Logging

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

Preface

This FAQ section provides help with logging-related issues.

As you read these questions, please keep in mind that Tomcat's internal logging is separate from your own webapp's logging. You would typically be concerned only with your own webapp's logging. You would modify Tomcat's internal logging settings if you are debugging a possible issue or running into other problems. It is anticipated that Tomcat's out-of-the-box logging configuration will be fine for the vast majority of users and environments.

Questions

Answers

Does Tomcat have built-in logging capabilities, and if so how do I use them?

The Servlet Specification requires Servlet Containers like Tomcat to provide at least a rudimentary implementation of the `ServletContext#log` method. Tomcat provides a much richer implementation than required by the Spec, as follows:

- Prior to Tomcat 5.5, Tomcat provided a Logger element that you could configure and extend according to your needs.
- Starting with Tomcat 5.5, Logger was removed and Apache Commons-Logging `Log` is used everywhere in Tomcat. Read the Commons-Logging documentation if you'd like to know how to better use and configure Tomcat's internal logging. See also https://tomcat.apache.org/tomcat-9.0-doc/logging.html
- In Tomcat 7 (and also 6), the logging code is based on a set of classes interacting with the java.util.logging API (JUL), which comes with Java since version 1.4. The Tomcat startup script configures the JVM to use a web-application-aware implementation of the JUL LogManager. This Tomcat logging infrastructure is called JULI, and one can still distinguish its Apache Commons Logging heritage, but the complex configuration has been edited out and the package name changed.

Web applications can get logging service by using the Servlet API logging (which not recommended), the JUL interface (which ultimately goes to JULI) or any other preferred interface for which they furnish the jar files and the appropriate configuration (see the respective descriptions for Log4J, SLF4J, logback or Apache Commons Logging for example).

To additionally log information about requests going to the web application, "Valves" can be configured in the server.xml file, as described in detail here. For example, inside the <Engine> tag:

```
<Valve className="org.apache.catalina.valves.AccessLogValve"
        directory="logs"  prefix="localhost_access_log." suffix=".log" pattern="common" resolveHosts="false"/>
```

This will produce a log file for each day, such as logs/localhost_access_log.2008-03-10.log, containing the files requested, IP address of the requester, and similar information.

```
128.34.123.121 - - [10/Mar/2008:15:55:57 -0500] "GET /upload/ClickPoints.jsp HTTP/1.1" 200 2725
```

In addition, Tomcat does not swallow the System.out and System.err JVM output streams. You may use these streams for elementary logging if you wish, but a more robust approach such as commons-logging or Log4J is recommended for production applications.

What role does commons-logging play in logging?

Tomcat wants to support multiple logging implementations, so it uses commons-logging. In case that's unclear, think of it like this. You are a Tomcat developer. The car you drive when logging is the commons-logging car. The engine of that car is either JULI or log4j. Without one of these engines, the car goes no where. However regardless of whether you use JULI or log4j, the steering wheel, break, gas pedal, etc. are the same.

Related FAQ: What role does JULI and log4j play in logging?

What role does JULI and log4j play in logging?

First see: What role does commons-logging play in logging?

Note in addition that in your own applications you could log directly with JULI or log4j. But once you choose one, you can't easily switch to the other later. If you use commons-logging you can.

How do I configure commons-logging for use with Tomcat?

You need to specify a commons-logging configuration file and, if you wish, a logging implementation that supports commons-logging. JDK 1.4 (and later) java.util.Logging and Log4j are the two most commonly used logging toolkits for Tomcat. Tomcat 5.5 and Tomcat 6.0 use java.logging as default implementation for commons-logging. So this *should* work by default, but sometimes it doesn't (see #Q9).

If you supply an external logging toolkit such as Log4J, it needs to be located in the $CATALINA_HOME/common/lib directory (for Tomcat 5.0 and earlier). Tomcat 5.5 and later uses commons-logging while bootstrapping so some people suggest adding Log4j to the bootstrap classpath by using the scripts in $CATALINA_HOME/bin (see Need for it to be in bootstrap classpath?). A better approach apparently working is:

1. Put log4j.jar in the $CATALINA_HOME/common/lib directory
2. Put the *full* commons-logging.jar in the $CATALINA_HOME/common/lib directory, even if you see the *reduced* API version there, named commons-logging-api.jar

Through some classloading voodoo during bootstrapping, if you have the full commons-logging.jar file in your common/lib directory, it replaces the classes from the commons-logging-api.jar file and will reinitialize the logging system and attempt to locate log4j or whatever other logging system you may be using. (see this thread).

The above recipe is for Tomcat 5.5. For Tomcat 7 — see Documentation.

See also the following mailing list discussions:

- A log4j 1.x example
- Logging Configuration
- Example with JSVC and running on port 80.
- Tomcat and Log4j Configuration (and Velocity), addressing and solving the bootstrap commons-logging.jar problem

How should I log in my own webapps?

While you can use System.out and System.err to log, we strongly recommend using a toolkit like Log4J or JDK 1.4's java.util.logging package. With these toolkits, you have significantly more functionality. For example, sending emails, logging to a database, controlling at runtime the logging level of different classes, inspecting the logs with a graphical viewer, etc.

We also recommend that you separate your logging from Tomcat's internal logging. That means you should bundle your logging toolkit with your webapp. If you are using Log4J, for example, place the Log4J jar in the WEB-INF/lib directory of your webapp and the Log4J configuration file in the WEB-INF/classes directory of your webapp. This way different web applications can have different logging configurations and you don't need to worry about them interfering with each other.

Where does System.out go?

System.out and System.err are both redirected to CATALINA_BASE/logs/catalina.out when using Tomcat's startup scripts (bin/startup.sh/.bat or bin/catalina.sh/.bat). Any code that writes to System.out or System.err will end up writing to that file.

If your webapp uses System.out and/or System.err a lot, you can suppress this via the 'swallowOutput' attribute in your <Context> configuration element and send to different log files (configured elsewhere: see the documentation for configuring logging).

How do I rotate catalina.out?

CATALINA_BASE/logs/catalina.out does not rotate. But it should not be an issue because nothing should be printing to standard output since you are using a logging package, right?

If you really must rotate catalina.out, here are some techniques you can use:

1. If you are using jsvc 1.0.4 or later (from Apache Commons Daemon project) to launch Tomcat, you can send SIGUSR1 signal to jsvc to get it to re-open its log files (Jira Ticket). You can couple this with 'logrotate' or your favorite log-rotation utility (including good-old 'mv') to re-name catalina.out at intervals and then get jsvc to re-open the original (catalina.out) file and continue writing to it.
2. Use 'logrotate' with the 'copytruncate' option. This allows you to externally rotate catalina.out without changing anything within Tomcat.
3. Modify bin/catalina.sh (or bin/catalina.bat) to pipe output from the JVM into a piped-logger such as cronolog or Apache httpd's rotatelogs (note that the previous reference is for Apache httpd documentation and *is not applicable to Tomcat* – it merely illustrates the concept).
   See also the patch in Bug 53930, "Allow capture of catalina stdout/stderr to a command instead of just a file".

References to mailing list discussions:

- tomcat-users thread from 2003
- tomcat-users thread from 2009
- tomcat-users thread from 2011
- tomcat-users thread from 2012

Where are the logs when running Tomcat as a Windows service?

See these mailing list archive threads:

- Where are the Tomcat logs when running as a Windows service?

How do I customize the location of the tomcat logging.properties file?

Set the following property when starting tomcat:

java.util.logging.config.file

Example: `-Djava.util.logging.config.file=/etc/tomcat/logging.properties`

For another example of how to set this look in catalina.sh for Tomcat 6.0.16 on lines 182-185. The statements look like this:

```
# Set juli LogManager if it is present
if [ -r "$CATALINA_BASE"/conf/logging.properties ]; then
  JAVA_OPTS="$JAVA_OPTS "-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager" "-Djava.util.
logging.config.file="$CATALINA_BASE/conf/logging.properties"
fi
```

Projects such as JPackage that repackage Tomcat for Linux typically move the configuration to a directory dictated by the FHS standard (http://www.pathname.com/fhs/), and therefore use the java.util.logging.config.file property to set the location of the logging.properties file in the Tomcat startup script.

On Fedora the startup script in typically located in /etc/rc.d/init.d/ and on Gentoo linux it is located in /etc/init.d/. On RedHat the startup script for Tomcat 5.5 is /etc/init.d/tomcat5 but eventually the real startup script is /usr/bin/dtomcat5.

Since java.logging is the default commons-logging implementation in Tomcat, why is it not working in my Linux distribution?

Yes, if you read Tomcat logging documentation, it says java.util.logging should work by default. But many Linux distribution repackage Tomcat and sometimes it does NOT work by default.

Here are some things you can check:

1. tomcat-juli.jar should be in your $CATALINA_HOME/bin directory
2. tomcat startup script should run java with `-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager`
3. tomcat startup script should run java with `-Djava.util.logging.config.file=<some_path>/logging.properties`
4. obviously, the logging.properties file must exist in the directory specified in the tomcat script at point #3

If you don't know where to look for your Tomcat startup script, see the previous How do I customize the location of the tomcat logging.properties file?

In RHEL5 (RedHat Enterprise Server 5) the Tomcat 5.5 rpm installation does not include the tomcat-juli.jar file. This is what I made:

- look for what Tomcat version you got installed with: `yum list installed tomcat5`

Since I had the 5.5.23, I downloaded the Tomcat Binaries 5.5.23 from https://archive.apache.org/dist/tomcat/, then:

```
tar xf apache-tomcat-5.5.23.tar.gz
cd apache-tomcat-5.5.23/bin
cp tomcat-juli.jar /usr/share/tomcat5/bin/
```

Restart Tomcat... and it's working!