Startup script for Tomcat on Centos | Redhat | Fedora

General

Author: Sebastian Hennebrueder
Date: August, 22 2008
Used software and frameworks
Tomcat 6
Java 6
Linux distribution Centos 5

Source code: http://www.lalilu na.de/download/tomcat-linux-startup-script.zip

PDF version of the tutorial: http://www.lalilu na.de/download/tomcat-linux-startup-script-en.pdf

Beginning

Why not using the catalina.sh startup.sh and shutdown.sh?

Apache Tomcat provides already scripts to start it up. They are powerful and correct but I wanted to have a script, I can understand. The scripts provided by Apache Tomcat have workarounds for IBM400, Cygwin and MacOs. After simplifying the script I arrived at about 185 lines versus 780.

Normal script behaviour

A script is called by the linux init process with parameters like start or stop. If **start** is passed as argument, the script should start up the service, note the PID (process ID) somewhere in /var/run/... and should create a lock file in /var/lock/subsys/

The PID is a convenient handle for the service. You can show the status of the process for example

```
cat "/proc/$(cat /var/run/tomcat.pid)/status"
```

Or you can kill a process brutally

kill -9 4711
or

```
kill -9  $(cat /var/run/tomcat.pid)
```

The lock file created in /var/lock/subsys is used by Linux when restarting the system or switching from a higher runlevel to a lower one. An existing file tells Linux that the service is running.

If a file exists and there is a symbolic link starting with K for the runlevel you are going to, the script is called with the stop parameter.

Sample

We are in runlevel 5 of Linux and there is a file **/var/lock/subsys/tomcatd** and a symbolic link in **/etc/rc1.d/K50tomcatd**.

We call init 1 to switch to single user mode in Linux.

Linux assumes that the service is running and will call /etc/init.d/tomcatd stop

It is very important that the name of the lock file corresponds to the script name: tomcatd

If a script is called with the 'stop' parameter, it shuts the service down identified by the PID in the PID file and removes the PID file and the lock file. We have the default behaviour defined so let us look at some border cases.

Further behaviour

If you check how other scripts work, you can see the following behaviour:

If a service has a pidfile and is running, a call with 'start' is ignored.

If a service has a pidfile and is NOT running, a call with 'start' will start it and update the PID file.

After the service was started successfully a script returns 0 else something greater than 0.

If a service has a pidfile and is not running, a call with 'stop' will exit with errors but removes the lock and the pidfile.

Scripts

We will need two scripts. The first one executes a command as unprivileged user and notes the PID in the pid file. The second is a normal startup script. We have to split the code into two files as the execution of Java programs does not provide a out of the box mechanism to start with root and then to downgrade privileges. You can do this with the 'commons daemon', see the link in further readings.

But as we are not using it, we use the bash command **runuser** which executes a subshell under the specified user. Our main shell script has no mean to get the PID of a subshell background process. Therefore the command passed to **runuser** must note the PID. As a consequence we cannot call java directly with **runuser** but need a simple script which is called from **runuser** and calls Java.and notes the PID in the pid file.

This approach has a limitation as well. The script starting Tomcat is already running as unprivileged user. As a consequence Tomcat cannot listen to a privileged port which are ports below 1024. You cannot have Tomcat listen on port 80 for example. Either start Tomcat as root, which I would not recommend or have a look at the Tanuki service wrapper or alternatively the commons daemon. See further readings below.

tomcatRunner

The script tomcatRunner is provided with the source download. Apart from some sanity checks it has only three lines being important:

```
# run command
$cmd $@ >> "$logfile"  2>&1 &
```

```
pid=$!
# write pid file
echo "$pid" > "$pidfile"
```

It executes the command as background process and writes the PID to the PID file.

Init script

The script includes the *functions* bash library which is used by most CentOS scripts. It provides the basic start, stop methods and in addition a status, restart and version method.

The status tells you, if the process is running or not. The version method tells you about which version of Java and Tomcat you are using.

Things you might change

There is a documented 'frequently changed settings' area right at the beginning of the main file. Here you can define your Tomcat directories and how much memory the Tomcat should consume.

The stop script currently waits for about 3 seconds before it shoots away the Tomcat with kill -9. You might adapt this to your need. If you like to wait for 6 seconds just change the -d option.

```
killproc -p $pidfile -d 6 $servicename
```

Tomcat cluster

If you have multiple Tomcats you normally reuse the same Tomcat installation for all Tomcats. In this case the CATALINA_HOME points to the installation whereas CATALINA_BASE points to the configuration and webapps folder of one instance.

You might use the variable **servicename** to specify the CATALINA_BASE directory.

```
CATALINA_BASE="$CATALINA_HOME/$servicename"
```

Further readings

/usr/share/doc/initscripts-8.45.19.EL/sysvinitfiles - Introduction on writing scripts

http://wrapper.tanukisoftware.org – a service wrapper which allows to restart Tomcat automatically in case of out of of memory errors
http://tille.garrels.be/training/bash/ - a very helpful guide when writing bash scripts

http://commons.apache.org/daemon/ and http://tomcat.apache.org/tomcat-6.0-doc/setup.html in case you want to start Tomcat as unprivileged user using privileged ports, for example port 80

Copyright and disclaimer