Building Apache, MySQL, and PHP on Mac OS X

By Lucas Marshall

Table of Contents

Introduction

Apple's newest operating system, Mac OS X, has been anticipated by Mac users for many years now. It's a fusion of a revised Mac OS, and a very fast and reliable BSD−based UNIX core.

The BSD core in Mac OS X allows for any program that is able to be compiled on Linux, BSD, or any other UNIX or UNIX derivative to be compiled and run in Mac OS X. The fusion of the Mac OS's ease of use with the power of UNIX has attracted much attention from the open source community, and many open source developers have worked to make sure their projects will work well on the new system.

Among the best open source programs are several that you may have read about here on DevShed: Apache, MySQL, and PHP. It just happens that all of these programs compile and run relatively painlessly on Mac OS X. This article details the steps for all these programs, and the few problems that crop up running them.

Prerequisites

This article assumes that you are a user with Administrator privileges and that the Mac OS X Developer Tools are installed. If the Developer Tools are not installed, they can be installed from the grey Developer Tools CD that came with your copy of Mac OS X.

To get everything you'll need to follow this article, open up a terminal. The Terminal program can be found in /Applications/Utilities and looks like this:

Type the following commands into your terminal (don't type the %s, they are just representative of the prompt).

```
% cd /tmp
% wget http://download.sourceforge.net/pub/mirrors/mysql/Downloads/MySQL-3.23/mysql-3.23.38.tar.gz
% wget http://httpd.apache.org/dist/httpd/apache_1.3.19.tar.gz
% wget http://www.php.net:8000/distributions/php-4.0.5.tar.gz
% wget
http://www.webdav.org/mod_dav/mod_dav-1.0.2-1.3.6.tar.gz
```

When you're done, don't close the Terminal window, you'll need it later. Now that you have everything, let's start getting things set up.

Making MySQL

Compiling MySQL is the most difficult part of our entire process. It doesn't just involve compiling and installing, but you also have to create a StartupItem for it so it will start whenever your Mac OS X box starts. You should compile it first so you'll have all the client libraries you'll need later for PHP.

First of all, you need to create a new user named mysql. The reason for having a user for a server process like this is security. When I have you actually compile MySQL later, I'll have you make sure it always runs under the mysql user, and then you'll be able to limit, via UNIX permissions, what files the process can modify.

To create a new user correctly, you'll need to get into NetInfo, the underlying database that keeps track of a lot of things in Mac OS X. To access NetInfo, you need to use the NetInfo Manager, which can be found in the Utilities folder that is in the Applications folder. It looks like this:

NetInfo is a very important part of Mac OS X, so there is quite a bit of security around it. Once you have NetInfo Manager open, the first thing you'll need to do is unlock it so you can make changes. Just click the lock in the lower left of the NetInfo Manager window. It looks like this:

A dialog box will come up asking for the username and password for an user with Administrator access.

Fill out the fields and click OK. The NetInfo manager should now be unlocked, and you'll be able to modify the NetInfo tree. The tree is displayed in a manner similar to the Browser view in the Finder. Click on users, then click on the new directory button.

This will create a new NetInfo directory. Use the New Property and New Value options in the Directory menu to create properties. Select New Property, set the name, select New Value, fill in the value, make sure that the property names and values match those of the directory in the screenshot below.

Once your NetInfo looks like mine, go ahead and click on one of the other directories and this dialog box will appear:

Click Save, then click Update this copy in the next dialog. You're now done with NetInfo Manager, so you can quit it.

You're not done yet, however. You still need to compile MySQL! You should still have the Terminal open from earlier. Type the following into it:

```
% gnutar zxf mysql−3.23.38.tar.gz
```

This will decompress the MySQL source into a directory in the current directory. Next, type the following:

```
% cd mysql−3.23.38
% ./configure −−enable−assembler −−with−mysqld−user=mysql
```

The −−enable−assembler flag tells the configure script that we want MySQL to be compiled using assembler code where possible − this has the advantage of speeding up our final MySQL server.

Note: Some people have reported problems compiling (the next step) when they have specified the −−enable−assembler flag. If it doesn't work for you, just re−run the configure script, omitting the −−enable−assembler flag.

The −−with−mysqld−user=mysql flag tells the configure script that we want the compiled MySQL to run under the mysql user we made earlier.

Now you compile.

```
% make
```

The "make" command will compile all the programs that are part of the MySQL distribution.

```
% sudo make install
```

The "make install" command installs the MySQL distribution, programs and libraries. Putting "sudo" before it runs it as root, the UNIX super−user. Sudo will ask you for your password (not root's).

```
% sudo /usr/local/bin/mysql_install_db
```

This will install the very important mysql permissions database.

```
% sudo chown −R mysql /usr/local/var
```

This command will change the owner of the newly installed MySQL databases to the mysql user.

```
% sudo /usr/local/bin/safe_mysqld r noshade size=1
color=#cccccc>
```

This will start MySQL. Now that MySQL is all installed, you may want to make a Mac OS X Startup Item for it so it will start every time the computer starts. Here's how you do it.

If there isn't a StartupItems folder in your Library folder, execute this command:

```
% mkdir /Library/StartupItems
```

It will make one. The reason we put it in Library and not System/Library is that everything in System/Library is reserved for use by the operating system only.

```
% mkdir /Library/StartupItems/MySQL
```

This will make a new directory for our MySQL startup item. Next you'll need to make a startup script.

Type this:

```
% pico /Library/StartupItems/MySQL/MySQL
```

and then paste the following into the window.

```
#!/bin/sh

. /etc/rc.common

##
# Start up MySQL server
##
```

```
if [ "${MYSQLSERVER:=−NO−}" = "−YES−" ]; then

ConsoleMessage "Starting MySQL Server"

/usr/local/share/mysql/mysql.server start

fi
```

Teaching you how to use pico or vi, or any other UNIX text editor is beyond the scope of this article. I'm having you use pico because it is easiest.

To save in pico type ctrl−O, then select the appropriate options that come up at the bottom of the Terminal. To exit pico, type ctrl−X.

Once you've saved and exited, type the following command to make the parameter list for your startup item.

```
% pico /Library/StartupItems/MySQL/StartupParameters.plist
```

and paste the following:

```
{
Description = "MySQL Server";
Provides = ("MySQL");
Requires = ("Resolver");
OrderPreference = "None";
Messages =
{
start = "Starting MySQL Server";
stop = "Stopping MySQL Server";
};
}
```

This parameter list tells Mac OS X what the Startup Item provides and what it requires.

```
% chmod −R −w /Library/StartupItems/MySQL/
```

This command makes it so no one can edit our new Startup Item.

```
% chmod +x /Library/StartupItems/MySQL/MySQL
```

This one tells the system that the MySQL startup script is executable.

```
% sudo chown −R root /Library/StartupItems/MySQL
```

And this one changes the owner of our new Startup Item to be the superuser, root. Now that we have a startup item, we need to tell the system that we want it run at startup.

Run the following command:

```
% sudo pico /etc/hostconfig
```

And paste the following at the bottom of the file.

```
MYSQLSERVER=−YES−
```

Whew. We're all done with MySQL now. There are some problems running MySQL that I will cover in the caveats section near the end of this article, but until then, we have some more (easier) compiling to do.

Making Apache

You may be asking "Why do I need to compile Apache when Mac OS X already comes with it?" Well, in my opinion it's always good to have a new version of such essential software. Also, it's best to ensure compatibility with the new modules you're going to be compiling, especially mod_dav.

```
% cd ..
```

This will take you out of the MySQL source directory.

```
% gnutar zxf apache_1.3.19.tar.gz
% cd apache_1.3.19
```

The Apache source is decompressed and ready to go.

```
% ./configure --with-layout=Darwin --server-uid=www \
--server-gid=www --enable-module=most --enable-shared=max
```

The −−with−layout=Darwin option tells the configure script that we want Apache installed with the correct layout for Mac OS X, the −−server−uid=www and −−server−gid=www tells configure to make it so the compiled Apache runs as user www in group www (this user and group are already in the system, Apple uses them for the built in Apache), −−enable−module=most tells the configure script to enable the standard Apache modules, and −−enable−shared=max tells the script to include mod_so (needed for dynamic modules, or DSOs) and compile all the standard modules as dynamic modules.

```
% make
% sudo make install
```

These commands compile and install Apache. You won't be needing to make a Startup Item for it because there was already one installed in the default Apache Apple included in Mac OS X.

That's it for Apache. Wasn't that easy?

Making WebDAV

WebDAV is a very useful protocol that allows for file transfer, versioning, and other things via HTTP. The standard installation of Apache comes with mod_dav, the module responsible for handling WebDAV requests, so I'll show you how to compile it for our new copy of Apache. You can find out more about WebDAV at http://www.webdav.org.

```
% cd ..
% gnutar zxf mod_dav−1.0.2−1.3.6.tar.gz
% cd mod_dav−1.0.2−1.3.6
```

Now that the mod_dav source is decompressed, we configure.

```
% ./configure −−with−apxs=/usr/sbin/apxs
```

The −−with−apxs option tells the configure script that we want mod_dav compiled as a DSO using Apache's APXS mechanism.

```
% make
% sudo make install
```

This will compile, install, and enable mod_dav. If you want it to disable it (as it is in Apple's default Apache) you'll need to edit /etc/httpd/httpd.conf and comment out (put # at the start of the line) these lines:

```
LoadModule dav_module libexec/httpd/libdav.so
AddModule mod_dav.c
```

One more step and you're done. On to PHP.

Making PHP

The last step is to get PHP set up.

```
% cd ..
% gnutar zxf php−4.0.5.tar.gz
% cd php−4.0.5
```

Now PHP is decompressed and ready to be configured.

```
% ./configure −−with−mysql=/usr/local
−−with−apxs=/usr/sbin/apxs \
−−with−zlib=/usr
```

The −−with−mysql option tells the configure script that we want to use the MySQL libraries that were installed with MySQL instead of the ones that come with PHP. The −−with−apxs option does the same thing it does in the mod_dav configuration, tells the configure script that we want PHP compiled as a DSO, and the −−with−zlib option tells the configure script where to find the zlib libraries it needs (These are included in Mac OS X).

Running the configure script strangely breaks one file in the distribution, main/internal_functions.c. In that file, some of the #include lines are run together into one line, and have the letter "n" separating them.

You can edit this file manually to fix this, but I've made a Perl script that will do the trick, if you'd rather use it.

Execute the following commands to use my script.

```
% wget http://www.devshed.com/Server_Side/Administration/BuildingOnOSX/fix_php.pl
% chmod +x fix_php.pl
```

This will get the script and allow you to execute it, and

```
% ./fix_php.pl
```

will run it, fixing the problem.

```
% make
% sudo make install
```

These commands will compile, install, and enable the PHP module.

You're not done yet, you need to tell Apache to use the PHP module for all files with a .php extension. Edit the Apache configuration file:

```
% sudo pico /etc/httpd/httpd.conf
```

and remove the #s from the lines that look like this:

```
# And for PHP 4.x, use:
#
#AddType application/x-httpd-php .php
#AddType application/x-httpd-php-source .phps
```

so they end up looking like this:

```
# And for PHP 4.x, use:
#
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

and restart your Apache server.

```
% sudo apachectl restart
```

Now, to test your PHP setup, you'll need to make a PHP script. Type the following:

```
% pico /Library/WebServer/Documents/phpinfo.php
```

and paste this in:

```
<?php
phpinfo();
?>
```

Save and exit, then open up your web browser and go to http://localhost/phpinfo.php If everything is working correctly, your browser will display a nicely formatted page with info about your PHP installation, and we're done!

Caveats

There are some problems with the built software when compiled as specified in this article, some because of the software itself, some not.

Firstly, the MySQL daemon that is compiled isn't able to shut down under Mac OS X. This is a known bug, but unfortunately there is no workaround or fix, so don't go trying to run mysqladmin shutdown expecting it to work.

Secondly, the way I have you build Apache in this article makes it so you can't load the mod_ssl module that comes with Mac OS X. This is because I felt the complexity of building an Apache with Extended API support as needed for mod_ssl, and compiling a new version of mod_ssl and it's support libraries, would add confusion to the article. I felt that this is a rarely needed feature, so this won't affect many of you.

If you need to use mod_ssl, you are able to build it by using the instructions that come with the module's source, if you are so inclined. Just be sure to use the same flags that you used in this article (−−with−layout, −−enable−shared, etc.) when compiling Apache, in addition to those required to make Apache work with mod_ssl. Mod_ssl can be found at http://www.mod−ssl.org.

This instructions in this article was tested and found to work on an iBook, an iMac DV, and a Power Macintosh G4 (PCI Graphics), all running Mac OS X 10.0.3. YMMV.