of type xxxxx

followed by one line for each resource. Each line is composed of fields delimited by colon characters (":"), as follows:

* Global Resource Name - The name of this global JNDI resource, which would be used in the global attribute of a <ResourceLink> element.
* Global Resource Type - The fully qualified Java class name of this global JNDI resource.

If an error occurs, the response will start with FAIL and include an error message. Possible causes for problems include:

* Encountered exception

An exception was encountered trying to enumerate the global JNDI resources. Check the Tomcat logs for the details.

* No global JNDI resources are available

The Tomcat server you are running has been configured without global JNDI resources.

#### Session Statistics

http://localhost:8080/manager/text/sessions?path=/examples

Display the default session timeout for a web application, and the number of currently active sessions that fall within one-minute ranges of their actual timeout times. For example, after restarting Tomcat and then executing one of the JSP samples in the /examples web app, you might get something like this:

OK - Session information for application at context path /examples

Default maximum session inactive interval 30 minutes

<1 minutes: 1 sessions

1 - <2 minutes: 1 sessions

#### Expire Sessions

http://localhost:8080/manager/text/expire?path=/examples&idle=num

Display the session statistics (like the above /sessions command) and expire sessions that are idle for longer than num minutes. To expire all sessions, use &idle=0 .

OK - Session information for application at context path /examples

Default maximum session inactive interval 30 minutes

1 - <2 minutes: 1 sessions

3 - <4 minutes: 1 sessions

>0 minutes: 2 sessions were expired

Actually /sessions and /expire are synonyms for the same command. The difference is in the presence of idle parameter.

#### Start an Existing Application

http://localhost:8080/manager/text/start?path=/examples

Signal a stopped application to restart, and make itself available again. Stopping and starting is useful, for example, if the database required by your application becomes temporarily unavailable. It is usually better to stop the web application that relies on this database rather than letting users continuously encounter database exceptions.

If this command succeeds, you will see a response like this:

OK - Started application at context path /examples

Otherwise, the response will start with FAIL and include an error message. Possible causes for problems include:

* Encountered exception

An exception was encountered trying to start the web application. Check the Tomcat logs for the details.

* Invalid context path was specified

The context path must start with a slash character. To reference the ROOT web application use "/".

* No context exists for path /foo

There is no deployed application on the context path that you specified.

* No context path was specified

The path parameter is required.

#### Stop an Existing Application

http://localhost:8080/manager/text/stop?path=/examples

Signal an existing application to make itself unavailable, but leave it deployed. Any request that comes in while an application is stopped will see an HTTP error 404, and this application will show as "stopped" on a list applications command.

If this command succeeds, you will see a response like this:

OK - Stopped application at context path /examples

Otherwise, the response will start with FAIL and include an error message. Possible causes for problems include:

* Encountered exception

An exception was encountered trying to stop the web application. Check the Tomcat logs for the details.

* Invalid context path was specified

The context path must start with a slash character. To reference the ROOT web application use "/".

* No context exists for path /foo

There is no deployed application on the context path that you specified.

* No context path was specified The path parameter is required.

#### Undeploy an Existing Application

http://localhost:8080/manager/text/undeploy?path=/examples

**WARNING - This command will delete any web application artifacts that exist within appBase directory (typically "webapps") for this virtual host**. This will delete the application .WAR, if present, the application directory resulting either from a deploy in unpacked form or from .WAR expansion as well as the XML Context definition from $CATALINA\_BASE/conf/[enginename]/[hostname]/ directory. If you simply want to take an application out of service, you should use the /stopcommand instead.

Signal an existing application to gracefully shut itself down, and remove it from Tomcat (which also makes this context path available for reuse later). In addition, the document root directory is removed, if it exists in the appBase directory (typically "webapps") for this virtual host. This command is the logical opposite of the /deploycommand.

If this command succeeds, you will see a response like this:

OK - Undeployed application at context path /examples

Otherwise, the response will start with FAIL and include an error message. Possible causes for problems include:

* Encountered exception

An exception was encountered trying to undeploy the web application. Check the Tomcat logs for the details.

* Invalid context path was specified

The context path must start with a slash character. To reference the ROOT web application use "/".

* No context exists named /foo

There is no deployed application with the name that you specified.

* No context path was specified The path parameter is required.

#### Finding memory leaks

http://localhost:8080/manager/text/findleaks[?statusLine=[true|false]]

**The find leaks diagnostic triggers a full garbage collection. It should be used with extreme caution on production systems.**

The find leaks diagnostic attempts to identify web applications that have caused memory leaks when they were stopped, reloaded or undeployed. Results should always be confirmed with a profiler. The diagnostic uses additional functionality provided by the StandardHost implementation. It will not work if a custom host is used that does not extend StandardHost.

Explicitly triggering a full garbage collection from Java code is documented to be unreliable. Furthermore, depending on the JVM used, there are options to disable explicit GC triggering, like -XX:+DisableExplicitGC. If you want to make sure, that the diagnostics were successfully running a full GC, you will need to check using tools like GC logging, JConsole or similar.

If this command succeeds, you will see a response like this:

/leaking-webapp

If you wish to see a status line included in the response then include the statusLine query parameter in the request with a value of true.

Each context path for a web application that was stopped, reloaded or undeployed, but which classes from the previous runs are still loaded in memory, thus causing a memory leak, will be listed on a new line. If an application has been reloaded several times, it may be listed several times.

If the command does not succeed, the response will start with FAIL and include an error message.

#### Connector SSL/TLS diagnostics

http://localhost:8080/manager/text/sslConnectorCiphers

The SSL Connector/Ciphers diagnostic lists the SSL/TLS ciphers that are currently configured for each connector. For NIO and NIO2, the names of the individual cipher suites are listed. For APR, the value of SSLCipherSuite is returned.

The response will look something like this:

OK - Connector / SSL Cipher information

Connector[HTTP/1.1-8080]

SSL is not enabled for this connector

Connector[HTTP/1.1-8443]

TLS\_ECDH\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA

TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA

TLS\_ECDH\_RSA\_WITH\_AES\_128\_CBC\_SHA

TLS\_ECDH\_ECDSA\_WITH\_AES\_128\_CBC\_SHA

...

#### Thread Dump

http://localhost:8080/manager/text/threaddump

Write a JVM thread dump.

The response will look something like this:

OK - JVM thread dump

2014-12-08 07:24:40.080

Full thread dump Java HotSpot(TM) Client VM (25.25-b02 mixed mode):

"http-nio-8080-exec-2" Id=26 cpu=46800300 ns usr=46800300 ns blocked 0 for -1 ms waited 0 for -1 ms

java.lang.Thread.State: RUNNABLE

locks java.util.concurrent.ThreadPoolExecutor$Worker@1738ad4

at sun.management.ThreadImpl.dumpThreads0(Native Method)

at sun.management.ThreadImpl.dumpAllThreads(ThreadImpl.java:446)

at org.apache.tomcat.util.Diagnostics.getThreadDump(Diagnostics.java:440)

at org.apache.tomcat.util.Diagnostics.getThreadDump(Diagnostics.java:409)

at org.apache.catalina.manager.ManagerServlet.threadDump(ManagerServlet.java:557)

at org.apache.catalina.manager.ManagerServlet.doGet(ManagerServlet.java:371)

at javax.servlet.http.HttpServlet.service(HttpServlet.java:618)

at javax.servlet.http.HttpServlet.service(HttpServlet.java:725)

...

#### VM Info

http://localhost:8080/manager/text/vminfo

Write some diagnostic information about Java Virtual Machine.

The response will look something like this:

OK - VM info

2014-12-08 07:27:32.578

Runtime information:

vmName: Java HotSpot(TM) Client VM

vmVersion: 25.25-b02

vmVendor: Oracle Corporation

specName: Java Virtual Machine Specification

specVersion: 1.8

specVendor: Oracle Corporation

managementSpecVersion: 1.2

name: ...

startTime: 1418012458849

uptime: 393855

isBootClassPathSupported: true

OS information:

...

#### Save Configuration

http://localhost:8080/manager/text/save

If specified without any parameters, this command saves the current configuration of the server to server.xml. The existing file will be renamed as a backup if required.

If specified with a path parameter that matches the path of a deployed web application then the configuration for that web application will be saved to an appropriately named context.xml file in the xmlBase for the current Host.

To use the command a StoreConfig MBean must be present. Typically this is configured using the [StoreConfigLifecycleListener](https://tomcat.apache.org/tomcat-8.5-doc/config/listeners.html" \l "StoreConfig_Lifecycle_Listener_-_org.apache.catalina.storeconfig.StoreConfigLifecycleListener).

If the command does not succeed, the response will start with FAIL and include an error message.

### Server Status

From the following links you can view Status information about the server. Any one of **manager-xxx** roles allows access to this page.

http://localhost:8080/manager/status

http://localhost:8080/manager/status/all

Displays server status information in HTML format.

http://localhost:8080/manager/status?XML=true

http://localhost:8080/manager/status/all?XML=true

Displays server status information in XML format.

First, you have the server and JVM version number, JVM provider, OS name and number followed by the architecture type.

Second, there is information about the memory usage of the JVM.

Then, there is information about the Tomcat AJP and HTTP connectors. The same information is available for both of them :

* Threads information : Max threads, min and max spare threads, current thread count and current thread busy.
* Request information : Max processing time and processing time, request and error count, bytes received and sent.
* A table showing Stage, Time, Bytes Sent, Bytes Receive, Client, VHost and Request. All existing threads are listed in the table. Here is the list of the possible thread stages :
  + "Parse and Prepare Request" : The request headers are being parsed or the necessary preparation to read the request body (if a transfer encoding has been specified) is taking place.
  + "Service" : The thread is processing a request and generating the response. This stage follows the "Parse and Prepare Request" stage and precedes the "Finishing" stage. There is always at least one thread in this stage (the server-status page).
  + "Finishing" : The end of the request processing. Any remainder of the response still in the output buffers is sent to the client. This stage is followed by "Keep-Alive" if it is appropriate to keep the connection alive or "Ready" if "Keep-Alive" is not appropriate.
  + "Keep-Alive" : The thread keeps the connection open to the client in case the client sends another request. If another request is received, the next stage will be "Parse and Prepare Request". If no request is received before the keep alive times out, the connection will be closed and the next stage will be "Ready".
  + "Ready" : The thread is at rest and ready to be used.

If you are using /status/all command, additional information on each of deployed web applications will be available.

### Using the JMX Proxy Servlet

#### What is JMX Proxy Servlet

The JMX Proxy Servlet is a lightweight proxy to get and set the tomcat internals. (Or any class that has been exposed via an MBean) Its usage is not very user friendly but the UI is extremely helpful for integrating command line scripts for monitoring and changing the internals of tomcat. You can do two things with the proxy: get information and set information. For you to really understand the JMX Proxy Servlet, you should have a general understanding of JMX. If you don't know what JMX is, then prepare to be confused.

#### JMX Query command

This takes the form:

http://webserver/manager/jmxproxy/?qry=STUFF

Where STUFF is the JMX query you wish to perform. For example, here are some queries you might wish to run:

* qry=\*%3Atype%3DRequestProcessor%2C\* --> type=RequestProcessor which will locate all workers which can process requests and report their state.
* qry=\*%3Aj2eeType=Servlet%2c\* --> j2eeType=Servlet which return all loaded servlets.
* qry=Catalina%3Atype%3DEnvironment%2Cresourcetype%3DGlobal%2Cname%3DsimpleValue --> Catalina:type=Environment,resourcetype=Global,name=simpleValue which look for a specific MBean by the given name.

You'll need to experiment with this to really understand its capabilities If you provide no qry parameter, then all of the MBeans will be displayed. We really recommend looking at the tomcat source code and understand the JMX spec to get a better understanding of all the queries you may run.

#### JMX Get command

The JXMProxyServlet also supports a "get" command that you can use to fetch the value of a specific MBean's attribute. The general form of the get command is:

http://webserver/manager/jmxproxy/?get=BEANNAME&att=MYATTRIBUTE&key=MYKEY

You must provide the following parameters:

1. get: The full bean name
2. att: The attribute you wish to fetch
3. key: (optional) The key into a CompositeData MBean attribute

If all goes well, then it will say OK, otherwise an error message will be shown. For example, let's say we wish to fetch the current heap memory data:

http://webserver/manager/jmxproxy/?get=java.lang:type=Memory&att=HeapMemoryUsage

Or, if you only want the "used" key:

http://webserver/manager/jmxproxy/

?get=java.lang:type=Memory&att=HeapMemoryUsage&key=used

#### JMX Set command

Now that you can query an MBean, its time to muck with Tomcat's internals! The general form of the set command is :

http://webserver/manager/jmxproxy/?set=BEANNAME&att=MYATTRIBUTE&val=NEWVALUE

So you need to provide 3 request parameters:

1. set: The full bean name
2. att: The attribute you wish to alter
3. val: The new value

If all goes ok, then it will say OK, otherwise an error message will be shown. For example, lets say we wish to turn up debugging on the fly for the ErrorReportValve. The following will set debugging to 10.

http://localhost:8080/manager/jmxproxy/

?set=Catalina%3Atype%3DValve%2Cname%3DErrorReportValve%2Chost%3Dlocalhost

&att=debug&val=10

and my result is (YMMV):

Result: ok

Here is what I see if I pass in a bad value. Here is the URL I used, I try set debugging equal to 'cow':

http://localhost:8080/manager/jmxproxy/

?set=Catalina%3Atype%3DValve%2Cname%3DErrorReportValve%2Chost%3Dlocalhost

&att=debug&val=cow

When I try that, my result is

Error: java.lang.NumberFormatException: For input string: "cow"

#### JMX Invoke command

The invoke command enables methods to be called on MBeans. The general form of the command is:

http://webserver/manager/jmxproxy/

?invoke=BEANNAME&op=METHODNAME&ps=COMMASEPARATEDPARAMETERS

For example, to call the findConnectors() method of the **Service** use:

http://localhost:8080/manager/jmxproxy/

?invoke=Catalina%3Atype%3DService&op=findConnectors&ps=

### Executing Manager Commands With Ant

In addition to the ability to execute Manager commands via HTTP requests, as documented above, Tomcat includes a convenient set of Task definitions for the Ant (version 1.4 or later) build tool. In order to use these commands, you must perform the following setup operations:

* Download the binary distribution of Ant from [http://ant.apache.org](http://ant.apache.org/). You must use version **1.4** or later.
* Install the Ant distribution in a convenient directory (called ANT\_HOME in the remainder of these instructions).
* Add the $ANT\_HOME/bin directory to your PATH environment variable.
* Configure at least one username/password combination in your Tomcat user database that includes the manager-script role.

To use custom tasks within Ant, you must declare them first with an <import> element. Therefore, your build.xml file might look something like this:

<project name="My Application" default="compile" basedir=".">

<!-- Configure the directory into which the web application is built -->

<property name="build" value="${basedir}/build"/>

<!-- Configure the context path for this application -->

<property name="path" value="/myapp"/>

<!-- Configure properties to access the Manager application -->

<property name="url" value="http://localhost:8080/manager/text"/>

<property name="username" value="myusername"/>

<property name="password" value="mypassword"/>

<!-- Configure the path to the Tomcat installation -->

<property name="catalina.home" value="/usr/local/apache-tomcat"/>

<!-- Configure the custom Ant tasks for the Manager application -->

<import file="${catalina.home}/bin/catalina-tasks.xml"/>

<!-- Executable Targets -->

<target name="compile" description="Compile web application">

<!-- ... construct web application in ${build} subdirectory, and

generated a ${path}.war ... -->

</target>

<target name="deploy" description="Install web application"

depends="compile">

<deploy url="${url}" username="${username}" password="${password}"

path="${path}" war="file:${build}${path}.war"/>

</target>

<target name="reload" description="Reload web application"

depends="compile">

<reload url="${url}" username="${username}" password="${password}"

path="${path}"/>

</target>

<target name="undeploy" description="Remove web application">

<undeploy url="${url}" username="${username}" password="${password}"

path="${path}"/>

</target>

</project>

Note: The definition of the resources task via the import above will override the resources datatype added in Ant 1.7. If you wish to use the resources datatype you will need to use Ant's namespace support to modify catalina-tasks.xml to assign the Tomcat tasks to their own namespace.

Now, you can execute commands like ant deploy to deploy the application to a running instance of Tomcat, or ant reload to tell Tomcat to reload it. Note also that most of the interesting values in this build.xml file are defined as replaceable properties, so you can override their values from the command line. For example, you might consider it a security risk to include the real manager password in your build.xml file's source code. To avoid this, omit the password property, and specify it from the command line:

ant -Dpassword=secret deploy

#### Tasks output capture

Using Ant version **1.6.2** or later, the Catalina tasks offer the option to capture their output in properties or external files. They support directly the following subset of the<redirector> type attributes:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Required** |
| output | Name of a file to which to write the output. If the error stream is not also redirected to a file or property, it will appear in this output. | No |
| error | The file to which the standard error of the command should be redirected. | No |
| logError | This attribute is used when you wish to see error output in Ant's log and you are redirecting output to a file/property. The error output will not be included in the output file/property. If you redirect error with the *error* or *errorProperty* attributes, this will have no effect. | No |
| append | Whether output and error files should be appended to or overwritten. Defaults to false. | No |
| createemptyfiles | Whether output and error files should be created even when empty. Defaults to true. | No |
| outputproperty | The name of a property in which the output of the command should be stored. Unless the error stream is redirected to a separate file or stream, this property will include the error output. | No |
| errorproperty | The name of a property in which the standard error of the command should be stored. | No |

A couple of additional attributes can also be specified:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Required** |
| alwaysLog | This attribute is used when you wish to see the output you are capturing, appearing also in the Ant's log. It must not be used unless you are capturing task output. Defaults to false. This attribute will be supported directly by*<redirector>*in Ant 1.6.3 | No |
| failonerror | This attribute is used when you wish to avoid that any manager command processing error terminates the ant execution. Defaults to true. It must be set to false, if you want to capture error output, otherwise execution will terminate before anything can be captured.  This attribute acts only on manager command execution, any wrong or missing command attribute will still cause Ant execution termination. | No |

They also support the embedded <redirector> element in which you can specify its full set of attributes, but input, inputstring and inputencoding that, even if accepted, are not used because they have no meaning in this context. Refer to [ant manual](http://ant.apache.org/) for details on <redirector> element attributes.

Here is a sample build file extract that shows how this output redirection support can be used:

<target name="manager.deploy"

depends="context.status"

if="context.notInstalled">

<deploy url="${mgr.url}"

username="${mgr.username}"

password="${mgr.password}"

path="${mgr.context.path}"

config="${mgr.context.descriptor}"/>

</target>

<target name="manager.deploy.war"

depends="context.status"

if="context.deployable">

<deploy url="${mgr.url}"

username="${mgr.username}"

password="${mgr.password}"

update="${mgr.update}"

path="${mgr.context.path}"

war="${mgr.war.file}"/>

</target>

<target name="context.status">

<property name="running" value="${mgr.context.path}:running"/>

<property name="stopped" value="${mgr.context.path}:stopped"/>

<list url="${mgr.url}"

outputproperty="ctx.status"

username="${mgr.username}"

password="${mgr.password}">

</list>

<condition property="context.running">

<contains string="${ctx.status}" substring="${running}"/>

</condition>

<condition property="context.stopped">

<contains string="${ctx.status}" substring="${stopped}"/>

</condition>

<condition property="context.notInstalled">

<and>

<isfalse value="${context.running}"/>

<isfalse value="${context.stopped}"/>

</and>

</condition>

<condition property="context.deployable">

<or>

<istrue value="${context.notInstalled}"/>

<and>

<istrue value="${context.running}"/>

<istrue value="${mgr.update}"/>

</and>

<and>

<istrue value="${context.stopped}"/>

<istrue value="${mgr.update}"/>

</and>

</or>

</condition>

<condition property="context.undeployable">

<or>

<istrue value="${context.running}"/>

<istrue value="${context.stopped}"/>

</or>

</condition>

</target>

**WARNING:** even if it doesn't make many sense, and is always a bad idea, calling a Catalina task more than once, badly set Ant tasks depends chains may cause that a task be called more than once in the same Ant run, even if not intended to. A bit of caution should be exercised when you are capturing output from that task, because this could lead to something unexpected:

* when capturing in a property you will find in it only the output from the first call, because Ant properties are immutable and once set they cannot be changed,
* when capturing in a file, each run will overwrite it and you will find in it only the last call output, unless you are using the append="true" attribute, in which case you will see the output of each task call appended to the file.

### Comments

**Notice:**This comments section collects your suggestions on improving documentation for Apache Tomcat.  
  
If you have trouble and need help, read [Find Help](http://tomcat.apache.org/findhelp.html) page and ask your question on the tomcat-users [mailing list](http://tomcat.apache.org/lists.html). Do not ask such questions here. This is not a Q&A section.  
  
The Apache Comments System is explained [here](https://tomcat.apache.org/tomcat-8.5-doc/comments.html). Comments may be removed by our moderators if they are either implemented or considered invalid/off-topic.

#### [RSS](https://comments.apache.org/rss.lua?site=tomcat&page=http://tomcat.apache.org/tomcat-8.5-doc/manager-howto.html)   [Log in / register](https://comments.apache.org/portal.lua?site=tomcat)

No comments have been made so far.

## JNDI Datasource HOW-TO 数据处理

### Table of Contents

* [Introduction](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Introduction)
* [DriverManager, the service provider mechanism and memory leaks](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#DriverManager,_the_service_provider_mechanism_and_memory_leaks)
* [Database Connection Pool (DBCP 2) Configurations](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Database_Connection_Pool_(DBCP_2)_Configurations)
  1. [Installation](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Installation)
  2. [Preventing database connection pool leaks](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Preventing_database_connection_pool_leaks)
  3. [MySQL DBCP Example](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#MySQL_DBCP_Example)
  4. [Oracle 8i, 9i & 10g](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Oracle_8i,_9i_&_10g)
  5. [PostgreSQL](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#PostgreSQL)
* [Non-DBCP Solutions](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Non-DBCP_Solutions)
* [Oracle 8i with OCI client](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Oracle_8i_with_OCI_client)
  1. [Introduction](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Oracle_8i_with_OCI_client/Introduction)
  2. [Putting it all together](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Putting_it_all_together)
* [Common Problems](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Common_Problems)
  1. [Intermittent Database Connection Failures](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Intermittent_Database_Connection_Failures)
  2. [Random Connection Closed Exceptions](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Random_Connection_Closed_Exceptions)
  3. [Context versus GlobalNamingResources](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#Context_versus_GlobalNamingResources)
  4. [JNDI Resource Naming and Realm Interaction](https://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html#JNDI_Resource_Naming_and_Realm_Interaction)

### Introduction

JNDI Datasource configuration is covered extensively in the JNDI-Resources-HOWTO. However, feedback from tomcat-user has shown that specifics for individual configurations can be rather tricky（微妙的）.

Here then are some example configurations that have been posted to tomcat-user for popular databases and some general tips for db usage.

You should be aware that since these notes are derived from configuration and/or feedback posted to tomcat-user YMMV :-). Please let us know if you have any other tested configurations that you feel may be of use to the wider audience, or if you feel we can improve this section in anyway.

**Please note that JNDI resource configuration changed somewhat between Tomcat 7.x and Tomcat 8.x as they are using different versions of Apache Commons DBCP library（7.x和8.x的jndi 配置改变了）.** You will most likely need to modify older JNDI resource configurations to match the syntax in the example below in order to make them work in Tomcat 8. See [Tomcat Migration Guide](http://tomcat.apache.org/migration.html) for details.

Also, please note that JNDI DataSource configuration in general, and this tutorial in particular, assumes that you have read and understood the [Context](https://tomcat.apache.org/tomcat-8.5-doc/config/context.html) and [Host](https://tomcat.apache.org/tomcat-8.5-doc/config/host.html)configuration references, including the section about Automatic Application Deployment in the latter reference.

### DriverManager, the service provider mechanism and memory leaks 驱动管理，服务提供机制和内存泄漏

java.sql.DriverManager supports the [service provider](http://docs.oracle.com/javase/6/docs/api/index.html?java/sql/DriverManager.html) mechanism. This feature is that all the available JDBC drivers that announce themselves by providing a META-INF/services/java.sql.Driver file are automatically discovered, loaded and registered, relieving you from the need to load the database driver explicitly before you create a JDBC connection. However, the implementation is fundamentally broken in all Java versions for a servlet container environment. The problem is thatjava.sql.DriverManager will scan for the drivers only once.

The [JRE Memory Leak Prevention Listener](https://tomcat.apache.org/tomcat-8.5-doc/config/listeners.html) that is included with Apache Tomcat solves this by triggering the drivers scan during Tomcat startup. This is enabled by default. It means that only libraries visible to the listener such as the ones in $CATALINA\_BASE/lib will be scanned for database drivers. If you are considering disabling this feature, note that the scan would be triggered by the first web application that is using JDBC, leading to failures when this web application is reloaded and for other web applications that rely on this feature.

Thus, the web applications that have database drivers in their WEB-INF/lib directory cannot rely on the service provider mechanism and should register the drivers explicitly.

The list of drivers in java.sql.DriverManager is also a known source of memory leaks. Any Drivers registered by a web application must be deregistered when the web application stops. Tomcat will attempt to automatically discover and deregister any JDBC drivers loaded by the web application class loader when the web application stops. However, it is expected that applications do this for themselves via a ServletContextListener.

### Database Connection Pool (DBCP 2) Configurations

The default database connection pool implementation in Apache Tomcat relies on the libraries from the [Apache Commons](http://commons.apache.org/) project. The following libraries are used:

* Commons DBCP
* Commons Pool

These libraries are located in a single JAR at $CATALINA\_HOME/lib/tomcat-dbcp.jar. However, only the classes needed for connection pooling have been included, and the packages have been renamed to avoid interfering with applications.

DBCP 2.0 provides support for JDBC 4.1.

#### Installation

See the [DBCP documentation](http://commons.apache.org/dbcp/configuration.html) for a complete list of configuration parameters.

#### Preventing database connection pool leaks

A database connection pool creates and manages a pool of connections to a database. Recycling and reusing already existing connections to a database is more efficient than opening a new connection.

There is one problem with connection pooling. A web application has to explicitly close ResultSet's, Statement's, and Connection's. Failure of a web application to close these resources can result in them never being available again for reuse, a database connection pool "leak". This can eventually result in your web application database connections failing if there are no more available connections.

There is a solution to this problem. The Apache Commons DBCP can be configured to track and recover these abandoned database connections. Not only can it recover them, but also generate a stack trace for the code which opened these resources and never closed them.

To configure a DBCP DataSource so that abandoned database connections are removed and recycled, add one or both of the following attributes to the Resourceconfiguration for your DBCP DataSource:

removeAbandonedOnBorrow=true

removeAbandonedOnMaintenance=true

The default for both of these attributes is false. Note that removeAbandonedOnMaintenance has no effect unless pool maintenance is enabled by setting timeBetweenEvictionRunsMillis to a positive value. See the [DBCP documentation](http://commons.apache.org/dbcp/configuration.html) for full documentation on these attributes.

Use the removeAbandonedTimeout attribute to set the number of seconds a database connection has been idle before it is considered abandoned.

removeAbandonedTimeout="60"

The default timeout for removing abandoned connections is 300 seconds.

The logAbandoned attribute can be set to true if you want DBCP to log a stack trace of the code which abandoned the database connection resources.

logAbandoned="true"

The default is false.

#### MySQL DBCP Example

##### 0. Introduction

Versions of [MySQL](http://www.mysql.com/products/mysql/index.html) and JDBC drivers that have been reported to work:

* MySQL 3.23.47, MySQL 3.23.47 using InnoDB,, MySQL 3.23.58, MySQL 4.0.1alpha
* [Connector/J](http://www.mysql.com/products/connector-j) 3.0.11-stable (the official JDBC Driver)
* [mm.mysql](http://mmmysql.sourceforge.net/) 2.0.14 (an old 3rd party JDBC Driver)

Before you proceed, don't forget to copy the JDBC Driver's jar into $CATALINA\_HOME/lib.

##### 1. MySQL configuration

Ensure that you follow these instructions as variations can cause problems.

Create a new test user, a new database and a single test table. Your MySQL user **must** have a password assigned. The driver will fail if you try to connect with an empty password.

mysql> GRANT ALL PRIVILEGES ON \*.\* TO javauser@localhost

-> IDENTIFIED BY 'javadude' WITH GRANT OPTION;

mysql> create database javatest;

mysql> use javatest;

mysql> create table testdata (

-> id int not null auto\_increment primary key,

-> foo varchar(25),

-> bar int);

**Note:** the above user should be removed once testing is complete!

Next insert some test data into the testdata table.

mysql> insert into testdata values(null, 'hello', 12345);

Query OK, 1 row affected (0.00 sec)

mysql> select \* from testdata;

+----+-------+-------+

| ID | FOO | BAR |

+----+-------+-------+

| 1 | hello | 12345 |

+----+-------+-------+

1 row in set (0.00 sec)

mysql>

##### 2. Context configuration

Configure the JNDI DataSource in Tomcat by adding a declaration for your resource to your [Context](https://tomcat.apache.org/tomcat-8.5-doc/config/context.html).

For example:

<Context>

<!-- maxTotal: Maximum number of database connections in pool. Make sure you

configure your mysqld max\_connections large enough to handle

all of your db connections. Set to -1 for no limit.

-->

<!-- maxIdle: Maximum number of idle database connections to retain in pool.

Set to -1 for no limit. See also the DBCP documentation on this

and the minEvictableIdleTimeMillis configuration parameter.

-->

<!-- maxWaitMillis: Maximum time to wait for a database connection to become available

in ms, in this example 10 seconds. An Exception is thrown if

this timeout is exceeded. Set to -1 to wait indefinitely.

-->

<!-- username and password: MySQL username and password for database connections -->

<!-- driverClassName: Class name for the old mm.mysql JDBC driver is

org.gjt.mm.mysql.Driver - we recommend using Connector/J though.

Class name for the official MySQL Connector/J driver is com.mysql.jdbc.Driver.

-->

<!-- url: The JDBC connection url for connecting to your MySQL database.

-->

<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"

maxTotal="100" maxIdle="30" maxWaitMillis="10000"

username="javauser" password="javadude" driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/javatest"/>

</Context>

##### 3. web.xml configuration

Now create a WEB-INF/web.xml for this test application.

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee

http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"

version="2.4">

<description>MySQL Test App</description>

<resource-ref>

<description>DB Connection</description>

<res-ref-name>jdbc/TestDB</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

</web-app>

##### 4. Test code

Now create a simple test.jsp page for use later.

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<sql:query var="rs" dataSource="jdbc/TestDB">

select id, foo, bar from testdata

</sql:query>

<html>

<head>

<title>DB Test</title>

</head>

<body>

<h2>Results</h2>

<c:forEach var="row" items="${rs.rows}">

Foo ${row.foo}<br/>

Bar ${row.bar}<br/>

</c:forEach>

</body>

</html>

That JSP page makes use of [JSTL](http://www.oracle.com/technetwork/java/index-jsp-135995.html)'s SQL and Core taglibs. You can get it from [Apache Tomcat Taglibs - Standard Tag Library](http://tomcat.apache.org/taglibs/standard/) project — just make sure you get a 1.1.x or later release. Once you have JSTL, copy jstl.jar and standard.jar to your web app's WEB-INF/lib directory.

Finally deploy your web app into $CATALINA\_BASE/webapps either as a warfile called DBTest.war or into a sub-directory called DBTest

Once deployed, point a browser at http://localhost:8080/DBTest/test.jsp to view the fruits of your hard work.

#### Oracle 8i, 9i & 10g

##### 0. Introduction

Oracle requires minimal changes from the MySQL configuration except for the usual gotchas :-)

Drivers for older Oracle versions may be distributed as \*.zip files rather than \*.jar files. Tomcat will only use \*.jar files installed in $CATALINA\_HOME/lib. Therefore classes111.zip or classes12.zip will need to be renamed with a .jar extension. Since jarfiles are zipfiles, there is no need to unzip and jar these files - a simple rename will suffice.

For Oracle 9i onwards you should use oracle.jdbc.OracleDriver rather than oracle.jdbc.driver.OracleDriver as Oracle have stated that oracle.jdbc.driver.OracleDriver is deprecated and support for this driver class will be discontinued in the next major release.

##### 1. Context configuration

In a similar manner to the mysql config above, you will need to define your Datasource in your [Context](https://tomcat.apache.org/tomcat-8.5-doc/config/context.html). Here we define a Datasource called myoracle using the thin driver to connect as user scott, password tiger to the sid called mysid. (Note: with the thin driver this sid is not the same as the tnsname). The schema used will be the default schema for the user scott.

Use of the OCI driver should simply involve a changing thin to oci in the URL string.

<Resource name="jdbc/myoracle" auth="Container"

type="javax.sql.DataSource" driverClassName="oracle.jdbc.OracleDriver"

url="jdbc:oracle:thin:@127.0.0.1:1521:mysid"

username="scott" password="tiger" maxTotal="20" maxIdle="10"

maxWaitMillis="-1"/>

##### 2. web.xml configuration

You should ensure that you respect the element ordering defined by the DTD when you create you applications web.xml file.

<resource-ref>

<description>Oracle Datasource example</description>

<res-ref-name>jdbc/myoracle</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

##### 3. Code example

You can use the same example application as above (assuming you create the required DB instance, tables etc.) replacing the Datasource code with something like

Context initContext = new InitialContext();

Context envContext = (Context)initContext.lookup("java:/comp/env");

DataSource ds = (DataSource)envContext.lookup("jdbc/myoracle");

Connection conn = ds.getConnection();

//etc.

#### PostgreSQL

##### 0. Introduction

PostgreSQL is configured in a similar manner to Oracle.

##### 1. Required files

Copy the Postgres JDBC jar to $CATALINA\_HOME/lib. As with Oracle, the jars need to be in this directory in order for DBCP's Classloader to find them. This has to be done regardless of which configuration step you take next.

##### 2. Resource configuration

You have two choices here: define a datasource that is shared across all Tomcat applications, or define a datasource specifically for one application.

###### 2a. Shared resource configuration

Use this option if you wish to define a datasource that is shared across multiple Tomcat applications, or if you just prefer defining your datasource in this file.

*This author has not had success here, although others have reported so. Clarification would be appreciated here.*

<Resource name="jdbc/postgres" auth="Container"

type="javax.sql.DataSource" driverClassName="org.postgresql.Driver"

url="jdbc:postgresql://127.0.0.1:5432/mydb"

username="myuser" password="mypasswd" maxTotal="20" maxIdle="10" maxWaitMillis="-1"/>

###### 2b. Application-specific resource configuration

Use this option if you wish to define a datasource specific to your application, not visible to other Tomcat applications. This method is less invasive to your Tomcat installation.

Create a resource definition for your [Context](https://tomcat.apache.org/tomcat-8.5-doc/config/context.html). The Context element should look something like the following.

<Context>

<Resource name="jdbc/postgres" auth="Container"

type="javax.sql.DataSource" driverClassName="org.postgresql.Driver"

url="jdbc:postgresql://127.0.0.1:5432/mydb"

username="myuser" password="mypasswd" maxTotal="20" maxIdle="10"

maxWaitMillis="-1"/>

</Context>

##### 3. web.xml configuration

<resource-ref>

<description>postgreSQL Datasource example</description>

<res-ref-name>jdbc/postgres</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

##### 4. Accessing the datasource

When accessing the datasource programmatically, remember to prepend java:/comp/env to your JNDI lookup, as in the following snippet of code. Note also that "jdbc/postgres" can be replaced with any value you prefer, provided you change it in the above resource definition file as well.

InitialContext cxt = new InitialContext();

if ( cxt == null ) {

throw new Exception("Uh oh -- no context!");

}

DataSource ds = (DataSource) cxt.lookup( "java:/comp/env/jdbc/postgres" );

if ( ds == null ) {

throw new Exception("Data source not found!");

}

### Non-DBCP Solutions

These solutions either utilise a single connection to the database (not recommended for anything other than testing!) or some other pooling technology.

### Oracle 8i with OCI client

#### Introduction

Whilst not strictly addressing the creation of a JNDI DataSource using the OCI client, these notes can be combined with the Oracle and DBCP solution above.

In order to use OCI driver, you should have an Oracle client installed. You should have installed Oracle8i(8.1.7) client from cd, and download the suitable JDBC/OCI driver(Oracle8i 8.1.7.1 JDBC/OCI Driver) from [otn.oracle.com](http://otn.oracle.com/).

After renaming classes12.zip file to classes12.jar for Tomcat, copy it into $CATALINA\_HOME/lib. You may also have to remove the javax.sql.\* classes from this file depending upon the version of Tomcat and JDK you are using.

#### Putting it all together

Ensure that you have the ocijdbc8.dll or .so in your $PATH or LD\_LIBRARY\_PATH (possibly in $ORAHOME\bin) and also confirm that the native library can be loaded by a simple test program using System.loadLibrary("ocijdbc8");

You should next create a simple test servlet or jsp that has these **critical lines**:

DriverManager.registerDriver(new

oracle.jdbc.driver.OracleDriver());

conn =

DriverManager.getConnection("jdbc:oracle:oci8:@database","username","password");

where database is of the form host:port:SID Now if you try to access the URL of your test servlet/jsp and what you get is a ServletException with a root cause of java.lang.UnsatisfiedLinkError:get\_env\_handle.

First, the UnsatisfiedLinkError indicates that you have

* a mismatch between your JDBC classes file and your Oracle client version. The giveaway here is the message stating that a needed library file cannot be found. For example, you may be using a classes12.zip file from Oracle Version 8.1.6 with a Version 8.1.5 Oracle client. The classesXXX.zip file and Oracle client software versions must match.
* A $PATH, LD\_LIBRARY\_PATH problem.
* It has been reported that ignoring the driver you have downloaded from otn and using the classes12.zip file from the directory $ORAHOME\jdbc\lib will also work.

Next you may experience the error ORA-06401 NETCMN: invalid driver designator

The Oracle documentation says : "Cause: The login (connect) string contains an invalid driver designator. Action: Correct the string and re-submit." Change the database connect string (of the form host:port:SID) with this one: (description=(address=(host=myhost)(protocol=tcp)(port=1521))(connect\_data=(sid=orcl)))

*Ed. Hmm, I don't think this is really needed if you sort out your TNSNames - but I'm not an Oracle DBA :-)*

### Common Problems

Here are some common problems encountered with a web application which uses a database and tips for how to solve them.

#### Intermittent Database Connection Failures

Tomcat runs within a JVM. The JVM periodically performs garbage collection (GC) to remove java objects which are no longer being used. When the JVM performs GC execution of code within Tomcat freezes. If the maximum time configured for establishment of a database connection is less than the amount of time garbage collection took you can get a database connection failure.

To collect data on how long garbage collection is taking add the -verbose:gc argument to your CATALINA\_OPTS environment variable when starting Tomcat. When verbose gc is enabled your $CATALINA\_BASE/logs/catalina.out log file will include data for every garbage collection including how long it took.

When your JVM is tuned correctly 99% of the time a GC will take less than one second. The remainder will only take a few seconds. Rarely, if ever should a GC take more than 10 seconds.

Make sure that the db connection timeout is set to 10-15 seconds. For the DBCP you set this using the parameter maxWaitMillis.

#### Random Connection Closed Exceptions

These can occur when one request gets a db connection from the connection pool and closes it twice. When using a connection pool, closing the connection just returns it to the pool for reuse by another request, it doesn't close the connection. And Tomcat uses multiple threads to handle concurrent requests. Here is an example of the sequence of events which could cause this error in Tomcat:

Request 1 running in Thread 1 gets a db connection.

Request 1 closes the db connection.

The JVM switches the running thread to Thread 2

Request 2 running in Thread 2 gets a db connection

(the same db connection just closed by Request 1).

The JVM switches the running thread back to Thread 1

Request 1 closes the db connection a second time in a finally block.

The JVM switches the running thread back to Thread 2

Request 2 Thread 2 tries to use the db connection but fails

because Request 1 closed it.

Here is an example of properly written code to use a database connection obtained from a connection pool:

Connection conn = null;

Statement stmt = null; // Or PreparedStatement if needed

ResultSet rs = null;

try {

conn = ... get connection from connection pool ...

stmt = conn.createStatement("select ...");

rs = stmt.executeQuery();

... iterate through the result set ...

rs.close();

rs = null;

stmt.close();

stmt = null;

conn.close(); // Return to connection pool

conn = null; // Make sure we don't close it twice

} catch (SQLException e) {

... deal with errors ...

} finally {

// Always make sure result sets and statements are closed,

// and the connection is returned to the pool

if (rs != null) {

try { rs.close(); } catch (SQLException e) { ; }

rs = null;

}

if (stmt != null) {

try { stmt.close(); } catch (SQLException e) { ; }

stmt = null;

}

if (conn != null) {

try { conn.close(); } catch (SQLException e) { ; }

conn = null;

}

}

#### Context versus GlobalNamingResources

Please note that although the above instructions place the JNDI declarations in a Context element, it is possible and sometimes desirable to place these declarations in the [GlobalNamingResources](https://tomcat.apache.org/tomcat-8.5-doc/config/globalresources.html) section of the server configuration file. A resource placed in the GlobalNamingResources section will be shared among the Contexts of the server.

#### JNDI Resource Naming and Realm Interaction

In order to get Realms to work, the realm must refer to the datasource as defined in the <GlobalNamingResources> or <Context> section, not a datasource as renamed using <ResourceLink>.

### Comments

**Notice:**This comments section collects your suggestions on improving documentation for Apache Tomcat.  
  
If you have trouble and need help, read [Find Help](http://tomcat.apache.org/findhelp.html) page and ask your question on the tomcat-users [mailing list](http://tomcat.apache.org/lists.html). Do not ask such questions here. This is not a Q&A section.  
  
The Apache Comments System is explained [here](https://tomcat.apache.org/tomcat-8.5-doc/comments.html). Comments may be removed by our moderators if they are either implemented or considered invalid/off-topic.

#### [RSS](https://comments.apache.org/rss.lua?site=tomcat&page=http://tomcat.apache.org/tomcat-8.5-doc/jndi-datasource-examples-howto.html)   [Log in / register](https://comments.apache.org/portal.lua?site=tomcat)

**Peter J Nelson** *66 days ago* Rating: 0 (register an account in order to rate comments)

Some of the information here seems to conflict with information posted here - <http://tomcat.apache.org/tomcat-8.5-doc/jdbc-pool.html>