HowTo

Permalink to this page: https://cwiki.apache.org/confluence/x/gSslBg

- Meta
  - How do I add a question to this page?
  - How do I contribute to Tomcat's documentation?
- Installation
  - How do I set up and run Tomcat on Macintosh OS X?
  - How do I set up and run Tomcat on Solaris 10?
  - How do I set up and run Tomcat on OpenVMS?
  - How do I set up another tomcat service on Windows, sharing the same Tomcat Home ?
  - How do I install Tomcat as a service under Unix?
  - How to run Tomcat without root privileges?
  - How to create native launchers for Tomcat
  - How do I rotate catalina.out?
    - Rotate catalina.out using logrotate (or similar)
    - Rotate catalina.out using rotatelogs or chronolog (or similar)
- Configuration
  - How do I set up multiple sites sharing the same war application/war file?
  - How do I log requests ?
  - Can I set Java system properties differently for each webapp?
  - How do I configure Tomcat Connectors?
  - How do I configure Tomcat to work with IIS and NTLM?
  - Setting up SSL
  - HowTo SSL Client Authentication with Fallback to FORM Authentication
  - How do I restrict the list of SSL ciphers used for HTTPS
  - How do I make Tomcat startup faster?
  - How do I override the default home page loaded by Tomcat?
  - How do I enable Server Side Includes (SSI)?
  - How do I install the Administration web app?
    - Tomcat 5.5
    - Tomcat 6.0 and later
  - How do I add JARs or classes to the common classloader without adding them to $CATALINA_HOME/lib?
  - How do I authenticate Manager access via JNDI to Active Directory for multiple Tomcat instances?
  - Where and how do I set memory related settings to improve Tomcat performance?
  - How do I make my web application be the Tomcat default application?
  - How do I set up Tomcat virtual hosts in a development environment?
  - How do I get started with a simple cluster?
- Programming
  - How do call tomcat ant tasks to deploy webapps?
  - How do I load a properties file?
  - How do I share sessions across web apps?
  - How can I access members of a custom Realm or Principal?
  - How do I get direct access to a Tomcat Realm?
  - How do I redirect System.out and System.err to my web page?
  - How do I connect to a Websphere MQ (MQ Series) server using JMS and JNDI?
  - How do I use DataSources with Tomcat?
  - How do I use Hibernate and database connection pooling with Tomcat?
  - How do I use DataSourceRealms for authentication and authorization?
- Troubleshooting
  - Tomcat crashed! What do I do now?
  - I'm encountering classloader problems when using JNI under Tomcat
  - How do I debug a Tomcat application?
  - How do I debug a Tomcat application when Tomcat is run as a Windows service ?
  - How do I check whether Tomcat is UP or DOWN? There is no status command
  - How do I obtain a thread dump of my running webapp ?
    - If you are running Oracle (Sun) JDK
    - If you are running on *NIX
    - If you are running on Microsoft Windows
    - If you are running Tomcat as a service on Microsoft Windows
    - If you have Tomcat running in a console
    - Using Java code
  - How do I read a Java thread dump ?
  - How do I obtain a heap dump?
  - How do I add my own custom MBean to monitor my application within Tomcat 6?

Meta

How do I add a question to this page?

Anyone may edit this page to add their own content. That is why this page is part of a Wiki and not a hardcoded static file in the FAQ.

However, *do not* add questions without answers to this page. If you have a question about how to do something in Tomcat which has not been addressed yet, ask the tomcat-user list. Once you've figured out how to fix your problem, come back and update the Wiki to allow the rest of us to benefit from what you've learned!

How do I contribute to Tomcat's documentation?

Download the source bundle or grab the source files from Tomcat Git repository (at GitHub).

The docs are in the `webapps/docs` subdirectory. They are in XML format and get processed into the HTML documentation as part of the Tomcat release.

Edit the documentation XML file(s) as you wish. The xdocs format is self-explanatory: use normal HTML markup, and add `<section>` or `<subsection>` tags as you see fit. Look at the existing docs as examples. Make sure you use valid XML markup.

If you're interested in previewing your changes, you will need to follow the directions for building Tomcat yourself. The docs will be generated in the `output/build/webapps/docs` directory.

Open a Bugzilla enhancement item with the explanation of your enhancements, and attach a `git diff` or `diff -u` format of your patch, or create a Pull Request at GitHub. We will evaluate and commit your patch as needed.

Note, that the Tomcat web site is updated with every release, so that documentation changes will not be visible until next Tomcat release. It is possible to view documentation for unreleased versions of Tomcat 7, Tomcat 8.5 and Tomcat 9 that is published by ASF Buildbot. See links on the buildbot page on Apache Tomcat web site.

Installation

How do I set up and run Tomcat on Macintosh OS X?

See TomcatOnMacOS

How do I set up and run Tomcat on Solaris 10?

See TomcatOnSolaris10

How do I set up and run Tomcat on OpenVMS?

See TomcatOnOpenVMS

How do I set up another tomcat service on Windows, sharing the same Tomcat Home ?

To install another Tomcat service using separate Home (binaries) and Base (configuration) paths you can use the `service.bat` script provided by Apache Tomcat. If your installation of Apache Tomcat does not have a `service.bat` script (in the `bin` directory), you can get one from a zip distributive for that version.

To install the service:

1. Set environment variables `CATALINA_HOME`, `CATALINA_BASE` and `JAVA_HOME` (or `JRE_HOME`) as usual, as documented in RUNNING.txt file.
2. Call the `service.bat` script to install the service, as shown in the Windows Service How-To in Tomcat documentation.
   `service.bat install NewServiceName --rename`

How do I install Tomcat as a service under Unix?

Create a shell program to start Tomcat automatically. Each UNIX varies in how it starts up automatic services, but there are two main variants:

BSD::In a typical BSD system, there are a series of start up scripts in `/etc` starting with `rc.`. Look for, or create, a file called `/etc/rc.local` and enter the appropriate instructions to start up Tomcat there as a shell script.

System V::In a typical UNIX System V setup, there is a directory containing startup scripts, and other directories which contain links to these startup scripts. Create the appropriate startup script for your setup, then create the appropriate links.

For more information on each, check your system documentation.

- 💡 It also makes a lot of sense to use the JavaServiceWrapper.

How to run Tomcat without root privileges?

The best way is to use jsvc, available as part of the Apache Commons Daemon project.

Other way is to put Apache httpd with mod_jk before your Tomcat servers, and use ports >=1024 in the Tomcat(s). However, if httpd is not needed for some other reason, this is the most inefficient approach.

An other method is to use SetUID scripts (assuming you have the capability) to do this. Here's how I do it.

Create a file called `foo.c` with this content (replace "/path/startupscript" with the tomcat startup script):

foo.c

```
#include <unistd.h>
#include <stdlib.h>
int main( int argc, char *argv[] ) {
  if ( setuid( 0 ) != 0 ) {
    perror( "setuid() error" );
    return 1;
  }
  printf( "Starting ${APPLICATION}\n" );
  execl( "/bin/sh", "sh", "/path/startupscript", 0 );
  return 0;
}
```

Run the following as root (replacing tmp with whatever you want the startup script to be and replacing XXXXX with whatever group you want to be able to start and stop tomcat:

```
gcc tmp.c -o tmp
chown root:XXXXX tmp
chmod ugo-rwx tmp
chmod u+rwxs,g+rx tmp
```

Now members of the tomcat group should be able to start and stop tomcat. One caveat though, you need to ensure that that your tomcat startup script is not writable by anyone other than root, otherwise your users will be able to insert commands into the script and have them run as root (very big security hole).

An other way is to use Iptables to redirect Port 80 and 443 to user ports (>1024)

```
/sbin/iptables -A FORWARD -p tcp --destination-port 443 -j ACCEPT
/sbin/iptables -t nat -A PREROUTING -j REDIRECT -p tcp --destination-port 443 --to-ports 8443
/sbin/iptables -A FORWARD -p tcp --destination-port 80 -j ACCEPT
/sbin/iptables -t nat -A PREROUTING -j REDIRECT -p tcp --destination-port 80 --to-ports 8080
/sbin/iptables-save or /etc/init.d/iptables save
```

If you'd like "localhost:443" to also redirect to "localhost:8443", you'll need this command as well:

```
/sbin/iptables -t nat -A OUTPUT -p tcp -o lo -destination-port 443 -j REDIRECT --to-ports 8443
```

Also note that if you have existing rules defined in your chains, you will need to make sure that the rules above are not ruled-out by another rule when using -A to add the above rules. For example, if you have an existing FORWARD rule that says "-j REJECT" than adding the FORWARD rule after it will have no effect.

BSD-based Unix systems such as Mac OS X use a tool similar to iptables, called ipfw (for Internet Protocol Fire Wall). This tool is similar in that it watches all network packets go by, and can apply rules to affect those packets, such as "port-forwarding" from port 80 to some other port such as Tomcat's default 8080. The syntax of the rules is different than iptables, but the same idea. For more info, google and read the man page. Here is one possible rule to do the port-forwarding:

```
sudo ipfw add 100 fwd 127.0.0.1,8080 tcp from any to any 80 in
```

Yet another way is to use authbind package (part of Debian- and CentOS based distributions) which allows a program that would normally require superuser privileges to access privileged network services to run as a non-privileged user.

How to create native launchers for Tomcat

See TomcatCreateNativeLaunchers

How do I rotate catalina.out?

Honestly, the first question is "why are you rotating catalina.out"? Tomcat logs very little to catalina.out so the usual culprit is web applications that stupidly send output to System.out or System.err. If that's the case, what you ought to do is set swallowOutput="true" on the application's <Context> configuration. That will send the output to a file configured (default) by conf/logging.properties. Once you've done that, get the application fixed to use a real logger, or at least use ServletContext.log().

If you've decided that you still absolutely positively need to rotate catalina.out, there is something that you have to understand: catalina.out is created by your shell's output redirection, just like when you type "ls -l > dir_listing.txt". So rotating the file needs to be done carefully.

You can't just re-name the file or you'll find that Tomcat will continue logging to the file under the new name. You also can't delete catalina.out and re-create it, or you'll never get anything logged to catalina.out after that, unless you restart Tomcat.

There are really only two ways to properly rotate catalina.out, and they both have downsides.

Rotate catalina.out using logrotate (or similar)

To use a tool like logrotate, you'll want to use the "copytruncate" configuration option. This will copy catalina.out to another file (like catalina.out.*datestamp*) and then truncates catalina.out to zero-bytes. There is a major downside to this if catalina.out is seeing a lot of action: some log messages written to the log file during the copy/truncate procedure may be lost.

Rotate catalina.out using rotatelogs or chronolog (or similar)

To use a tool like Apache httpd's rotatelogs or chronolog, you'll have to modify Tomcat's catalina.sh (or catalina.bat) script to change the output redirection from a redirect to a pipe. The existing code in catalina.sh looks like this:

```
    eval "\"$_RUNJAVA\"" "\"$LOGGING_CONFIG\"" $LOGGING_MANAGER $JAVA_OPTS $CATALINA_OPTS \
      -Djava.endorsed.dirs="\"$JAVA_ENDORSED_DIRS\"" -classpath "\"$CLASSPATH\"" \
      -Djava.security.manager \
      -Djava.security.policy=="\"$CATALINA_BASE/conf/catalina.policy\"" \
      -Dcatalina.base="\"$CATALINA_BASE\"" \
      -Dcatalina.home="\"$CATALINA_HOME\"" \
      -Djava.io.tmpdir="\"$CATALINA_TMPDIR\"" \
      org.apache.catalina.startup.Bootstrap "$@" start \
      >> "$CATALINA_OUT" 2>&1 "&"
```

You'll need to change that to something which looks more like this:

```
    eval "\"$_RUNJAVA\"" "\"$LOGGING_CONFIG\"" $LOGGING_MANAGER $JAVA_OPTS $CATALINA_OPTS \
      -Djava.endorsed.dirs="\"$JAVA_ENDORSED_DIRS\"" -classpath "\"$CLASSPATH\"" \
      -Djava.security.manager \
      -Djava.security.policy=="\"$CATALINA_BASE/conf/catalina.policy\"" \
      -Dcatalina.base="\"$CATALINA_BASE\"" \
      -Dcatalina.home="\"$CATALINA_HOME\"" \
      -Djava.io.tmpdir="\"$CATALINA_TMPDIR\"" \
      org.apache.catalina.startup.Bootstrap "$@" start \
      | "$PATH_TO_CHRONOLOG" $CATALINA_BASE/logs/catalina.out.%Y-%m-%d
```

This will be somewhat similar for catalina.bat, but the actual launch command will look different.

Also note that there are currently two places in catalina.sh (and catalina.bat) where Tomcat is launched, depending upon whether you are using a security manager or not. You should read the whole catalina.sh (or catalina.bat) file to make sure you have handled every case where Tomcat is launched.

Configuration

How do I set up multiple sites sharing the same war application/war file?

See CreateVirtualHosts

How do I log requests ?

See AccessLogValve

Can I set Java system properties differently for each webapp?

No. If you can edit Tomcat's startup scripts (or better create a `setenv.sh` file), you can add "-D" options to Java. But there is no way in Java to have different values of system properties for different classes in the same JVM.

There are some other methods available, like using `ServletContext.getContextPath()` to get the context name of your web application and locate some resources accordingly, or to define `<context-param>` elements in `WEB-INF/web.xml` file of your web application and then set the values for them in Tomcat context file (`META-INF/context.xml`). See https://tomcat.apache.org/tomcat-9.0-doc/config/context.html .

How do I configure Tomcat Connectors?

On the Tomcat FAQ, there is a list of Other Resources which should have information pointing you to the relevant pages.

Each connector has its own configuration, and its own set up. Check them for more information.

In particular, here are a number of locations for Tomcat Connectors:

- Tomcat Connectors Documentation
- Tomcat Connectors FAQ
- Configuring Tomcat Connectors for Apache
- Connectors How To
- AJP Connector in Tomcat 9 Configuration Reference

The following **excellent** article was written by Mladen Turk. He is a Developer and Consultant for JBoss Inc in Europe, where he is responsible for native integration. He is a long time commiter for Jakarta Tomcat Connectors, Apache Httpd and Apache Portable Runtime projects.

- *Fronting Tomcat with Apache or IIS - Best Practices*
  https://people.apache.org/~mturk/docs/article/ftwai.html

Over the time several different connectors have been built, and some connector projects have been abandoned (so beware of old documentation).

How do I configure Tomcat to work with IIS and NTLM?

See TomcatNTLM

Setting up SSL

Threads from the tomcat-user list

Using VeriSign:

- https://marc.info/?l=tomcat-user&m=106285452711698&w=2
- https://marc.info/?l=tomcat-user&m=107584265122914&w=2

Using OpenSSL:

- https://marc.info/?l=tomcat-user&m=106293430225790&w=2
- https://marc.info/?l=tomcat-user&m=106453566416102&w=2
- https://marc.info/?l=tomcat-user&m=106621232531781&w=2

A description of "what SSL is all about anyway":

- https://marc.info/?l=tomcat-user&m=106692394104667&w=2

HowTo SSL Client Authentication with Fallback to FORM Authentication

See SSLWithFORMFallback

How do I restrict the list of SSL ciphers used for HTTPS

See HowTo SSLCiphers.

How do I make Tomcat startup faster?

See HowTo FasterStartUp

How do I override the default home page loaded by Tomcat?

After successfully installing Tomcat, you usually test it by loading http://localhost:8080 . It is quite easy to override that page. Inside `$TOMCAT_HOME/conf/web.xml` there is a section called `<welcome-file-list>` and it looks like this:

```
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
```

The default servlet attempts to load the `index.*` files in the order listed. You may easily override the `index.jsp` file by creating an index.html file at `$TOMCAT_HOME/webapps/ROOT`. It's somewhat common for that file to contain a new static home page or a redirect to a servlet's main page. A redirect would look like:

```
<html>

<head>
<meta http-equiv="refresh" content="0;URL=http://mydomain.com/some/path/to/servlet/homepage/">
</head>

<body>
</body>

</html>
```

This change takes effect immediately and does not require a restart of Tomcat.

How do I enable Server Side Includes (SSI)?

See https://tomcat.apache.org/tomcat-7.0-doc/ssi-howto.html

How do I install the Administration web app?

Tomcat 5.5

If you install Tomcat 5.5 binaries, the Administration web app is not bundled with it; this describes how to add the Administration web app to your Tomcat 5.5 installation. (Tomcat 4.1 comes with the Administration web app as part of the binary).

The following refers to a Tomcat 5.5 set up on Windows 2000, so your path names will be different on *nix platforms. In this example, Tomcat 5.5.17 in installed in *c:\Program Files\Apache Software Foundation\Tomcat 5.5* (this is my **CATALINA_HOME**).

1. Unzip or untar (be careful to use GNU tar) the file containing the administration web app files (eg. *apache-tomcat-5.5.17-admin.zip*) to a temporary directory, eg. *c:\temp*.
2. Copy *c:\temp\apache-tomcat-5.5.17\conf\Catalina\localhost\admin.xml* to the directory *c:\Program Files\Apache Software Foundation\Tomcat 5.5\conf\Catalina\localhost*.
3. Copy the entire directory tree *c:\temp\apache-tomcat-5.5.17\server\webapps\admin*

to the directory *c:\Program Files\Apache Software Foundation\Tomcat 5.5\server\webapps*. This is an overlay, so \server\webapps is just pointing you to the \server\webapps, and the admin directory with its contents will be the only thing you see added there.

1. Add a line to your *c:\Program Files\Apache Software Foundation\Tomcat 5.5\conf\tomcat-users.xml* file so that you have a user who has **admin** role. For example, add this line just before the last line (containing **</tomcat-users>**) of the file:
    - <user username="admin" password="makesomethingup" roles="admin,manager"/>
2. Restart Tomcat.
3. Now when you visit *http://localhost:8080/admin* you should see a page that asks for a user name and password. If you still see the "no longer loaded" error message in your browser, you must either force a full reload of the web page (in Firefox, hold down Shift key while clicking on the Reload button) or just restart your browser completely.

Tomcat 6.0 and later

Development of Administration web app was ceased and it is no longer provided for Tomcat 6.0 and later versions.

An alternative is to use 3-rd party applications, such as PSI Probe. See AddOns page for links.

How do I add JARs or classes to the common classloader without adding them to $CATALINA_HOME/lib?

Either

a) Configure Tomcat to run with separate `$CATALINA_BASE` and `$CATALINA_HOME` directories (as documented in `RUNNING.txt`), and place your JARs and classes into `$CATALINA_BASE/lib`, or

b) Edit the file *catalina.properties* under `$CATALINA_BASE/conf`; there is a property called *common.loader* to which you can add additional paths to find JARs or classes for the common classloader.

How do I authenticate Manager access via JNDI to Active Directory for multiple Tomcat instances?

ADS insists that the CN of every group be unique, but the Manager app. always uses the group CN=manager. The default can be changed, but it's hard to find and you have to do it over every time you upgrade. Instead, pick an attribute other than the common name – for example, "description" – that doesn't have to be unique, name it as the `RoleName` attribute of the `Realm` (in server.xml, which you'll be editing anyway), and set that attribute to "manager" in each group you create. Create an OU for each Tomcat instance's groups and give that OU's DN as the `RoleBase` in that instance's server.xml. Create a uniquely-named group in each instance's OU with the chosen attribute ("description" for example) set to "manager".

Where and how do I set memory related settings to improve Tomcat performance?

When your web application is using large memory as this memory size default setting can be too small. One way to address this problem is to set a larger heap size.

If you start Tomcat by using the standard **script files** (such as `CATALINA_HOME/bin/catalina.bat` or `catalina.sh`), this can be done by setting `CATALINA_OPTS` environment variable. The recommended way to do so is to create a `setenv.bat` or `setenv.sh` file, — read RUNNING.txt for details.

Let say you want to increase it to 256 MB (as you required but make sure you have enough amount of physical memory/RAM and for 32bit system, use no more than 1.0-1.1 GB heap space size). Set the `CATALINA_OPTS` to the value of `-Xms256m -Xmx256m`. In some cases it is better to set slightly lower size for `-Xms`.

For `setenv.bat` use the following line:

```
set CATALINA_OPTS=-Xms256m -Xmx256m
```

For `setenv.sh` use the following:

```
CATALINA_OPTS='-Xms256m -Xmx256m'
```

There are other parameters that can be added, depending on your application and requirements, e.g: `'-XX:MaxPermSize'`.

For other parameters, look at the following pages:

- FAQ/Memory
- OutOfMemory

If you are running Tomcat as a **Windows service**, then environment variables and `setenv.bat` script have no effect. The relevant settings for the service wrapper application are stored in the Windows registry. They can be edited via Configuration application (`tomcat<N>w.exe`). See "Java" tab in the configuration dialog. The{{-Xms}} and `-Xmx` options are configured in fields named "Initial memory pool" and "Maximum memory pool". Other options can be added to "Java Options" field as if they were specified on the command line of java executable.

How do I make my web application be the Tomcat default application?

Congratulations. You have created and tested a first web application (traditionally called "mywebapp"), users can access it via the URL "http://myhost.company.com/mywebapp". You are very proud and satisfied. But now, how do you change the setup, so that "mywebapp" gets called when the user enters the URL "http://myhost.company.com" ?

The pages and code of your "mywebapp" application currently reside in (CATALINA_BASE)/webapps/mywebapp/. In a standard Tomcat installation, you will notice that under the same directory (CATALINA_BASE)/webapps/, there is a directory called `ROOT` (the capitals are important, even under Windows). That is the residence of the *current* Tomcat default application, the one that is called right now when a user calls up "http://myhost.company.com[:port]". The trick is to put your application in its place.

First stop Tomcat.
Then before you replace the current default application, it may be a good idea to make a copy of it somewhere else.
Then delete everything under the ROOT directory, and move everything that was previously under the (CATALINA_BASE)/webapps/mywebapp/ directory, toward this (CATALINA_BASE)/webapps/ROOT directory. In other words, what was previously .../mywebapp/WEB-INF should now be .../ROOT/WEB-INF (and not .../ROOT/mywebapp/WEB-INF).

Just by doing this, you have already made you webapp into the Tomcat *default webapp*.

Restart Tomcat and you're done.
Call up "http://myhost.company.com/" and enjoy.

Addendum 1: If you are deploying your application as a war file..

The above instructions relate to the situation where you are "manually" deploying your application as a directory-and-files structure under the /webapps directory. If instead you are using the "war" method to deploy your application, the principle is about the same :

- delete the ROOT directory
- name your war file "ROOT.war" (capitals mandatory)
- drop the ROOT.war file directly in the /webapps directory.
  Tomcat will automatically deploy it.

For more information about this topic in general, consult this page : "Configuration Reference / Context"

Addendum 2: If for some reason you want another method..

If, for some reason, you do not want to deploy your application under the CATALINA_BASE/webapps/ROOT subdirectory, or you do not want to name your war-file "ROOT.war", then read on. But you should first read this : "Configuration Reference / Context" and make sure you understand the implications.

The method described above is the simple method. The two methods below are more complex, and the second one has definite implications on the way you manage and run your Tomcat.

Method 2.1

- Place your war file outside of CATALINA_BASE/webapps (it **must** be outside to prevent double deployment).
- Place a context file named ROOT.xml in CATALINA_BASE/conf/<engine name>/<host name>. The single <Context> element in this context file MUST have a **docBase** attribute pointing to the location of your war file. The path element should not be set - it is derived from the name of the . xml file, in this case ROOT.xml. See the Context Container above for details.

Method 2.2

If you really know what you are doing..

- leave your war file in CATALINA_BASE/webapps, under its original name
- turn off autoDeploy **and** deployOnStartup in your Host element in the server.xml file.
- explicitly define **all** application Contexts in server.xml, specifying both path and docBase. You must do this, because you have disabled all the Tomcat auto-deploy mechanisms, and Tomcat will not deploy your applications anymore unless it finds their Context in the server.xml.

Note that this last method also implies that in order to make any change to any application, you will have to stop and restart Tomcat.

How do I set up Tomcat virtual hosts in a development environment?

See TomcatDevelopmentVirtualHosts

How do I get started with a simple cluster?

See ClusteringOverview

Programming

How do call tomcat ant tasks to deploy webapps?

See AntDeploy

How do I load a properties file?

Here are the three most popular ways::

- Use a classloader's `getResource()` method to get an url to the properties file and load it into the Properties. The properties file must be located within the webapp classpath (i.e. either `WEB-INF/classes/...` or in a jar in `WEB-INF/lib/`).

A challenge is to get the classloader when you are in a static initializer:

```
  public class Config {
     private static java.util.Properties prop = new java.util.Properties();
     private static loadProperties() {
          // get class loader
          ClassLoader loader = Config.class.getClassLoader();
          if(loader==null)
            loader = ClassLoader.getSystemClassLoader();

          // assuming you want to load application.properties located in WEB-INF/classes/conf/
          String propFile = "conf/application.properties";
          java.net.URL url = loader.getResource(propFile);
          try{prop.load(url.openStream());}catch(Exception e){System.err.println("Could not load configuration
file: " + propFile);}
     }

     //....
     // add your methods here. prop is filled with the content of conf/application.properties

     // load the properties when class is accessed
     static {
        loadProperties();
     }
  }
```

This method even works in a standalone java application. So it is my preferred way.

- Use a `ResourceBundle`. See the Java docs for the specifics of how the `ResourceBundle` class works. Using this method, the properties file must go into the `WEB-INF/classes` directory or in a jar file contained in the `WEB-INF/lib` directory.
- Another way is to use the method `getResourceAsStream()` from the `ServletContext` class. This allows you update the file without having to reload the webapp as required by the first method. Here is an example code snippet, without any error trapping:

```
// Assuming you are in a Servlet extending HttpServlet
// This will look for a file called "/more/cowbell.properties" relative
// to your servlet Root Context
InputStream is = getServletContext().getResourceAsStream("/more/cowbell.properties");
Properties  p  = new Properties();
p.load(is);
is.close();
```

How do I share sessions across web apps?

You cannot share sessions directly across web apps, as that would be a violation of the Servlet Specification. There are workarounds, including using a singleton class loaded from the common classloader repository to hold shared information, or putting some of this shared information in a database or another data store. Some of these approaches have been discussed on the tomcat-user mailing list, whose archives you should search for more information.

Sharing sessions across containers for clustering or replication purposes is a different matter altogether.

How can I access members of a custom Realm or Principal?

When you create a custom subclass of `RealmBase` or `GenericPrincipal` and attempt to use those classes in your webapp code, you'll probably have problems with `ClassCastException`. This is because the instance returned by `request.getUserPrincipal()` is of a class loaded by the server's classloader, and you are trying to access it through you webapp's classloader. While the classes maybe otherwise exactly the same, different (sibling) classloaders makes them different classes.

This assumes you created a `MyPrincipal` class, and put in Tomcat's server/classes (or lib) directory, as well as in your webapp's webinf/classes (or lib) directory. Normally, you would put custom realm and principal classes in the server directory because they depend on other classes there.

Here's what you would like to do, but it throws `ClassCastException`:

```
MyPrincipal p = request.getUserPrincipal();
String emailAddress = p.getEmailAddress();
```

Here are 4 ways you might get around the classloader boundary:

1) *Reflection*

```
Principal p = request.getUserPrincipal();
String emailAddress = p.getClass().getMethod("getEmailAddress", null).invoke(p, null);
```

2) *Move classes to a common classloader*

You could put your custom classes in a classloader that is common to both the server and your webapp - e.g., either the "common" or bootstrap classloaders. To do this, however, you would also need to move the classes that your custom classes depend on up to the common classloader, and that seems like a bad idea, because there a many of them and they a core server classes.

3) *Common Interfaces*

Rather than move the implementing custom classes up, you could define interfaces for your customs classes, and put the interfaces in the common directory. You're code would look like this:

```
public interface MyPrincipalInterface extends java.security.Principal {
  public String getEmailAddress();
}

public class MyPrincipal implements MyPrincipalInterface {
...
  public String getEmailAddress() {
    return emailAddress;
  }
}

public class MyServlet implements Servlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    MyPrincipalInterface p = (MyPrincipalInterface)request.getUserPrincipal();
    String emailAddress = p.getEmailAddress();
...
}
```

Notice that this method gives you pretty much the webapp code you wanted in the first place

4) *Serializing / Deserializing*

You might want to try serializing the response of `request.getUserPrincipal()` and deserialize it to an instance of webapp's MyPrincipal.

How do I get direct access to a Tomcat Realm?

Credit: This code is from a post by Yoav Shapira https://www.yoavshapira.com/ in the user list

Sometimes access directly into the Tomcat realm object is needed; to do, this the following code can be used. Be aware, however, that by using this, your application is relying on a Tomcat extension and is therefore non-standard.

Note that in order for this to work the Context of the web application in question needs to have its privileged attribute set to "true", otherwise web apps do not have access to the Tomcat classes.

```
Server server = ServerFactory.getServer();
//Note, this assumes the Container is "Catalina"
Service service = server.findService("Catalina");
Engine engine = (Engine) service.getContainer();
Host host = (Host) engine.findChild(engine.getDefaultHost());
//Note, this assumes your context is "myContext"
Context context = (Context) host.findChild("myContext");
Realm realm = context.getRealm();
```

Warning: The above recipe on how to obtain a `Context` for a web application is a bit obsolete and does not work in Tomcat 7 and later (as Server is no longer a singleton). There are other ways to achieve that. An easy one is to add a `Valve` or `Listener` to a context, as those classes have access to Tomcat internals. There may be other ways mentioned in the archives of the users mailing list.

How do I redirect System.out and System.err to my web page?

I have met a situation where I needed to redirect a portion of standard ouput (`System.out`, STDOUT) and standard error (`System.err`, STDERR) to my web page instead of a log file. An example of such an application is a compiler research platform that our resarch team is putting online for anybody to be able to quickly compile-test their programs on line. Naturally, the compilers dump some of their stuff to STDERR or STDOUT and they are not web application `.jar`. Thus, I needed badly these streams related to the compiler output to be redirected to my web editor interface. Having found no easy instructions on how to do that lead me writing up this quick HOWTO. The HOWTO is based on Servlets, but similar arrangements can be done for JSPs. The below example shows the essentials, with most non-essentials removed.

```
public class WebEditor extends HttpServlet
{
 ...
        public void doGet
        (
                HttpServletRequest poHTTPRequest,
                HttpServletResponse poHTTPResponse
        )
        throws IOException, ServletException
        {
                poHTTPResponse.setContentType("text/html");

                ServletOutputStream out = poHTTPResponse.getOutputStream();

                out.println("<html>");
                out.println("<body>");
                out.println("<head>");
                out.println("<title>WebEditor Test $Revision: 1.6 $</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("<h3>WebEditor Test $Revision: 1.6 $</h3>");
                out.println("<hr />");

                // Backup the streams
                PrintStream oStdOutBackup = System.out;
                PrintStream oStdErrBackup = System.err;

                try {

                  // Redired STDOUT and STDERR to the ServletOutputStream
                  System.setOut(new PrintStream(out));
                  System.setErr(new PrintStream(out));

                  try {
                        // ... call compiler here that produces
                        // tons of STDOUT/STDERR messages ...
                  } catch(Exception e) {
                        out.println(e);
                  }

                } finally {

                  // Restore original STDOUT and STDERR
                  System.setOut(oStdOutBackup);
                  System.setErr(oStdErrBackup);

                }

                out.println("<hr />");
                out.println("</body>");
                out.println("</html>");
        }
}
```

A few caveats arise, as for instance while the `System.out` and `System.err` are redirected as per above, no logging of these is done to files. You will need more legwork to do to make the additional logging. It is important to backup and restore the original streams as the above example does. The servlet should not be used to process several requests in parallel (some synchronization should be added to the above code to prevent that). Also, notice the use of `getOutputStream()`: when this method is called, the `getWriter()` method can no longer be used in the same response object.

Corrections and comments are most welcome!

How do I connect to a Websphere MQ (MQ Series) server using JMS and JNDI?

Basically, this works just as described in https://tomcat.apache.org/tomcat-9.0-doc/jndi-resources-howto.html: Within your application, you are using the standard JNDI and JMS API calls. In web.xml (the container independent application descriptor), you specify resource references (stub resources). And in context.xml (the container specific application descriptor), you are actually configuring the JMS connection.

More to the point. Here's some example code, which might be added to a Servlet. The example is sending a message to an MQ server:

```
import javax.jms.Queue;
import javax.jms.QueueConnection;
import javax.jms.QueueConnectionFactory;
import javax.jms.QueueSender;
import javax.jms.QueueSession;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.naming.Context;
import javax.naming.InitialContext;

Context ctx = (Context) new InitialContext().lookup("java:comp/env");
QueueConnectionFactory qcf = (QueueConnectionFactory) ctx.lookup("jms/MyQCF");
QueueConnection qc = qcf.createQueueConnection();
Queue q = (Queue) ctx.lookup("jms/MyQ");
QueueSession qs = qc.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
TextMessage tm = qs.createTextMessage();
tm.setText("Hi, there!");
QueueSender sender = qc.createSender();
sender.send(tm);
sender.close();
qs.close();
qc.close();
```

Note the following:

1. I have intentionally omitted proper resource handling. For example, one ought to ensure that qc.close() is always called by using a try { .. } finally { ..} block.
2. The code contains absolutely no references to com.ibm.mq*.jar.
3. There are only two items, which need configuration: "jms/MyQCF", and "jms/MyQ". We'll find them again in web.xml, and context.xml.

We have now written the code. Additionally, our web application needs the following files, and directories:

```
+--META-INF
|  +--- context.xml
+--WEB-INF
   +--- web.xml
   +--- lib
        +--- com.ibm.mq.jar
        +--- com.ibm.mqjms.jar
        +--- connector.jar
        +--- dhbcore.jar
        +--- geronimo-j2ee-management_1.0_spec-1.0.jar
        +--- geronimo-jms_1.1_spec-1.0.jar
```

The application descriptor web.xml looks just the same as usual, with the exception of the following lines:

```
<resource-env-ref>
  <resource-env-ref-name>jms/MyQCF</resource-env-ref-name>
  <resource-env-ref-type>javax.jms.QueueConnectionFactory</resource-env-ref-type>
</resource-env-ref>

<resource-env-ref>
  <resource-env-ref-name>jms/MyQ</resource-env-ref-name>
  <resource-env-ref-type>javax.jms.Queue</resource-env-ref-type>
</resource-env-ref>
```

This is simply telling, that the items "jms/MyQCF", and "jms/MyQ" exist, and are instances of QueueConnectionFactory, and Queue, respectively. The actual configuration is in context.xml:

```
<Resource
   name="jms/MyQCF"
   auth="Container"
   type="com.ibm.mq.jms.MQQueueConnectionFactory"
   factory="com.ibm.mq.jms.MQQueueConnectionFactoryFactory"
   description="JMS Queue Connection Factory for sending messages"
   HOST="<mymqserver>"
   PORT="1414"
   CHAN="<mychannel>"
   TRAN="1"
   QMGR="<myqueuemanager>"/>
<Resource
   name="jms/MyQ"
   auth="Container"
   type="com.ibm.mq.jms.MQQueue"
   factory="com.ibm.mq.jms.MQQueueFactory"
   description="JMS Queue for receiving messages from Dialog"
   QU="<myqueue>"/>
```

Basically, you just have to enter your values for <myqserver> (the WebSphere MQ servers host name), <mychannel> (the channel name), <myqueuemanager> (the queue manager name), and <myqueue> (the queue name). Both these values, the associated names (HOST, PORT, CHAN, ...), and their collection is truly MQ specific. For example, with ActiveMQ, you typically have a broker URL, and a broker name, rather than HOST, PORT, CHAN, ...

The main thing to know (and the reason why I am writing this, because it took me some hours to find out): How do I know the property names, their meaning, and possible values? Well, there is an excellent manual, called "WebSphere MQ Using Java". It should be easy to find by entering the title into Google. The manual contains a section, called "Administering JMS objects", which describes the objects being configured in JNDI. But the most important part is the subsection on "Properties", which contains all the required details.

How do I use DataSources with Tomcat?

See UsingDataSources

How do I use Hibernate and database connection pooling with Tomcat?

See TomcatHibernate

How do I use DataSourceRealms for authentication and authorization?

See TomcatDataSourceRealms

Troubleshooting

Tomcat crashed! What do I do now?

These steps are in no particular order ...

1. Read the Tomcat FAQ
2. Read the Tomcat RELEASE NOTES - there is something about Linux in it
3. First look at the stack traces. I hope a stack trace was produced before the failure aborted the JVM process. After you get a few stack traces, see if a pattern appears. Trace back to source code if needed.
4. Patch (or *unpatch*!) the operating system as needed.
5. Patch (or *unpatch*!) the JVM (Java Virtual Machine).
6. Linux Problem? - read the RELEASE NOTES!
7. Look at commercial vendor support for other servlet engines. Sometimes the problem is universal regardless of servlet engine and may be a JVM /OS/application code issue
8. Search Google for web pages - maybe someone else had this problem. I'll bet they did.
9. Search Google news groups
10. If the JVM is from a commercial vendor, (eg: IBM, HP) check their release notes and news groups
11. Using a database? Make sure JDBC type 4 drivers are used. Check their release notes.
12. Tweak JVM memory parameters. Setting memory too high can be as bad as having memory too low. If your memory settings are set too high, Java 1.3 JVMs may freeze while waiting for the entire garbage collection to finish. Also if the JVM has too much memory, if may be starving other resources on the machine which are needed which may be causing unforeseen exceptions. In a nutshell, throwing more memory doesn't always solve the problem!
13. Turn off the Java JIT compiler. See the Java Docs on how to do this.

I'm encountering classloader problems when using JNI under Tomcat

The important thing to know about using JNI under Tomcat is that one cannot place the native libraries OR their JNI interfaces under the WEB-INF/lib or WEB-INF/classes directories of a web application and expect to be able to reload the webapp without restarting the server. The class that calls System. loadLibrary(String) must be loaded by a classloader that is not affected by reloading the web application itself.

Thus, if you have JNI code that follows the convention of including a static initilaizer like this:

```
class FooWrapper {
    static {
        System.loadLibrary("foo");
    }

    native void doFoo();
}
```

then both this class and the shared library should be placed in the `$CATALINA_HOME/shared/lib` directory.

Note that under Windows, you'll also need to make sure that the library is in the `java.library.path`. Either add `%CATALINA_HOME%\shared\lib` to your Windows PATH environment variable, or place the DLL files in another location that is currently on the `java.library.path`. There may be a similar requirement for UNIX based system (I haven't checked), in which case you'd also have to add `$CATALINA_HOME/shared/lib` to the PATH environment variable. (Note: I'm not the original author of this entry.)

The symptom of this problem that I encountered looked something like this -

```
java.lang.UnsatisfiedLinkError: Native Library WEB-INF/lib/libfoo.so already loaded in another classloader
        at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1525)
```

If the `UnsatisfiedLinkError` is intermittent, it may be related to Tomcat's default session manager. It restored previous sessions at startup. One of those objects may load the JNI library. Try stopping the Tomcat JVM, deleting the SESSIONS.ser file, then starting Tomcat. You may consider changing the session persistence manager at this time.

Note that Tomcat 6.0.14 the $CATALINA_HOME/shared/lib directory does not exist. You will need to add this and you will need to edit $CATALINA_HOME /conf/catalina.properties so that the shared.loader line looks like this shared.loader=$CATALINA_HOME/shared/lib

How do I debug a Tomcat application?

There is nothing magical about debugging a Tomcat application. All you need is an IDE and two environment variables.

- If you have not already done so begin by creating a new Tomcat context for your application. Navigate to **TOMCAT_HOME\conf\Catalina\localhost** and create a new file, say, myapp.xml. This will become part of your url, so to access your app you'll have to type http://localhost:8080/myapp.
- Enter the following in myapp.xml:

```
<Context docBase="c:/workspace/myapp/WebRoot" />
```

- This assumes you have a web application containing WEB-INF in **c:/workspace/myapp/WebRoot**
- Create two environment variables:

```
C:\>set JPDA_ADDRESS=1044
C:\>set JPDA_TRANSPORT=dt_socket
```

- Now, you can launch Tomcat with these debug options:

```
TOMCAT_HOME\bin\>catalina jpda start
```

- Use your IDE to connect to Tomcat through port 1044

See also: FAQ/Developing

How do I debug a Tomcat application when Tomcat is run as a Windows service ?

You can debug the tomcat service by editing the service parameters as follows.

- Launch a command prompt
- Set the proper CATALINA_HOME environment variable: pointing to tomcat home

- Run the following command:

```
%CATALINA_HOME%\bin\tomcat6w.exe //ES//tomcat6
```

- Select the Java tab in the properties dialog box,
- Add the following two lines to the Java Options text box:

```
-Xdebug
-Xrunjdwp:transport=dt_socket,address=127.0.0.1:1044,server=y,suspend=n
```

If you want to allow remote debugging, replace 127.0.0.1 by your server IP address.

- Click on "Apply" and close the dialog by clicking on "OK"
- Restart the Apache Tomcat service
- Use your IDE to connect to Tomcat through port 1044

For IntelliJ IDEA you choose a remote debug target and set transport to "socket" and mode to "attach" , then you specify the host (127.0.0.1) and port (1044)

See also: FAQ/Developing

How do I check whether Tomcat is UP or DOWN? There is no status command

Unfortunately, the `org.apache.catalina.util.ServerInfo` class does not determine if Tomcat is UP or DOWN. It is possible to do an HTTP GET on the root url but this is not accurate. In my case I sometimes use a regular Apache HTTPd to display a maintainence message while upgrading, etc. and using that method would give false positives.

The correct way to do determine status is to parse the admin port from server.xml and see if we can connect to it. If we can then the Tomcat is UP otherwise it is DOWN.

Here is my code to do this. Consider it public domain and use it as you see fit. Tomcat makes a note of this connection with something like this on the console.

```
May 1, 2007 5:10:35 PM org.apache.catalina.core.StandardServer await
WARNING: StandardServer.await: Invalid command '' received
```

Ideally this should be incorporated into `org.apache.catalina.util.ServerInfo` by some committer. In addition to the shutdown command they should add commands like status (UP or DOWN) and uptime in the await method of `org.apache.catalina.core.StandardServer`

```
import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.xml.sax.SAXException;

/**
 * Check to see if Tomcat is UP/DOWN.
 *
 * This parses the server.xml file for the Tomcat admin port and see if
 * we can connect to it. If we can, then the Tomcat is UP otherwise it
 * is DOWN
 *
 * It is invoked as follows:
 *    java -Dcatalina.base=c:/tomcat-6.0.10 CatalinaStatus
 *
 * It can also (optionally) shutdown the Tomcat by adding the shutdown
 * command line parameter as follows:
 *
 *    java -Dcatalina.base=c:/tomcat-6.0.10 CatalinaStatus shutdown
 *
 * @author Shiraz Kanga <skanga at yahoo.com>
 */
public class CatalinaStatus
{
```

```
/**
 * Pathname to the server configuration file.
 */
protected static String configFile = "conf/server.xml";
protected static String serverShutdown;
protected static int serverPort;

/**
 * The application main program.
 *
 * @param args Command line arguments
 */
public static void main (String args[])
{
  Document configDom = getXmlDom (configFile ());
  parseDocument (configDom);
  // System.out.println ("Catalina.serverPort: " + serverPort);
  // System.out.println ("Catalina.serverShutdown: " + serverShutdown);

  // Stop the existing server
  try
  {
    Socket localSocket = new Socket ("127.0.0.1", serverPort);
    System.err.println ("Server status:  UP");
    if ((args.length > 0) && (args[0].equalsIgnoreCase ("shutdown")))
    {
      System.out.println ("Tomcat shutdown initiated" );
      doShutdown (localSocket);
    }

    localSocket.close ();
  }
  catch (IOException e)
  {
    System.err.println ("Server status:  DOWN");
    System.exit(1);
  }
}

/**
 * Return a File object representing our configuration file.
 */
protected static File configFile ()
{
  File confFile = new File (configFile);
  if (!confFile.isAbsolute())
    confFile = new File (System.getProperty ("catalina.base"), configFile);
  return (confFile);
}

/**
 * Parses an XML file and returns a DOM document.
 */
public static Document getXmlDom (File fileName)
{
  try
  {
    // Create a builder factory
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance ();

    // Create the builder and parse the file
    Document doc = factory.newDocumentBuilder ().parse (fileName);
    return doc;
  }
  catch (SAXException e)
  {
    // A parsing error occurred; the xml input is not valid
    e.printStackTrace ();
  }
  catch (ParserConfigurationException e)
  {
```

```
      e.printStackTrace ();
    }
    catch (IOException e)
    {
      e.printStackTrace ();
    }
    return null;
  }

  /**
   * Extract the server port & shutdown command from the DOM
   */
  private static void parseDocument (Document configDom)
  {
    //get the root element which is Server Eg: <Server port="8005" shutdown="SHUTDOWN">

    Element docEle = configDom.getDocumentElement ();
    serverPort = Integer.parseInt (docEle.getAttribute ("port"));
    serverShutdown = docEle.getAttribute ("shutdown");
  }

  /**
   * Send the shutdown command to the server
   */
  private static void doShutdown (Socket localSocket)
  {
    try
    {
      OutputStream outStream = localSocket.getOutputStream ();

      for (int i = 0; i < serverShutdown.length (); i++)
        outStream.write (serverShutdown.charAt (i));
      outStream.flush ();
      outStream.close ();
    }
    catch (IOException e)
    {
      System.out.println ("ERROR: I/O Exception during server shutdown.");
      e.printStackTrace ();
    }
  }
}
```

How do I obtain a thread dump of my running webapp ?

You can only get a thread dump of the entire JVM, not just your webapp. This shouldn't be a big deal, but should be made clear: you are getting a dump of all JVM threads, not just those "for your application", whatever that means.

Getting a thread dump depends a lot on your environment. Please choose the section below that matches your environment best. The more universal and convenient options are presented first, while the more difficult ones or those for specific setups are provided later. Generally, you should start at the top of the list and work your way down until you find a technique that works for you.

If you are running Oracle (Sun) JDK

Oracle JDK (not the JRE) (formerly Sun JDK) since version 1.6 (and since 1.4 on *nix systems) ships with a program called *jstack* (or *jstack.exe* on Microsoft Windows) which will give you a thread dump on standard output. Redirect the output into a file and you have your thread dump. You will need the process id ("pid") of the process to dump. Use of the program *jps* (*jps.exe* on Microsoft Windows) can help you determine the pid of a specific Java process.

See Tools page in JDK documentation for usage reference.

If you are running on *NIX

Send a SIGQUIT to the process. The thread dump will be sent to stdout which is likely to be redirected to CATALINA_BASE/logs/catalina.out.

To send a SIGQUIT, use *kill -3 <pid>* from the command line.

If you are running on Microsoft Windows

You can try to use SendSignal, developed specifically for this purpose. Make sure you read the comments for certain sitautions (e.g. running as a service, RDP connections, etc.). http://www.latenighthacking.com/projects/2003/sendSignal/

If you are running Tomcat as a service on Microsoft Windows

Tomcat service has a monitoring application with it. When it is running it puts an icon in the Windows system tray area. Right-click the icon, a menu will appear. Select "Thread Dump" command from the menu. It will cause thread dump to be printed to stdout. The service captures stdout into a log file (`logs /tomcat`*NN*`-stdout.`*DATE*`.log`).

If the monitoring application is not running, you can start it manually. The command is

```
Tomcat9w.exe //MS//
```

or

```
Tomcat9w.exe //MS//servicename
```

If you installed Tomcat with an "exe" installer, "Apache Tomcat *version servicename*" group in the Windows menu has shortcut "Monitor Tomcat" that starts the monitoring application.

For details, see Windows service page in Tomcat documentation.

If you have Tomcat running in a console

*NIX: Press CRTL-\ Microsoft Windows: press CRTL-BREAK

This will produce a thread dump on standard output, but may not be possible to capture to a file.

Using Java code

If you cannot use any of the above methods, it is also possible to obtain a thread dump programmatically, with a Servlet or JSP page that runs within Tomcat. For example, you can use `java.lang.Thread.getAllStackTraces()` method.

Tomcat Manager web application starting with Tomcat 7.0.58 / 8.0.0 supports a command that outputs a thread dump. (Tomcat 9 documentation, BZ 57261)

`StuckThreadDetectionValve` valve logs stacktraces of request processing threads that are busy for longer than configured time limit. It is available starting with Tomcat 6.0.36 / 7.0.14. (Tomcat 9 documentation)

How do I read a Java thread dump ?

Java thread dumps are just text files, so you can read them with any text editor. There are some tools that can make your life easier, especially if you need to look at more than one thread dump at once.

One such tool is the Thread Dump Viewer (TDV), which you can find here: https://sourceforge.net/projects/tdv/. It is a bit old (last release: 2007) but it can be somewhat helpful.

How do I obtain a heap dump?

See Getting a Heap Dump on the help pages of Eclipse Memory Analysis Tool.

How do I add my own custom MBean to monitor my application within Tomcat 6?

First of all, you can read this great tutorial from Christopher Blunck (chris@wxnet.org). I will just add my comments and improvements.

1. Start your Tomcat and check that you have access to http://localhost:8080/manager/jmxproxy/.

It means that JMX is enabled on your Tomcat configuration (if not, check if the following line is in your /conf/server.xml file:
`<Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" />`
Otherwise, check the Tomcat documentation to activate it). Let this page opened to check further if your custom MBean is detected by Tomcat.

2. Build your custom MBean by following the Christopher Blunck's example:

ServerMBean.java :

```
  package org.wxnet.mbeans;

  public interface ServerMBean {
    public void setLoggingLevel(int level);
    public long getUptime();
  }
```

Server.java :

```
package org.wxnet.mbeans;

  import javax.management.MBeanServer;
  import javax.management.MBeanServerFactory;
  import javax.management.ObjectName;

  import java.util.ArrayList;

  public class Server implements ServerMBean {
    private int _logLevel = 1;
    private long _startTime = 0L;

    public Server() {
      MBeanServer server = getServer();

      ObjectName name = null;
      try {
        name = new ObjectName("Application:Name=Server,Type=Server");
        server.registerMBean(this, name);
      } catch (Exception e) {
        e.printStackTrace();
      }

      _startTime = System.currentTimeMillis();
    }


    private MBeanServer getServer() {
      MBeanServer mbserver = null;

      ArrayList mbservers = MBeanServerFactory.findMBeanServer(null);

      if (mbservers.size() > 0) {
        mbserver = (MBeanServer) mbservers.get(0);
      }

      if (mbserver != null) {
        System.out.println("Found our MBean server");
      } else {
        mbserver = MBeanServerFactory.createMBeanServer();
      }

      return mbserver;
    }


    // interface method implementations
    public void setLoggingLevel(int level) { _logLevel = level; }
    public long getUptime() { return System.currentTimeMills() - _startTime; }
  }
```

In this implementation, firstly notice the *ObjectName* representing the MBean (in the constructor):
*name = new ObjectName("**Application**:Name=**Server**,Type=**Server**");*
Do not hesitate to change the domain name (the first parameter) by your own to easily find your MBean reference in the http://localhost:8080/manager/jmxproxy page.

Secondly, take a look at your MBean constructor:

1. First step is to get a reference to the Tomcat's MBeanServer with *MBeanServer server = getServer();*.

2. The *getServer()* method returns the first MBean server in the list of MBean servers registered in JVM, which is the one used by Tomcat.

In my application architecture, I placed the 2 MBeans files (the interface and its implementation) in a particular package (I don't think its compulsary but definitely more aesthetic). Compile those one in a jar archive and place it in the Tomcat's library folder (`/lib`).

3. Build your **ContextListener**: According to the Tomcat's documentation, a Listener is a *a component that performs actions when specific events occur, usually Tomcat **starting** or Tomcat stopping.*. We need to instantiate and load our MBean at Tomcat's start. So we build a ContextListener.java file which is placed wherever you want in your project architecture:

```
package org.bonitasoft.context;

/**
 * @author Christophe Havard
 *
 */

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import org.bonitasoft.mbeans.Server;

public final class ContextListener  implements ServletContextListener {

  public void contextInitialized(ServletContextEvent event) {
    Server mbean = new Server();
  }

  public void contextDestroyed(ServletContextEvent event) { }

}
```

Take a look especially at the contextInitialized method. It just instantiates our custom MBean. Don't forget that the MBean register itself in the Tomcat's MBeanServer in its constructor. DO NOT FORGET to change the *package* line according to your application architecture.

Then, you have to modify your WEB-INF/web.xml file to make Tomcat execute your ContextListener.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <display-name>My Web Application</display-name>
 '''''bla bla bla...'''''
  <listener>
    <listener-class>org.bonitasoft.context.ContextListener</listener-class>
  </listener>
</web-app>
```

In his tutorial, Christopher Blunck suggests to compile the ContextListener.java file in a jar archive and then place it into our WEB-INF/lib folder. In my own experiments, I never found any difference without doing this.

4. The ***mbeans-descriptor.xml*** file: The only entry in the Tomcat documentation about custom MBean is about this file. It says "*You may also add MBean descriptions for custom components in a mbeans-descriptor.xml file, located in the same package as the class files it describes.*". Unfortunately, instead of reading this file, Tomcat applied its own templates to replace my MBeans attributes and operations descriptions... I really didn't figure out what is the correct way of using and placing this file. So I don't use it.

5. The configuration should be over. You should have done the following operations:

1. Build your MBean,
2. Compile it and place the .jar archive in the Tomcat's /lib folder,
3. Build your ContextListener.java,
4. Add a reference to your ContextListener inside your WEB-INF/web.xml file

You can try to run your project. Open the http://localhost:8080/manager/jmxproxy page and find your custom MBean (with a simple ctrl+f). You can see its domain, name, type and its attributes and methods.

You can now use this MBean in your application by getting a reference to the Tomcat's MBean server:

```
MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
//call operations with invoke(...) and attributes with getAttributes(...)
```

Do not hesitate to check the ManagementFactory class javadoc.