CHAPTER 1

Setting Up the PHP Development Environment

Getting a working development environment put together can be intimidating, especially for the absolute beginner.

To follow along with the project in this book, you'll need to have access to a working installation of Apache, PHP, and MySQL, preferably on your local machine. It's always desirable to test locally, both for speed and security. Doing this both shelters your work-in-progress from the open Internet and decreases the amount of time spent uploading files to an FTP server and waiting for pages to reload.

Why You Need Apache, MySQL, and PHP

PHP is a powerful scripting language that can be run by itself in the command line of any computer with PHP installed. However, PHP alone isn't enough in order to build dynamic web sites.

To use PHP on a web site, you need a server that can process PHP scripts. Apache is a free web server that, once installed on a computer, allows developers to test PHP scripts locally; this makes it an invaluable piece of your local development environment.

Additionally, dynamic websites are dependent on stored information that can be modified quickly and easily; this is the main difference between a dynamic site and a static HTML site. However, PHP doesn't provide a simple, efficient way to store data. This is where a relational database management system like MySQL comes into play. This book's examples rely on MySQL; I chose this database because PHP provides native support for it and the database is free, open-source project.

Note An open-source project is available for free to end users and ships with the code required to create that software. Users are free to inspect, modify, and improve the code, albeit with certain conditions attached. The Open Source Initiative lists 10 key provisions that define open-source software; you can view this list at www.opensource.org/docs/osd.

Drilling Down on PHP

PHP is a general-purpose scripting language that was originally conceived by Rasmus Lerdorf in 1995. Lerdorf created it to satisfy the need for an easy way to process data when creating pages for the World Wide Web.

■ **Note** PHP was born out of Rasmus Lerdorf's desire to create a script that would keep track of how many visits his online resume received. Due to the wild popularity of the script he created, Lerdorf continued developing the language. Over time other developers joined him in creating the software; today, PHP is one of the most popular scripting languages in use on the Internet.

PHP originally stood for "Personal Home Page" and was released as a free, open source project. Over time, the language was reworked to meet the needs of its users. In 1997, PHP was renamed to the current "PHP: Hypertext Preprocessor."

At the time I write this, PHP 5.2.9 is the latest stable release available, but versions 5.3 and 6 are both scheduled for release in the near future. PHP 4 is still in use on a number of servers, but support has been discontinued. Many hosting companies let developer use either PHP 4 or PHP 5 on their sites.

Stable/Production vs. Development Releases

Many software products will have a *development release* available, alongside the current, stable release. This is a way for the development community to test an upcoming version of a product; this helps the product's creators work bugs out of the system.

After a proposed release has been tested to a satisfactory level, its creators mark it as the current production release. Users reasonably expect the production software they use to be free of major defects or bugs; calling a version a stable/production release is a way for the product's creators to let potential users know that, in the opinion of the product's creators, all major issues have been worked out, and that it is safe to use this software for mission-critical applications.

How PHP Works

PHP is generally used as a server-side scripting language; it is especially well-suited for creating dynamic web pages. The scripting language features integrated support for interfacing with databases such as MySQL, which makes it a prime candidate for building all manner of web applications, from simple personal web sites to complex enterprise-level applications.

Unlike HTML, which is parsed by a browser when a page loads, PHP is *preprocessed* by the machine that serves the document (this machine is referred to as a server). All PHP code contained with the document is processed by the server *before* the document is sent to the visitor's browser.

PHP is a scripted language, which is another great advantage for PHP programmers. Many programming languages require that you compile files into machine code before they can be run, which is a time-consuming process. Bypassing the need to compile means you're able to edit and test code much more quickly.

Because PHP is a server-side language, running PHP scripts on your local machine requires installing a server on your local machine, as well. The examples in this book rely on the Apache web server to deliver your web pages.

Server-Side vs. Client-Side Scripting

The Internet features two main kinds of scripting: *server-side* and *client-side.* Client-side scripting is comprised mainly of JavaScript, which is responsible for many of the web features that you can actually *see* happening, such as pop-up windows, some animations, and other site features like drop-down menus. The reason this is called "client-side" scripting because the code is executed on the user's machine, after the page has been loaded.

Using client-side scripts enables you to make changes to a page without requiring a page refresh; it also facilitates initial form validation and simplifies making improvements to the user interface.

However, using client-side scripts requires that the users have JavaScript turned on or that their browsers support the script you have written. This means you should not use client-side scripts for user authentication or the handling of anything sensitive, due to the user's ability to modify and/or disable your client-side scripts.

Server-side scripting is performed on the site's hosting server before the page is delivered to the user. This means that any changes that must be made by the script require a page refresh.

Using server-side scripting is great for user authentication, saving changes to database information, retrieving entries for display, and many other tasks.

The downside of server-side scripts lies mainly in the required page refresh. Because the script is processed before it is delivered to the browser, the user doesn't have access to the inner workings of the code. This makes server-side scripts the best choice for handling any sensitive information.

■ **Caution** Server-side scripting is better suited to handling sensitive information than client-side scripts, but you still must take care to protect sensitive data. We'll spend more time on basic security later in the book.

Serving web pages with Apache HTTP Server is the most popular web server on the web; it hosts nearly half of all web sites that exist today. Apache is an open-source project that runs on virtually all available operating systems.[1] Apache server is a community-driven project, with many developers contributing to its progress. Apache's open-source roots also means that the software is available free of charge, which probably contributes heavily to Apache's overwhelming popularity relative to its competitors, including Microsoft's IIS and Google's GWS, among others.

On the Apache web site (www.apache.org), Apache HTTP Server is described as "an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient, and extensible server that provides HTTP services in sync with the current HTTP standards."

What Apache Does

Like all web servers, Apache accepts an HTTP request and serves an HTTP response. The World Wide Web is founded on web servers, and every website you visit demonstrates the functionality of web servers.

I've already mentioned that, while HTML can be processed by a web browser, programming languages such as PHP need to be handled by a web server. Due to its overwhelming popularity, Apache is used for testing purposes throughout this book.

Store Info with MySQL

MySQL is a relational database management system (DBMS). Essentially, this means that MySQL allows users to store information in a table-based structure, using rows and columns to organize different pieces of data. This structure is similar to that of Microsoft's Access database.

The examples in this book rely on MySQL to store the information you'll use in your PHP scripts, from blog entries to administrator information; it is that approach that allows your site to be *dynamic*.

---

1 Wikipedia, "Apache HTTP Server," `http://en.wikipedia.org/wiki/Apache_(HTTP_server)`

■ **Note** *Blog* is short for *weblog*, which is an online journal for an individual or business.

Installing PHP, Apache, and MySQL (the Hard Way)

One of the biggest hurdles for new programmers is *starting*. Before you can write your first line of PHP, you must first download Apache and PHP, and usually MySQL, and then fight through installation instructions that are full of technical jargon you might not understand yet. This experience can leave many developers feeling unsure of themselves, doubting whether they've installed the required software correctly.

In my own case, this hurdle kept me from learning programming for *months*, even though I desperately wanted to move beyond plain ol' HTML. I unsuccessfully attempted to install PHP on my local machine not once, but *three* different times before I was able to run my first PHP command successfully.

Installation Made Easy

Fortunately, the development community has responded to the frustration of beginning developers with several options that take all the pain out of setting up your development environment, whether you create applications for Windows, Mac, or Linux machines. These options include all-in-one solutions for setting up Apache, MySQL, and PHP installations.

The most common all-in-one solution is a program called "XAMPP" (`www.apachefriends.org/en/xampp.html`), which rolls Apache, MySQL, PHP, and a few other useful tools together into one easy installer.

XAMPP is free and available for Windows, Mac, and Linux, so this book assumes you will use it as your development environment.

■ **Note** Most Linux distributions ship with one flavor or another of the LAMP stack (Linux-specific software that functions similarly to XAMPP) bundled in by default. Certain versions of Mac OS X also have PHP and Apache installed by default.

Installing XAMPP

Enough background: You're now ready to install XAMPP on your development machine. This process should take about five minutes and is completely painless.

■ **Note** A good habit to get into is to create separate *development* and *production* environments. A development environment is for testing projects for bugs and is generally sheltered from the world at large. A production environment is reserved for fully functional, publicly available projects.

Step 1: Download XAMPP

Your first task is to obtain a copy of the XAMPP software. Head over to the XAMPP site (`www.apachefriends.org/en/xampp.html`) and download the latest version (0.7.4 for Mac, 1.7.1 for Windows, and 1.7 for Linux at the time I write this).

Step 2: Open the Installer and Follow the Instructions

After downloading XAMPP, find the newly downloaded installer and run it. You should be greeted with a screen similar to the one shown in Figure 1-1.

■ **Note** All screenshots used in this book were taken on a computer running Mac OS X 10.4.11. Your installation might differ slightly if you use a different operating system. XAMPP for Windows offers additional options, such as the ability to install Apache, MySQL, and Filezilla (an FTP server) as services. This is unnecessary and will consume computer resources even when they are not being used, so it's probably best to leave these services off. Additionally, Windows users should keep the c:\xampp install directory for the sake of following this book's examples more easily.

Figure 1-1. *The introductory screen for the XAMPP installer on Mac OS X*

Click the Continue button to move to the next screen (see Figure 1-2), where you can choose the destination drive you want to install XAMPP on.

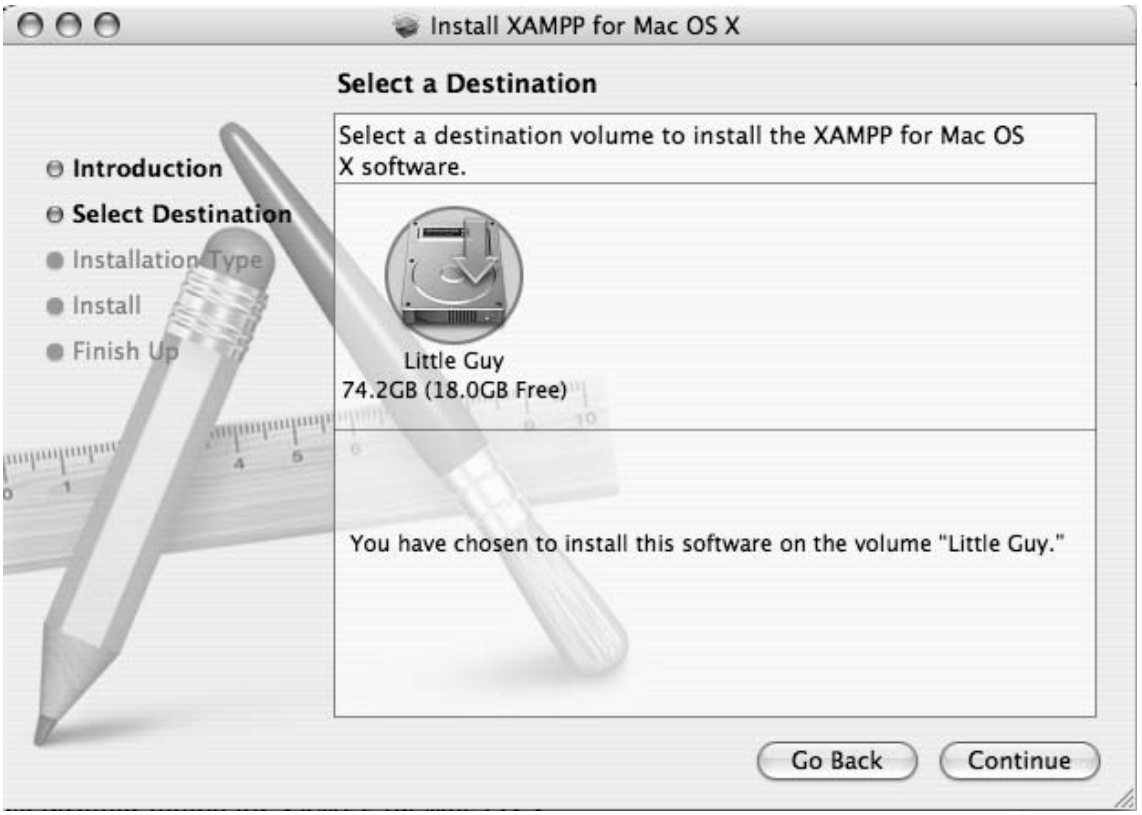

Figure 1-2. *Select a destination drive on which to install XAMPP*

The installation wizard's next screen (see Figure 1-3) asks what type of installation you prefer. This is your first time installing XAMPP, so the only available option is a basic installation of XAMPP.

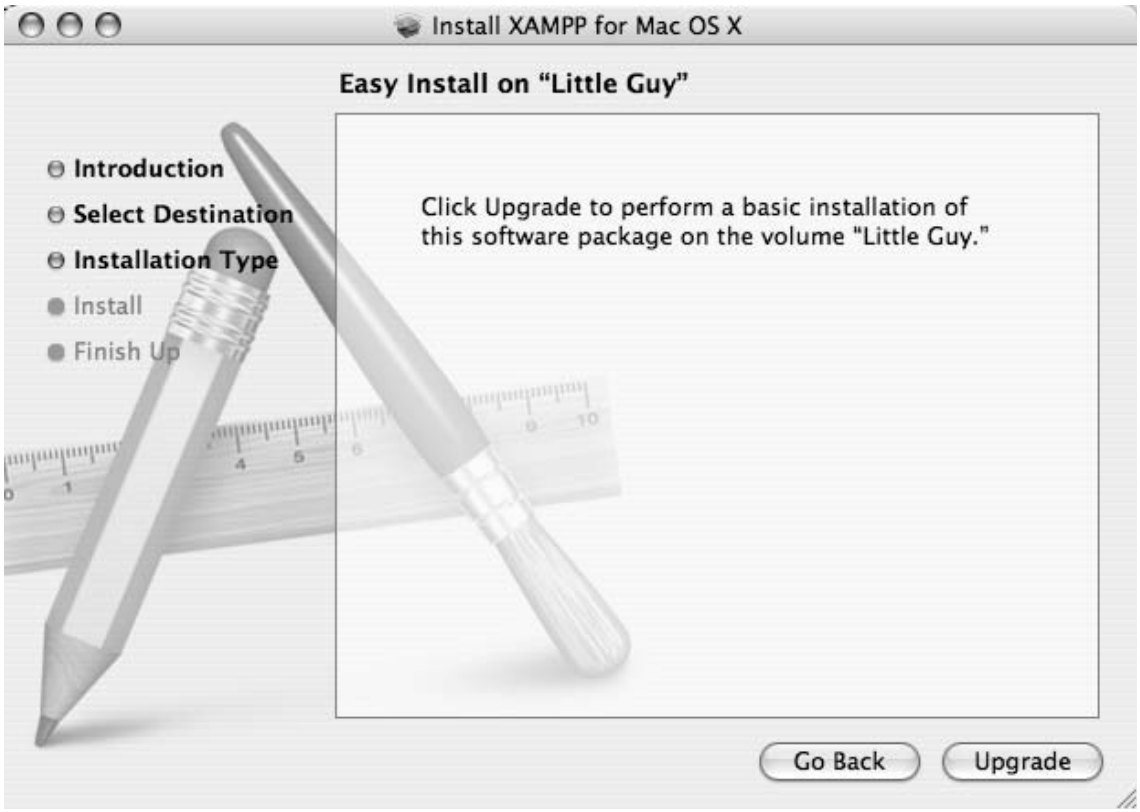

Figure 1-3. *XAMPP gives you only one option the first time you install it*

Clicking "Upgrade" brings up a screen that shows the progress of XAMPP as it installs on the selected drive (see Figure 1-4).

Figure 1-4. *Watch the installer's progress for XAMPP for Mac OS X*

Installation requires a minute or two to complete, whereupon the installer displays the final screen (see Figure 1-5), which confirms that the installation was successful.

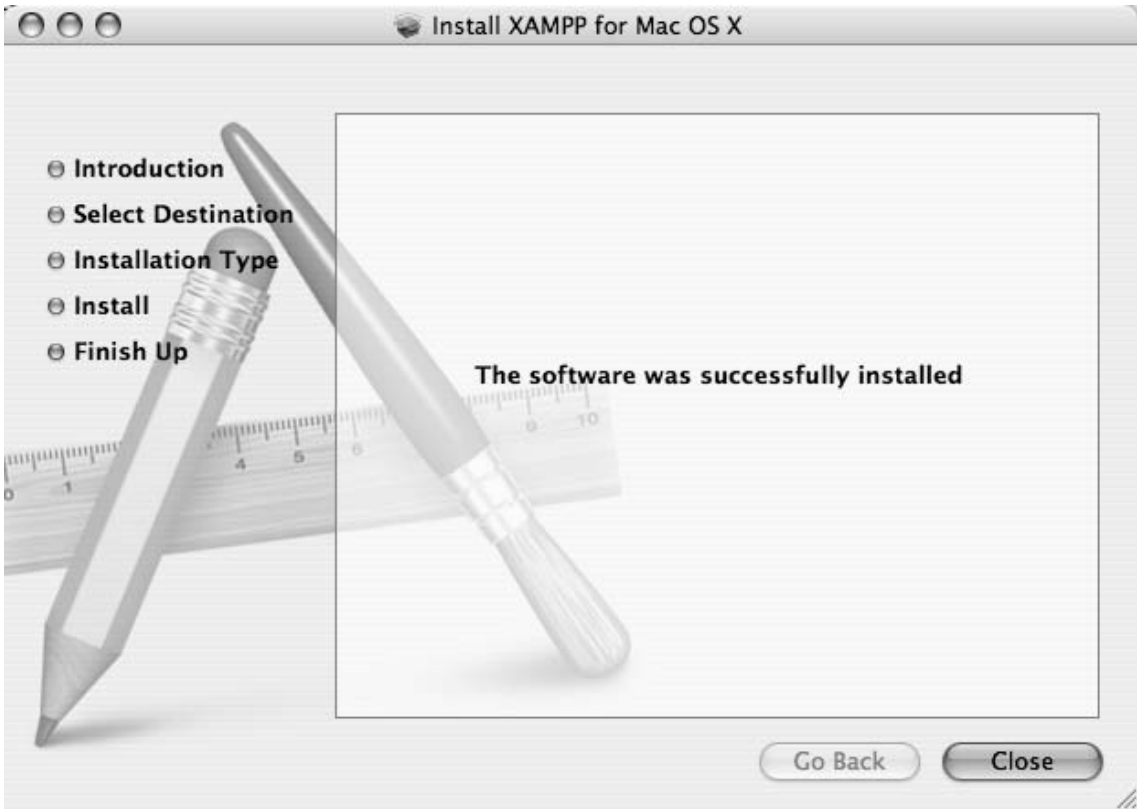

Figure 1-5. *Successful installation of XAMPP brings up this screen*

Step 3: Test XAMPP to Ensure Proper Installation

So far you've used the XAMPP wizard to install Apache, PHP, and MySQL. The next step is to activate the trio of applications.

Open the XAMPP Control Panel

You can activate the just-installed applications by navigating to the newly installed xampp folder and opening the XAMPP Control Panel (see Figure 1-6).

■ **Note** When opening the XAMPP Control Panel you may be prompted for your password. This has no effect on the services themselves and should not affect the projects covered in this book.

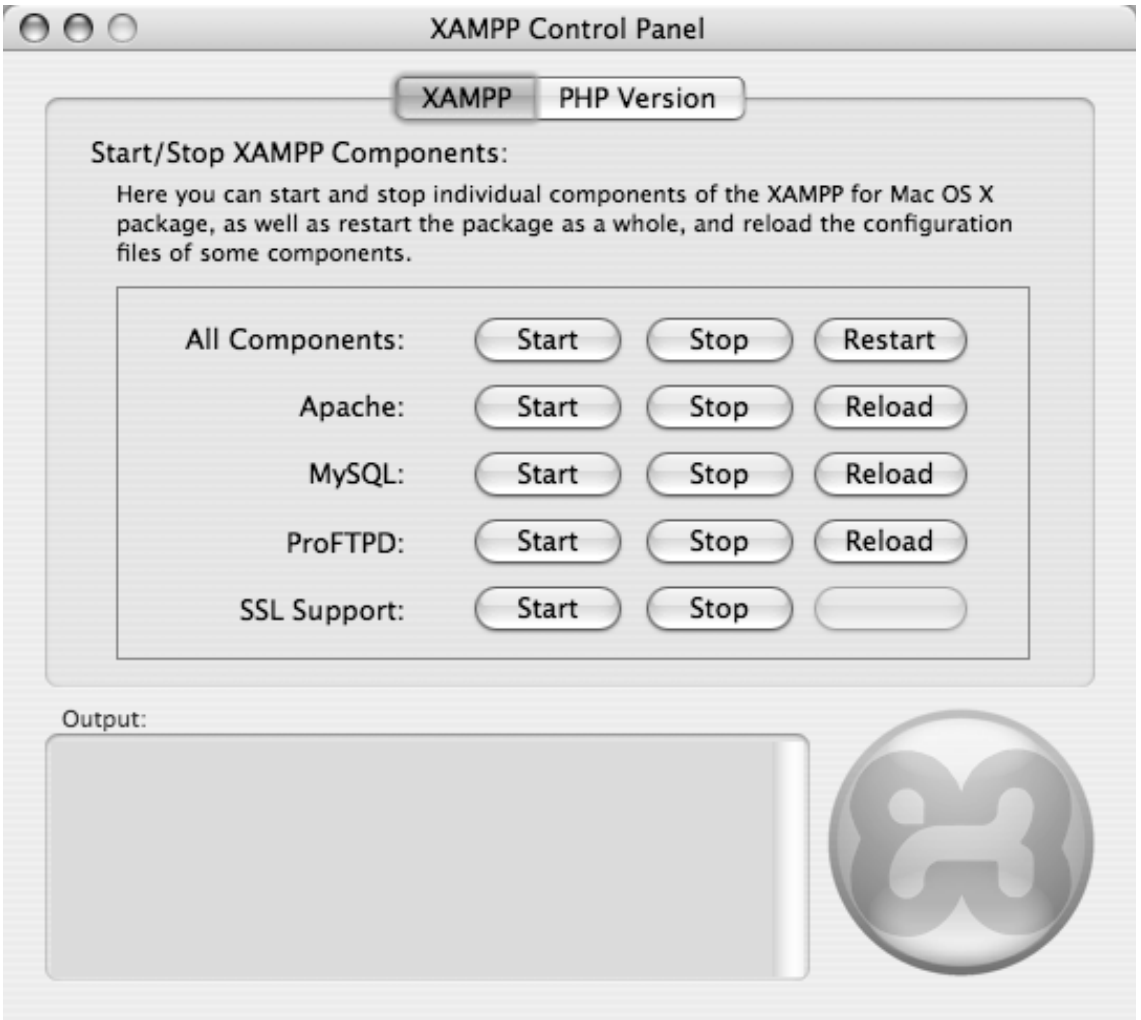

Figure 1-6. *Inside the XAMPP control panel*

Activating Apache, PHP, and MySQL on your development machine is as simple as clicking the "Start" button next to both Apache and MySQL on the XAMPP Control Panel. You might be prompted to confirm that the server is allowed to run on your computer, and you might be required to enter your system password. After you do this, the "Output" panel should start displaying a series of messages (see Figure 1-7); the final message displayed should say, "XAMPP for (*operating system here*) started."

Figure 1-7. *Starting XAMPP services*

■ **Note** There is also an FTP (file transfer protocol) option available in XAMPP. FTP provides a method for moving files between networks. The examples in this book don't require this option, so there is no need to activate it in the XAMPP control panel.

Verify That Apache and PHP Are Running

It's a simple matter to check whether all the services are running properly on your development machine. Simply open a browser and go to this address: `http://localhost`. If everything has gone correctly, you'll be redirected to `http://localhost/xampp/index.php` (see Figure 1-8).

If this screen loads, you've installed Apache and PHP on your development machine successfully!

If you do *not* see this screen, the XAMPP online community is extremely helpful and most installation issues have been addressed in the Apache Friends forum at `http://www.apachefriends.org/f/viewforum.php?f=34`.

The address, `http://localhost`, is an alias for the current computer you're working on. When using XAMPP, navigating to `http://localhost` in a browser tells the server to open the root web directory; this is the `htdocs` folder contained in the XAMPP install directory.

Another way to use your server to access the root web directory on your local machine is to navigate to the *IP address*—a numerical identifier assigned to any device connected to a computer network—that serves as the "home" address for all HTTP servers: `http://127.0.0.1`.

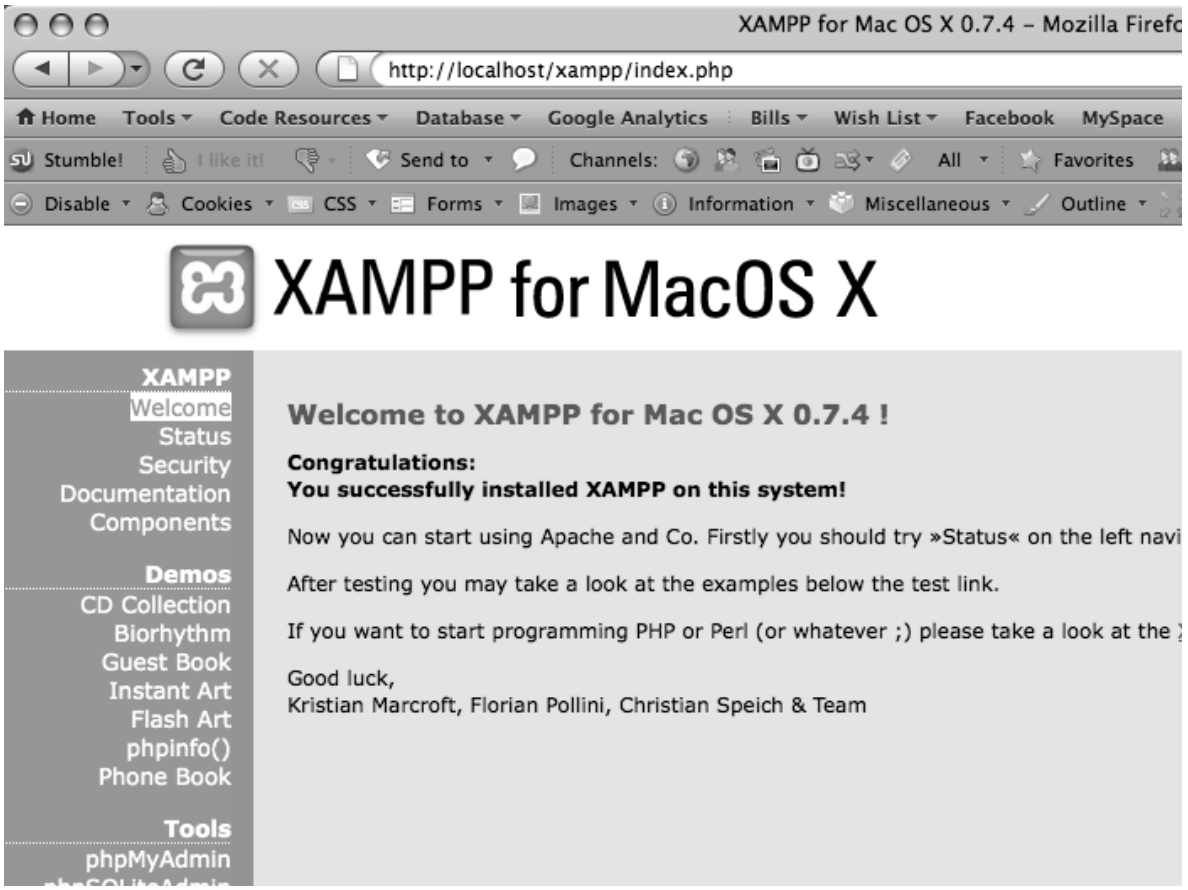

Figure 1-8. *Visit the XAMPP homepage at* `http://localhost`

Verify That MySQL Is Running

You can verify that MySQL is also running by going to the Tools menu and choosing "phpMyAdmin." This should bring up the screen shown in Figure 1-9.

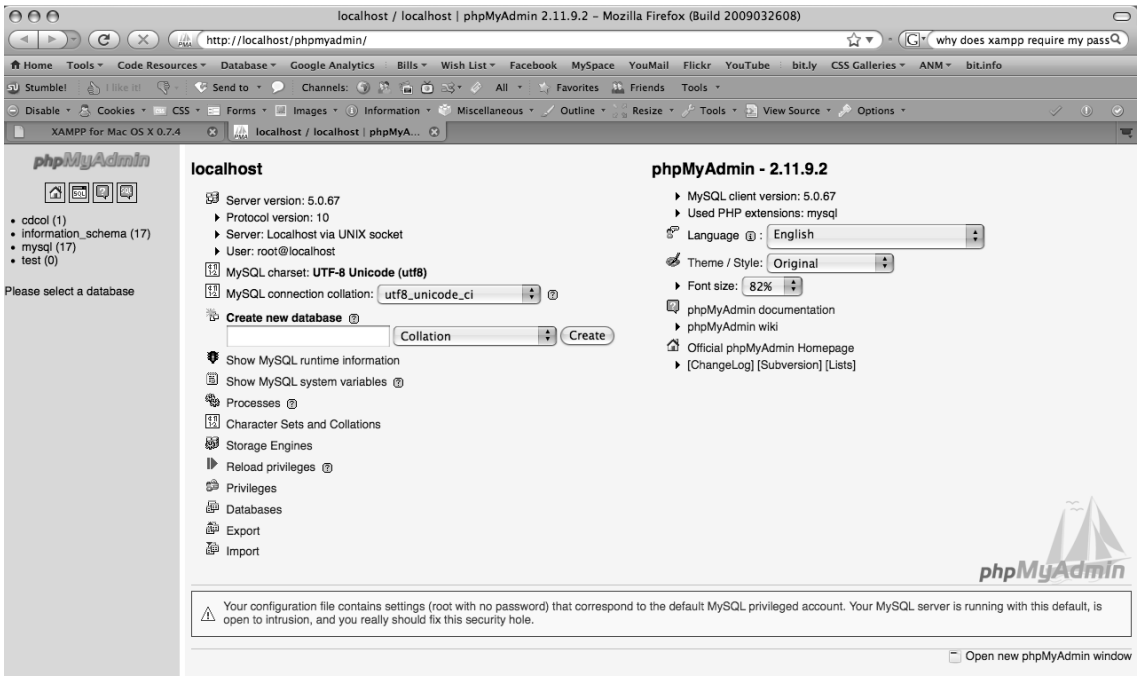

Figure 1-9. *MySQL is running if phpMyAdmin loads without error*

Now that have MySQL running on your development machine, you're ready to start running PHP scripts. Note that if you're a Windows user, you might need to navigate to `C:\xampp\php\php.ini` and locate the following lines to verify that `magic_quotes_gpc` is set to `Off`:

```
; Magic quotes for incoming GET/POST/Cookie data
magic_quotes_gpc = Off
```

Choosing a Development Kit

Your development machine is now running all the necessary programs for programming with PHP. The next step is to decide how you're going to write your scripts. PHP scripts are text-based, so you have myriad options, ranging from the simple Notepad.exe and text-edit programs to highly specialized software development kits (SDKs) and integrated development environments (IDEs).

Benefiting from SDKs and IDEs

There's nothing wrong with coding in a plain text editor, but using an SDK and/or IDE for development can bring you many benefits, including:

- *Syntax highlighting*: This is the ability to recognize certain words in a programming language, such as variables, control structures, and various other special text. This special text is highlighted or otherwise differentiated to make scanning your code much easier. For example, the color coding on the left makes this code less daunting to look at (see Figure 1-10).
- *Built-in function references:* When you enter the name of a function, this feature displays available parameters, as well as the file that declares the function, a short description of what the function does, and a more in-depth breakdown of parameters and return values (see Figure 1-11). This feature proves invaluable when dealing with large libraries, and it can save you trips to the PHP manual to check the order of parameters or acceptable arguments for a function.
- *Auto-complete features*: Common to most SDKs and IDEs, this feature adds available variables to a drop-down list, allowing you to select the intended variable from the list quickly and easily, saving you the effort of typing it out every time (see Figure 1-12). When it comes to productivity, every second counts, and this feature is a great way to contribute to saved time.
- *Code Folding:* This feature lets you collapse snippets of code (see the plus and minus toggles on the left-hand side of Figure 1-13), reducing workspace clutter-free and making it easy to navigate your code. This feature proves especially helpful for reducing the confusing clutter that springs up as your scripts become increasingly complicated.

```
static function buildMenu($menu_array, $url_array, $is_sub=FALSE)
{
    $attr = ($is_sub) ? ' id="menu"' : ' class="submenu"';
    $menu = "<ul$attr>";
    foreach($menu_array as $id => $properties) {
        echo 'ID: ', $id, '<br />';
        foreach($properties as $key => $val) {
            if(is_array($val)) {
                $sub = buildMenu($val, $url_array, TRUE);
            } else {
                $sub = NULL;
                $$key = $val;
            }
        }
        $sel = ($id==$url_array[0]) ? ' class="selected"' : NULL;
        $menu .= "<li><a href=\"$url\"$sel>$display</a></li>$sub";

    }
    $menu .= '</ul>';

    return $menu;
}
```

```
static function buildMenu($menu_array, $url_array, $is_sub=FALSE)
    {
            $attr = ($is_sub) ? ' id="menu"' : ' class="submenu"';
            $menu = "<ul$attr>";
            foreach($menu_array as $id => $properties) {
                    echo 'ID: ', $id, '<br />';
                    foreach($properties as $key => $val) {
                            if(is_array($val)) {
                                    $sub = buildMenu($val, $url_array, TRUE);
                            } else {
                                    $sub = NULL;
                                    $$key = $val;
                            }
                    }
                    $sel = ($id==$url_array[0]) ? ' class="selected"' : NULL;
                    $menu .= "<li><a href=\"$url\"$sel>$display</a></li>$sub";

            }
            $menu .= '</ul>';

            return $menu;
    }
```

Figure 1-10. *The left-hand side shows code with syntax highlighting; the right-hand side shows the same code with no syntax highlighting*

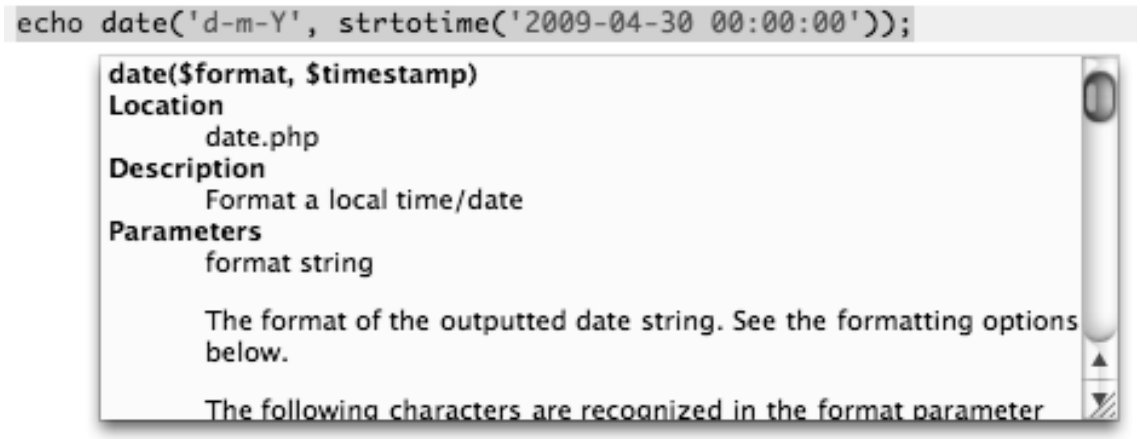

Figure 1-11. *Viewing a function reference in the Eclipse PDT*

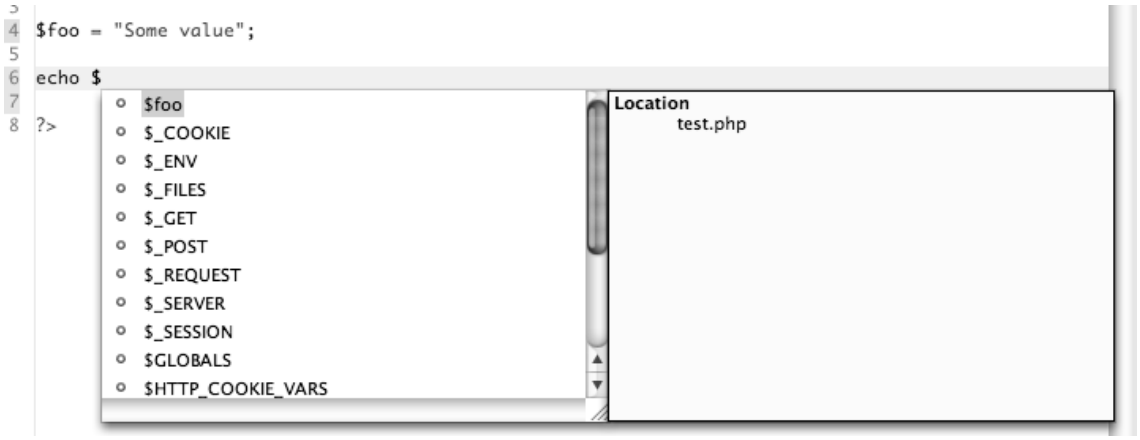

Figure 1-12. *Taking advantage of autocomplete in the Eclipse PDT*

```
 3⊖class Utilities
 4 {
 5⊕     static function textPreview($body, $limit=45)..
39⊕     static function formatImageThumb($e, $class=NULL)..
53⊕     static function formatImage($e, $link=FALSE)..
85⊕     static function buildMenu($menu_array, $url_array, $is_sub=FALSE)..
108⊕    static function curry($func, $arity) {..
123⊕    static function replaceTags($transformations, $matches)..
128⊖    static function copyrightYear($created)
129     {
130         $current = date('Y', time());
131         return ($current>$created) ? $created.'-'.$current : $current;
132     }
133 }
134
135 ?>
```

Figure 1-13. *Code folding in the Eclipse PDT*

Choosing the Right SDK

You have many choices when it comes to choosing development tools, and it should be noted that there's no wrong answer when selecting yours. If the tool feels right, and makes sense to you, that's really all that matters.

The Eclipse Foundation and PDT

The exercises in this book rely on the popular open source Eclipse SDK and more specifically on the PHP-specific PDT IDE. PDT stands for PHP Development Tools, and this IDE provides a free IDE option for beginning developers.

The team responsible for overseeing Eclipse notes, "Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle."

Essentially, this means that Eclipse is a group of people working together to create the best available tools for developers—at no cost to the developer.

Installing and Using the Eclipse PDT

Installing and setting up the Eclipse PDT requires six steps.

Step 1: Downloading the PDT

Get started by navigating to the PDT download page (http://www.eclipse.org/pdt/downloads/) and scrolling down to the All-In-One downloads section. Select your operating system, then choose a mirror for downloading (generally, the default mirror will be highlighted and will work just fine).

Step 2: Unpacking the PDT Archive

After the file finishes downloading, unzip the file. A folder called "eclipse" should appear in the same directory as the downloaded archive.

Drag this folder into your Programs or Applications folder (or wherever you want to keep it) and open it up. There will be a purple icon that says "Eclipse"; double-clicking the icon launches the IDE and brings up the loading screen (see Figure 1-14).

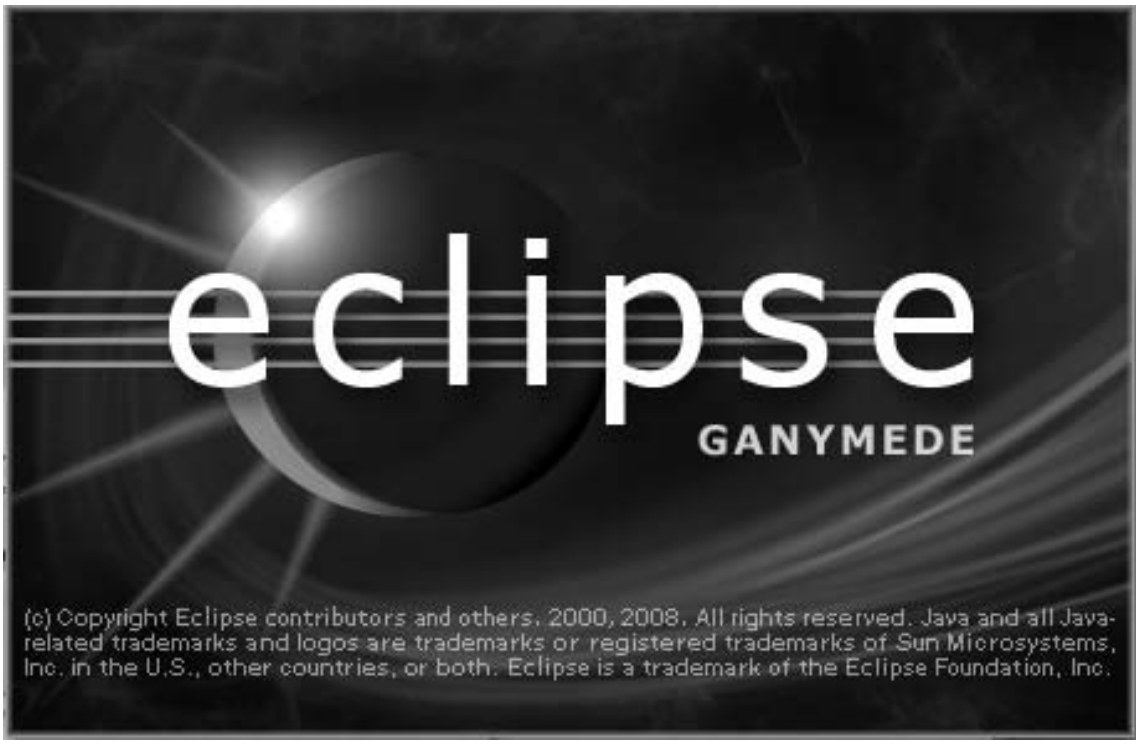

Figure 1-14. *Loading the Eclipse PDT screen*

Step 3: Choosing Your Project Workspace

After a moment, a dialog box will pop up (see Figure 1-15) that asks you to select your workspace. You'll be working with XAMPP, so set the path to the XAMPP `htdocs` folder (see Figure 1-16). You can find this in the `xampp` folder you installed previously.

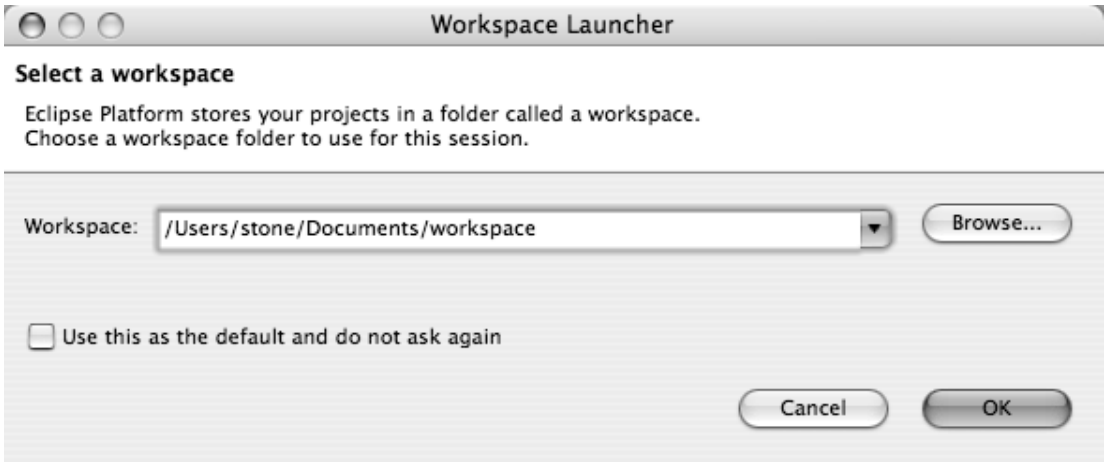

Figure 1-15. *Selecting your workspace in the Eclipse PDT*

Figure 1-16. *Selecting the* `htdocs` *folder from the XAMPP installation*

Selecting this folder and checking the "Use this as the default and do not ask me again" box enables you to tell Eclipse to create new projects in the `htdocs` folder automatically, which simplifies testing your code.

After clicking "Choose," close the welcome screen that shows up by clicking the "X" in the tab at the top, next to "Welcome" (see Figure 1-17), which takes you to the editor.

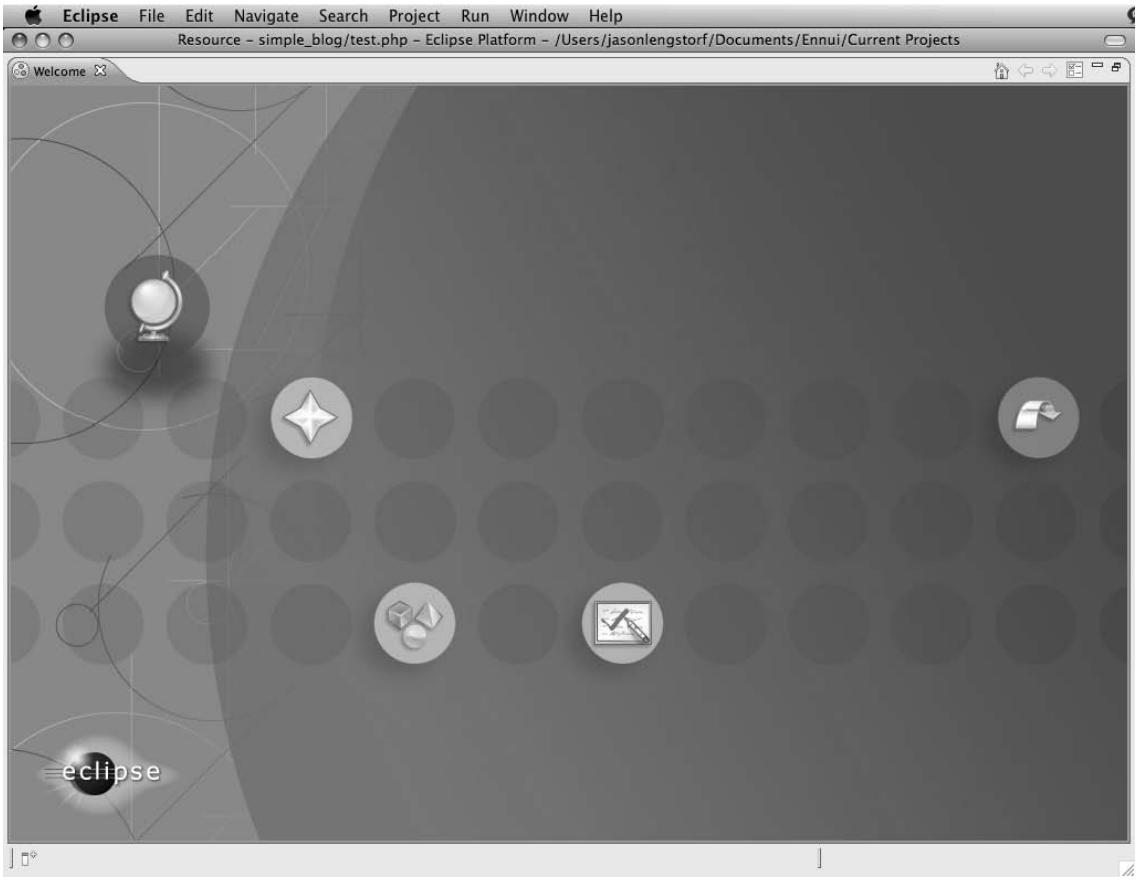

Figure 1-17. *Clicking the "X" next to "Welcome" closes the welcome screen*

Step 4: Creating Your First Project

You're almost to the point where you can code a simple blog. The next step is to create a project. Do this by clicking the New Project icon, which is at the top left of the Eclipse toolbar (see Figure 1-18).

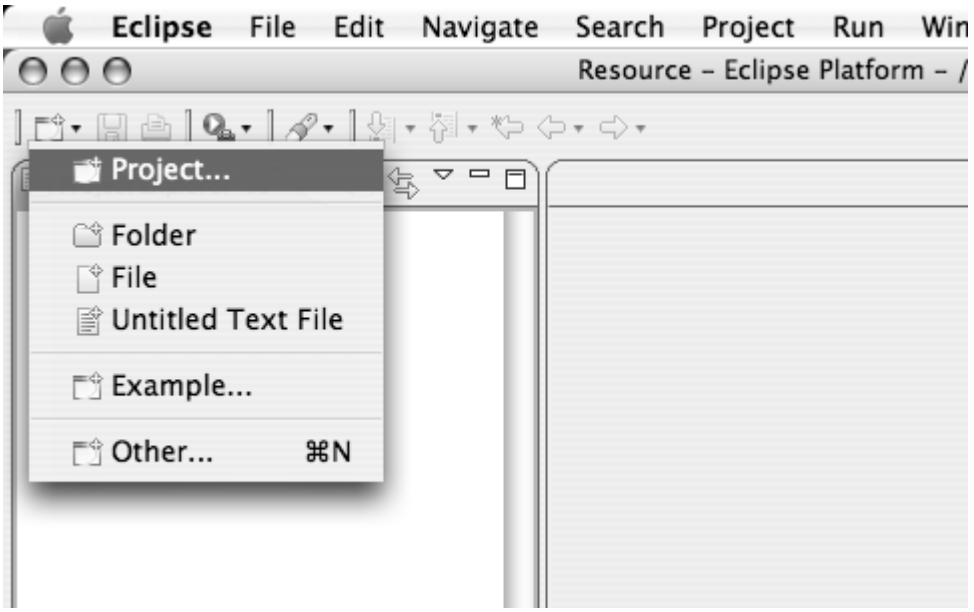

Figure 1-18. *Creating a new project in Eclipse*

Select "New Project..." from the drop-down menu that pops up. This brings up a new dialog (see Figure 1-19); select "PHP Project" from the list of project types and click "Next."

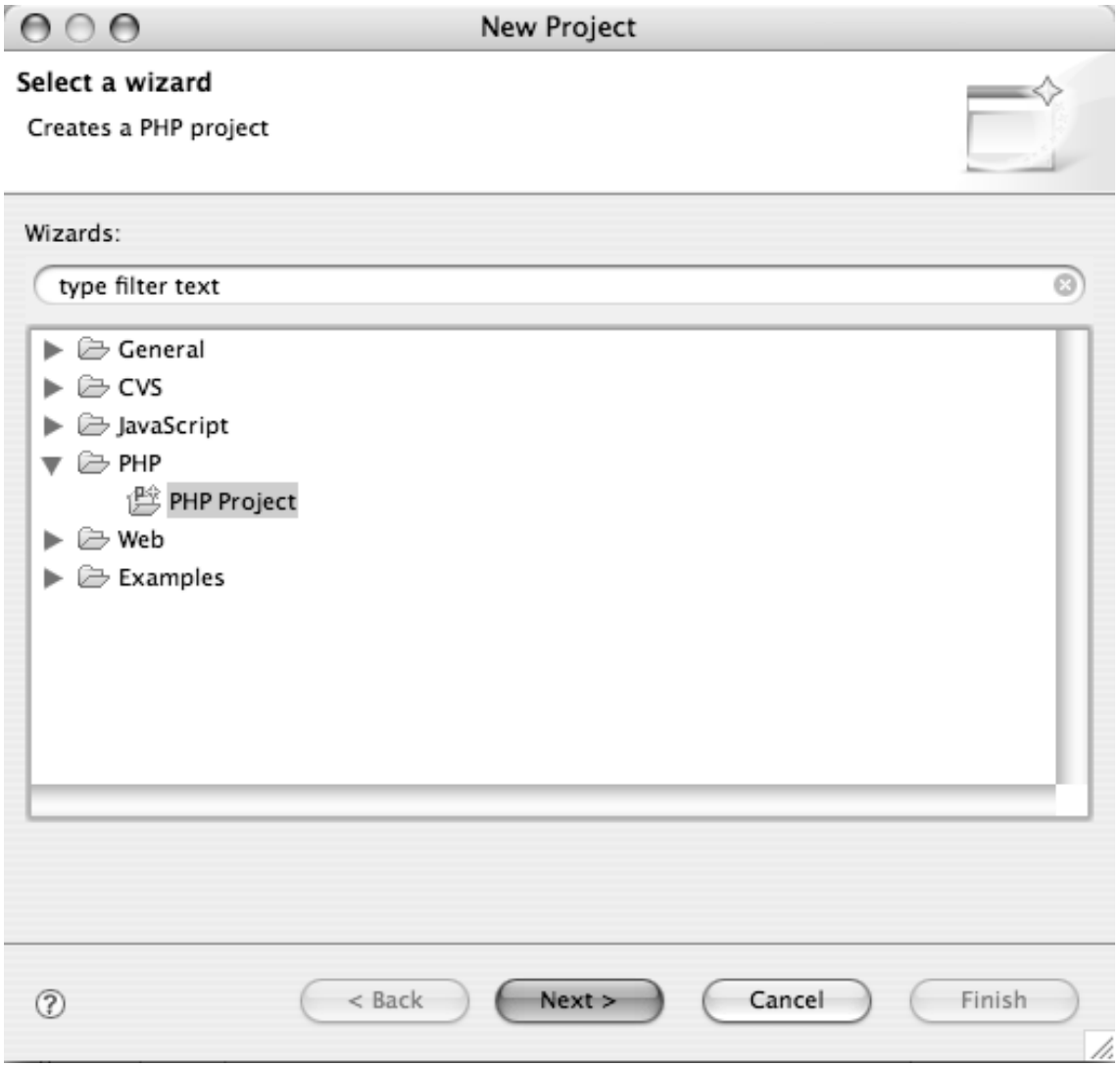

Figure 1-19. *Creating a PHP project*

The next dialog presented asks you to name your project and provides some customization options for the project (see Figure 1-20). Name the project "simple_blog" and leave all the settings at their default values.

Figure 1-20. *Creating the* `simple_blog` *project*

Clicking "Finish" will bring you back to the editor; your newly created project will be listed in the left-hand panel. At this point, Eclipse lets you know that this project type is associated with the PHP perspective, and it asks whether you'd like to switch to the PHP perspective. It makes sense to do this because you're working with PHP, so select, "Yes."

Step 5: Creating a File

The final step is to create a file that you can start coding in. At first you'll be doing basic exercises to get a feel for the language, so call your first file `test.php`.

To create the file, right click the `simple_blog` project, hover over New, and then click PHP File from the resulting drop-down menu (see Figure 1-21).

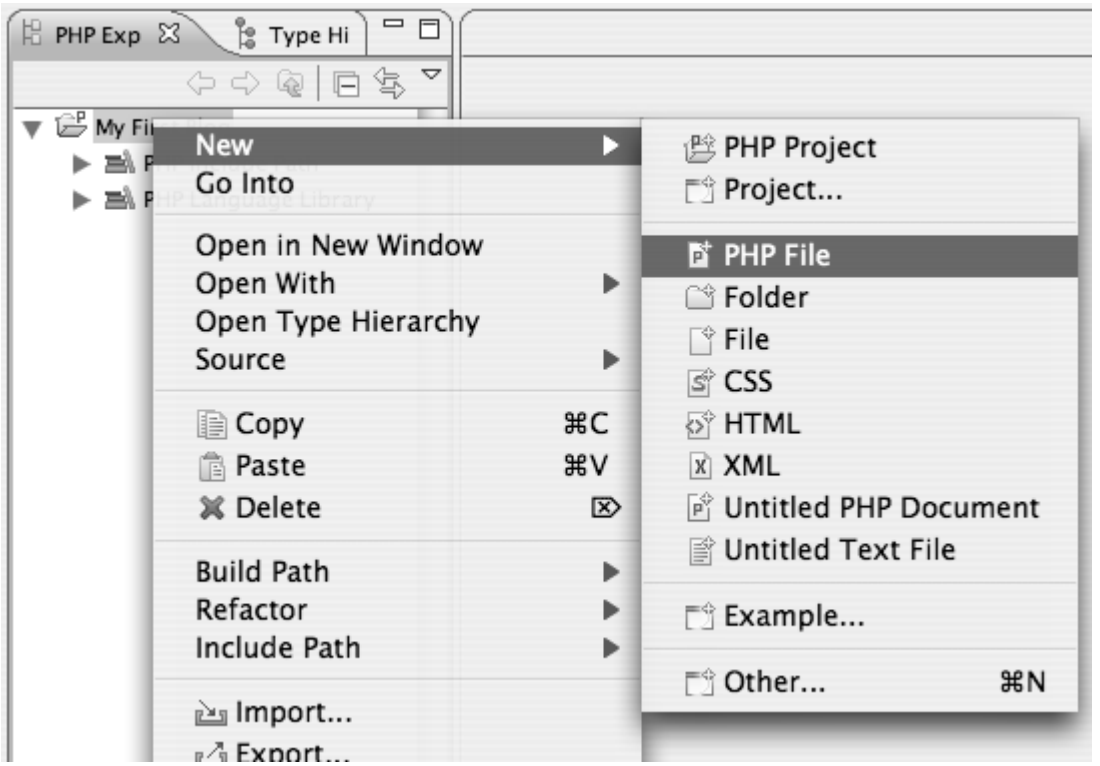

Figure 1-21. *Creating a PHP file in Eclipse*

This brings up a new dialog where you can name your file (see Figure 1-22).

Figure 1-22. *Create* `test.php` *and click "Finish"*

Clicking Finish creates your first file and means you can now start coding.

Step 6: Writing Your First Script

The final step is to make sure that everything is set up correctly and ready for you to start developing your blog. You can do this by writing a short script to ensure that PHP is working in your `test.php` file. Add this code to the `test.php` file and save:

```
<?php
    echo "Hello world!";
?>
```

Now open any browser and navigate to `http://localhost/simple_blog/test.php`; what you see should look like Figure 1-23).

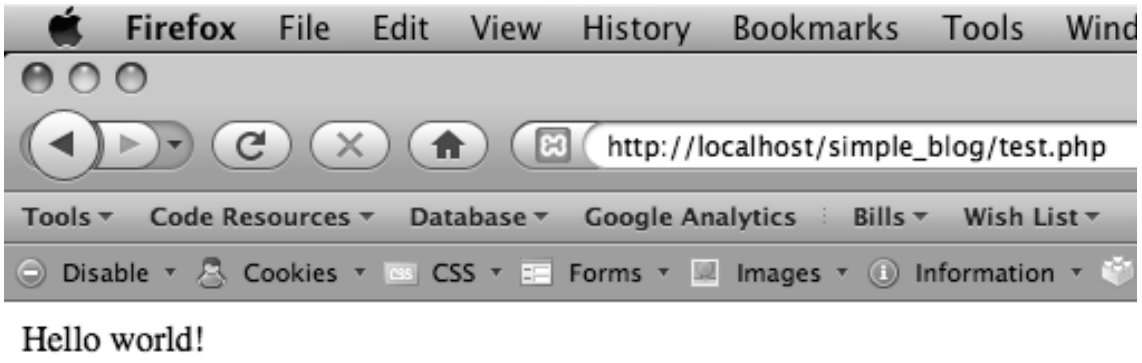

Figure 1-23. *`test.php` loaded in a browser, displaying your first script!*

If the above text is visible, you have installed XAMPP and Eclipse successfully, and you're ready to start building your blog.

Summary

In this chapter, you learned what PHP, MySQL, and Apache are, and what role they play in the development of dynamic web sites. You also learned a quick and easy way to install a fully functional development environment on your local computer by installing XAMPP and Eclipse PDT.

In the next chapter, you'll learn the basics of PHP, including variables, control structures, and functions. Nearly everything you learn will be tested in your new development environment, so keep `test.php` open and ready to edit.