

# Configuring a Mac Intel Development Environment with multiple ColdFusion instances

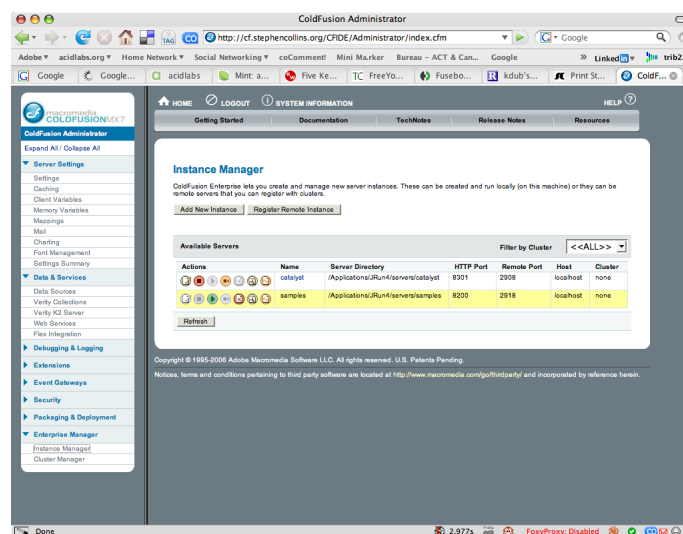
This document assumes several things:

1. You have already configured your Intel Mac as a functioning development workstation according to the *CFMX on Intel Macs - The Definitive Guide* (<http://webmages.com/geek/cfm-x-on-intel-macs>).
2. That your development workstation is running on the built-in or some other version of Apache httpd server and *not* on the built-in JRun web server.
3. You are at least comfortable with using tools such as Terminal (and Unix commands), Lingon and text editing tools. In this document I use the excellent SubEthaEdit (<http://www.codingmonkeys.de/subethaedit/>) for reasons that will become clear.
4. You understand and can cope with the concepts behind using `sudo` on the command line and editing files owned by root.

## Create the server instance

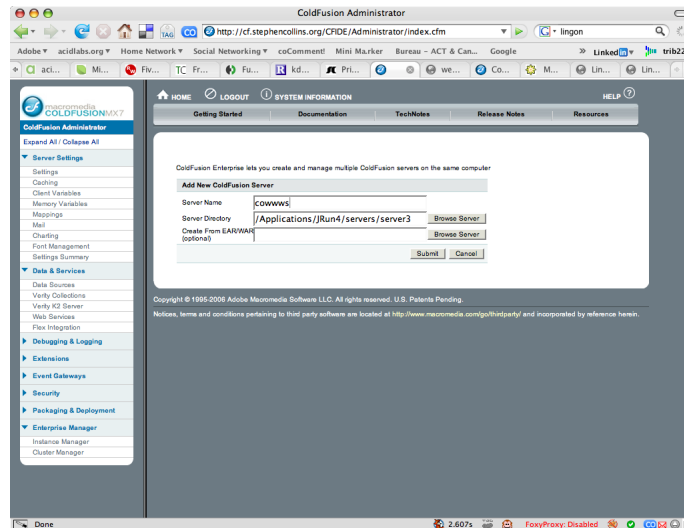
The first thing you need to do is to create the server instance.

1. Open a web browser.
2. Log into your master ColdFusion Administrator. This is the ColdFusion server instance you created when you installed ColdFusion for the first time. It's probably at <http://localhost/CFIDE/administrator/index.cfm>.
3. Open the **Instance Manager**.



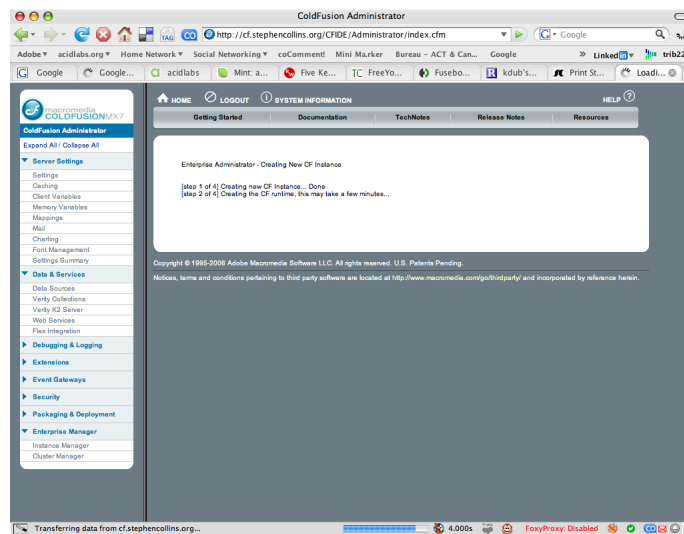
4. Create a new server instance by clicking on the **Add New Instance** button.

5. Complete the form by filling in appropriate details for your new server instance. For the purposes of this document, and ACME environments in general, you need only complete the first field, **Server Name**.

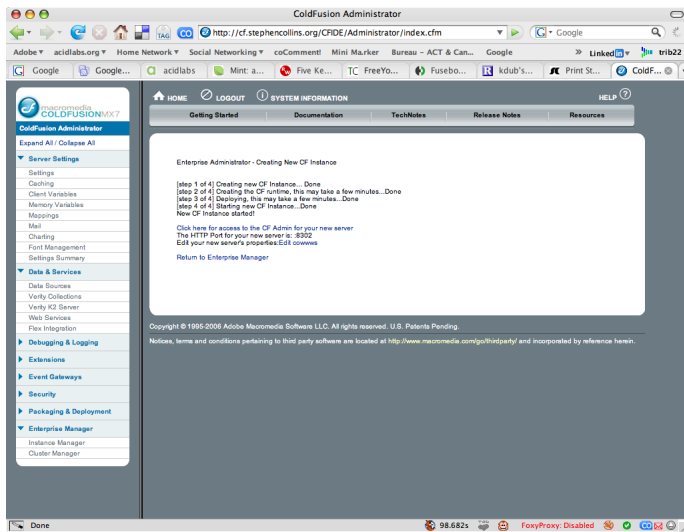


Choose a name for your new server instance, and enter it here. From this point forward, we'll use the alias `{instancename}`. I've used, `cwwwws`, although you can use anything you like (as you'll see throughout the screen shots, it just happened to be the name of the client environment I was setting up when I took the shots).

6. Click the **Submit** button.
7. Wait as the server is built and deployed.



- When the instance is built, the ColdFusion Administrator will display a set of messages offering to allow you to administer the server. Don't try to do this at the moment, as since we're using an ACME build, and shouldn't have the default JRun web server installed, it won't work.



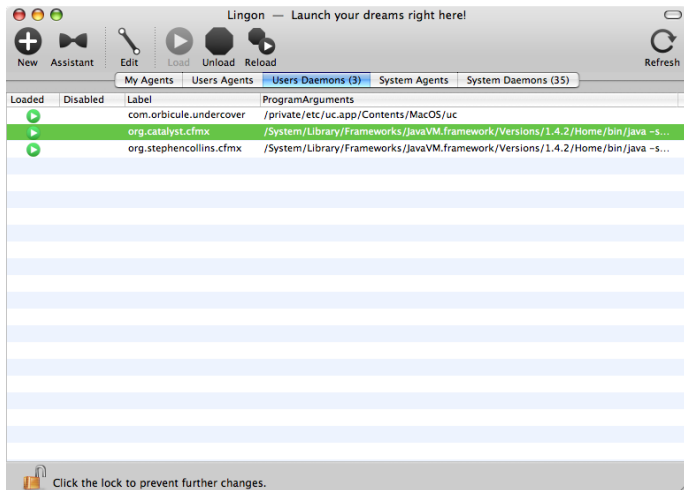
- Minimise the web browser. We'll come back to it later.

Your server is now built, and works. You just need to complete a few more tasks to get it to a state where you can actually use it.

## Add the server instance to Lingon

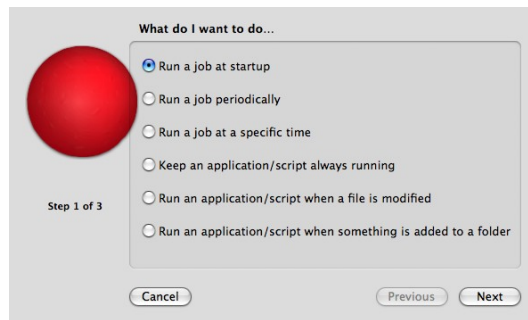
Mark Andrachek's *Definitive Guide* shows us how to use Lingon (<http://lingon.sourceforge.net/>) to configure and control services in OS X. The new ColdFusion instance we just created now needs to be added to Lingon, just as we did when we first set up ColdFusion.

- Open Lingon.



- Using the **Assistant**, add the new server instance as a service. This is exactly the same procedure as in the *Definitive Guide*.

Select **Run a job at startup** and click **Next**.

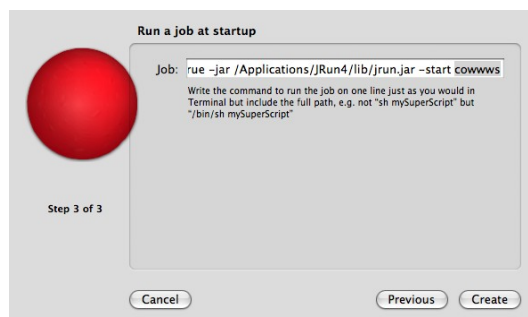


- Give the service a label. Again, it doesn't matter what you use, but some sort of sensible naming system is always a good idea. I use a system that uses `org.{instancename}.cfmx`. So, in this case, `org.cowwws.cfm`. Enter the label you chose in the **Label** box and click **Next**.



- In the **Job** box, enter or copy/paste the following string:

```
/System/Library/Frameworks/JavaVM.framework/Versions/1.4.2/Home/bin/java
-server -Djava.awt.headless=true -jar /Applications/JRun4/lib/jrun.jar -start
{instancename}
```



- Replace `{instancename}` with the actual instance name you chose. Click **Create**.

- Minimise Lingon. We're going to need it again a little later.

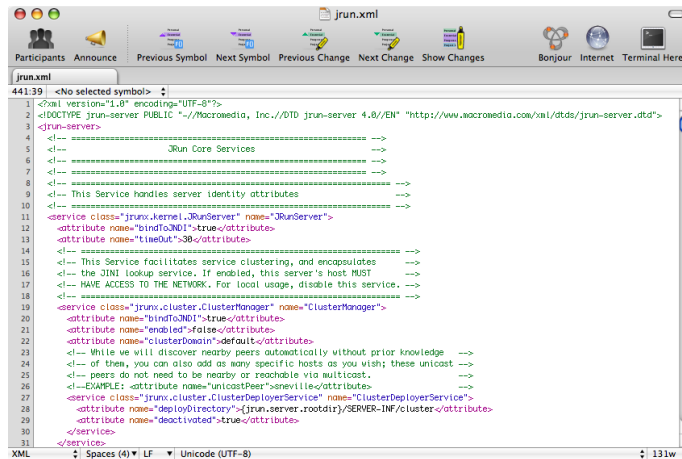
Your service to be managed by Lingon is now properly set up. You can start and stop it as you see fit. There are still a few more steps to complete before we can use the new server, however.

## Edit jrun.xml

Now, we need to edit the `jrun.xml` for the new server instance we've created. This process will allow us to tie its use to Apache server and to configure the Apache `httpd.conf` to use the server instance (coming up in a little bit).

- Open Finder.

2. Browse to `/Applications/JRun4/servers/{instancename}/SERVER-INF/` and open `jrun.xml` in SubEthaEdit (or whatever text editor you use).



3. At line 408, there is a section that begins with the line:

```
<service class="jrun.servlet.http.WebService" name="WebService">
```

Add the following line after line 411:

```
<attribute name="deactivated">true</attribute>
```

This deactivates the server instance's connection to the built-in JRun web server.

4. At line 438, there is a section that begins with the line:

```
<service class="jrun.servlet.jrpp.JRunProxyService" name="ProxyService">
```

Change line 441 to read:

```
<attribute name="deactivated">>false</attribute>
```

This activates the server instance's connection to Apache.

5. Save and close `jrun.xml`. You should note that this file is owned by root and can't normally be edited by a user. SubEthaEdit is smart, however. In such cases, both when editing and saving, SubEthaEdit allows you to enter the password for an admin user to allow the action. This is pretty much the same as editing on the command line using `sudo`.

We've now finished with `jrun.xml`, except for one thing. Make sure you note down the number on line 445. This is the JRun port number that the server is tied to and we will need it when we edit `httpd.conf` a little later.

## Configure the JRun server instance

We now need to create the config files for the server instance to use when running. I've heard that on some environments, these actually get created when making a new server instance, but not on my machine. As such, the steps are here for you to follow.

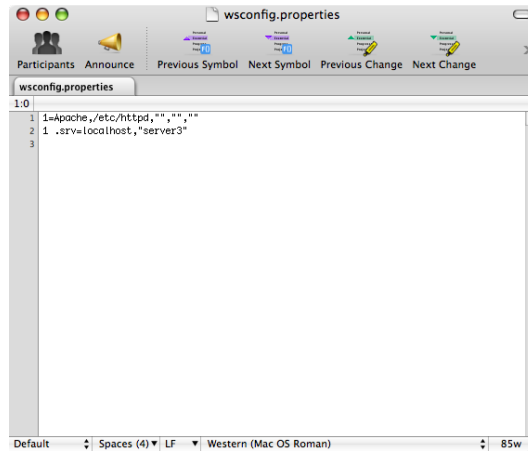
1. Open Terminal.
2. `cd` to `/Applications/JRun4/lib/wsconfig`. Enter `ls` to get a listing of the directories here.
3. Enter `sudo mkdir {nextnumber}`, where `{nextnumber}` is the next number greater than the directory number that is here. This isn't very clear... is it? So, let's make it easy. If the directory listing reveals only a directory with the named 1, create a directory named 2, and so on.
4. Copy the files from the 1 directory to your new directory by entering `sudo cp 1/* {nextnumber}`.
5. Go into the new directory by entering `cd {nextnumber}`.
6. Open `wsconfig.properties` in SubEthaEdit (you didn't close it, did you?).

7. Change the line that reads:

```
1 .srv=localhost,"server1"
```

to read

```
1 .srv=localhost,"server{nextnumber}"
```



8. Save and close `wsconfig.properties`.

9. Open `jrun.serverstore` in SubEthaEdit.

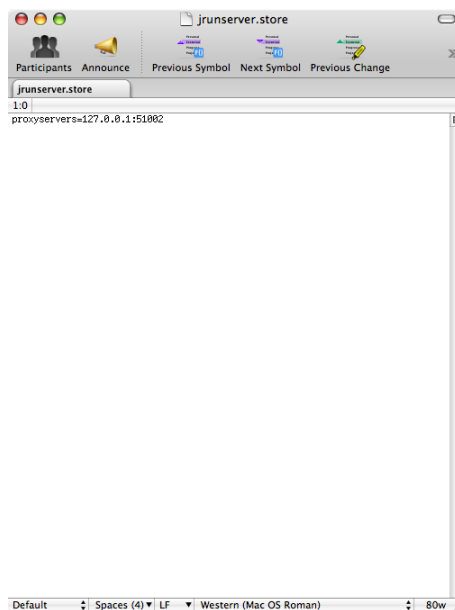
10. Change the line that reads:

```
proxyservers=127.0.0.1:51020
```

to read

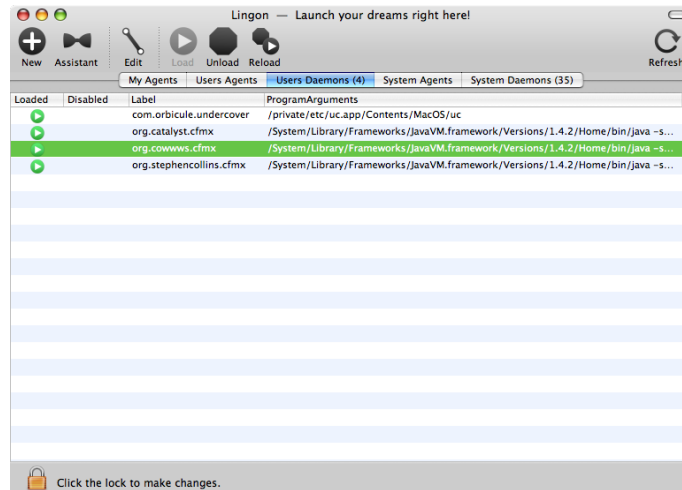
```
proxyservers=127.0.0.1:{numberfromjrun.xml}
```

where `{numberfromjrun.xml}` is the number you noted down in the previous section.



11. Save and close `jrun.serverstore`.

12. Go to Lingon and reload the service.



13. Minimise or quit Lingon. I keep Lingon open and minimised whenever I'm developing with ColdFusion. You just never know when you might do something foolish and need to restart a ColdFusion instance.

## Modify httpd.conf

httpd.conf, as we know, is the file that Apache uses to manage the configuration of the httpd server. I have a fairly granular setup for httpd.conf, with a series of included files, which is principally due to my being anal. By no means should you follow what I do here unless it rocks your world in the (nerdy and definitely scary) way it rocks mine. Here is my setup:

- » main httpd.conf in /etc/httpd with no modifications;
- » a user file, /private/etc/httpd/users/trib\_cf.conf that is an alias (ln -s) for a file I keep in /Users/trib/Sites/httpd. I keep all my additional httpd includes in this directory;
- » /Users/trib/Sites/httpd/trib\_cf.conf, which contains general ColdFusion configuration information, and uses an Include statement to include all files in /Users/trib/Sites/httpd/clientconfig, and;
- » files in /Users/trib/Sites/httpd/clientconfig containing individual client additions to the httpd configuration.

I told you it was anal... There's *absolutely* no reason you can't just edit /etc/httpd/httpd.conf and do everything there. So, now you've been enlightened (or just frightened) by my httpd.conf setup, here's what you need to do. Let's assume you choose just to have a single .conf file; httpd.conf.

1. Open httpd.conf in SubEthaEdit.
2. Update the global JRun configuration to reflect multiple server instances. To do this, edit the JRun configuration block by commenting out the Serverstore and Bootstrap values so that it looks like the following:

```
# Global JRun Settings
LoadModule jrun_module libexec/httpd/mod_jrun.so
AddModule mod_jrun.c

<IfModule mod_jrun.c>
JRunConfig Verbose false
JRunConfig Apialloc false
JRunConfig Ssl false
JRunConfig Ignoresuffixmap false
#JRunConfig Serverstore /Applications/JRun4/lib/wsconfig/1/jrunserver.store
#JRunConfig Bootstrap 127.0.0.1:51020
#JRunConfig Errorurl <optionally redirect to this URL on errors>
#JRunConfig ProxyRetryInterval 600
#JRunConfig ConnectTimeout 15
#JRunConfig RecvTimeout 300
#JRunConfig SendTimeout 15
AddHandler jrun-handler .jsp .jws .cfm .cfml .cfc .cfr .cfswf

<IfModule mod_dir.c>
DirectoryIndex index.cfm index.php index.html
</IfModule>
</IfModule>
```

3. In each virtual server you want to use ColdFusion, add a block reflecting the ColdFusion server instance you want to work with. So, for instance, my localhost block reads:

```
# localhost
<VirtualHost *>
ServerAdmin webmaster@stephencollins.org
# DocumentRoot /Library/WebServer/Documents
DocumentRoot /Users/trib/Sites/localhost/htdocs
ServerName localhost
ErrorLog /Users/trib/Sites/logs/localhost-error_log
CustomLog /Users/trib/Sites/logs/localhost-access_log common

<IfModule mod_jrun.c>
Alias /CFIDE/ "/Applications/JRun4/servers/cfusion/cfusion-ear/cfusion-war/CFIDE/"
Alias /cfdocs/ "/Applications/JRun4/servers/cfusion/cfusion-ear/cfusion-war/cfdocs/"
JRunConfig Serverstore /Applications/JRun4/lib/wsconfig/1/jrunserver.store
JRunConfig Bootstrap 127.0.0.1:51020
</IfModule>
</VirtualHost>
```

localhost uses the default cfusion ColdFusion instance. However, for the purposes of this guide, the server instance we've created is added to a virtual server with the name cowwws.acidlabs.org, as shown:

```
# cowwws.acidlabs.org
<VirtualHost *>
ServerAdmin webmaster@stephencollins.org
DocumentRoot /Users/trib/Sites/cowwws.acidlabs.org/htdocs
ServerName cowwws.acidlabs.org
ErrorLog /Users/trib/Sites/logs/cowwws.acidlabs.org-error_log
CustomLog /Users/trib/Sites/logs/cowwws.acidlabs.org-access_log common

<IfModule mod_jrun.c>
Alias /CFIDE/ "/Applications/JRun4/servers/cowwws/cfusion.ear/cfusion.war/CFIDE/"
JRunConfig Serverstore /Applications/JRun4/lib/wsconfig/3/jrunserver.store
JRunConfig Bootstrap 127.0.0.1:51002
</IfModule>
</VirtualHost>
```

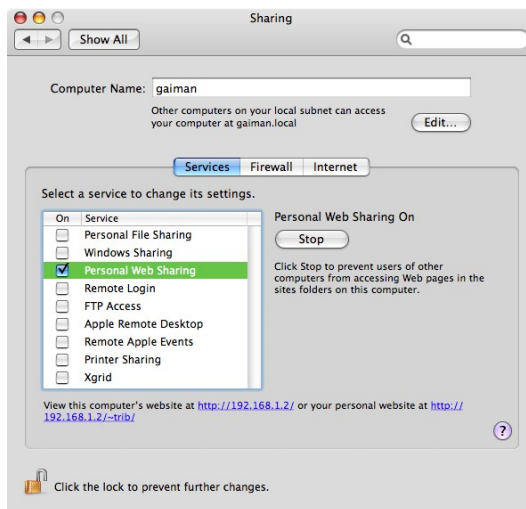
You can see that this virtual server uses the new ColdFusion server instance we have created and configured in all the previous sections above. Make sure you add this new virtual server to /etc/hosts.

4. Save and close httpd.conf and quit SubEthaEdit. You don't need to use it any more.

5. Open **System Preferences**.



6. Restart Apache through the **System Preferences > Sharing** panel.



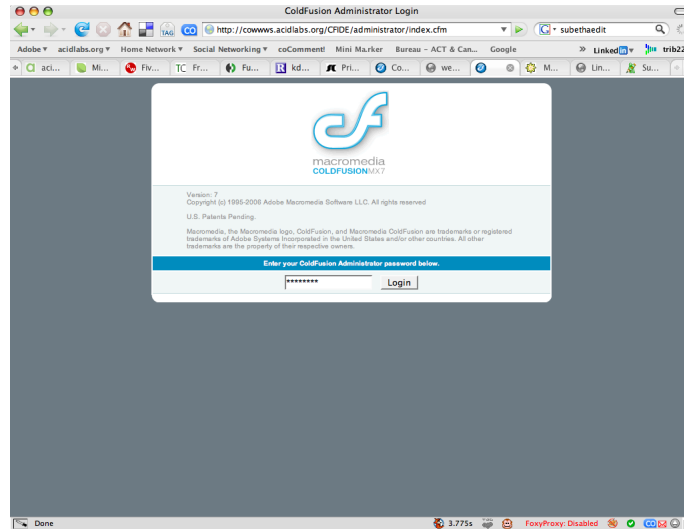
7. Close **System Preferences**.

That's it. You're done! Everything is ready to go.

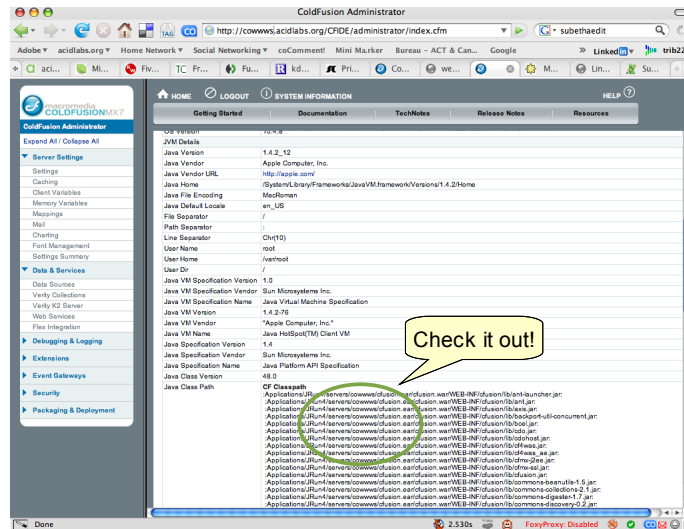
## Check the new server instance

Assuming you've done all the right things, you can now check that the server is working.

1. Open <http://cowwww.acidlabs.org/CFIDE/administrator/index.cfm> in a web browser. It will use the same password as your master ColdFusion server instance.



2. Have a look around. All the same settings you have for your master ColdFusion instance will be here. The first thing I usually do is remove any Mappings and Custom Tag paths I don't need.
3. Click on the System Information link in the top of the window. You will see the full configuration information for the new server instance. In particular, you will be able to see that this virtual server is using your new ColdFusion server instance!



## Conclusion

I consider the ability to be able to develop on multiple, differently configured ColdFusion servers critical to my work. Adobe (and Macromedia before them) have actually made it pretty easy to do this. There's absolutely no reason you shouldn't be using this technique in your own work.

Hopefully, the arrival of Scorpio sometime later in 2007 will bring full, native Mac Intel compatibility to the ColdFusion platform. The current situation with developers using Mac Intel (and the number is growing) is something of a pain in the backside. It certainly made me hesitate to do the install until others had done it. Mark Andrachek's *Definitive Guide* got me there. I consider this document to be an addendum to that material.

Have fun!