CHAPTER 3

Installing and Configuring mod_rewrite

As with any Apache module, there are a number of ways to install mod_rewrite. Fortunately, the vast majority of third-party distributions of Apache come with mod_rewrite installed and enabled. This is a reflection of the popularity and power of the module.

However, since mod_rewrite was added to the main Apache source distribution several years after the initial release, it is not part of what is enabled by default in an installation from source. Thus, whether you already have mod_rewrite installed and what you will need to do to get it working will vary depending on how you installed Apache.

Third-Party Distributions

A great amount of complication stems from the fact that there are dozens of different ways you might have installed Apache. Simplistically, however, you might have installed Apache from source code, downloaded from `http://httpd.apache.org/`, or you might have installed Apache from a binary package downloaded from `http://httpd.apache.org/`, or you might have installed Apache from a binary package obtained either with the operating system that you installed or from some third-party source as an add-on package for your particular operating system.

It is in this last case (i.e., third-party distribution of Apache) that causes the most frustration. The license of the Apache Software Foundation allows this sort of thing—even encourages it. But it means that those installations of Apache will differ from the documentation sufficiently to cause confusion on even the simplest task.

That doesn't mean that using third-party distributions of Apache is a bad thing;[1] it just means that these unofficial distributions make the documentation less reliable, and you may need to consult the documentation for your particular distribution.

1. You'll find a great deal of disagreement on this particular point, and I stubbornly (and cowardly) refuse to take a position on it in this book. Obviously, though, some third-party distributions of Apache do a better job of being "standard" and compliant with the documentation than do others.

Having said that, the following installation instructions should be correct in most situations. While some readers might find this a bit frustrating, it must be assumed that the makers of these third-party distributions thought that their decisions were the right ones for some reason, so let's give them the benefit of the doubt.

Installing mod_rewrite

Since there are a number of different ways to install Apache, and, thus, a number of different ways to install mod_rewrite, this section attempts to cover those various options. If you know how you installed Apache, you really only need to look at that particular portion of this section. If you're unsure, each subsection will try to clarify what kind of installation it is talking about.

We'll consider installing Apache from source, using both a static module build and a shared-object approach. Next, we'll discuss installing via a binary package.

This section does not constitute complete documentation of how to install the Apache web server. For that, you should consult the installation documentation at one of the URLs listed in Table 3-1.

Table 3-1. *Installation Documentation*

Version	Documentation
1.3	`http://httpd.apache.org/docs/1.3/install.html`
2.0	`http://httpd.apache.org/docs/2.0/install.html`

Static vs. Shared Objects

When installing Apache, you will need to decide whether you will compile modules statically or build them as shared objects. It's worthwhile to spend a few moments on this distinction before we delve into the various ways of installing mod_rewrite.

When a module is compiled statically, that just means the module is built into the main Apache executable file. Conversely, when a module is built as a shared object, the module is in a separate file (an `.so` file), which can be loaded into the Apache server when the server starts up.

In the case of statically compiled modules, you have no choice as to what modules are loaded: everything that was compiled statically will be loaded. The trade-off is that your server will run slightly faster, and there will never be any ambiguity as to what modules are or are not loaded.

In the case of modules that are built as shared objects, each one is stored in its own .so file, which must be loaded at server startup time. Most third-party binary distributions of Apache are built this way. With this kind of installation, you can pick which modules you want to have installed and leave out the ones you don't need, without having to recompile Apache. This is handled by directives in your configuration file.

Of the two options, building modules as shared objects is far more common, due to the convenience of adding and removing modules at will. It also makes it far easier to add third-party modules to the server later on.

The loading of shared object modules is handled by mod_so. It is thus recommended that you always install mod_so on your server, just in case you need it.

Installing from Source: Static

If you perform a default installation of Apache and accept the default selection of modules, mod_rewrite will not be installed. Thus, if you want to have mod_rewrite installed as a statically compiled module, you'll need to add an additional flag at build time.

If you are installing Apache 1.3, this flag will look like this: `--enable-module=rewrite`. So, when you configure your Apache installation, the `configure` command might look something like the following:

```
./configure --prefix=/usr/local/apache --enable-module=rewrite [other options]
```

This will add the mod_rewrite module to the list of those being installed already, and it will (when you type `make` and `make install`) build the module into the httpd binary executable file.

If you are installing Apache 2.0, the flag will look instead like this: `--enable-rewrite`. In this case, the configure like will look as follows:

```
./configure --prefix=/usr/local/apache2 --enable-rewrite [other options]
```

In either case (1.3 or 2.0), you can include other command-line arguments as well, in order to build Apache exactly as you need it. You can find out more about the available configuration command-line options by typing

```
./configure --help
```

After running `./configure` with these options, you will need to `make` and `make install` to get Apache installed and ready to run. Once again, you may need to consult the installation documentation referenced in Table 3-1.

Installing from Source: Shared

If you wish to install mod_rewrite as a shared object, either because you've already built Apache and don't wish to have to rebuild it, or because you just happen to like running your modules as shared objects, this section is for you.

You can install a module as a shared object either at the time that you build Apache or after the fact. The effect is the same: you end up with a file called `mod_rewrite.so` in your `modules` directory, in the case of Apache 2.*x*, or in your `libexec` directory, in the case of Apache 1.3.

This module can be loaded as necessary using the `LoadModule` directive in your Apache configuration file. This allows you to decide whether you want to load the module, and you can also change your mind as often as you want, without having to recompile anything. Modules can be compiled as shared objects during the initial installation of Apache, using the techniques outlined in the sections that follow.

Installing mod_rewrite As a Shared Object on Apache 1.3

For Apache 1.3, the syntax is `--enable-shared=rewrite` in the `./configure` command line, like this:

```
./configure --prefix=/usr/local/apache --enable-shared=rewrite [other options]
```

This will automatically include `mod_so` into the configuration and configure mod_rewrite to be compiled as a shared object. It will also add the necessary `AddModule` and `LoadModule` directives to the default configuration file, so that the module will get loaded when the server starts up.

The following directives will appear for mod_rewrite:

```
LoadModule rewrite_module     libexec/mod_rewrite.so
AddModule mod_rewrite.c
```

Note that you can also tell the configure script to build everything as a shared object, thus eliminating the inevitable confusion when some modules are shared and others are not:

```
./configure --prefix=/usr/local/apache --enable-module=most --enable-shared=max
```

On the other hand, if you already have Apache 1.3 installed and only want to add mod_rewrite to it as a shared object without recompiling Apache from scratch, you can do so with the utility called `apxs`, which comes with Apache.

First, you will need to locate the source code of mod_rewrite itself. You can find it in the downloaded Apache source code, in the `src/modules/standard` directory. Change into that directory, and type

```
/usr/local/apache/bin/apxs -cia mod_rewrite.c
```

This will build the `mod_rewrite.so` shared object file, copy it into your Apache `modules` directory, and append the necessary configuration directives to your Apache configuration file.

Note that if you installed Apache in some location other than /usr/local/apache, apxs will be located at a different path.

Installing mod_rewrite As a Shared Object on Apache 2.*x*

For Apache 2.*x*, since the configure script is entirely different, so too are the options for building mod_rewrite as a shared object. To build an individual module (such as mod_rewrite) as a shared object, you may use the --enable-mods-shared argument:

```
./configure --prefix=/usr/local/apache --enable-mods-shared='rewrite'
```

Several modules can be configured as shared objects by making this a space-separated list:

```
./configure --prefix=/usr/local/apache --enable-mods-shared='rewrite dav dav-fs'
```

And, as with Apache 1.3, you can simply specify that all of the modules should be built as shared objects by specifying the following:

```
./configure --prefix=/usr/local/apache --enable-mods-shared=most
```

In this case, only one line will be added to your configuration file to load mod_rewrite:

```
LoadModule rewrite_module modules/mod_rewrite.so
```

If you already have Apache 2.*x* installed, you can add mod_rewrite, or any other module, as a shared object using the apxs utility that comes with Apache.

The mod_rewrite source is located in the modules/mappers subdirectory of your Apache 2.0 source code directory. Change into that directory and type the following:

```
/usr/local/apache2/bin/apxs -cia mod_rewrite.c
```

This will build the mod_rewrite shared object file, copy it into your Apache modules directory, and modify your configuration file to load mod_rewrite on its next restart.

If you installed Apache in some location other than /usr/local/apache2, then apxs will be located in that different path.

Enabling mod_rewrite: Binary Installation

If you didn't install Apache from source, then you've probably installed a binary distribution, which you obtained either from the binaries/ directory on the Apache distribution site or from your operating system vendor. For example, you may have installed an RPM, or you installed via the apt-get, yum, or urpmi installation manager. Alternatively, if you are running Apache on Windows, you may have download a binary installation package either from the http://apache.org website or from any of a variety of third-party sites that offer Apache with an assortment of add-on features.

If you installed a binary distribution of Apache, or if Apache was installed by default when you installed your operating system, it is certain that the modules have been built as shared objects. This is done by default with all binary distributions of Apache, so that a single distribution may satisfy everyone's requirements. It is then up to the system administrator to determine which modules should be loaded and to modify the configuration file accordingly.

One point on which the various binary distributions differ is which modules they enable by default. It's increasingly common to enable *everything* by default and leave it up to system administrators to disable those modules they do not want. Since it seems that the majority of administrators are unaware of this, many web servers are running with modules enabled that are not actually necessary.

Enabling—or disabling—an installed module in this situation is a matter of uncommenting, or commenting out, a line or two in the configuration file.

As you saw earlier, in the case of Apache 1.3, each module requires two configuration lines to enable it. You will need to locate these lines in your configuration file, and, if they are commented out (the line starts with a # character) you will need to uncomment them (remove that # character) in order to enable the module. The lines you are looking for are those listed previously, namely

```
LoadModule rewrite_module     libexec/mod_rewrite.so
AddModule mod_rewrite.c
```

Conversely, if there are any modules you don't want loaded, you should locate the directives for those modules and comment them out:

```
# LoadModule imap_module        libexec/mod_imap.so
# AddModule mod_imap.c
```

These directives will probably not appear together, as shown here. All of the LoadModule directives will be together in a block, and all of the AddModule directives will be together in a block.

Keep in mind this isn't always the case, as makers of third-party distributions of Apache are at complete liberty to do whatever they want with their default configuration files, and some of them are quite creative. Therefore, be sure to keep an eye out for discrepancies should you be using any such distribution.

The most common deviation from this convention is to put the module load statements in a separate file, often one per module. For example, there might be a file named rewrite.conf that contains the LoadModule directive for mod_rewrite as well as other mod_rewrite-related directives. These files (or this file) will then be loaded via an `Include` directive, such as

```
Include modules.d
```

or perhaps

```
Include rewrite.conf
```

In the case of Apache 2.*x*, you will need only the `LoadModule` directive, which should look like this:

```
LoadModule rewrite_module modules/mod_rewrite.so
```

In the event that you can't find any of these files or directives, you might want to learn to use the command-line utility `grep`, which is a powerful search tool. If you change into the directory where your configuration files are located and type the following, you'll learn which files contain the directives you're looking for:

```
grep -ri loadmodule *
```

This command will search through all the files in that directory and in subdirectories for any occurrences of the `loadmodule` directive. The `-r` command-line option tells it to recurse through subdirectories. The `-i` option tells it to search in a case-insensitive manner—that is, to find occurrences whether they are upper- or lowercase. And the * tells it to look in all files. If you are running Apache on Windows, you will need to use the Windows search tool to perform this function.

Testing Whether mod_rewrite Is Correctly Installed

While there are a variety of ways to test if mod_rewrite is correctly installed, we'll opt for the simplest one right now. In your Apache configuration file, add this one line:

```
RewriteEngine On
```

Then run the configuration test argument to `apachectl`:

```
apachectl configtest
```

If Apache does not return an error message, then mod_rewrite is installed correctly.

If mod_rewrite is not correctly installed, you will receive an error message that looks like the following:

```
Syntax error on line 265 of /usr/local/apache/conf/httpd.conf:
Invalid command 'RewriteEngine', perhaps misspelled or defined by a module not
included in the server configuration
```

In that event, you should go back and read the preceding sections again to see which bit you missed.

If You’re Not the System Administrator

Many of you have likely found much of this discussion to be rather frustrating, because you’re not the system administrator on your server. For example, you may have hosted web space at some website provider, and you don’t have access to make the kinds of changes and configurations we’ve been discussing.

In your case, mod_rewrite is either installed or it’s not, and there isn’t much you can do about it, other than ask your sysadmin nicely and hope that they’re having a good day.

Let’s start by checking to see if `.htaccess` files are enabled on your server. You probably already know whether or not they are and can skip to the next bit. But if you’re unsure, then follow these steps.

First, create a `test` directory in which you can safely experiment without breaking any of the content you’re already serving from your website. In that directory, create two files. The first one, which you’ll name `index.html`, will contain nothing more than some sample content, such as, perhaps, the word “Hello”. Assuming you created this `test` directory at the root level of your website, you should now be able to load that file in your browser with the URL `http://your.server.com/test/index.html`.

Next, you’ll create a file in that directory called `.htaccess` that contains the following line:

```
InvalidDirective Here
```

Return to your browser and reload the URL you were looking at a moment ago. One of two things will happen. Hopefully, you’ll get an “Internal Server Error” page, which will indicate that Apache did indeed parse the `.htaccess` file and didn’t understand what it found there because it was an invalid directive.

If, on the other hand, the `index.html` file loads without any error message, this is an indication that Apache is entirely ignoring your `.htaccess` file, and you’ll need to contact your sysadmin to ask them to enable `.htaccess` files. Hopefully, they know how to do this, but, if they do not, you should refer them to the `.htaccess` tutorial, which can be found at `http://httpd.apache.org/docs/2.0/howto/htaccess.html`.

Next, we'll check to see if mod_rewrite is in fact installed. Remove the `InvalidDirective` line from your `.htaccess` file and replace it with this line:

```
RewriteEngine On
```

Once again, you'll see one of two things. Either the page will load without a problem, which indicates that mod_rewrite is installed, or you'll receive the "Internal Server Error" page again.

If you do receive the error message, it can, in fact, mean one of two things: it might mean that mod_rewrite is not installed, or it might mean that mod_rewrite is installed, but the server is configured in such a way as to not permit you to use it. If you have access to your error log, check for the following error message:

```
RewriteEngine not allowed here
```

This message indicates that mod_rewrite is installed, but your server administrator has not set the `AllowOverride` directive to a level sufficient to allow you to use it. In particular, `AllowOverride` needs to be set to `FileInfo` (or to `All`) in order to permit the use of mod_rewrite directives.

Alternatively, you may see the following error message:

```
Invalid command 'RewriteEngine', perhaps misspelled or defined by a module
not included in the server configuration
```

This message indicates that mod_rewrite is not even installed.

In either case, you will need to contact your system administrator, explain to them what you did and what error message you received, and ask if they might configure the server correctly so that you can use mod_rewrite directives.

Enabling the RewriteLog

We're going to talk more about the rewrite log in Chapter 12, but it's important that you're at least aware of it at this point. This will help you to troubleshoot your own mistakes as you go along (if you're at a more advanced level).

To turn on the rewrite log, you need to add the following two directives to your main server configuration file. These cannot be set in an `.htaccess` file.

```
RewriteLog /var/log/rewrite.log
RewriteLogLevel 9
```

The location you select for your rewrite log should probably be where your other logs are located, although it can be just about anywhere. And you should be aware that having the `RewriteLog` enabled causes a pretty significant performance drain, as well as creating enormous log files in a very short time period. Thus, you should enable this feature only for testing and debugging, rather than having it enabled on your production server.

■**Note** In Chapter 12, we'll discuss in greater detail what you'll see in the rewrite log and how to decipher it.

Summary

Hopefully, you now have mod_rewrite installed and you're ready to start using it. In the course of the next few chapters, I'm going to introduce the various directives and instruct you in their use. It is therefore important, as we move forward, that you actually have mod_rewrite installed and operational, since there will be hands-on examples you'll want to try.