CHAPTER 2

Getting Ready to Work with PHP

Now you've decided to use PHP to enrich your web pages, you need to make sure that you have everything you need to get on with the rest of this book. Although you can test everything on your remote server, it's usually more convenient to test PHP pages on your local computer. Everything you need to install is free. In this chapter, I'll explain the various options for Windows and macOS. The necessary components are normally installed by default on Linux.

This chapter covers

- Checking if your web site supports PHP
- Understanding why you should no longer use PHP 5
- Deciding whether to create a local testing setup
- Using a ready-made package in Windows and macOS
- Deciding where to store your PHP files
- Checking the PHP configuration on your local and remote servers

Checking Whether Your Web Site Supports PHP

The easiest way to find out whether your web site supports PHP is to ask your hosting company. The other way to find out is to upload a PHP page to your web site and see if it works. Even if you know that your site supports PHP, do the following test to confirm which version is running:

1. Open your script editor, and type the following code into a blank page:

   ```
   <?php echo phpversion();
   ```

2. Save the file as `phpversion.php`. It's important to make sure that your operating system doesn't add a `.txt` filename extension after the `.php`. If you're using TextEdit on a Mac, make sure that it doesn't save the file in Rich Text Format (RTF). If you're at all unsure, use `phpversion.php` from the `ch02` folder in the files accompanying this book.

3. Upload `phpversion.php` to your web site in the same way you would an HTML page and then type the URL into a browser. Assuming you upload the file to the top level of your site, the URL will be something like `www.example.com/phpversion.php`.

D. Powers, *PHP 7 Solutions*, https://doi.org/10.1007/978-1-4842-4338-1_2

If you see a three-part number like 7.2.0 displayed onscreen, you're in business: PHP is enabled. The number tells you which version of PHP is running on your server. *This book assumes you're running PHP 7.2.0 or later.*

4. If you get a message that says something like "Parse error", it means PHP is supported but that you have made a mistake in typing the code in the file. Use the version in the `ch02` folder instead.

5. If you just see the original code, it means PHP is not supported.

If your server is running a version of PHP prior to 7.2.0, contact your host and tell them you want the most recent stable version of PHP. If your host refuses, it's time to change your hosting company.

WHY YOU SHOULD NO LONGER USE PHP 5

PHP versions comprise three numbers separated by dots. The first number is the major version; the second is the branch; and the last one is the point release. Major versions introduce significant changes that might break existing sites, including the removal of outdated features. Branches introduce new features; but it's extremely rare for a branch to break compatibility with the same major version. Point releases contain fixes for bugs and security issues.

When PHP 5 was released in 2004, the development team continued to release security updates for PHP 4. This not only delayed adoption of PHP 5, it slowed vital improvements to the new major version. A change in policy led to PHP releasing a new branch once a year. Each branch receives two years of active support for bug and security fixes, followed by a third year of critical security fixes. Thereafter it's no longer supported.

The final version of PHP 5 (5.6) received nearly two and a half years of extra support; but it reached its end of life on December 31, 2018. Using PHP 5 puts your site and valuable data at risk because security vulnerabilities won't be fixed. Moreover, you won't get the benefit of PHP 7's new features, not to mention that it's twice as fast as PHP 5. All support for PHP 7.0 and active support for PHP 7.1 ended before this book was published. That's why this book requires a minimum of PHP 7.2. Although most of the code in this book will run on older versions of PHP, some of it won't.

Deciding Where to Test Your Pages

Unlike ordinary web pages, you can't just double-click PHP pages in Windows File Explorer or Finder on a Mac and view them in your browser. They need to be **parsed**, or processed, through a web server that supports PHP. If your hosting company supports PHP, you can upload your files to your web site and test them there. However, you need to upload the file every time you make a change. In the early days, you'll probably find you have to do this often because of some minor mistake in your code. As you become more experienced, you'll still need to upload files frequently because you'll want to experiment with different ideas.

If you want to get working with PHP straight away, by all means use your own web site as a test bed. However, you'll soon discover the need for a local PHP test environment. The rest of this chapter is devoted to showing you how to do this, with instructions for both Windows and macOS.

What You Need for a Local Test Environment

To test PHP pages on your local computer, you need to install the following:

- A web server, which is a piece of software that displays web pages, not a separate computer
- PHP
- A MySQL or MariaDB database, and phpMyAdmin, a web-based front end for administering the database

---

■ **Tip** MariaDB (`https://mariadb.org/`) is a community-developed drop-in replacement for MySQL. The code in this book is fully compatible with both MySQL and MariaDB.

All the software you need is free. The only cost to you is the time it takes to download the necessary files, plus, of course, the time to make sure everything is set up correctly. In most cases, you should be up and running in less than an hour, probably considerably less. As long as you have at least 1GB of free disk space, you should be able to install all the software on your computer—even one with modest specifications.

■ **Tip** If you already have a PHP 7 test environment on your local computer, there's no need to reinstall. Just check the section at the end of this chapter titled "Checking Your PHP Settings."

Individual Programs or an All-in-one Package?

For many years, I advocated installing each component of a PHP testing environment separately, rather than using a package that installs Apache, PHP, MySQL, and phpMyAdmin in a single operation. My advice was based on the dubious quality of some early all-in-one packages, which installed easily but were next to impossible to uninstall or upgrade. However, the all-in-one packages currently available are excellent, and I have no hesitation in now recommending them.

On my computers, I use XAMPP for Windows (`www.apachefriends.org/index.html`) and MAMP for macOS (`www.mamp.info/en/`). Other packages are available; it doesn't matter which you choose.

Setting Up on Windows

Make sure that you're logged on as an administrator before proceeding.

Getting Windows to Display Filename Extensions

By default, most Windows computers hide common three- or four-letter filename extensions, such as `.doc` or `.html`, so all you see in dialog boxes and Windows File Explorer is `thisfile` instead of `thisfile.doc` or `thisfile.html`.

Use these instructions to enable the display of filename extensions in Windows 10 and 8:

1. Open File Explorer.
2. Select View to expand the ribbon at the top of the File Explorer window.
3. Select the "File name extensions" check box.

Displaying filename extensions is more secure—you can tell if a virus writer has attached an `.exe` or `.scr` executable file to an innocent-looking document.

Choosing a Web Server

Most PHP installations run on the Apache web server. Both are open source and work well together. However, Windows has its own web server, Internet Information Services (IIS), which also supports PHP. Microsoft has worked closely with the PHP development team to improve the performance of PHP on IIS to roughly the same level as Apache. So, which should you choose?

Unless you need IIS for ASP or ASP.NET, I recommend that you install Apache, using XAMPP or one of the other all-in-one packages, as described in the next section. If you need to use IIS, the most convenient way to install PHP is to use the Microsoft Web Platform Installer (Web PI), which you can download from `www.microsoft.com/web/downloads/platform.aspx`.

Installing an All-in-one Package on Windows

There are three popular packages for Windows that install Apache, PHP, MySQL or MariaDB, phpMyAdmin, and several other tools on your computer in a single operation: XAMPP (`www.apachefriends.org/index.html`), WampServer (`www.wampserver.com/en/`), and EasyPHP (`www.easyphp.org`). The installation process normally takes only a few minutes. Once the package has been installed, you might need to change a few settings, as explained later in this chapter.

Versions are liable to change over the lifetime of a printed book, so I won't describe the installation process. Each package has instructions on its web site.

Setting Up on macOS

The Apache web server and PHP are preinstalled on macOS, but they're not enabled by default. Rather than using the preinstalled versions, I recommend that you use MAMP, which installs Apache, PHP, MySQL, phpMyAdmin, and several other tools in a single operation.

To avoid conflicts with the preinstalled versions of Apache and PHP, MAMP locates all the applications in a dedicated folder on your hard disk. This makes it easier to uninstall everything by simply dragging the MAMP folder to the Trash if you decide you no longer want MAMP on your computer.

Installing MAMP

Before you begin, make sure you're logged in to your computer with administrative privileges.

1. Go to `www.mamp.info/en/downloads/` and select the link for MAMP & MAMP PRO. This downloads a disk image that contains both the free and paid-for versions of MAMP.
2. When the download completes, launch the disk image. You'll be presented with a license agreement. You must click Agree to continue with mounting the disk image.

3. Follow the onscreen instructions.
4. Verify that MAMP has been installed in your Applications folder.

■ **Note** MAMP automatically installs both the free and paid-for versions in separate folders called MAMP and MAMP PRO. The paid-for version makes it easier to configure PHP and to work with virtual hosts, but the free version is perfectly adequate, especially for beginners. If you want to remove the MAMP PRO folder, don't drag it to the Trash. Open the folder and double-click the MAMP PRO uninstall icon. The paid-for version requires both folders.

Testing and Configuring MAMP

By default, MAMP uses nonstandard ports for Apache and MySQL. Unless you're using multiple installations of Apache and MySQL, change the port settings as described in the following steps:

1. Double-click the MAMP icon in Applications/MAMP. If you're presented with a panel inviting you to access your data from multiple Macs with MAMP Cloud Functions, click the Close button at the top left to dismiss the panel. The MAMP Cloud Functions are a paid-for service that's not required for this book. You can get details by clicking the Learn More button. There's also a check box to prevent the panel from being displayed each time you start MAMP.
2. Click Start Servers in the MAMP control panel (see Figure 2-1). Tiny green lights should come on alongside Apache Server and MySQL Server to indicate that they're running. Your default browser should also launch and present you with the MAMP welcome page.

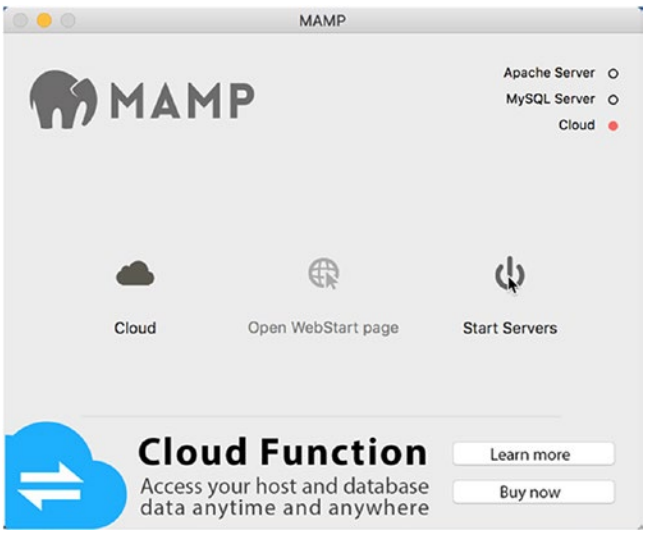


Figure 2-1. *Starting the servers in the MAMP control panel*

3. If your browser doesn't launch automatically, click Open WebStart page in the MAMP control panel.
4. Check the URL in the browser address bar. It begins with `localhost:8888`. The `:8888` indicates that Apache is listening for requests on the nonstandard port 8888.
5. Minimize the browser and click anywhere in the MAMP control panel to make it the active application.
6. Go to the main MAMP menu at the top of the screen and select Preferences (or use the keyboard shortcut Cmd+,).
7. Select Ports at the top of the panel that opens. It shows that Apache and MySQL are running on ports 8888 and 8889 (see Figure 2-2).

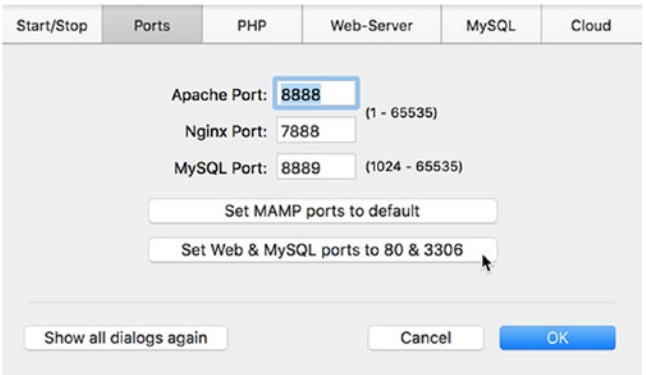


Figure 2-2. *Changing the Apache and MySQL ports*

8. Click "Set Web & MySQL ports to 80 & 3306" as shown in Figure 2-2. The numbers change to the standard ports: 80 for Apache and 3306 for MySQL.

■ **Note** MAMP now supports Nginx as an alternative web server. When I clicked "Set Web & MySQL ports to 80 & 3306," both Apache Port and Nginx Port changed to 80, which prevented the settings from being accepted. If this happens, manually reset Nginx Port to 7888.

---

9. Click OK and enter your Mac password when prompted. MAMP restarts both servers.

■ **Tip** If any other program is using port 80, Apache won't restart. If you can't find what's preventing Apache from using port 80, open the MAMP preferences panel and click "Set MAMP ports to default."

---

10. When both lights are green again, the MAMP welcome page reloads into your browser. This time, the URL shouldn't have a colon followed by a number appearing after `localhost` because Apache is now listening on the default port.

Where to Locate Your PHP Files (Windows and Mac)

You need to create your files in a location where the web server can process them. Normally, this means that the files should be in the server's document root or in a subfolder of the document root. The default location of the document root for the most common setups is as follows:

- **XAMPP**: `C:\xampp\htdocs`
- **WampServer**: `C:\wamp\www`
- **EasyPHP**: `C:\EasyPHP\www`
- **IIS**: `C:\inetpub\wwwroot`
- **MAMP**: `/Applications/MAMP/htdocs`

To view a PHP page, you need to load it in a browser using a URL. The URL for the web server's document root in your local testing environment is `http://localhost/`.

■ **Caution** If you needed to reset MAMP back to its default ports, you will need to use `http://localhost:8888` instead of `http://localhost`.

If you store the files for this book in a subfolder of the document root called `phpsols-4e`, the URL is `http://localhost/phpsols-4e/` followed by the name of the folder (if any) and file.

■ **Tip** Use `http://127.0.0.1/` if you have problems with `http://localhost/`. `127.0.0.1` is the loopback IP address all computers use to refer to the local machine.

Using Virtual Hosts

The alternative to storing your PHP files in the web server's document root is to use a virtual host. A **virtual host** creates a unique address for each site and is how hosting companies manage shared hosting. MAMP PRO simplifies setting up virtual hosts through its control panel. EasyPHP also has a plug-in module for administering virtual hosts.

Manually setting up virtual hosts involves editing one of your computer's system files to register the host name on your local machine. You also need to tell the web server in your local testing environment where the files are located. The process isn't difficult, but it needs to be done each time you set up a new virtual host.

The advantage of setting up each site in a virtual host is that it matches more accurately the structure of a live web site. However, when learning PHP, it's probably more convenient to use a subfolder of your testing server's document root. Once you have gained experience with PHP, you can advance to using virtual hosts. Instructions for manually setting up virtual hosts in Apache are on my web site at the following addresses:

- **Windows**: `http://foundationphp.com/tutorials/apache_vhosts.php`
- **MAMP**: `http://foundationphp.com/tutorials/vhosts_mamp.php`

■ **Tip** Remember to start the web server in your testing environment to view PHP pages.

Checking Your PHP Settings

After installing PHP, it's a good idea to check its configuration settings. In addition to the core features, PHP has a large number of optional extensions. Both the all-in-one packages and the Microsoft Web PI install all the extensions that you need for this book. However, some of the basic configuration settings might be slightly different. To avoid unexpected problems, adjust your PHP configuration to match the settings recommended in the following pages.

Displaying the Server Configuration with phpinfo()

PHP has a built-in command, `phpinfo()`, that displays details of how PHP is configured on the server. The amount of detail produced by `phpinfo()` can feel like massive information overload, but it's invaluable for determining why something works perfectly on your local computer yet not on your live web site. The problem usually lies in the remote server having disabled a feature or not having installed an optional extension.

The all-in-one packages make it easy to run `phpinfo()`:

- **XAMPP**: Click the phpinfo link in the menu on the top of the XAMPP welcome screen.
- **MAMP**: Click phpinfo in the main menu at the top of the MAMP welcome page.
- **WampServer**: Open the WampServer menu and click Localhost. The link for `phpinfo()` is under Tools.

Alternatively, create a simple test file and load it in your browser using the following instructions:

1. Make sure that Apache or IIS is running on your local computer.
2. Type the following in a script editor:

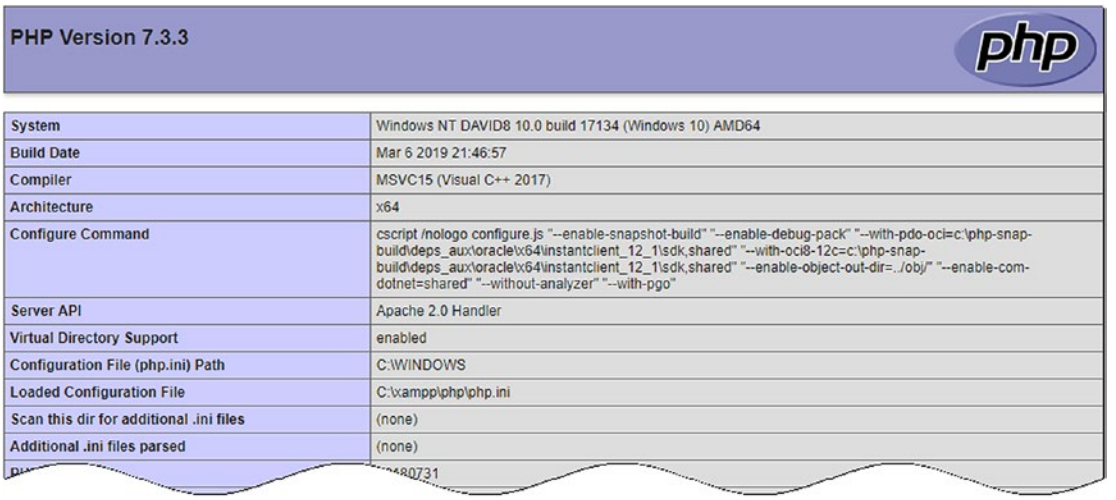
   ```
   <?php phpinfo();
   ```

   There should be nothing else in the file.
3. Save the file as `phpinfo.php` in the server's document root (see "Where to Locate Your PHP Files (Windows and Mac)" earlier in this chapter).

■ **Caution** Make sure your editor doesn't add a `.txt` or `.rtf` extension after `.php`.

---

4. Type `http://localhost/phpinfo.php` in your browser address bar and press Enter.
5. You should see a page similar to that in Figure 2-3 displaying the version of PHP followed by extensive details of your PHP configuration.



PHP Version 7.3.3

System	Windows NT DAVID8 10.0 build 17134 (Windows 10) AMD64
Build Date	Mar 6 2019 21:46:57
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x64
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PH	1680731

Figure 2-3. *Running the* `phpinfo()` *command displays full details of your PHP configuration*

6. Make a note of the value for the Loaded Configuration File item. This tells you where to find `php.ini`, the text file that you need to edit in order to change most settings in PHP.

7. Scroll down to the section labeled Core and compare the settings with those recommended in Table 2-1. Make a note of any differences so you can change them as described later in this chapter.

Table 2-1. *Recommended PHP configuration settings*

Directive	Local value	Remarks
`display_errors`	On	Essential for debugging mistakes in your scripts. If set to Off, errors result in a completely blank screen, leaving you clueless as to the possible cause.
`error_reporting`	32767	This sets error reporting to the highest level.
`file_uploads`	On	Allows you to use PHP to upload files to a web site.
`log_errors`	Off	With `display_errors` set on, you don't need to fill your hard disk with an error log.

8. The rest of the configuration page shows you which PHP extensions are enabled. Although the page seems to go on forever, the extensions are all listed in alphabetical order. To work with this book, make sure the following extensions are enabled:

   - **`gd`**: Enables PHP to generate and modify images and fonts.
   - **`mysqli`**: Connects to MySQL/MariaDB (note the "i," which stands for "improved." PHP 7 doesn't support the older `mysql` one, which should no longer be used).

- **PDO**: Provides software-neutral support for databases (optional).
- **pdo_mysql**: Alternative method of connecting to MySQL/MariaDB (optional).
- **session**: Sessions maintain information associated with a user and are used, among other things, for user authentication.

You should also run `phpinfo()` on your remote server to check which features are enabled. If the listed extensions aren't supported, some of the code in this book won't work when you upload your files to your web site. `PDO` and `pdo_mysql` aren't always enabled on shared hosting, but you can use `mysqli` instead. The advantage of `PDO` is that it's software-neutral, so you can adapt scripts to work with a database other than MySQL by changing only one or two lines of code. Using `mysqli` ties you to MySQL/MariaDB.

If any of the Core settings in your setup are different from the recommendations in Table 2-1, you will need to edit the PHP configuration file, `php.ini`, as described in the next section.

■ **Caution** The output displayed by phpinfo() reveals a lot of information that could be used by a malicious hacker to attack your web site. Always delete the file from your remote server after checking your configuration. Don't leave it there for future convenience. What's convenient for you is even more convenient for the bad guys.

Editing php.ini

The PHP configuration file, `php.ini`, is a very long file, which tends to unnerve newcomers to programming, but there's nothing to worry about. It's written in plain text, and one reason for its length is that it contains copious comments explaining the various options. That said, it's a good idea to make a backup copy before editing `php.ini` in case you make a mistake.

How you open `php.ini` depends on your operating system and how you installed PHP:

- If you used an all-in-one package, such as XAMPP, on Windows, double-click `php.ini` in Windows Explorer. The file opens automatically in Notepad.
- If you installed PHP using the Microsoft Web PI, `php.ini` is normally located in a subfolder of Program Files. Although you can open `php.ini` by double-clicking it, you won't be able to save any changes you make. Instead, right-click Notepad and select Run as Administrator. Inside Notepad, select File ➤ Open and set the option to display All Files (*.*). Navigate to the folder where `php.ini` is located, select the file, and click Open.
- On macOS, use a plain text editor to open `php.ini`. If you use TextEdit, make sure it saves the file as plain text, not Rich Text Format.

Lines that begin with a semicolon (`;`) are comments. The lines you need to edit do not begin with a semicolon.

Use your text editor's Find functionality to locate the directives you need to change to match the recommendations in Table 2-1. Most directives are preceded by one or more examples of how they should be set. Make sure you don't edit one of the commented examples by mistake.

For directives that use `On` or `Off`, just change the value to the recommended one. For example, if you need to turn on the display of error messages, edit this line:

```
display_errors = Off
```

by changing it to this:

```
display_errors = On
```

To set the level of error reporting, you need to use PHP constants, which are written in uppercase and are case-sensitive. The directive should look like this:

```
error_reporting = E_ALL
```

After editing `php.ini`, save the file and then restart Apache or IIS so that the changes take effect. If the web server won't start, check the server's error log file. It can be found in the following locations:

- **XAMPP**: In the XAMPP Control Panel, click the Logs button alongside Apache and then select Apache (error.log).
- **MAMP**: In /Applications/MAMP/logs, double-click apache_error.log to open it in Console.
- **WampServer**: In the WampServer menu, select Apache ➤ Apache error log.
- **EasyPHP**: Right-click the EasyPHP icon in the system tray and select Log Files ➤ Apache.
- **IIS**: The default location of log files is C:\inetpub\logs.

The most recent entry in the error log should give you an indication of what prevented the server from restarting. Use that information to correct the changes you made to `php.ini`. If that doesn't work, be thankful you made a backup of `php.ini` before editing it. Start again with a fresh copy and check your edits carefully.

What's Next?

Now that you've got a working test bed for PHP, you're no doubt raring to go. The last thing I want to do is dampen any enthusiasm, but before using PHP in a live web site, you should have a basic understanding of the rules of the language. So, before jumping into the cool stuff, read the next chapter, which explains how to write PHP scripts. Don't skip it, even if you've already dabbled with PHP—it's really important.