Web Server HOWTO

Because Spk provides its services via the World Wide Web, an important component of the overall system is the web server. This document describes the installation and configuration of web server software to support Spk.

To complete the configuration for secure https connections, you will need two authentication certificates. If you do not have these certificates, it might be a good idea to obtain them before proceeding with the web server configuration. See the Authentication and Encryption HOWTO [1] for details.

Table of Contents

Introduction **1**
Installing Tomcat **1**
Defining Environment Variables **2**
Creating Two Run-Time Instances **2**
Configuring User Groups and Permissions **3**
Configuring the Instances **3**
Setting the Production Instance of Tomcat to Run Automatically **6**
Setting the Test Instance of Tomcat to Run Automatically **6**
Installing Authentication Certificates **7**
Verifying the Configuration **7**
Stopping and Starting **8**

Introduction

The client component of Spk, called the Model Design Agent (MDA), is written in the Java language. It communicates with the rest of the system via the Internet, using the *http* and *https* protocols. On the server side of this communication is a web server, which is software that supports World Wide Web functionality and runs continuously in a server machine. The fact that the MDA is a Java application makes it very desirable that the components that communicate with the MDA from the server side be also written in Java. Server-side Java components such as those in Spk are known as *Java servlets*.

In order to run servlets within a web server, software known as a *servlet container* is required. Tomcat, a free product of the Catalina project, which in turn is part of the Jakarta project sponsored by Apache, the leading developer of web server software, serves this purpose for Spk. Tomcat can be added to an Apache web server, in which case it provides only the servlet container function, or it can be installed as a complete stand-alone web server and servlet container. For the purposes of Spk, the latter is the simplest solution.

The rest of this document describes the installation and configuration of tomcat as the web-server and servlet container for Spk.

Installing Tomcat

In a terminal window, as an ordinary user, download the tomcat tarball from whitechuck, then **su** to become *root*:

```
scp 'whitechuck:/opt/download/jakarta-tomcat*.tar.gz' .
su
```

If you do not already have a directory called `/usr/local/tomcat`, create one:

```
mkdir /usr/local/tomcat
```

Move the tarball to `/usr/local/tomcat`, and expand it:

```
mv jakarta-tomcat*.tar.gz /usr/local/tomcat
cd /usr/local/tomcat
tar xvzf *
```

Defining Environment Variables

It is convenient to define an environment variable, CATALINA_HOME, which contains the path to the directory in which tomcat is installed. Edit your `~/.bash_profile` file, adding lines to define CATALINA_HOME and to export it.

For example, suppose that the tarball expanded to `jakarta-tomcat-5.0.18`. Then you would add these lines to `~/.bash_profile`:

```
CATALINA_HOME=/usr/local/tomcat/jakarta-tomcat-5.0.18
export CATALINA_HOME
```

After saving your modified `~/.bash_profile`, restart your desktop by logging out and then logging back in again.

Creating Two Run-Time Instances

In this section, we will set things up so that two independent instances of tomcat can run simultaneously. Onc instance will be for test and the other for production.

When changes are made to the web site or to the MDA, the process used to migrate these changes to production is as follows:

1. Copy the changes from the workstation(s) of software developers, content authors and content editors to the test to the file hierarchy associated with the test instance.
2. Test and proof read the changes.
3. Copy the changes to the file hierarchy associated with the production instance.

The following steps will create the file hierarchies for the two instances:

```
su
cd $CATALINA_HOME/..
mkdir instance
cd instance
mkdir test
mkdir prod
cp -r $CATALINA_HOME/conf test
cp -r $CATALINA_HOME/conf prod
```

```
cd test
mkdir logs temp webapps work
cd ../prod
mkdir logs temp webapps work
```

Configuring User Groups and Permissions

It is most convenient for tomcat to be started automatically by the *init* process when the system boots. Because the init process is owned by root, tomcat also runs as root, through inheiritance. When we tried to run tomcat under an ordinary user name, we could not get it to stop and restart reliably. Running tomcat as apache is run, using an ordinary user name for security reasons, may not be worth the trouble. Unlike apache, which is written in C, tomcat is written in Java. Java applications are not vulnerable to buffer overflows, which are the vulnerability underlying most security breaches in conventional web servers.

Although root is the tomcat process owner, it is preferable for its files to belong to a user or users other than root. That way, users can be given the authority to modify these files without having to know the root password.

First we create a user and a group. The user *tomcat* will be the owner of the files, while the group *webadmin* will have group access to them.

```
su
/usr/sbin/useradd -r tomcat
/usr/sbin/groupadd -r webadmin
/usr/bin/gpasswd -a tomcat webadmin
```

At this point, you should also add *your user name* to the webadmin group:

```
/usr/bin/gpasswd -a your-user-name webadmin
```

Now set permissions on the file hierarchies for the instances:

```
cd $CATALINA_HOME/../instance
chown -R tomcat.webadmin *
find . -type d -exec chmod 2775 {} \;
find . -type f -exec chmod g+rw {} \;
```

Configuring the Instances

We need to make changes to `server.xml`, which is the principal configuration file for tomcat.

Changing the Shutdown Port for the Production Instance

In the default configuration, tomcat receives shutdown requests on port 8005. We will leave 8005 as the shutdown value for the test server, but change that of the production server to 8006, to avoid a clash.

Before editing, `$CATALINA_HOME/../instance/prod/conf/server.xml` will look like this:

```
<Server port="8005" shutdown="SHUTDOWN" debug="0">
```

Change the port, so that after editing the element will look like this:

```
<Server port="8006" shutdown="SHUTDOWN" debug="0">
```

Changing the HTTP Port and the Redirect Port for the Production Instance

For the production server, only, we will change the port for receiving http requests from 8080 to 80, and the port to which https requests are redirected, from 8443 to 443.

Before editing, `$CATALINA_HOME/../instance/prod/conf/server.xml` will look like this:

```
<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8080 -->
<Connector port="8080"
   maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
   enableLookups="false" redirectPort="8443" acceptCount="100"
   debug="0" connectionTimeout="20000"
   disableUploadTimeout="true" />
```

After editing it should look like this:

```
  <!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 80 -->
  <Connector port="80"
   maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
   enableLookups="false" redirectPort="443" acceptCount="100"
   debug="0" connectionTimeout="20000"
   disableUploadTimeout="true" />
```

Note that both the *port* and the *redirectPort* attributes must be changed.

Setting Up HTTPS for the Test Instance

In the default configuration, the https port is disabled, because the xml element that defines it has been commented out. We will restore the https port by removing the xml comment brackets. We will also tell tomcat where to find our authentication certificates and what password it needs to decrypt them.

Before editing, `$CATALINA_HOME/../instance/test/conf/server.xml` will look like this:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<!--
<Connector port="8443"
   maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
   enableLookups="false" disableUploadTimeout="true"
   acceptCount="100" debug="0" scheme="https" secure="true"
   clientAuth="false" sslProtocol="TLS" />
-->
```

AFter editing out the comment brackets, it will look like this:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector port="8443"
   maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
   enableLookups="false" disableUploadTimeout="true"
   acceptCount="100" debug="0" scheme="https" secure="true"
   keystoreFile="/root/.keystore" keystorePass="Make@!This&*Good"
   clientAuth="false" sslProtocol="TLS" />
```

Note that the password must be the one that you used when creating your key pair (see the Authentication and Encryption Howto [2]).

Setting Up HTTPS for the Production Instance

For the production server, only, we will change the port for processing https requests from 8443 to 443, and add the attributes for authentication:

Before editing, `$CATALINA_HOME/../instance/prod/conf/server.xml` will look like this:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<!--
<Connector port="8443"
   maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
   enableLookups="false" disableUploadTimeout="true"
   acceptCount="100" debug="0" scheme="https" secure="true"
   clientAuth="false" sslProtocol="TLS" />
-->
```

After editing, this is how it looks:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 443 -->
<Connector port="443"
   maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
   enableLookups="false" disableUploadTimeout="true"
   acceptCount="100" debug="0" scheme="https" secure="true"
   keystoreFile="/root/.keystore" keystorePass="Make@!This&*Good"
   clientAuth="false" sslProtocol="TLS" />
```

Note that the *port* attribute has been changed and that the comment brackets around the element have been removed. Note also the addition of the *keystoreFile* and the *keystorePass* attributes.

Disabling the Apache Connection for both Test and Production

For both the production and the test servers, we will comment out the element in `server.xml` which defines a connector between tomcat and apache, because we are using tomcat as a complete stand-alone server that does not depend on apache.

Edit both `$CATALINA_HOME/../instance/test/conf/server.xml` and `$CATALINA_HOME/../instance/prod/conf/server.xml`.

Before editing, both files will look like this:

```
<!-- Define a Coyote/JK2 AJP 1.3 Connector on port 8009 -->
<Connector port="8009"
   enableLookups="false" redirectPort="8443" debug="0"
   protocol="AJP/1.3" />
```

afterward, both files should look like this:

```
<!-- Define a Coyote/JK2 AJP 1.3 Connector on port 8009 -->
<!--
<Connector port="8009"
   enableLookups="false" redirectPort="8443" debug="0"
   protocol="AJP/1.3" />
-->
```

Setting the Production Instance of Tomcat to Run Automatically

Tomcat should be run automatically from the **init** process whenever the system boots, and be capable of being stopped or restarted manually from the command line by the *root* user, just like other system services.

We need to create a script called **tomcat** and place it in the standard location for scripts that start and stop services.

```
su
cd /etc/rc.d/init.d
emacs tomcat &
```

Cut and past the following lines into the editing window and save.

```
#!/bin/bash
#
# Init file for production tomcat
#
# chkconfig: 2345 63 37
# description: tomcat server for ports 80 and 443
#
CATALINA_HOME=/usr/local/tomcat/jakarta-tomcat-5.0.18
CATALINA_BASE=$CATALINA_HOME/../instance/prod
export CATALINA_HOME CATALINA_BASE

exec $CATALINA_HOME/bin/catalina.sh $*
```

Still as *root* and in the same directory, install your new shell script:

```
chmod +x tomcat
/sbin/chkconfig --add tomcat
```

Setting the Test Instance of Tomcat to Run Automatically

We also need to create a script called **tomtest** and place it in the standard location for scripts that start and stop services.

```
su
cd /etc/rc.d/init.d
emacs tomtest &
```

Cut and past the following lines into the editing window and save.

```
#!/bin/bash
#
# Init file for production tomcat
#
# chkconfig: 2345 63 37
# description: tomcat server for ports 8080 and 8443
#
CATALINA_HOME=/usr/local/tomcat/jakarta-tomcat-5.0.18
CATALINA_BASE=$CATALINA_HOME/../instance/test
export CATALINA_HOME CATALINA_BASE

exec $CATALINA_HOME/bin/catalina.sh $*
```

Still as *root* and in the same directory, install your new shell script:

```
chmod +x tomtest
/sbin/chkconfig --add tomtest
```

Installing Authentication Certificates

If you have not obtained the certificates that tomcat will need for https connections, get them by following the instructions in the Authentication and Encryption Howto [3]. If you obtained the certificates on a workstation rather than on the web server, transfer a copy of the `cert` directory from the ~root directory on the workstation to the ~root directory on the server. To install the the certificates, do the following:

```
su
cd cert
$JAVA_HOME/bin/keytool -import -alias root -keystore keystore -trustcacerts -file uw_rc
$JAVA_HOME/bin/keytool -import -alias tomcat -keystore keystore -trustcacerts -file spk
```

Verifying the Configuration

Tomcat provides a number of test and demonstrations which we will now copy in to the test and production instances.

As an *ordinary* user, member of the *webadmin* group:

```
cd $CATALINA_HOME/../instance
cp -r $CATALINA_HOME/webapps/* test/webapps
cp -r $CATALINA_HOME/webapps/* prod/webapps
```

Now you should be ready to start tomcat manually, and see if it is working.

```
su
/etc/rc.d/init.d/tomcat  start
/etc/rc.d/init.d/tomtest start
```

should get both instances of tomcat running. You can check that that they are running by using the **ps** command:

```
ps -ef | grep tomcat
```

should list two tomcat processes in the system run queue.

There are four different ports which, when requested via your browser, should return the tomcat welcome page. The ports are 80 and 443, for the production server; 8080 and 8443 for the test server. URLs are provided for the two following cases:

i. Your workstation is on the LAN, behind the firewall:

http://192.168.2.2/ [4]

https://192.168.2.2/ [5]

http://192.168.2.2:8080/ [6]

https://192.168.2.2:8443/ [7]

ii. Your workstation is on the public internet, outside the firewall:

http://spk.rfpk.washington.edu/ [8]

https://spk.rfpk.washington.edu/ [9]

http://spk.rfpk.washington.edu:8080/ [10]

https://spk.rfpk.washington.edu:8443/ [11]

Stopping and Starting

Tomcat is a daemon. It will run until the *root* user tells it to stop, or until the computer is rebooted.

To tell the production tomcat to stop (as root):

```
/etc/rc.d/init.d/tomcat stop
```

Similarly, to tell the test tomcat to stop:

```
/etc/rc.d/init.d/tomtest stop
```

Important! After issuing one or the other of the above commands, do not try to start that tomcat again, until you are certain that it has terminated. Use the command

```
ps -ef | grep tomcat
```

to verify this.

Notes

1. http://whitechuck.rfpk.washington.edu/soft/howto/rhel3/authentication/authentication.html
2. http://whitechuck.rfpk.washington.edu/soft/howto/rhel3/authentication/authentication.html
3. http://whitechuck.rfpk.washington.edu/soft/howto/rhel3/authentication/authentication.html
4. http://192.168.2.2/
5. https://192.168.2.2/
6. http://192.168.2.2:8080/
7. https://192.168.2.2:8443/
8. http://spk.rfpk.washington.edu/
9. https://spk.rfpk.washington.edu/
10. http://spk.rfpk.washington.edu:8080/
11. https://spk.rfpk.washington.edu:8443/