TomcatDevelopmentVirtualHosts

Tomcat Virtual Hosts Development Environment

Introduction

This document describes a clean way to create Tomcat virtual hosts in a development environment. This setup will not work as is for a production environment, since the Tomcat virtual hosts described by this document will not be visible except on the local host. However, this document can be used as a basis for creating a production Tomcat virtual host environment.

This document is based on a response I originally wrote concerning logging issues and Tomcat virtual hosts.

Environments

I use primarily the following two environments when developing concepts and testing ideas.

Component	Version
OS	Fedora 11 32 bit
JDK/JRE	1.6.0_20
Tomcat	6.0.26
IDE	NetBeans 6.8
Spring	2.5 (provided with NetBeans)
OS	Windows/XP Professional SP 4 32 bit
JDK/JRE	1.6.0_20
Tomcat	6.0.26
IDE	NetBeans 6.8
Spring	2.5 (provided with NetBeans)

I am reworking the Spring Developer's Notebook bike store example so I can understand Spring testing and the use of mock objects. I've also added logging to this application. Logging is provided by Apache's commons-logging, since Spring ships with the commons-logging package. The back end logging provider is Apache's log4j.

I've run into a lot of "interesting" issues, mostly having to do with Spring test libraries versus JUnit versus mock objects. However, that's a topic for another discussion.

Directory Structure

In order to create virtual hosts, do the following:

1. Create a separate directory for each host **outside of $CATALINA_HOME/webapps**
2. Underneath each directory, create a webapps directory.

For example, here is a sample directory structure for hosts foo and bar.

Windows Environment

Tomcat is located in C:\Apache\apache-tomcat-6.0.26. While recent versions of Tomcat and Apache HTTPD manage spaces in directory names, I prefer to have my services live outside of C:\Program Files.

Hostname	Location
localhost	C:\Apache\apache-tomcat-6.0.26\webapps
foo	C:\Apache\hosts\foo-host\webapps
bar	C:\Apache\hosts\bar-host\webapps

Linux Environment

Tomcat is located in ~/Apache/apache-tomcat-6.0.26. This is a development copy and lives in my home directory. I can easily start, stop, deploy, undeploy, and make changes to this Tomcat server both from the command line and from within my development environment.

Hostname	Location
localhost	~/Apache/apache-tomcat-6.0.26/webapps
foo	~Apache/hosts/foo-host/webapps
bar	~Apache/hosts/bar-host/webapps

Environment Notes

Putting the virtual host directory structure outside of the Tomcat installation provides the following advantages..

- When Tomcat is upgraded, virtual hosts are not a concern
- A pristine Tomcat environment is preserved, which can be useful for debugging
- Creating new virtual hosts is easy, and does not disturb a running Tomcat

Creating a hosts/<host-name>/webapps structure provides several advantages as well.

- All the hosts are located in one directory, making it easy to move, change permissions, or otherwise manage a collection of virtual hosts
- Provides a consistent and easily understood naming convention
- the hosts/<host-name> location provides a good place to locate cross-context web applications such as Solr

Tomcat Configuration

Server.xml

To create virtual hosts, I just copied the original locahost entry for the foo and bar hosts. I changed the name and appBase values to fit the targeted hosts. The resulting files (Windows and Linux) are shown below. Adjust those locations to reflect your directory structure.

Server.xml fragment for Windows

```
<Host name="localhost"  appBase="webapps"
  unpackWARs="true" autoDeploy="true"
  xmlValidation="false" xmlNamespaceAware="false">
</Host>

<Host name="foo"  appBase="C:/Apache/hosts/foo-host/webapps"
  unpackWARs="true" autoDeploy="true"
  xmlValidation="false" xmlNamespaceAware="false">
</Host>

<Host name="bar"  appBase="C:/Apache/hosts/bar-host/webapps"
  unpackWARs="true" autoDeploy="true"
  xmlValidation="false" xmlNamespaceAware="false">
</Host>
```

Server.xml fragment for Linux

```
<Host name="localhost"  appBase="webapps"
  unpackWARs="true" autoDeploy="true"
  xmlValidation="false" xmlNamespaceAware="false">
</Host>

<Host name="foo"  appBase="/home/mdeggers/Apache/hosts/foo-host/webapps"
  unpackWARs="true" autoDeploy="true"
  xmlValidation="false" xmlNamespaceAware="false">
</Host>

<Host name="bar"  appBase="/home/mdeggers/Apache/hosts/bar-host/webapps"
  unpackWARs="true" autoDeploy="true"
  xmlValidation="false" xmlNamespaceAware="false">
</Host>
```

Note that I didn't change any of the logging prefixes and suffixes, so all common logging gets mixed together. Since this is just a proof of concept, I don't really mind.

Application Setup

I like having both the default web page and the manager application available for each virtual host in a development environment. This provides me with a couple of advantages.

- Applications can be managed on a virtual host basis without copying war files around
- I know that the Tomcat virtual host is running even if I've broken the web application for that particular virtual host

To do this, use the following steps.

1. Copy manager.xml from $CATALINA_HOME/conf/Catalina/localhost to each of the virtual host configuration directories:
    a. For virtual host **foo**, this means **manager.xml** gets copied into $CATALINA_HOME/conf/Catalina/**foo**
    b. For virtual host **bar**, this means **manager.xml** gets copied into $CATALINA_HOME/conf/Catalina/**bar**
2. Copy the ROOT application from $CATALINA_HOME/webapps to each of the virtual host webapp directories
    a. For virtual host **foo**, this means copying **ROOT** to **hosts/foo-host/webapps** (full path depends on actual location to match the **server.xml entry**)
    b. For virtual host **bar**, this means copying **ROOT** to **hosts/bar-host/webapps** (full path depends on actual location to match the **server.xml entry**)
3. Copy the manager application from $CATALINA_HOME/webapps to each of the virtual host webapp directories
    a. For virtual host **foo**, this means copying **manager** to **hosts/foo-host/webapps** (full path depends on actual location to match the **server.xml entry**)
    b. For virtual host **bar**, this means copying **manager** to **hosts/bar-host/webapps** (full path depends on actual location to match the **server.xml entry**)

Network Setup

In order to browse to **http://foo:<port>/<application>** and **http://bar:<port>/<application>**, hosts foo and bar have to be resolved to an IP address. The simplest way to accomplish this is to create entries for foo and bar in the appropriate hosts file and associate the entries with 127.0.0.1 (locahost). Note that by setting up the IP addresses in this fashion, **the virtual hosts will only be visible on your local machine**.

If these virtual hosts need to be visible from some other machine, then an externally visible IP address will have to be used (not 127.0.0.1) and an entry into the remote host machine's host file will have to be made. Alternatively, the IP address and fully qualified domain name will have to be entered into a domain name server. Explaining these options are beyond the scope of this document.

Windows Network Setup for Virtual Hosts

Edit the hosts file for Windows. This is normally found in %windir%\system32\drivers\etc. You will need administrator rights in order to do this. For my machine this is C:\WINNT\system32\drivers\etc since this machine was upgraded from Windows/2000 Professional.

Add a line for each of your virtual hosts. The line looks like the following:

```
127.0.0.1    <hostname>
```

For the example above, there are two lines which read:

```
127.0.0.1    foo
127.0.0.1    bar
```

Linux Network Setup for Virtual Hosts

The hosts file for Linux is located in /etc. You will need root permission in order to do this. The entries are the same as those for the Windows hosts file.

There is an additional consideration for Linux. Linux has a file called /etc/nsswitch.conf which controls how various name resolution operations are performed. Make sure that for hosts, the files directive is present. For example:

```
hosts:dns files
```

If you do not use DNS, then obviously do not have the dns entry present. The important entry is files, so that Linux will (eventually) look at /etc/hosts for name to IP address resolution.

Results and Testing

Results

Upon starting Tomcat, all three hosts should be visible at localhost:8080, foo:8080, and bar:8080 (if you have not changed the default connector port). This is where having the default ROOT application is handy. If you do not see the Tomcat home page for each of the virtual hosts, then something is wrong with the setup.

The manager application should be available for each virtual host as well. The passwords will all be the same, since the web.xml is pointing to the same global resource reference. Having different passwords for each virtual host can be accomplised by pointing either to different global resources or by pointing to a suitable entry in virtual host's manager.xml file. Both changes are beyond the scope of this document.

Log Testing

Application Notes

In order to test that each virtual host was acting independently, I made use of the Spring Developer's Notebook bike shop application. I added some logging via commons-logging to BikeValidator.java. The exact syntax isn't important. What is important is that I added logging statements for several different failed validations. I could then cause a specific validation failure on a particular virtual host. By looking at the log files, I could make sure that the validation error was only logged on the virtual host where it occured.

To accomplish this, I created three war files, one for each virtual host. The only change that was made was to the log4j.properties file. The complete log4j.properties file for the localhost virtual host is shown below.

```
### direct messages to file dnb-02.log ###
log4j.appender.file=org.apache.log4j.FileAppender log4j.appender.file.File=${catalina.home}/logs/dnb-02.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout log4j.appender.file.layout.ConversionPattern=%d
{ABSOLUTE} %5p %c{1}:%L - %m%n

### set log levels - for more verbose logging change 'info' to 'debug' ###
log4j.rootLogger=warn, file

### Spring Framework logging
log4j.logger.org.springframework=info
```

Important issues to note:

- Change the log4j.appder.file name for each log4j.properties file. I used foo-dnb-02.log and bar-dnb-02.log for virtual hosts foo and bar.
- **DO NOT PUT commons-logging or log4j jars in $CATALINA_HOME/lib.**

Test Results

After starting Tomcat with three virtual hosts (localhost, foo, and bar), I caused specific warnings to occur by making invalid entries for editing a bike (no serial number, frame too small, weight too light, etc.). To verify that the virtual hosts were working as expected, I verified the following:

- Three log files were created - dnb-02.log, foo-dnb-02.log, and bar-dnb-02.log
- Time stamps in the log files were consistent with browser access
- No duplicate Spring info messages were logged
- Warning messages from the application were logged
    - Warning messages were in the appropriate files
    - Warning messages had time stamps consistent with the warning

Conclusion

This document demonstrates a simple way to set up virtual hosts in a development environment. The resulting setup works as expected, maintaining separate management and application logging. Virtual hosts are easily added, modified, enabled, or disabled. Tomcat upgrades can occur without impacting the virtual host structure.