1

Getting Up and Running

PHP, which stands for HyperText Preprocessor, is widely used for creating programmed features for Web sites because it is easy to learn and also because PHP syntax is drawn from other widely used languages, making it familiar to many programmers. In this chapter we present a very brief history of PHP, and then discuss the nature of PHP as it relates to the Web.

Before you can get into the nitty-gritty of programming with PHP5, you need a clear understanding of how PHP programs work across the Web, and that obviously implies knowledge of the Web protocol called HyperText Transfer Protocol (HTTP). HTTP is the language or format for communications from browser to Web server and back, and is therefore fundamental to many aspects of PHP. HTTP gets some coverage in this chapter, and quite a bit more in Chapter 3.

You'll see how to properly setup PHP on a Linux server, and on a Windows server as well. PHP programs run in conjunction with Web pages, which in turn run (or are distributed by) Web server software (such as Apache or IIS), which in turn run on top of an operating system (such as Linux or Windows). Although it's not strictly necessary to know everything about network operating systems to build good PHP programs, there are many aspects of PHP that are controlled or affected by the Web server. If you're unfamiliar with server computers, Web servers, and the like, don't worry. You'll soon see how they work, and look at the requirements and process of installing basic Web server software.

This chapter leads you through installing PHP on a Red Hat Linux machine running Apache, and through installing PHP on a Windows 2000 machine running IIS. Just pick the one that's right for you.

You'll also examine the contents of the PHP configuration file `php.ini` with you, and test your PHP installation.

Obviously there's a lot of work for you in this chapter, so let's get started.

The Roots of PHP

PHP is a programming language designed to work with HTML, but unlike HTML, PHP has data processing capabilities. If you are familiar with HTML, you know that it is not really a programming language, but more of a rendering language—that is, HTML enables you to write Web pages using

code that creates a pleasing (hopefully) display of text, graphics, and links within a browser. Although there are a few helpful features of HTML (such as the capability to cause a form submission), for the most part HTML does nothing programmatically. For example, there are no HTML commands that enable you to add two numbers together, or access a database.

If you remember the Web back in the early '90s, you may recall that early Web pages were made from HTML code written as plain text source files. When you made a connection to a Web site with your browser, the Web server software sent these plain text HTML files to be processed and rendered into Web pages. Your browser actually did the rendering process (and still does, to be sure), but if you clicked View ➪ Page Source, you'd see the raw HTML code.

Javascript (and a few other almost unknown programming languages) improved the situation for Web designers in that it provided for programmatic functionality within Web pages. However, it was limited to programmatic functionality on the user's computer, not on the back-end (on the Web server), where all the really cool data processing and database access takes place. Practical Extraction and Reporting Language (PERL) was one of the first widely used languages for programming on the back-end, but has limitations of its own, such as an inability to be mixed in with HTML for easy in-page programming.

So where does PHP fit in with HTML? PHP began as PHP/FI, developed in 1995 by Rasmus Lerdorf from some Perl scripts he had created for tracking accesses to his online resume. Eventually, Rasmus wrote an implementation in C, released the source code to the public, and by the beginning of 1998 version 3.0 of PHP was released (written by Rasmus Lerdorf, Andi Gutmans, and Zeev Suraski), the first version that is very similar to the current releases of PHP.

The main goal of PHP is to enable users to easily develop dynamic Web pages. The difference between dynamic Web pages and static Web pages is that the content and structure of dynamic Web pages may change each time they are accessed (that's what the back-end programming is for) whereas the content and structure of static Web pages is fixed and does not change unless the designer manually changes them.

Unlike many other languages, PHP can be embedded directly into HTML, making it quite easy for those familiar with HTML to grasp how to add back-end, programmatic functionality to their Web pages. This single capability is one of the most important factors in PHP's usefulness, and thereby its popularity. But have no doubt that PHP is growing into a much more full-features language going well beyond the initial intentions of its authors. PHP intends to be the primary language for a great variety of online and offline applications, and PHP5 is showing every sign of doing just that.

And you shouldn't forget how well PHP works with HTTP (HyperText Transfer Protocol), the communications protocol (pre-agreed format for data communications) for Web. Whenever you click a link or enter a Web address into your browser, a request in HTTP format is sent to the Web server, which responds by sending back the Web page. If the Web page isn't found, you'll probably get the "404 Not Found" error. Sending back the correct page or sending an error if the page is not found are HTTP functions. We discuss HTTP thoroughly in Chapter 2 because several important aspects of PHP applications depend on HTTP.

Installing, Configuring, and Running PHP

Before you can write a PHP application that works with your Web pages, you need to have PHP installed and configured. Because you'll be writing a Web application, it's a given that you'll need a Web server and some Web pages (a short HTML primer is provided in Chapter 3, although it's assumed that you

know or can easily pick up how to make basic Web pages). You'll also need to download, install, and configure PHP, so we provide complete instructions about how to do these things in the coming sections. Note that some configuration options for PHP are related to very specific application requirements (you don't need to worry about them unless you need them) so many of the options aren't discussed until you reach the appropriate chapter.

System Requirements

To run the code in this book you will need at least the following software:

- Server software (an operating system such as Windows 2000 or Linux)
- A PHP-compatible Web server (such as Apache or Internet Information Server (IIS)
- PHP5 (get the download from `www.php.net`)
- A relational database system (starting at Chapter 9, we use SQLite or MySQL)
- A Web browser (such as IE, Mozilla, and so on)
- A text editor, such as Notepad, Emacs, vi, BBEdit, and so on.

You shouldn't have to worry about hard drive space or RAM, unless you are working on a very old system, or one that is overloaded. PHP doesn't take up much room, and runs very efficiently.

You can run all of the software listed here on the same computer, for development purposes. If you have access to several networked computers, you may want to install all of your server software on one (typically either a UNIX or Windows NT/2000 computer), and use another networked computer as your client machine. For the purposes of this book, we will generally assume you are running all of the software on a single computer. This is the configuration used by most Web developers.

php.ini, the PHP Configuration File

There are two examples of PHP configuration files that come with PHP when you download it: `php.ini-dist` and `php.ini-recommended`. After you download and install PHP, there will be one file named `php.ini` strategically placed on your system, and each time PHP starts it will read this file and set itself up accordingly. The `php.ini` file can be written out by hand, but of course most of us just modify either the `dist` or `recommended` file to suit our needs, and then copy and rename it into the appropriate folder.

However, you should note the following lines in the top of the `dist` file:

```
; This is the default settings file for new PHP installations.
; By default, PHP installs itself with a configuration suitable for
; development purposes, and *NOT* for production purposes.
```

The settings in the `dist` file are used for nearly all of the examples in this book and we'll let you know whenever the configuration settings are changed. But you will want to use the `recommended` file when you complete your applications and copy them over to your production server, and you should be aware

that you may need to rewrite your code a little bit to work properly with the `recommended` file's configuration settings. We'll discuss this more as we go along.

Setting Up a Test Machine

In this chapter, we'll walk through setting up PHP5 on a Red Hat Linux machine running the Apache Web server, as well as on a Windows 2000 machine running Internet Information Server (IIS). You can run PHP5 with many other operating systems and Web servers, so see the PHP5 documentation for installation and configuration on other servers. And there are a variety of installation methods you can use. For example, there is an automatic installer for the Windows version, whereas you can install the Linux version using RPMs (for some versions of Linux), and you can also download and compile the Linux versions from the original source code if you like. None of the installations are all that difficult if you follow procedures correctly, and the examples we provide are a good starting point for many of the installations available.

There are some third-party installers (often open-source and free) out there, if you want to look for them. For instance, you might try PHPTriad or Foxserv in Google.

Network Connections

If you don't already know, a computer doesn't need to be attached to the Internet, or even to a network, to run Web server software. If you install a Web server on a computer, it's always possible to access that Web server from a Web browser running on the same machine, even if it doesn't have a network card or modem. Of course, to download and install the software you need, you have to have access to an Internet connection. But you don't need it to be active just because you're running your Web server.

Once you have a Web server installed and running, you'll install PHP5 alongside it. There's some configuration required to tell the Web server how to run PHP programs, and we'll walk through that process before we start PHP. There is an automatic installer to be found with most distributions of PHP; we'll use a primarily manual process to illustrate what's happening during installation.

What if it goes wrong? The `README` and `INSTALL` files that are included in most PHP downloads, as well as the PHP manual at `www.php.net/manual/`, provide detailed information which may be more up-to-date than the information here, which covers the PHP5.0.2 release.

Where Do You Start?

There are two main installation paths from which to choose, and each simply depends on which operating system you're using:

- Installing PHP5 with the Apache Web Server on Linux (we use Red Hat Fedora Linux)
- Installing PHP5 with Microsoft Internet Information Server on Windows (we use Windows 2000)

PHP5 can be installed on a great variety of Web server/operating system combinations, including under Apache on Windows. The two systems we're using are the easiest to get working. If neither of them suits you, of course you can install whatever other configuration you want—you should still be able to run all of the examples in the book. Refer to the PHP5 manual for more general installation instructions.

Running PHP5

One of the basic choices to make when installing PHP5 with your Web server is whether to run it as a CGI binary or as a separate static or dynamic module. CGI (Common Gateway Interface) is a very useful way to run interpreters such as PHP5. Because of security risks (see the "*Running as a CGI*" section later in this chapter for more information), compiling PHP5 as a static or dynamic module is recommended under most circumstances. Our installations (on Linux and on Windows) load PHP as a separate SAPI (Server Application Programming Interface) module. On Windows, the ISAPI filter was used to run PHP as a SAPI module.

Although it is most common to run PHP in conjunction with a Web server, so that Web pages with a file extension such as `.php` are processed through the PHP interpreter before the finished page is sent back to the browser, there is also a command line utility that enables you to run PHP code from the command line. It is present from any of the installation types we demonstrate. You can find plenty of documentation about it on the PHP site (`www.php.net`).

Creating and running PHP Web applications in a satisfactory way implies that you are running (or have access to) a Web server upon which PHP is (or can be) installed, and that the installation has been tested and runs properly. It also implies that PHP has been (or can be) configured to support the needs of your PHP programs. There are a couple scenarios under which these requirements can be achieved:

- You are running a desktop or server machine, operating system, and Web server compatible with PHP, and PHP has been installed and configured.
- You are running a desktop or server machine connected to the Internet, with access to a Web hosting account supported by a Web server with which PHP has been installed and configured.

The vast majority of desktop machines run Windows 98, NT, 2000, 2003, and XP. In many cases you can get a free copy of Personal Web Server (PWS) and install it on a machine running one of these operating systems. PHP is compatible with PWS, so you can install and configure PHP on desktop machines running basic operating systems such as Windows 98. Server operating systems such as Windows NT, 2000, and 200, come with Internet Information Server (IIS). PHP is compatible with IIS, and you can install and configure PHP on these machines. Our Windows 2000 installation of PHP5 uses IIS as a Web server.

The majority of Web-hosting computers run some version of Linux, such as Debian, RedHat, FreeBSD, and so on. The Web server of choice for these machines is Apache. PHP is compatible with Linux and Apache, and you can install and configure PHP on these systems, but if you are not in charge of the Web-hosting computer (and many times you won't be) you'll probably have little control over the installation and configuration. If you find yourself in this position (for example, if you've been hired to work on an existing Web site running on someone else's server), you can simply verify the operating system, Web server software, and PHP version so you can cope with whatever you've have to work with as you develop your PHP programs.

Chapter 1

Installing PHP5 with Linux and Apache

At the time of this writing, the very first release candidate of PHP5 was available, and that's the one we're using. But you may want to check the PHP site for more recent versions, and any notes about changes.

The combination of Linux, Apache, MySQL, and PHP is probably the most common production environment for running PHP Web servers. This combination of open-source software has been referred to by the acronym LAMP. If you run the same combination of software, you can benefit from the experiences of the many other people who've used this setup.

The PHP developers work very closely with the Apache and MySQL teams to ensure that advances in the three server systems are fully supported by the other components. However, at the time of this writing PHP5 is being distributed with SQLite rather than MySQL, because there is some concern about whether MySQL is still open source. This may not be a concern when you read this and begin developing, but it's worth noting.

Choosing Your Installation Method

As with other open-source software, you have the option of downloading the PHP and Apache source code (which, in both cases, is written in the C programming language) and compiling the programs yourself. If that sounds daunting (it's not actually as scary as it sounds), you can obtain precompiled versions in one of two forms: binary downloads, which are precompiled versions of the software that typically come with installation scripts to put all the required pieces into the necessary parts of your file system, and binary packages, which are available for systems that have a software package management system, such as the Red Hat Package Manager (RPM) for Linux, and are the easiest to install.

Here's a quick overview of the three methods:

Installation Method	Advantages	Disadvantages
Source	Most flexible solution for custom installations. Additional tests and examples are included in the source distribution	Needs to be compiled. Slightly more difficult than the other options. Harder to remove once it's been done
Binary (compiled)	No need to mess around with trying to compile the server. Takes less time to install	Less flexible than doing an installation from source
Binary RPMs	Fastest and easiest installation method. Very easy to uninstall or upgrade later	Must be using an RPM-based Linux distribution such as Red Hat. Least flexible installation method

An RPM Installation of PHP4

The version of Red Hat we're using is actually called Fedora, because Red Hat has split off development into two parts: Fedora and the enterprise version of Red Hat Linux. Currently, the Fedora site doesn't

have an RPM for PHP5, so we'll provide the instructions for getting and installing the RPM for PHP4 here, and then show how to download and compile PHP5 for Fedora later. By the time you read this, in all likelihood there will be an RPM available for PHP5 for your Linux distribution, so the RPM installation presented here should provide good guidance for installing PHP5 via the RPM method.

A number of popular Linux distributions use the Red Hat Package Manager, including Red Hat, SuSE, Mandrake, Definite, TurboLinux, Caldera, and Yellow Dog. If your system uses an alternative package management system, such as Debian's deb packages, refer to your distribution's manual for installation instructions.

Obtaining RPMs

The best place to get RPMs is almost always the disks from which you installed your Linux system. Red Hat 7 and SuSE 7 both include PHP4 (although it isn't installed by default)—by the time you read this, the same should be true of most current Linux distribution versions.

If your distribution doesn't include PHP4, or it doesn't include all the required functionality or support RPMs, then the next place to check is your Linux distribution vendor's Web site, which should have a download area or FTP site from which you can obtain the latest RPMs.

Finally, `www.rpmfind.net` provides a comprehensive search service for RPMs. When you download RPMs, though, make certain that they are compatible with your Linux distribution and your computer hardware. Different distributions put important files in different places, and this can lead to RPMs from different vendors not working on other systems. Most RPMs are available compiled to run on the different hardware systems that Linux supports. The following table shows the most common abbreviations used in RPM names (you need the abbreviation to search on the rpmfind site):

Abbreviation	Compatible with
i386	PCs based on Intel and 100% compatible processors: Intel 80386, 486, Pentium, Pentium II, Pentium III, and Celeron; AMD 5x86, K-series, and Athlon; and Cyrix 6x86
i586	PCs based on Intel Pentium and 100% compatible processors: Intel Pentium II, III, and Celeron; AMD K-Series and Athlon; and Cyrix 6x86
PPC	Computers built around Motorola PowerPC (and compatible) chips, such as Apple's Power Macs, G3s, G4s, and iMacs. You can still only use the RPMs on Macintosh hardware with Linux installed, though
alpha	Servers and workstations running the Compaq Digital 64-bit Alpha processor
sparc	Servers and workstations running the processors which use the 64-bit SPARC architecture, such as Sun Microsystems' UltraSPARC
m68k	Computers built around Motorola's older 68000 series processors, such as Amigas, and older Apple Macintoshes, for which various Linux ports exist

Refer to your distribution's manual if you want to use the graphical installation tools that come with your specific distribution. These differ widely, so they can't all be covered here. However, any RPM-based

system can be controlled using the rpm command-line tool, and you'll see how to install the required components using this interface.

Which RPM Packages Do You Need?

The RPM packages you will need are:

- zlib
- libpng
- libjpeg
- gd
- gd-devel
- apache
- mod_php4

You can find out which of them are already installed on your system by typing the following at a command prompt, substituting in the name of each of these packages in turn:

```
> rpm -q zlib
zlib-1.1.3-6-i386
> rpm -q libng
Package libpng is not installed
```

As you can see, if the package is installed, it gives you a random-looking string. If it isn't installed, you get a helpful error message. The string actually tells you which version of the software you installed using the package (1.1.3 in this case), which release of the package it is (this example has the sixth public release installed), and the architecture for which the RPM was compiled (Intel 386 compatible, which is just as well, because the package is installed on a Pentium III for this book).

Note which of the packages you already have, and which versions they are (the version number is more important than the release number).

Apache is at version 1.3.29 if you want to remain at the old versions of GD, or 2.0.48 if you want to be current with the latest version of GD. Of course, if you are installing PHP5, GD is now bundled with PHP and is up to version 2.0.17

Then locate suitably up-to-date versions of all the packages that you don't have already, or have old versions for. As suggested, try your install CDs, your distributor's Web site, and `www.rpmfind.net`.

Once you have current versions of all the packages you need, you can install them. The command for upgrading an existing installation or installing a package for the first time is exactly the same. Navigate your command prompt to the location of the files on the CD or the directory into which you downloaded the RPMs. As root, type:

```
> rpm -Uh libpng-1.0.5-3-i386.rpm
##################
```

For each package you need to upgrade or install, just substitute the name of the package file you downloaded. The line of # signs extends across the screen as each installation progresses.

If you install the packages in the order listed previously, you should find that all the prerequisite files are installed in the necessary order.

Installing PHP5 by Compiling from Source Files

The installation method we'll use for installing PHP5 on Red Hat Fedora running Apache is downloading the source files and compiling them. You use command-line commands in Linux, but also make use of some of the visual tools (such as Konqueror) included in your Red Hat installation. If you are running Linux visually (for example, with KDE), you get to the command prompt by going to the Red Hat button, and then choosing System Tools ➪ Terminal. Figure 1-1 shows the terminal window you'll see.

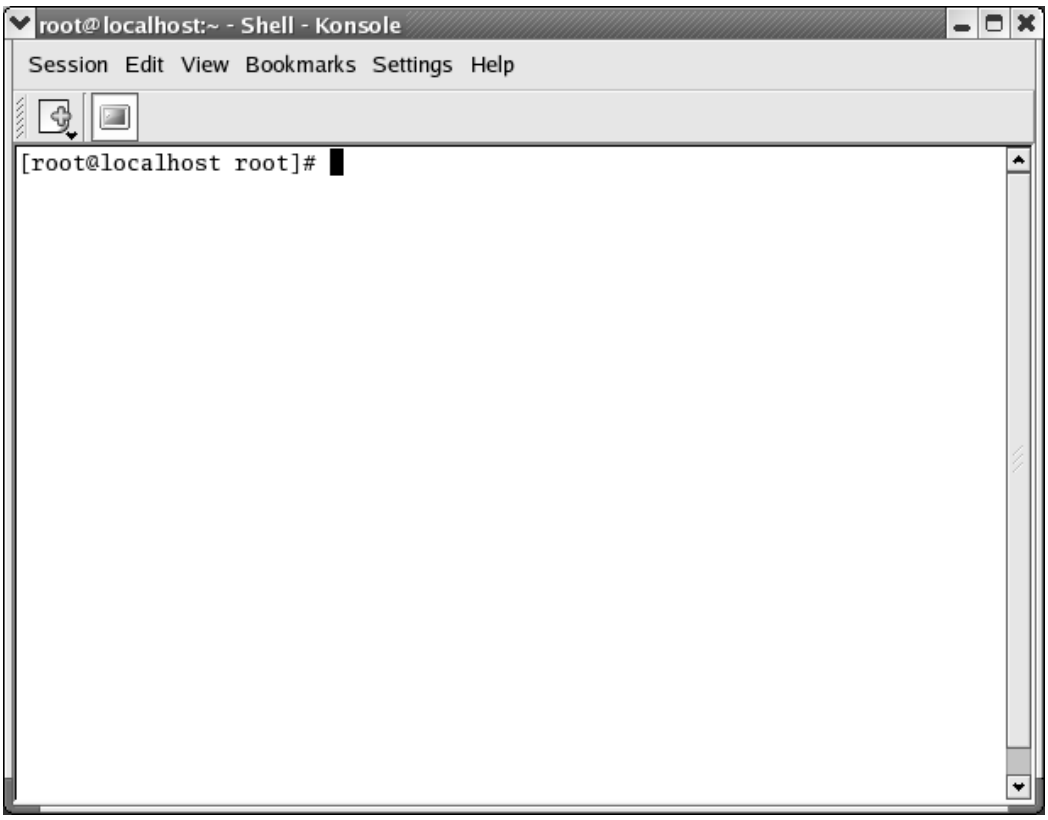

Figure 1-1

You must have a compiler installed (an ANSI C compiler). Sometimes such a compiler is installed as part of your Linux installation, but if you need one, a good one (and free), named gnugcc, can be found at `www.gnu.org`. Figure 1-2 shows the GNU Web site.

And Figure 1-3 shows a bit of the documentation for GCC.

With your compiler installed, download the source file from `www.php.net`. This file is archived as a tar file and compressed with gzip, so you will need to uncompress it. There is also a `.bz2` file you can download, but you need only one of these files—either the gzip or a `.bz2` file.

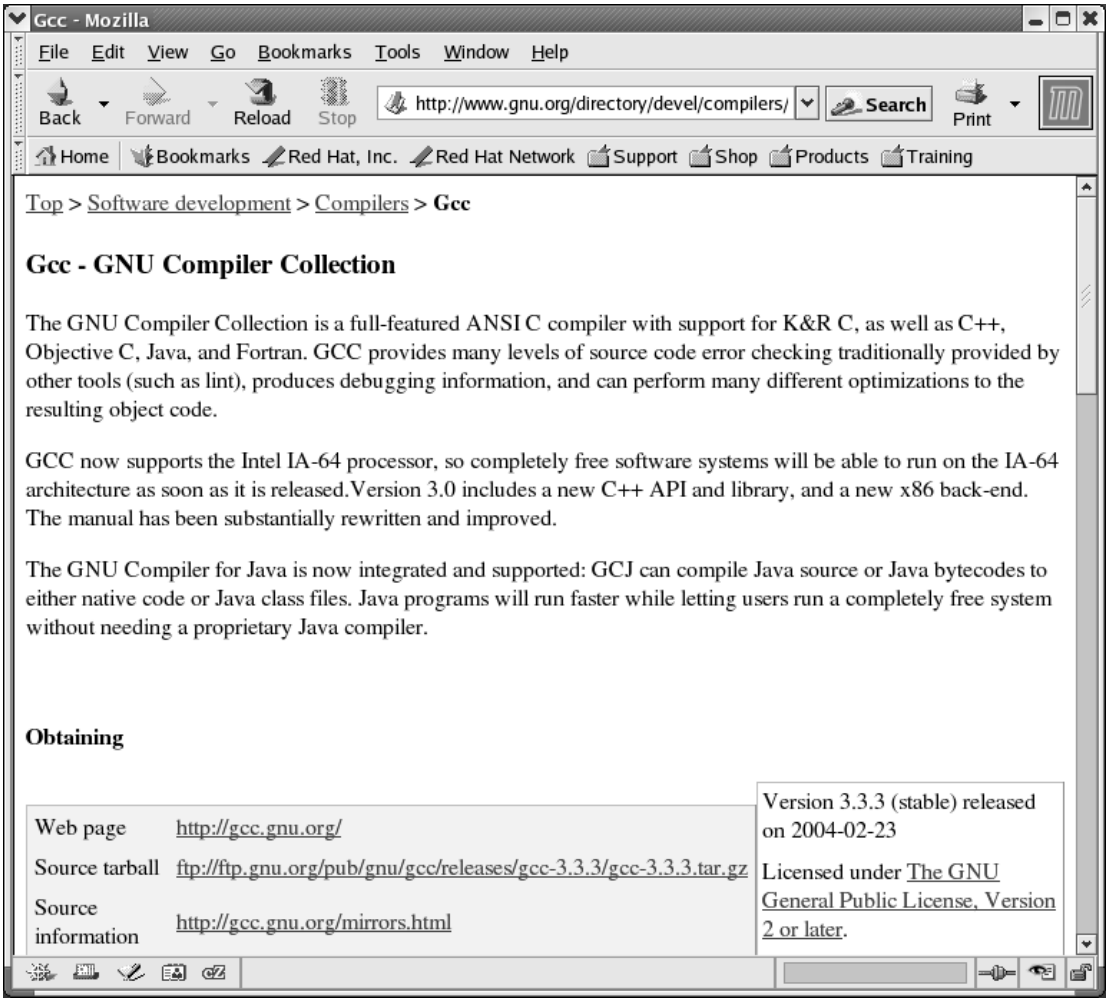

Figure 1-2

You can use the Konquerer file-management tool to view the contents of the compressed file. Figure 1-4 shows some of the contents in the tar file.

You also could use Konquerer to copy all of the compressed file's contents directly to another folder, but doing so will make your compilation fail cryptically (meaning you'll get strange error messages that won't help you figure out what's wrong). Instead, make sure to use the following command from the terminal (see Figure 1-5) to uncompress the files:

```
tar -xvzf php-5.0(insert the rest of the version number here).tar.gz
```

Next, use the `cd` command to change to the PHP5 distribution directory:

```
cd php-5.0(insert the rest of the version number here)
```

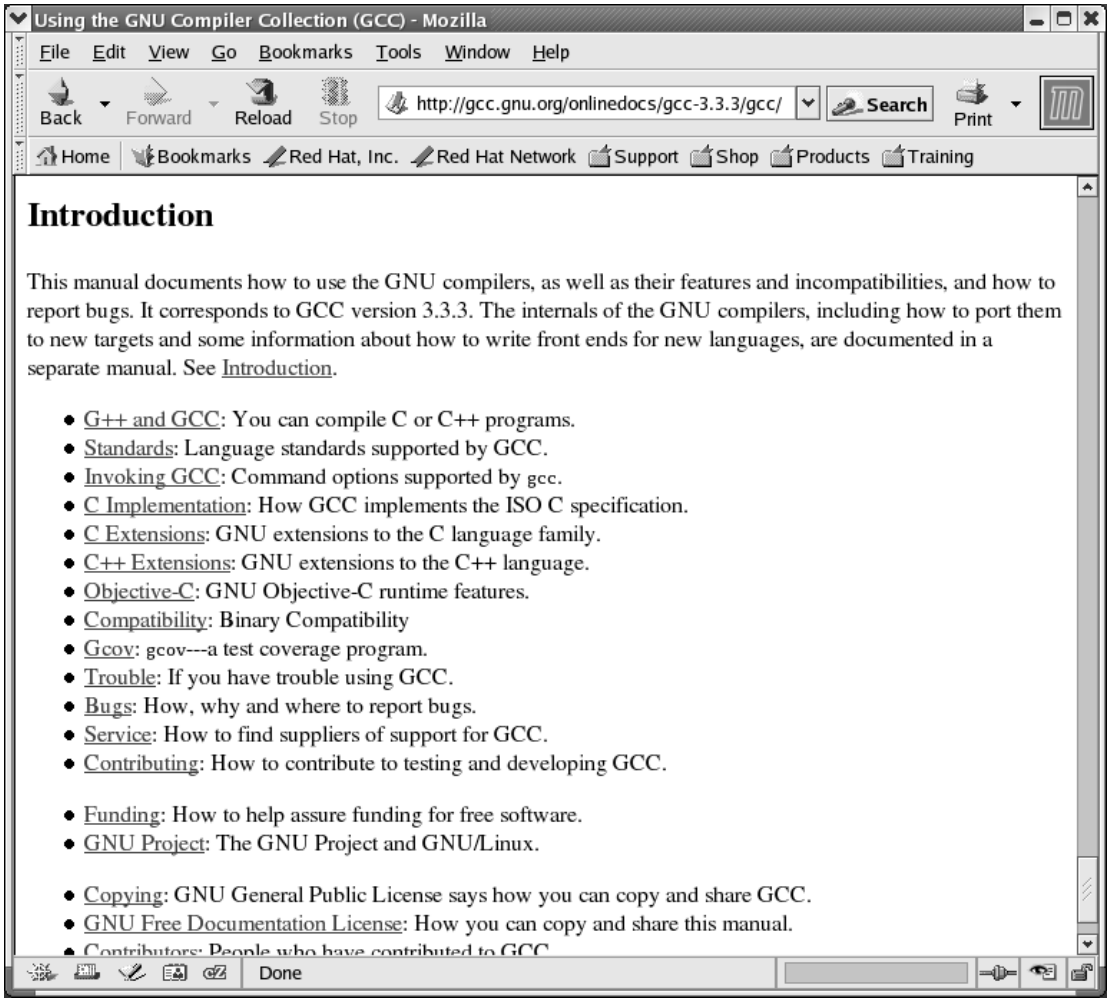

Figure 1-3

Now that you've cd'd to the `php-5.0.0RC1` folder, you'll see quite a few folders and files there. Open the `INSTALL` text file (see Figure 1-6) to find many of the instructions related to your installation.

Folder *and* directory *are equivalent terms and can be used interchangeably.*

For this book, PHP is installed as a Dynamic Shared Object (DSO), and that's what you'll also do, so that the entire Apache Server won't need to be recompiled.

The latest versions of Apache support DSOs, and shared objects can be used by other programs, such as PostGreSQL. Although you could compile PHP5 as a static module, that isn't recommended. If PHP is statically linked to, say, Apache or PostGreSQL, each of those programs would need to be recompiled before they would recognize PHP. The programs' configuration files can be easily changed in shared objects (DSOs) without any recompiling.

Chapter 1

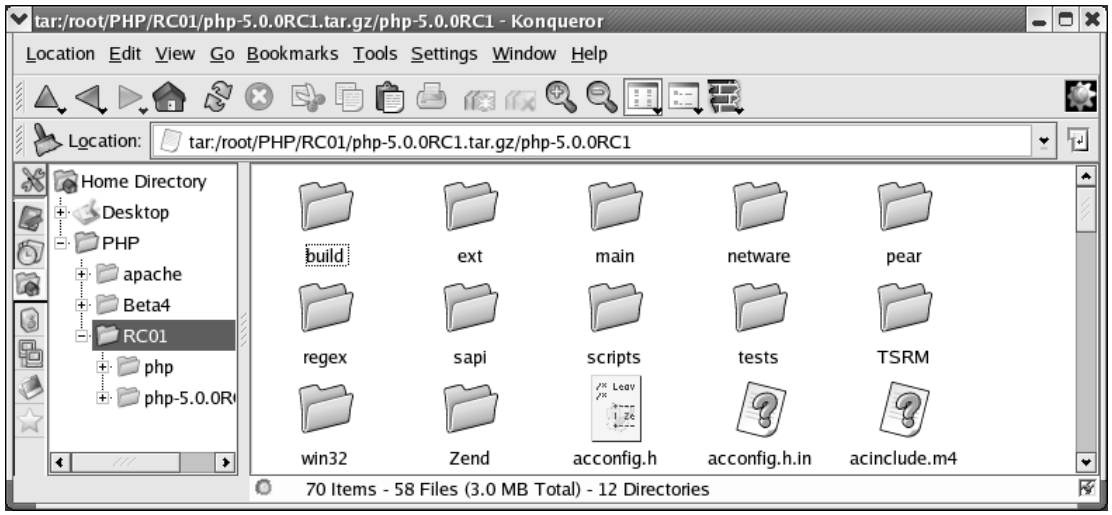

Figure 1-4

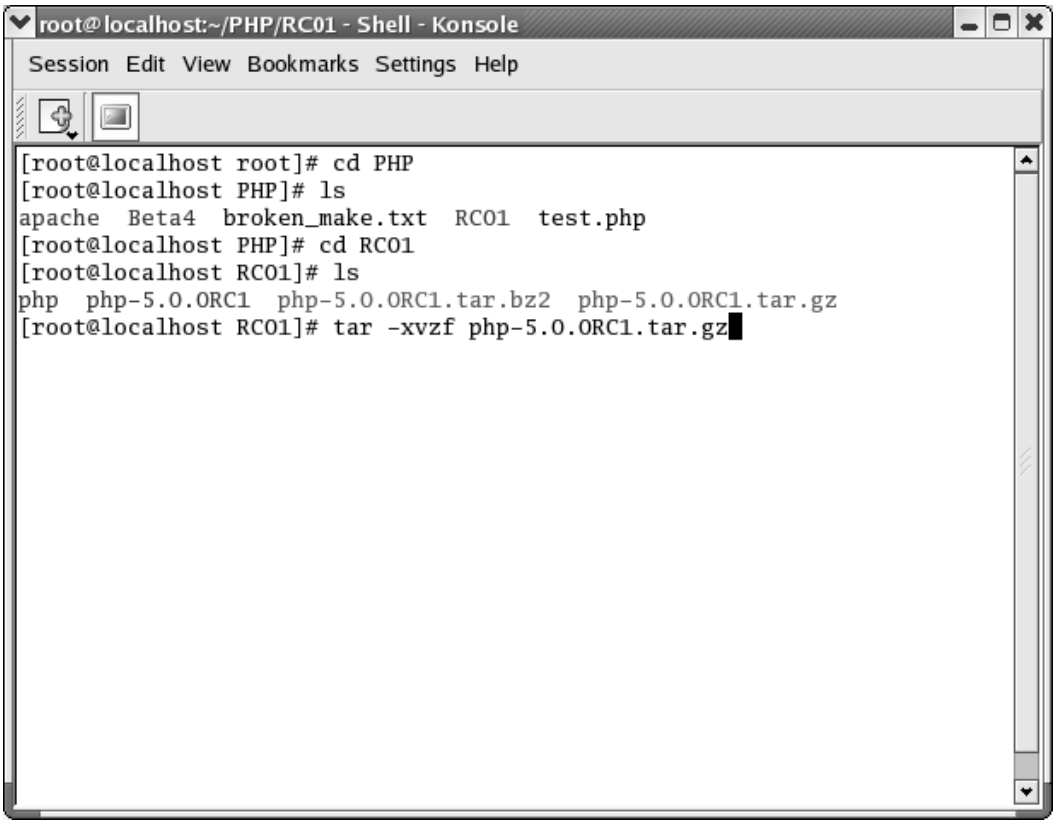

Figure 1-5

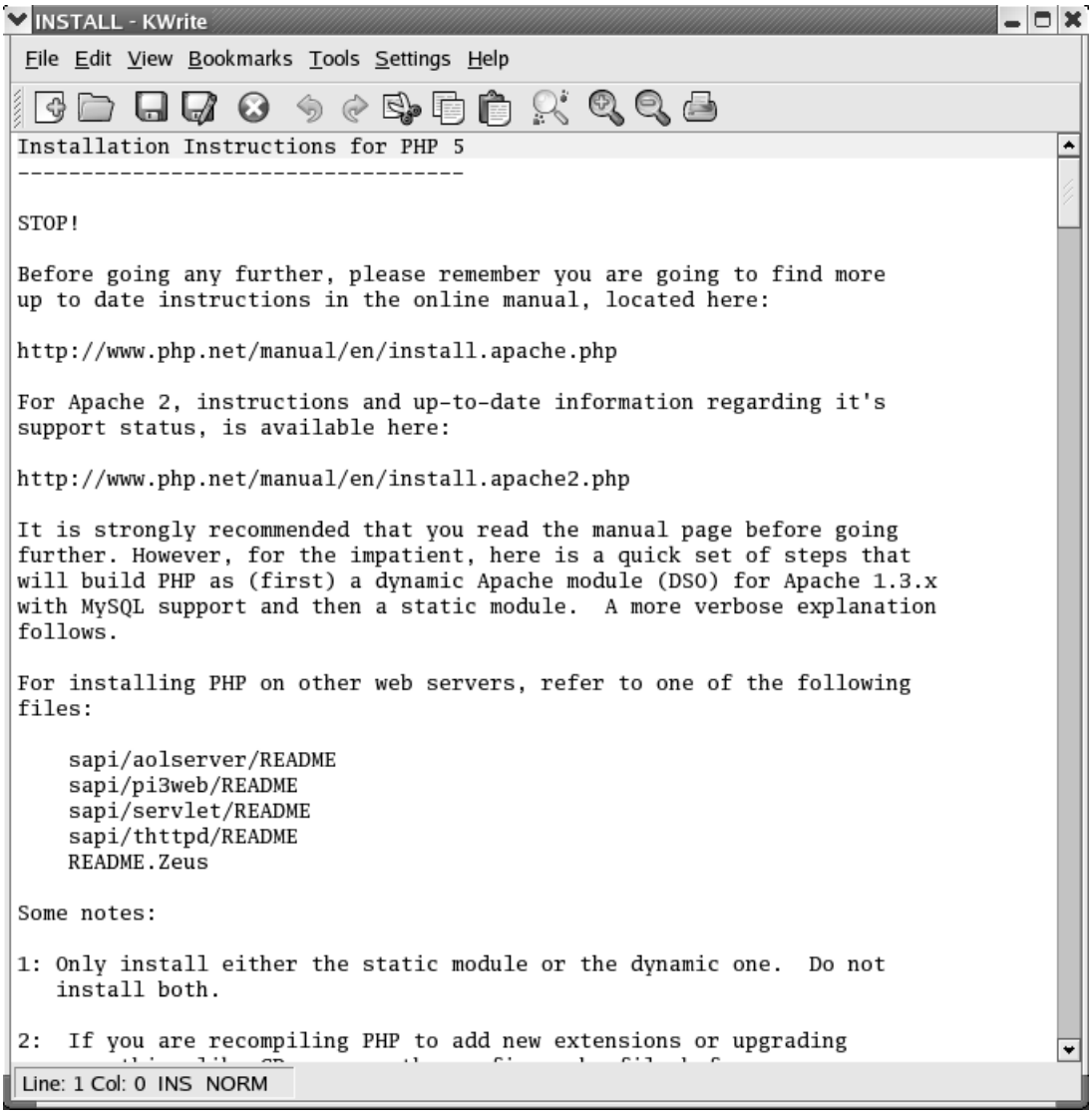

Figure 1-6

Checking Apache for DSO installation

You must have Apache installed and set up for dynamic modules before compiling PHP5 as a DSO. Use the following command from the terminal to make sure Apache is ready:

```
httpd -l
```

You see a terminal window like the one shown in Figure 1-7.

As long as `mod_so.c` is present, you're OK to proceed.

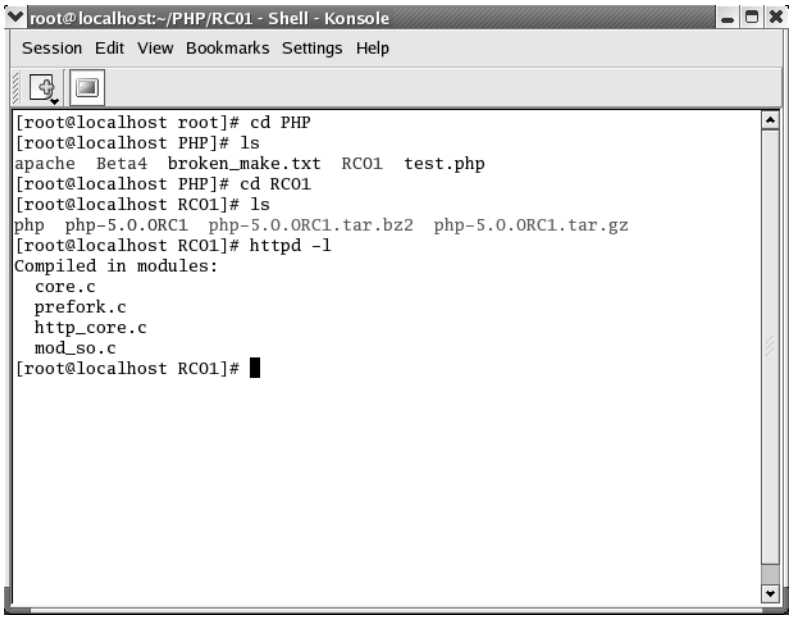

Figure 1-7

Running the configure Script

Within your PHP5 directory (probably named something like `php-5.0.0RC1.`), you'll find a shell script named `configure`. This script accepts arguments to control the features that PHP will support. You'll need to run this script to configure compilation, not PHP5 itself (PHP configuration with `php.ini` will come later).

Commands Available for the configure Script

The default configure is to compile PHP5 as a CGI binary. Use the `–with-apache` option to compile PHP5 as a static module; use the `–with-apxs` option to compile PHP5 as a DSO. For this installation, use the `–with_apxs` option (actually `–with_apxs2` because you're running Apache 2).

Here are some of the command line arguments that you may use when you compile PHP5. The configure command is `./configure` (the `./` lets it run) followed by a single space and then as many of the following options as you like:

- `–enable-track-vars`: Automatically populates associative arrays with values submitted as part of `GET` and `POST` requests or provided in a cookie.
- `–with-gd = /path/to/directory`: Enables support for the GD library that allows you to create dynamic GIF and PNG images from your scripts. You'll either want to compile with this or add this module later to do the graphics work in Chapter 16.
- `–with-mysql = /path/to/directory`: With MySQL support.
- `–with-pgsql = /path/to/directory`: With PostgreSQL support.

For your quick install use only –with-mysql and –with-apxs2. If you get any error messages telling you something couldn't be found, provide the full path to the folder in which the appropriate files can be found. For example, our configure command found the path to mysql. If it hadn't we would have provided the full path to mysql as part of the command to run the configure script.

Other Configure Options

There are many more command line arguments that you can use as well. For example, type in the command

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

and you'll see the complete range of arguments that you can use, along with their descriptions.

Performing the QUICK INSTALL (DSO)

The quick installation in the PHP INSTALL text file recommends starting with just two commands: –with-mysql and –with-apxs. Run the configure script like this for quick installation:

```
./configure --with-mysql --with-apxs2
```

You need to use –with-apxs2 rather than –with-apxs because you're running a later version of Apache. The script actually came back and informed me of this when it ran, which was very helpful. If you read through the commands that appear in your terminal as the script runs, you'll see that it does quite a few such checks as it gets ready for the make command.

After the configure script has run, you need to enter two more commands:

```
make
make install
```

Install makes a directory in /user/local/lib named php where it places a copy of PEAR (php Extension Add-on Repository) and the php.ini file. The Location bar on the screen in Figure 1-8 shows that the main panel's contents are in the php directory.

Running Additional Configure Options

You may use some of the other options of the configure script to compile PHP5 with enable_track_vars, with-gd, and with_pgsql. But if you'd like to use the configure options for gd (the graphics module) and pgsql (the database) you must make sure these programs are also loaded for everything to work properly, and you must provide the full path to the installations as required.

Running as a CGI

PHP5 is compiled as a module if you used the configure script with the –with-apache or —with-apxs2 options. It's compiled as a CGI executable binary file if you used the configure script without any reference to Apache or apxs. And if you compiled PHP5 as a CGI, then when you installed PHP5, it most likely put the actual binary in /usr/local/bin. Copy the CGI executable binary to your cgi-bin directory using:

```
cp /usr/local/bin/php /usr/local/apache/cgi-bin/php.cgi
```

This enables Apache users to run different PHP-enabled pages under different user-ids. However, CERT advisory CA-96.11 recommends against placing any interpreters (such as PHP5) into cgi-bin because

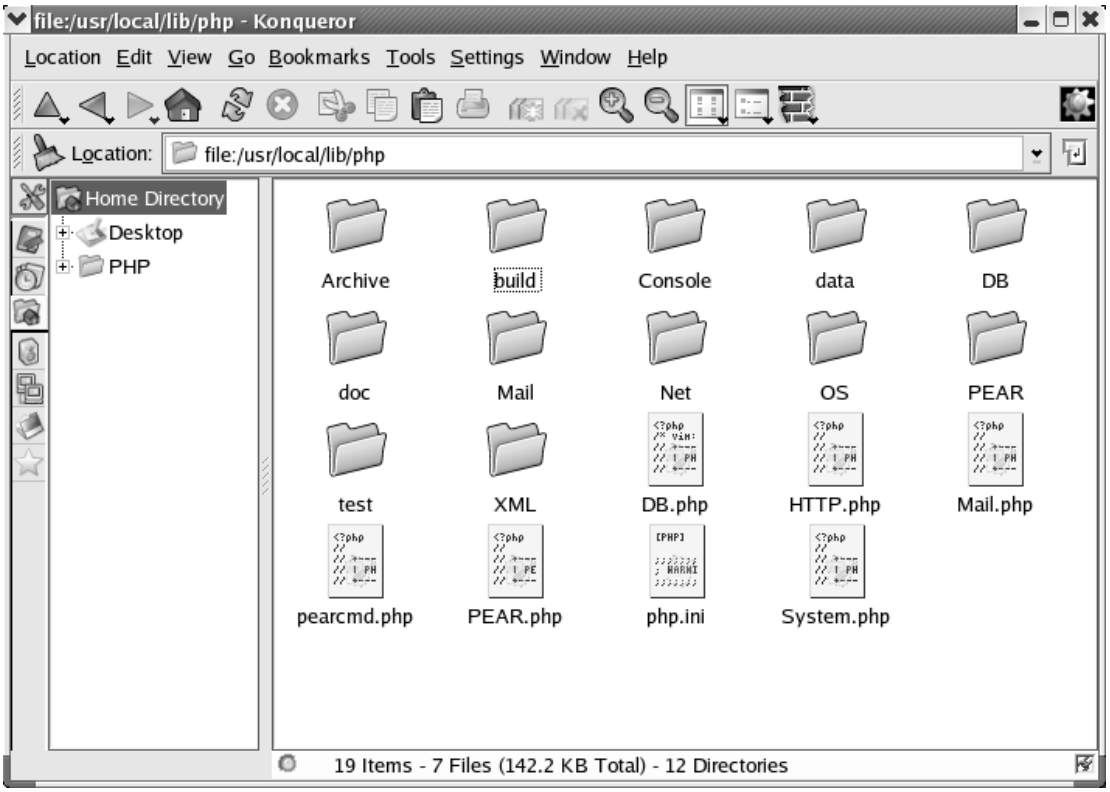

Figure 1-8

attacks such as accessing system files and Web documents can be launched against your server. When you compile PHP5 with the `–with-apache` option, you will have a Server Application Programming Interface (SAPI), which provides a higher level of performance and security over the CGI method.

Setting up Apache for PHP

To install Apache, use RPMs or download and compile the source code. But Apache probably comes with your Linux distribution, and may already be properly installed. On my Red Hat Fedora installation, Apache was already installed, and all that was needed to verify this was go to the Red Hat button, choose System Settings ➪ Server Settings ➪ Services, and look for `httpd`, as shown in Figure 1-9.

Httpd (all lowercase) means HTTP daemon, and daemon is the name of services running in the background on Linux machines. So httpd means the HTTP daemon running in the background, for example, the Web server.

If you're using Linux visually (running KDE, for example), click the httpd service to see whether it's running. If it isn't, start it, and then open your browser and enter `http://localhost`. You should see a Web page such as the one shown in Figure 1-10.

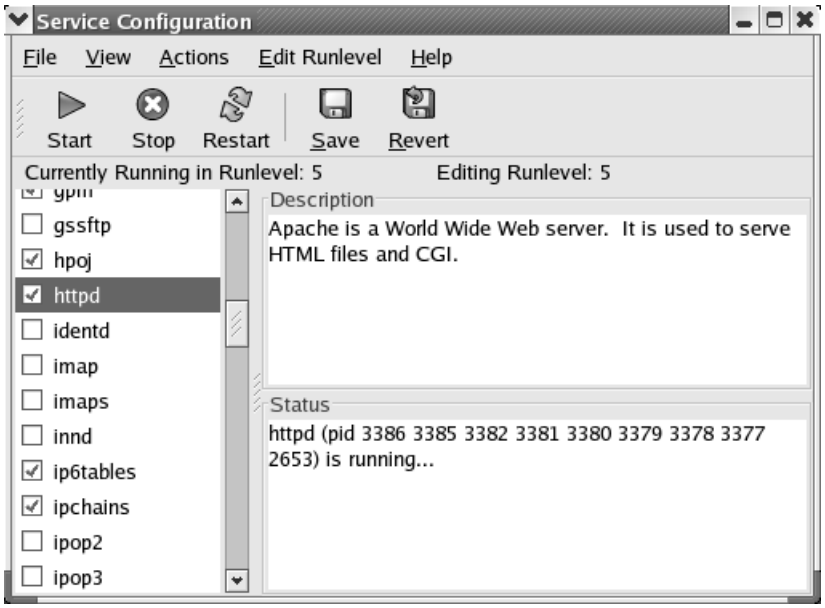

Figure 1-9

If Apache is not already installed and running, you can use these commands for installing it from the Terminal:

```
lynx http://httpd.apache.org/download.cgi
gzip -d httpd-2_0_NN.tar.gz
tar xvf httpd-2_0_NN.tar
./configure --prefix=PREFIX
make
make install
vi PREFIX/conf/httpd.conf
PREFIX/bin/apachectl start
```

You must replace the `NN` with the minor version number, and `PREFIX` with the correct path in which you'd like to install Apache (the default is `/usr/local/apache2`).

Configuring Apache to Run PHP5

If PHP5 is installed as a DSO (as the example installation was), you need to check the Apache configuration file (named `httpd.conf`) to make sure it has several entries. In the Fedora installation, you can find the `httpd.conf` file in the /etc/httpd/conf folder. Open any text editor and let's modify the `httpd.conf` file.

First, ensure that PHP is enabled on your Apache server. Look for a lot of lines that begin with the word `LoadModule`. Among them you should find a line like this:

```
LoadModule php5_module /usr/local/apache/lib/libphp5.so
```

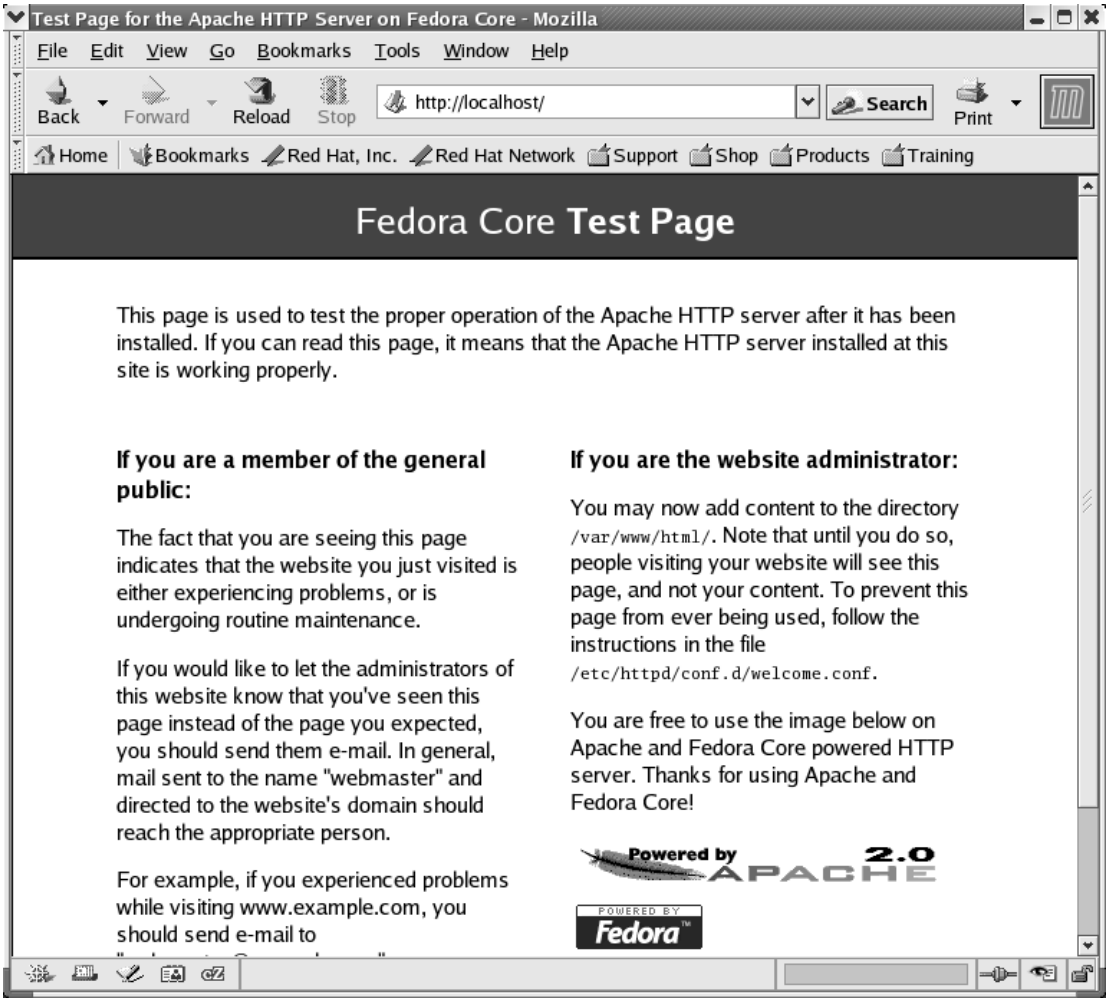

Figure 1-10

If there isn't such a line, you need to add one; or if the path is wrong, you need to correct it. Find out where the PHP compile put your `libphp5.so` file. Open Konqueror and choose Tools ➪ Find file. My installation put the `libphp5.so` file in

```
/usr/lib/httpd/modules
```

You need this information to tell Apache how to run PHP scripts. In the Apache configuration file `httpd.conf`, add the `LoadModule` instruction to load PHP5. Put it after any of the other `LoadModule` lines, using the path you just obtained:

```
LoadModule php5_module      /usr/lib/httpd/modules/libphp5.so
```

Now that Apache knows how to load PHP5, you need to activate PHP5 in Apache. There's a section farther down the file consisting of a lot of lines beginning with `AddModule`. If you find a

`ClearModulesList` line in the file, you need to add the following line to the file. Although it doesn't matter where you put it, it makes sense to locate it near other `AddModule` lines for easy access in the future:

```
AddModule mod_php5.c
```

The `AddModule` line is not required unless you have a `ClearModulesList` line.

Finally, you tell Apache how to recognize a PHP program file by its extension. Further down the document are some directives that begin `AddType`. To the end of these, add the following line:

```
AddType application/x-httpd-php .php
```

This tells Apache that all files that end in `.php` are PHP programs. Now you're done, so save the file.

Starting or Restarting Apache

Check to see if Apache is running by going into Services once again and checking the httpd service. If it's not running, start it. Verify that it's running by opening the `http://localhost` test page. If everything seems to be working OK (and you're not also installing on Windows), move on to the "*Testing Your Installation*" section to test that PHP is working properly.

The majority of Web hosting computers runs some version of Linux, such as Debian, RedHat, FreeBSD, and so on. The Web server of choice for these machines is Apache. PHP is compatible with Linux and Apache, so you can install and configure PHP on these systems. However, if you are not in charge of the Web hosting computer (and many times you won't be) you'll probably have little control over the installation and configuration. If you find yourself in this position (for example, if you've been hired to work on an existing Web site running on someone else's server) you can simply verify the operating system, Web server software, and PHP version so you can cope with whatever you've got as you develop your PHP programs.

Installing PHP5 on Windows 2000/Internet Information Server (IIS) 5

Before beginning the installation process, let's take a look at IIS. If it's been properly installed using the default settings, it should already be running. Select Start ➪ Programs ➪ Administrative Tools ➪ Services and scroll down to World Wide Web Publishing Service to see if IIS is currently running. If it isn't, start it up

If you need to install IIS, go to Settings ➪ Control Panel and open Add/Remove Programs. Then click the Components button to find the list of components that can be installed in Windows 2000, including Internet Information Server. Select IIS and then click the Details button to see all the services (such as FTP, SMTP, and so on) that can be installed. Choose any you'd like, but be sure to include World Wide Web Publishing, and click the Finish button. IIS should be installed and running.

You can examine the installation of IIS by opening its documentation from your browser, using `http://localhost/iisHelp` as the URL. You should see something like Figure 1-11, which shows IIS 5.0 running on Windows 2000.

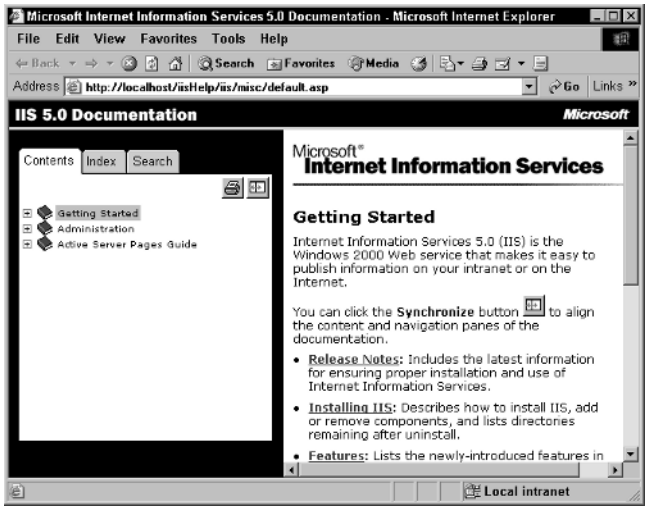

Figure 1-11

Now let's take a look at the Internet Service Manager, a console application for managing IIS, to see how IIS has been configured. Select Start ➪ Programs ➪ Administrative Tools ➪ Internet Services Manager. The IIS snap-in displays in Microsoft Management Console (MMC), as Figure 1-12 shows.

The MMC provides an easy way to examine and manage the services provided by IIS, as well as the Web sites and FTP sites set up under IIS. Figure 1-13 shows the hierarchical view of the default installation of IIS:

To do the installation of PHP you need to turn off Internet Information Services and then restart it after you're done making some changes. Right-click on the Default Web site and chose Stop from the short-cut menu. When you've got your installation of PHP files complete, you'll turn the Web server back on. For now, close the Internet Services Manager.

Downloading PHP5

To get the most recent version of PHP5, go to `www.php.net` and find the downloads section, as shown in Figure 1-14 (it may look a bit different by the time you read this).

Download the Windows binary file (it's zipped) and set it aside. Create a folder (such as `C:\PHP5RC01`) on your hard drive, put the downloaded file in the folder, and then unzip the file. You should end up with something like Figure 1-15 (as seen in Windows Explorer).

The filenames are current at the time of this writing, but of course may change by the time this book is published and in your hands. But the version of PHP5 that this book is using is very nearly ready for final release, so don't despair—your PHP5 installation should work just the same as the book's does. And while your installation of Windows may be on your C: drive, the book's is on the D: drive. You may have to change some of the path names to reflect your C: drive (or whatever drive Windows is loaded on) to make your installation work.

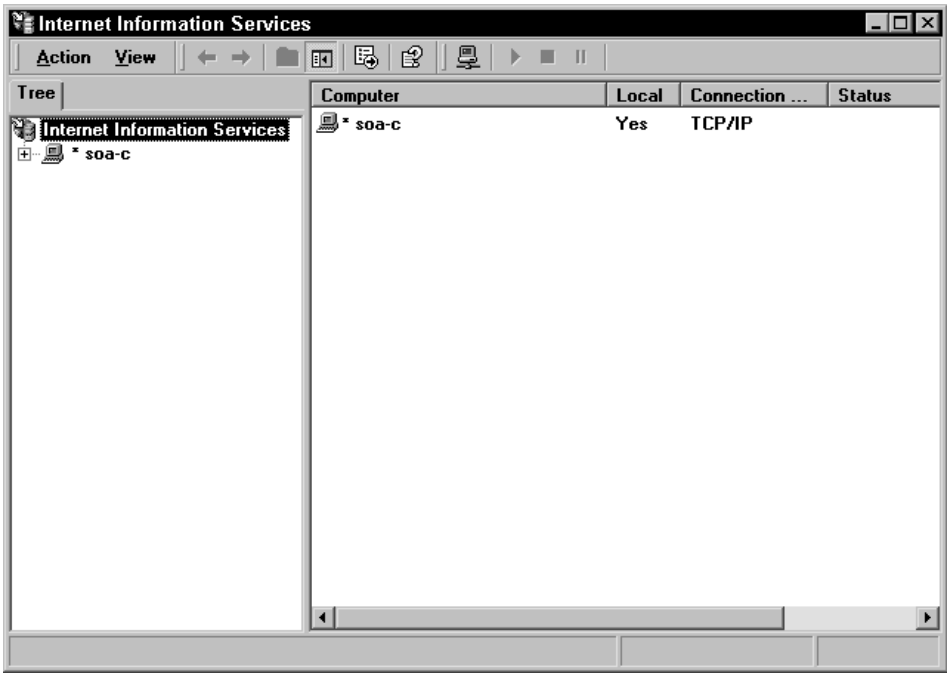

Figure 1-12

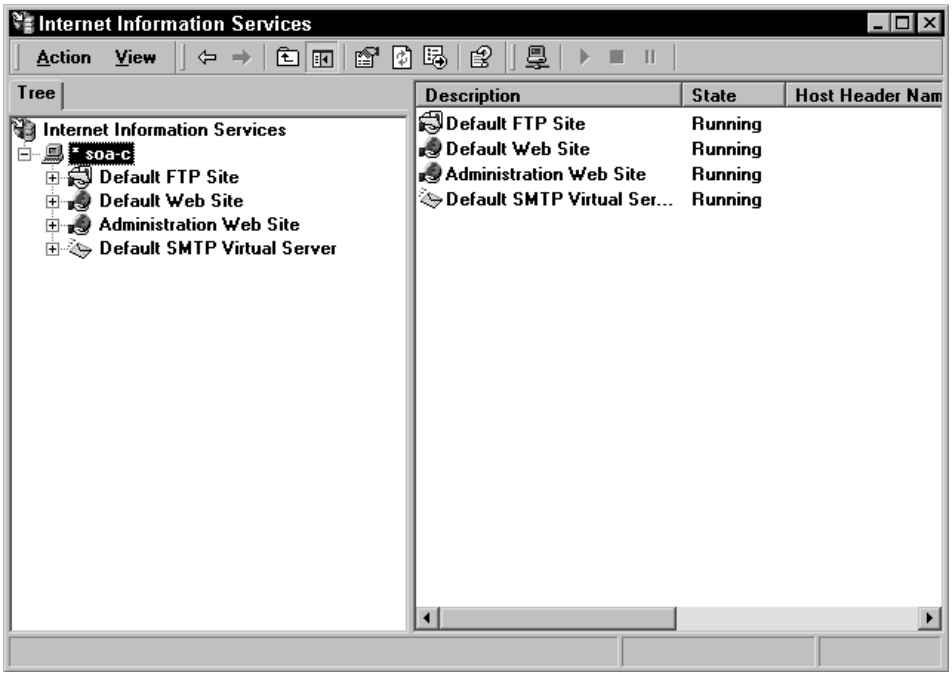

Figure 1-13

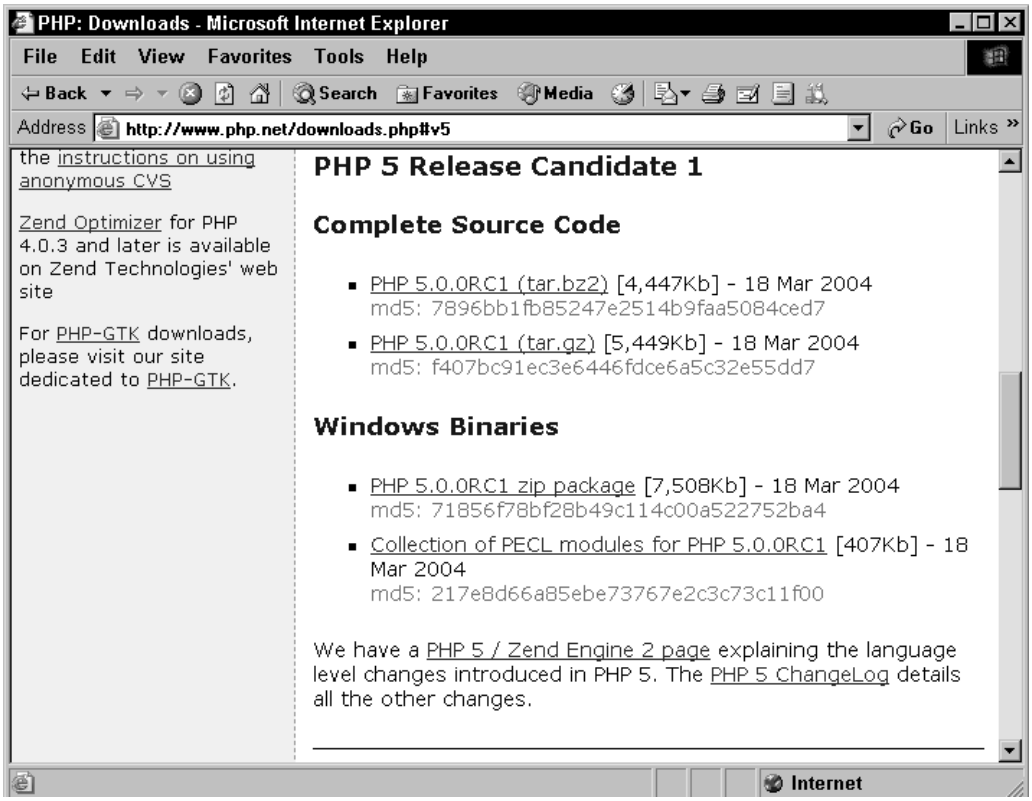

Figure 1-14

Now, the directory you created (such as PHPRC01) contains several subdirectories, and a few text files. It also contains a program file called `php.exe`, which you won't actually be using, and a library file called `php5ts.dll` (in the dev folder). Copy this `.dll` to your D:\WINNT\System32 directory (using the correct drive letter for your machine). Now copy the rest of the `.dll` files from here to your D:\WINNT\System32 directory as well. If Windows complains that you've already got a file by one or other of these names, then keep your old one—don't overwrite it with the newly downloaded files.

If you prefer not to copy all your `.dll`s into your System32 directory and you are installing PHP as an SAPI (like this book does), you may create another folder for them and make a change to your PATH environment variable so that they can be found. (A PATH environment variable contains a list of directories—paths—in which Windows will look for things, such as your `.dll`s. If, for example, you have a C:\php5\dlls directory, you'd add the string `C:\php5\dlls` to your variable, and then anything in that folder could be used by your Windows environment. To set the variable, select Start ➪ Control Panel ➪ System ➪ Advanced ➪ Environment Variables, and locate and set the PATH variable.)

php.ini and Extensions

As mentioned earlier, the `php.ini` file contains instructions to PHP such that, when it is running, certain configuration settings are in place and certain extensions to PHP are running. Configuration settings are

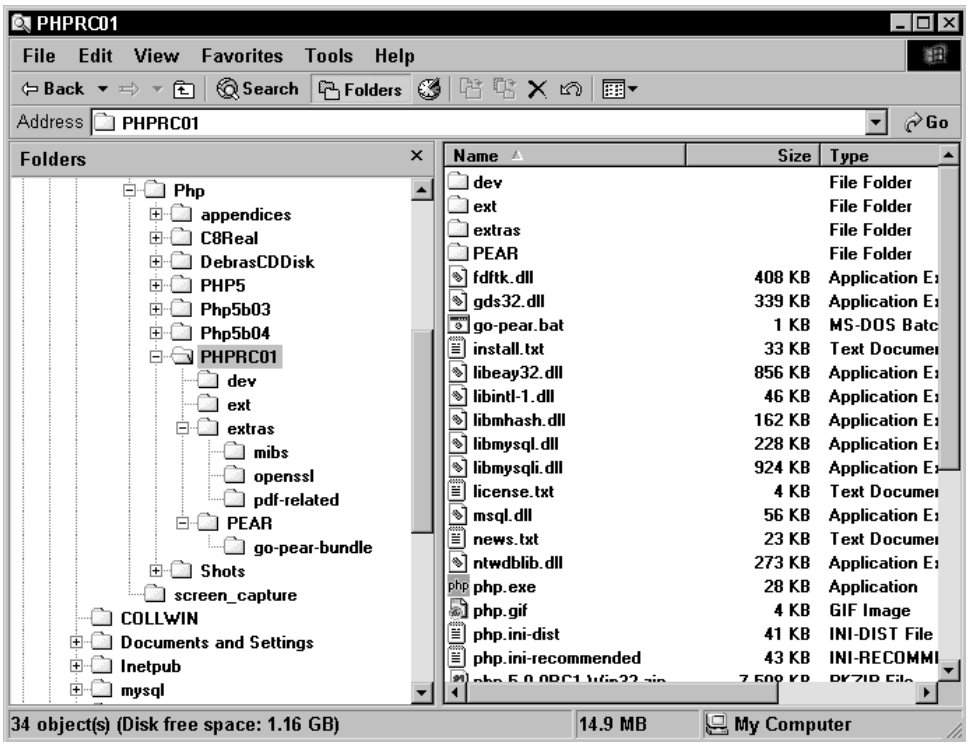

Figure 1-15

like switches, turning a variety of PHP behaviors on and off. Extensions provide added or enhanced built-in capabilities to PHP.

At the top of your PHP directory should be files called `php.ini-dist` and `php.ini-recommended`. Copy the `php.ini-dist` file to D:\WINNT (using the appropriate drive letter), rename it `php.ini`, and open it up with Notepad. Scroll down the document until you find a line that looks like:

```
extension_dir = C:\php\extensions ; directory in which the loadable extensions
(modules) reside
```

Make sure that this path is the correct path to the extensions directory of the unzipped PHP5 installation. If it isn't, change it to point to the right place (look for an ext folder under your unzipped PHP folder). The extensions directory is the one that contains a large number of files whose names begin with `php_` and end with `.dll`.

The next section in your `php.ini` file tells PHP which extensions to load. There are semicolons at the beginning of all the lines that load extensions you don't need—a semicolon means that PHP will ignore the directive on that line. Remove the semicolon from `extension=php_gd.dll`, so that you have text like this:

```
;extension=php-filepro.dll
extension=php-gd.dll
;extension=php_mssql.dll
```

This gives you access to the functionality of the GD library, which enables you to generate images using PHP programs (you'll see how in Chapter 16, "*Generating Graphics*"). Save your modified `php.ini` file.

Again, select Start ➪ Programs ➪ Administrative Tools ➪ Internet Services Manager, and open the hierarchy of services. Right-click Default Web Site, and bring up its Properties (see Figure 1-16).

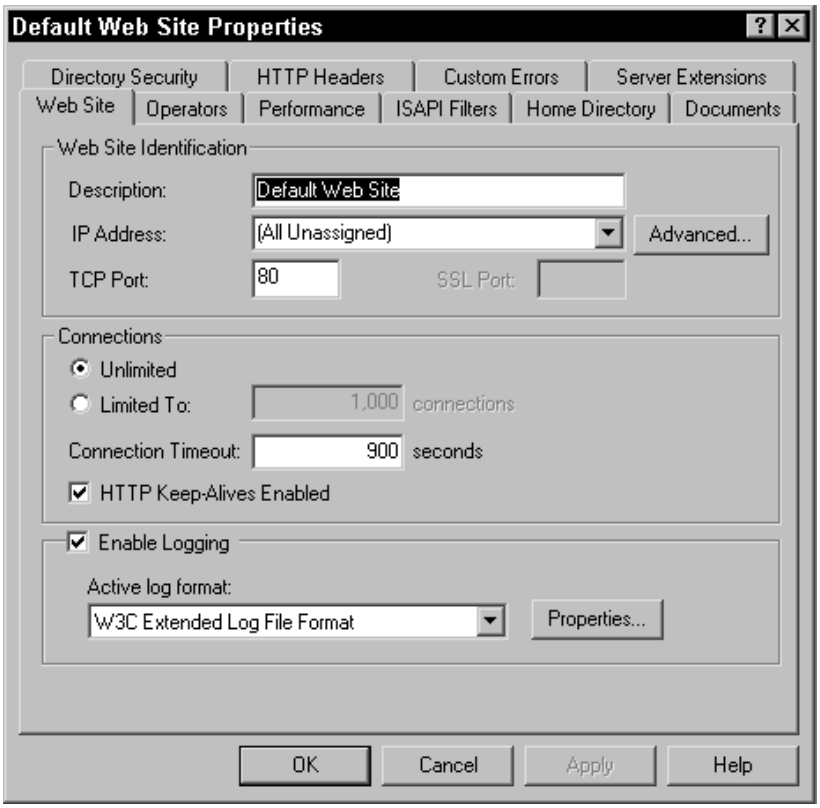

Figure 1-16

There are two changes to make. First, you need to register the PHP5 ISAPI filter, because you want to install PHP with its own SAPI rather than as a CGI binary. Click the ISAPI Filters tab. Click the Add button, and create a new filter called PHP. The folder of PHP files you downloaded contains `php5isapi.dll`, a PHP ISAPI filter, in the sapi directory. Put in the correct path for your `php5isapi.dll` file, as shown in Figure 1-17.

And second, you need to tell IIS which files to apply the PHP5 filter to. You want it to treat all files that end with `.php` as PHP programs. On the Home Directory tab, click the Configuration button. Click the Add button in the next dialog box, and the Add/Edit Application Extension Mapping dialog box opens (see Figure 1-18).

Click the Browse button and specify the path to `php5isapi.dll`. Tell IIS to apply it to `.php` files by entering `.php` in the Extension text box. Click OK, and then click OK again. Now you need to restart IIS. Close the Properties box, right-click the Default Web site, and choose Start from the shortcut menu.

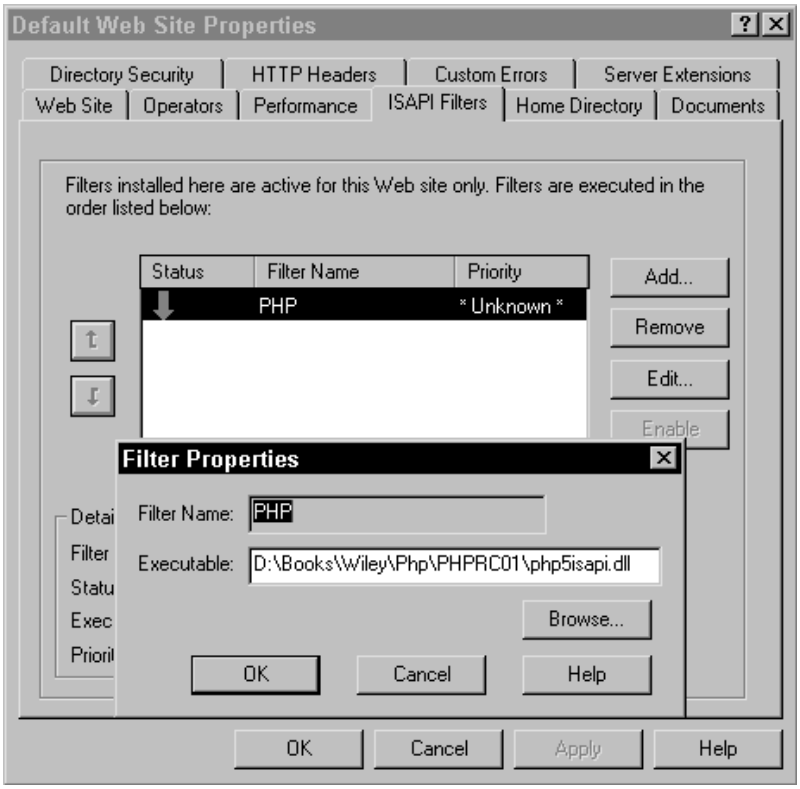

Figure 1-17

Provided the message in the MMC reports that the World Wide Web Publishing service was started successfully, you now have PHP5 installed. Remember the name of your Web site's root directory (the book's is D:\Inetpub\wwwroot).

Create a folder in the wwwroot folder. Name it anything you want (but something helpful, like `php_files`), and place files in it with a `.php` extension. These files will be processed through the PHP scripting engine when requested by a browser.

Now open your text editor and create a text file with the following code in it:

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

Save this file as `test01.php` (or something like that, so long as the filename extension is `.php`) in the folder you just created under the wwwroot folder. Open the file in your browser, using `http://localhost` plus the name of your new folder and the filename (for example, `http://localhost/php_file/myfile.php`). You should now see something like Figure 1-19 in your browser (although your PHP version number might be a bit different if you have a more recent release):

Assuming it works, you're now in business. If not, check out the following section.

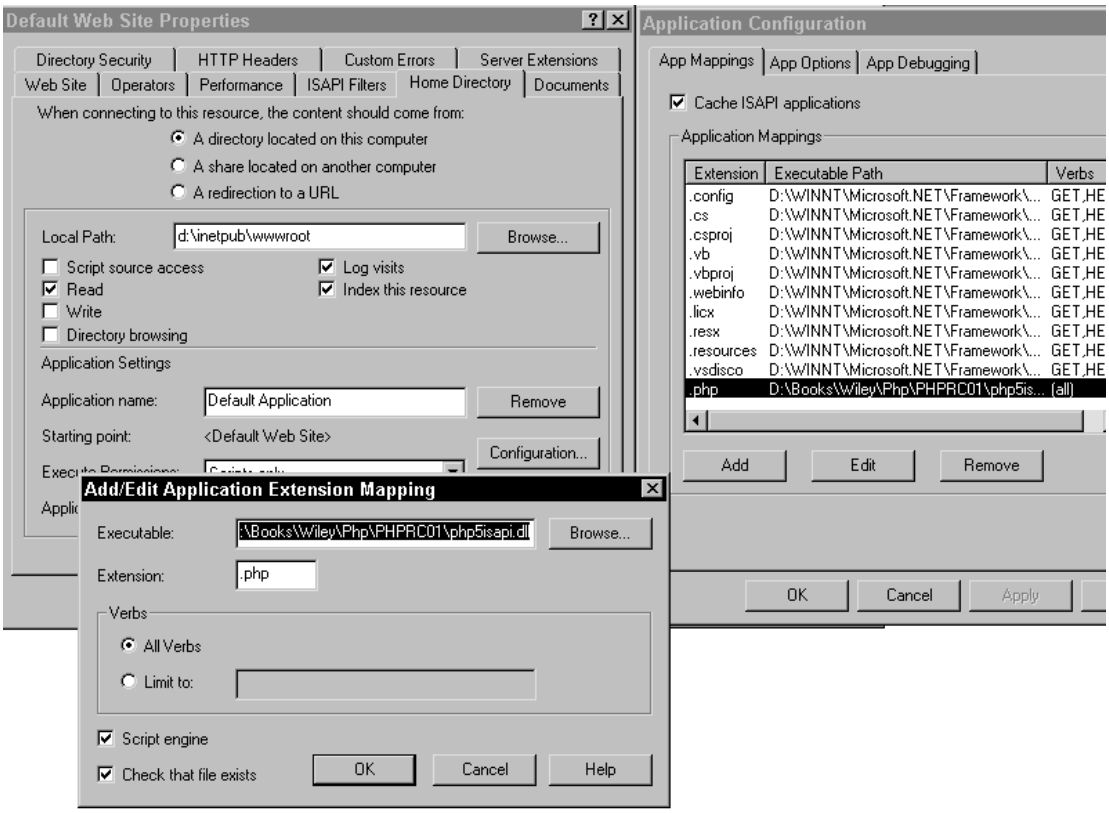

Figure 1-18

Testing and Troubleshooting

Testing your installation of PHP is really as simple as writing a small PHP program and running it. Create a small PHP file to use for tests. Write the following code in it:

```
<?php
echo "Hey, it worked";
?>
```

Save the file as `test02.php` in any folder within wwwroot (or an appropriate folder if you happen to be running some other OS/Web server combination than Windows 2000/IIS).

Open the file in your browser. You should see the words "Hey, it worked" in your browser. If you see "Page cannot be displayed," there's a problem with your PHP installation or with finding the file, or with the Web server. If you see something that talks about a parse error, you may have made a mistake in entering the code. Coding errors are discussed in more detail in the next few chapters, but in the next section you'll explore some ways you can troubleshoot your basic installation of PHP.

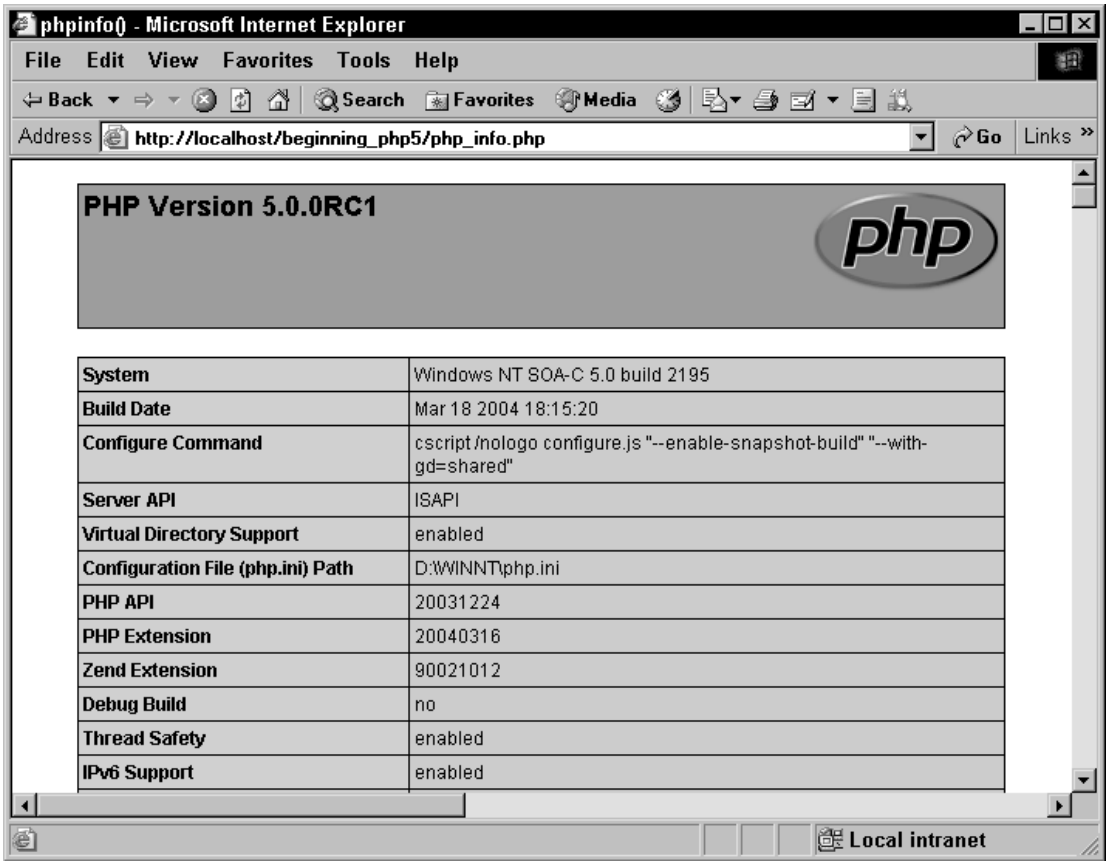

Figure 1-19

Did the file you tested work? Or are you reading this section right now because the file didn't work? Don't worry; it's very common for problems to occur anytime you try something new, especially when it comes to computers and programming. In fact, consider yourself lucky if it didn't work because you'll learn a lot more about PHP and programming from your mistakes than from your successes.

So let's start at the beginning. Troubleshooting and debugging comprise the process of identifying problems, deducing possible causes, logically isolating those causes until you've identified the most likely culprit, and then trying solutions. The end result is that the problem is fixed and if you've done a good job, you'll have fixed the problem (in an elegant, robust way) without causing other problems.

Chapter 5 covers debugging in more detail; for now, though, let's look at the steps for troubleshooting your installation of PHP:

1. Check that the Web server is on and running properly. In Windows 2000, do this by checking Services (Start ➪ Programs ➪ Administrative Tools ➪ Services), specifically Internet Admin Services and World Wide Web Publishing Service. Just because these are set to `Automatic`

within Services doesn't mean they are on. Stop and restart them if you need to reassure yourself. You can also check to make sure the Web server is running on the default Web site in Internet Services Manager. For Apache on Linux, check the httpd service (you can also test it by entering `http://localhost` in your browser).

2. Place a simple HTML Web page in the wwwroot folder, making sure that it has `.htm` or `.html` as the extension for the filename (such as `test01.htm`), and bring the page up in your browser. Make sure you are using `http://localhost/test01.htm` to bring up the file, not the file location (such as `D:\inetpub\wwwroot\test01.htm`).
3. If the HTML Web page displays properly, you can be sure your Web server is functioning. This implies (assuming your PHP page cannot be displayed) that something is wrong with your PHP installation. Of course, if you see other messages (such as 404 Page Not Found) you may simply not be finding the file properly, so you'd need to take a second look at the file name you chose, the name of your Web folder, and so on.
4. If you think that something is wrong with your PHP installation, reexamine the installation process you used, going carefully through each step, and make sure you placed all the PHP files in the places they belong. Pay particular attention to the names of Windows and System folders because these may differ depending upon your installation of Windows or Linux.
5. Check file permissions. File permissions are very important in Linux systems, and to a lesser degree in Windows 2000 or desktop Windows operating systems. You should be logged in as root or administrator on Linux and Windows systems, and should be able to change permissions as necessary to run scripts from within Web server folders. For external hosting accounts any good FTP utility can modify file permissions on Linux systems.

It's likely that following these steps will isolate the problem so that you can fix it.

Configuring PHP

During installation you modified the `php.ini` file to affect the way PHP runs and what features it includes. Appendix F, "*Configuring PHP5*," at the end of the book discusses the major settings in the `php.ini` file, as well as some of the extensions currently available for PHP, but in this section we'll go over a few of the most important PHP configurations settings and extensions.

php.ini

The `php.ini` file is parsed when PHP is first loaded and executed, so that PHP behaves (for any script running on that Web server) in a particular way. All lines that are not preceded by a semicolon are working commands; think of everything else in the file as a comment. Following is the text of several sections of the `php.ini-recommended` file. The settings shown are important because they have a direct effect on how PHP behaves (under common operating circumstances), will affect your code, or may affect the security of your applications:

```
;;;;;;;;;;;;;;;;;;;
; Resource Limits ;
;;;;;;;;;;;;;;;;;;;
```

```
max_execution_time = 30   ; Maximum execution time of each script, in seconds
max_input_time = 60    ; Maximum amount of time each script may spend parsing
request data
memory_limit = 8M      ; Maximum amount of memory a script may consume (8MB)

; Whether or not to register the EGPCS variables as global variables. You may
; want to turn this off if you don't want to clutter your scripts' global scope
; with user data. This makes most sense when coupled with track_vars - in which
; case you can access all of the GPC variables through the $HTTP_*_VARS[],
; variables.
;
; You should do your best to write your scripts so that they do not require
; register_globals to be on; Using form variables as globals can easily lead
; to possible security problems, if the code is not very well thought of.
register_globals = Off
```

There's much more information regarding the settings in `php.ini` in Appendix F.

PHP Extensions

PHP extensions are programmatic capabilities that add to or enhance PHP's built-in capabilities for performing useful work in your PHP programs. Although no special extensions are used in the early chapters of this book, you'll run across some later. Meanwhile, Appendix F covers all of the available extensions.

Caching

Caching is a method by which some results are stored temporarily, so that all processing does not have to be repeated each time a new request is made to the server. One potential disadvantage of running all your code on the server is that if the client (or some machine in between the end user and your site) has a cache going, the user may not get the most recently processed page To work around caching (at least for most browsers and servers), you can place the following code in your programs:

```
<?php
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache");
header("Expires: Mon,26 Jul 1997 05:00:00 GMT");
?>
```

The first line works well with HTTP 1.1, the second line with HTTP 1.0, and the third works by specifying a date in the past (more about HTTP in Chapter 2).

Summary

This chapter covered a bit of the history of PHP and several ways to install PHP alongside common Web server software.

You learned how to install PHP on both Windows and Linux platforms, some of the differences about installing PHP as a CGI or as a separate module, the basic meaning of some PHP settings, where PHP

files go upon installation, and how to test your installation. The basic definition of troubleshooting and debugging was covered, as well as a series of steps to take if your installation of PHP isn't working.

Exercise

You're not really ready to begin programming exercises at this point, so let's do a different type of exercise that you'll find handy whenever you have to install with or work with PHP on a new platform. To complete this exercise, do the following:

Create a document that summarizes all of the following:

- What are the hardware capabilities of the computer on which PHP is running? Describe the CPU, hard drive, RAM, and so on, and any particular limitations you perceive.
- What operating system is running on the hardware? List the version, as well as any current patches and known bugs.
- What Web server software is running on the machine? List the version, patches, and known bugs. Also, list how the Web server is configured, the root folder, how PHP is set up to work with the Web server, and any file permissions you've set.
- What version of PHP was installed? List the version, the files installed, the folders in which they were installed, and any Registry settings that were set or created to support the PHP installation.
- What configuration settings were set or changed (from the default) to install PHP? List them.
- What extensions were enabled? List them all, and why you enabled them.