PHP 8 Solutions

Dynamic Web Design and Development Made Easy

Fifth Edition

David Powers

Apress®

PHP 8 Solutions

Dynamic Web Design and Development Made Easy

Fifth Edition

David Powers

Apress®

PHP 8 Solutions: Dynamic Web Design and Development Made Easy

David Powers
London, UK

ISBN-13 (pbk): 978-1-4842-7140-7 ISBN-13 (electronic): 978-1-4842-7141-4
https://doi.org/10.1007/978-1-4842-7141-4

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Steve Anglin
Development Editor: Matthew Moodie
Coordinating Editor: Mark Powers

Cover designed by eStudioCalamar

Cover image by Hazel Clifton on Unsplash (www.unsplash.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484271407. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

In memory of Toshiko, my friend, companion, and wife of many years.

Table of Contents

About the Author

David Powers is the author of more than 30 highly successful video training courses and books on PHP. He began his professional career as a radio and TV journalist for the BBC, spending a large part of it in Japan reporting on the rise and collapse of the bubble economy. His background of reporting on complex issues in plain, jargon-free language reveals itself in his writing about PHP and web development.

David first became involved with web development in the early 1990s as Editor of BBC Japanese TV. With no marketing budget, he developed a bilingual web site to promote the channel. After leaving the BBC, he went on to develop a bilingual online database for an international consultancy, as well as teaching web development courses at two universities in the United Kingdom. In addition to writing and creating video training courses, he's a trustee of a charity in North London that provides educational facilities for retired people and those no longer in full-time employment.

About the Technical Reviewer

Satej Kumar Sahu works in the role of Senior Enterprise Architect at Honeywell. He is passionate about technology, people, and nature. He believes through technology and conscientious decision-making, each of us has the power to make this world a better place. In his free time, he can be found reading books, playing basketball, and having fun with friends and family.

Acknowledgments

Many people have contributed to this book, each one helping improve it as it has moved over five editions. I'm particularly grateful to Chris Mills, the editor of the first edition, whose idea it was to move away from the cookbook formula of isolated solutions that left the reader with little or no idea about the practical use of a technique. Chris's successors, Ben Renow-Clarke (for the second and third editions) and Mark Powers (for the fourth and fifth editions), have both provided a light touch, nudging me in the right direction and forgiving my late delivery times. By the way, if you think we're keeping it in the family, Mark is no relation in spite of his splendid surname.

A big thank you is due to the technical reviewers of this edition, Matt Wade and Satej Sahu. By the time a book gets to its fifth edition, an author hopes to get something of a free ride, expecting all the problems to have been sorted out in previous editions. Fortunately for you, the reader, they subjected my code and text to detailed analysis, making many helpful suggestions. As a result, the book has been greatly improved. Any errors or inconsistencies that remain are my responsibility alone.

Thanks are also due to everyone involved in the production chain at Apress. A book would never see the light of day without their diligent work behind the scenes.

Finally, I would like to pay tribute to my late wife, Toshiko, who put up with me disappearing for hours on end working on the first three editions of this book. We should have spent more time together. Miss you.

Introduction

PHP 8 is a major update of one of the most widely used languages for developing dynamic web sites. It was released in November 2020. So how could a book released less than 12 months later have managed to get to its fifth edition? Quite simply, this is the fifth iteration of my book *PHP Solutions* that was first published in 2006. When the fourth edition came out in 2019, it was felt important to indicate which version of PHP it covered. So, although the structure of the book remains close to the original, the code has gone through major revision each time.

The fact that *PHP Solutions* has remained so popular owes a great deal to the concept of the book's first editor, Chris Mills. He wanted a book that dealt with practical problems in easily digestible bites; but we agreed that it shouldn't be yet another code "cookbook," a format that was popular at the time. The problem with the cookbook approach is that the reader is presented with a potentially useful block of code but no indication of how it might be used in a real-world situation. *PHP Solutions* aims to provide solutions to practical problems rather than a series of meaningless exercises.

How Easy Is It?

I've always felt concerned about unduly raising readers' expectations with the subtitle of this book, Dynamic Web Design and Development Made Easy. PHP is not difficult, but nor is it like an instant cake mix: just add water and stir. Every web site is different, so it's impossible to grab a script, paste it into a web page, and expect it to work. My aim is to help web designers with little or no knowledge of programming gain the confidence to dive into the code and adjust it to their own requirements.

You don't need any previous experience of PHP or another programming language to be able to use this book; but it does move at a fast pace. After the first few chapters, you start working with relatively advanced features of the language. Don't let that put you off. Regard it as a challenge.

How you use the book will depend on your level of experience. If you're new to PHP and programming, start at the beginning and work your way gradually through the book. It's organized as a logical sequence with each chapter building on knowledge and skills gained in previous ones. When describing the code, I try to explain what it does in plain language. I avoid jargon, but not technical terms (each new term is described briefly when it's first encountered). If you have more experience with PHP, you can probably jump straight into whatever interests you. Even if the code makes sense to you without my explanations, I hope the text throws light onto my thought processes when solving a problem with PHP.

A Word of Caution About PHP Versions

Because hosting companies are often slow to upgrade the version of PHP that they offer, previous editions of this book provided workarounds for older versions of PHP. This time, I don't. In some respects, this is a gamble. As of mid-2021, less than one percent of web servers running PHP were using PHP 8. This means code that works perfectly in a local testing environment is likely to break when it's uploaded to a remote server unless you have upgraded to PHP 8. However, active support for the last version of PHP 7 (7.4) ends in November 2021, shortly after this book's publication.

PHP isn't like that old car you've been running for years and doesn't need changing as long as you give it sufficient love and oil. PHP is constantly being updated, not only to add new features but also to fix bugs and security issues. Even if you're not interested in the new features, you should be interested in security fixes. The Internet can be a wild place with lots of unsavory characters trying to find exploitable holes in web sites. This book contains a lot of advice on security, but it can't protect you from security issues that are uncovered in the PHP core. Making sure that your remote server is kept up to date is an indispensable insurance policy to minimize your risks. And it shouldn't cost you any extra because PHP is free (although hosting companies charge for their services).

If you really need code that's compatible with PHP 7, check out the fourth edition of this book. Better still, make the move to the most up-to-date version of PHP.

What's New in This Edition?

All the code has been extensively reviewed and updated to take advantage of time-saving new features in PHP 8, including named arguments, constructor property promotion, and the `match` expression. This particularly affects the custom classes in Chapters 9–11. They have been radically rewritten using named arguments to avoid the need for public methods to modify their behavior. There are fewer changes in the second half of the book from Chapter 12 onward because the only significant change PHP 8 makes to interacting with a database is that PDO (PHP Data Objects) now throws an exception by default when it encounters an error. Nevertheless, each chapter has been thoroughly reviewed and revised.

Using the Example Files

All the files necessary for working through this book can be downloaded from the Apress web site via the **Download Source Code** button located at `www.apress.com/9781484271407`.

Set up a PHP development environment, as described in Chapter 2. Unzip the files and copy the `php8sols` folder and all its contents into your web server's document root. The code for each chapter is in a folder named after the chapter: `ch01`, `ch02`, and so on. Follow the instructions in each PHP solution, and copy the relevant files to the site root or the work folder indicated.

Where a page undergoes several changes during a chapter, I have numbered the different versions like this: `index_01.php`, `index_02.php`, and so on. When copying a file that has a number, remove the underscore and number from the filename, so `index_01.php` becomes `index.php`. If you are using a program that prompts you to update links when moving files from one folder to another, do not update them. The links in the files are designed to pick up the right images and style sheets when located in the target folder. I have done this so you can use a file comparison utility to check your files against mine.

If you don't have a file comparison utility, I strongly urge you to install one. It will save you hours of head-scratching when trying to spot the difference between your version and mine. A missing semicolon or mistyped variable can be hard to spot in dozens of lines of code. Windows users can download WinMerge for free from `http://winmerge.org/`. I use the file comparison utility built into my favorite script editor, PhpStorm. BBEdit on a Mac includes a file comparison utility. If you're comfortable using Terminal on a Mac, the diff utility is installed by default.

Layout Conventions

To keep this book as clear and easy to follow as possible, the following text conventions are used throughout:

Important words or concepts are normally highlighted on the first appearance in **bold type**.

Code is presented in `fixed-width font`.

New or changed code is normally presented in **`bold fixed-width font`**.

Pseudocode and variable input are written in *`italic fixed-width font`*.

Menu commands are written in the form Menu ➤ Submenu ➤ Submenu.

Where I want to draw your attention to something, I've highlighted it, like this:

Ahem, don't say I didn't warn you.

CHAPTER 1

What Is PHP 8?

PHP 8, released in late November 2020, is a major update of one of the most popular programming languages. According to Web Technology Surveys (https://w3techs.com/technologies/details/pl-php/all/all), PHP is deployed on more than four in every five web sites that use a server-side language. In spite of its popularity, PHP has a lot of detractors, mainly because of the way the language evolved in the early years. This resulted in the names of related functions and the order of arguments being sometimes inconsistent. And some of its features posed a security risk in inexperienced hands. Concerted efforts to improve the language since 2012 have eliminated most of the problems.

PHP is now a mature, powerful language that's become the most widely used technology for creating dynamic web sites. It's used by major enterprises, including Wikipedia, Mailchimp, and Tumblr, as well as powering the popular WordPress, Drupal, and Joomla content management systems. PHP brings web sites to life in the following ways:

- Sends feedback from your web site directly to your mailbox
- Uploads files through a web page
- Generates thumbnails from larger images
- Reads and writes to files
- Displays and updates information dynamically
- Uses a database to display and store information
- Makes web sites searchable
- And much more…

By reading this book, you'll be able to do all that. Not only is PHP easy to learn; it's platform-neutral, so the same code runs on Windows, macOS, and Linux. All the software you need to develop with PHP is open source and free.

In this chapter, you'll learn about the following:

- How PHP has grown into the most widely used technology for dynamic web sites
- How PHP makes web pages dynamic
- How difficult—or easy—PHP is to learn
- Whether PHP is safe
- What's new in PHP 8
- What software you need to write PHP

D. Powers, *PHP 8 Solutions*, https://doi.org/10.1007/978-1-4842-7141-4_1

How PHP Has Grown

PHP started out in 1995 with rather modest ambitions. It was originally called Personal Home Page Tools (PHP Tools). One of its main goals was to create a guestbook by gathering information from an online form and displaying it on a web page. Within three years, it was decided to drop Personal Home Page from the name, because it sounded like something for hobbyists and didn't do justice to the range of sophisticated features that had since been added. That left the problem of what the initials PHP should stand for. In the end, it was decided to call it PHP Hypertext Preprocessor; but most people simply call it PHP.

PHP has continued to develop over the years, adding new features all the time. One of the language's great attractions is that it remains true to its roots. Although it has support for sophisticated object-oriented programming, you can start using it without diving into complex theory. PHP's original creator, Rasmus Lerdorf, once described it as "a very programmer-friendly scripting language suitable for people with little or no programming experience as well as the seasoned web developer who needs to get things done quickly." You can start writing useful scripts right away, yet be confident in knowing that you're using a technology with the capability to develop industrial-strength applications.

■ **Note** Much of the code in this book uses features that are new to PHP 8. It is not guaranteed to work on older versions of PHP.

How PHP Makes Pages Dynamic

PHP was originally designed to be embedded in the HTML of a web page, and that's the way it's often still used. For example, to display the current year in a copyright notice, you could put this in your footer:

```
<p>&copy; <?php echo date('Y'); ?> PHP 8 Solutions</p>
```

On a PHP-enabled web server, the code between the <?php and ?> tags is automatically processed and displays the year like this:

© 2021 PHP 8 Solutions

This is only a trivial example, but it illustrates some of the advantages of using PHP:

- The year is automatically updated at the stroke of midnight on New Year's Day.
- The date is calculated by the web server, so it's not affected if the clock in the user's computer is set incorrectly. However, as you'll learn later, PHP follows the server's time zone; but this can be adjusted programmatically.

Although it's convenient to embed PHP code in HTML like this, it's repetitive and can lead to mistakes. It can also make your web pages difficult to maintain, particularly once you start using more complex PHP code. Consequently, it's common practice to store a lot of dynamic code in separate files and then use PHP to build your pages from the different components. The separate files—or *include files*, as they're usually called—can contain only PHP, only HTML, or a mixture of both.

As a simple example, you can put your web site's navigation menu in an include file and use PHP to include it in each page. Whenever you need to change the menu, you edit only the include file, and the changes are automatically reflected in every page that includes the menu. Just imagine how much time that saves on a web site with dozens of pages!

With an ordinary HTML page, the content is fixed by the web developer at design time and uploaded to the web server. When somebody visits the page, the web server simply sends the HTML and other assets, such as images and the style sheet. It's a simple transaction—the request comes from the browser, and the fixed content is sent back by the server. When you build web pages with PHP, much more goes on. Figure 1-1 shows what happens.

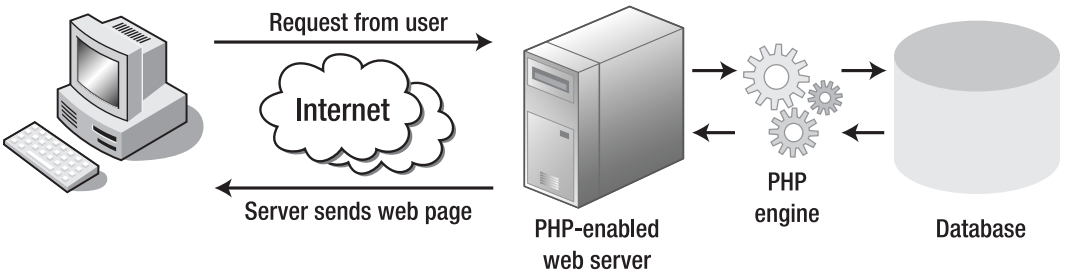

Figure 1-1. *The web server builds each PHP page dynamically in response to a request*

When a PHP-driven web site is visited, it sets in motion the following sequence of events:

1. The browser sends a request to the web server.
2. The web server passes the request to the PHP engine embedded in the server.
3. The PHP engine processes the code in the requested page. In many cases, it might also query a database before building the page.
4. The server sends the completed page back to the browser.

This process usually takes only a fraction of a second, so the visitor to a PHP web site is unlikely to notice any delay. Because each page is built individually, PHP sites can respond to user input, displaying different content when a user logs in or showing the results of a database search.

Creating Pages That Think for Themselves

PHP is a server-side language. The PHP code remains on the web server. After it has been processed, the server sends only the output of the script. Normally, this is HTML, but PHP can also be used to generate other web languages, such as JSON (JavaScript Object Notation) or XML (Extensible Markup Language).

PHP enables you to introduce logic into your web pages that is based on alternatives. Some decisions are made using information that PHP gleans from the server: the date, the time, the day of the week, information in the page's URL, and so on. If it's Wednesday, it will show Wednesday's TV schedules. At other times, decisions are based on user input, which PHP extracts from online forms. If you have registered with a site, it will display personalized information—that sort of thing.

How Hard Is PHP to Use and Learn?

PHP isn't rocket science, but don't expect to become an expert in 5 minutes. Perhaps the biggest shock to newcomers is that PHP is far less tolerant of mistakes than browsers are with HTML. If you omit a closing tag in HTML, most browsers will still render the page. If you omit a closing quote, semicolon, or brace in PHP, you'll get an uncompromising error message like the one shown in Figure 1-2. This affects all programming languages, such as JavaScript and C#, not just PHP.

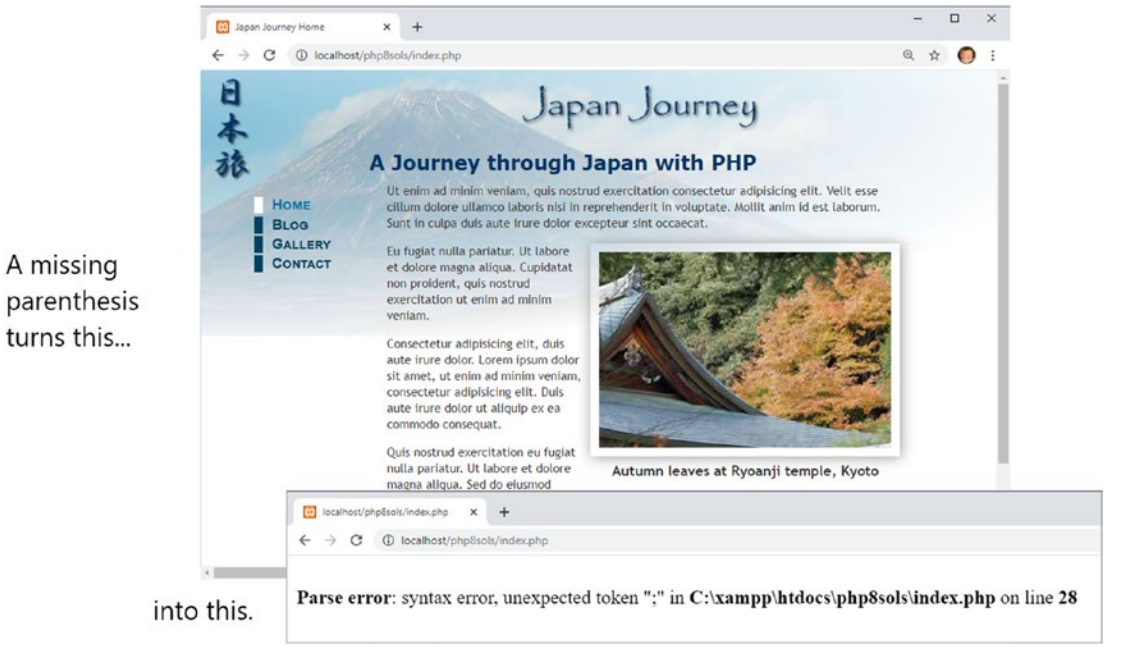

Figure 1-2. *Server-side languages like PHP are intolerant of most coding errors*

If you use a visual design tool and never look at the underlying code, it's time to rethink your approach. Mixing PHP with poorly structured HTML is likely to lead to problems. PHP uses loops to perform repetitive tasks, such as displaying the results of a database search. A loop repeats the same section of code—usually a mixture of PHP and HTML—until all results have been displayed. If you put the loop in the wrong place or if your HTML is badly structured, your page is likely to collapse like a house of cards.

If you're not already in the habit of doing so, it's a good idea to check your pages using the World Wide Web Consortium's (W3C) Nu HTML Checker (`https://validator.w3.org/nu/`).

■ **Note** The W3C is the international body that develops standards such as HTML and CSS to ensure the long-term growth of the Web. It's led by the inventor of the World Wide Web, Tim Berners-Lee. To learn about the W3C, see `www.w3.org/Consortium/mission`.

Can I Just Copy and Paste the Code?

There's nothing wrong with copying the code in this book. That's what it's there for. I've structured this book as a series of practical projects. I explain what the code is for and why it's there. Even if you don't understand exactly how it all works, this should give you sufficient confidence to know which parts of the code to adapt to your own needs and which parts are best left alone. But to get the most out of this book, you need to start experimenting and then come up with your own solutions.

PHP has thousands of built-in functions that perform all sorts of tasks, such as converting text to uppercase, generating thumbnail images from full-sized ones, or connecting to a database. The real power comes from combining these functions in different ways and adding your own conditional logic.

How Safe Is PHP?

PHP is like the electricity or kitchen knives in your home: handled properly, it's very safe; handled irresponsibly, it can do a lot of damage. One of the inspirations for the first edition of this book was a spate of attacks that exploited a vulnerability in email scripts, turning web sites into spam relays. The solution is quite simple, as you'll learn in Chapter 6, but even many years later, I still see people using the same insecure techniques, exposing their sites to attack.

PHP is not unsafe, nor does everyone need to become a security expert to use it. What is important is to understand the basic principle of PHP safety: *always check user input before processing it*. You'll find that to be a constant theme in this book. Most security risks can be eliminated with very little effort.

The best way to protect yourself is to understand the code you're using.

What's New in PHP 8?

The list of new features and changes in PHP 8 is extensive; but the most important is the adoption of a Just In Time (JIT) compiler. This changes the way PHP code is converted into machine code that the server can understand. As the name implies, JIT is designed to speed up performance. Zeev Surasky, one of the authors of the JIT proposal, has created a short video (`https://youtu.be/dWH65pmnsrI`) to demonstrate the dramatic improvement JIT is capable of. However, such improvements in speed affect only processor-intensive calculations—at least for the time being. WordPress runs no faster on PHP 8 than on PHP 7, which is where the real speed gains over previous versions were made.

Many of the new features in PHP 8 are designed to make code more concise and efficient. For example, named arguments eliminate the need to repeat the default values of multiple arguments to a function if you want to change only one of them. Constructor property promotion greatly simplifies the declaration of properties in a class definition, typically reducing the number of lines by one third. The new nullsafe operator similarly reduces the amount of code needed to call a method or fetch the property on the result of an expression only if it's not null. Details of these and other new features are covered in Chapters 3 and 4.

An important consideration of migrating existing code to a new version of PHP is whether incompatible changes will break your application. If you have been following recommended best practices, you're unlikely to have problems. However, there are some important changes that you should be aware of, as follows:

- The error control operator (`@`) will no longer silence fatal errors.
- Non-strict comparisons between numbers and non-numeric strings using two equal signs (`==`) now convert the number to a string and compare the strings. This means that some comparisons that previously equated to `true` are now `false`.
- Methods with the same name as the class are no longer interpreted as constructors. You must use `__construct()` instead.
- You can no longer define case-insensitive constants.
- `match` is now a reserved keyword.
- `#[` is no longer regarded as the start of a comment because this syntax is used for a new feature called attributes. This affects only an opening square bracket after #.

■ **Note** See `www.php.net/manual/en/migration80.incompatible.php` for a complete list of backward-incompatible changes in PHP 8.

What Software Do I Need to Write PHP?

Strictly speaking, you don't need any special software to write PHP scripts. PHP code is plain text and can be created in any text editor, such as Notepad on Windows or TextEdit on macOS. Having said that, your life will be a lot easier if you use a program that has features designed to speed up the development process. There are many available—both free and on a paid-for basis.

What to Look for When Choosing a PHP Editor

If there's a mistake in your code, your page will probably never make it as far as the browser, and all you'll see is an error message. You should choose a script editor that has the following features:

- **PHP syntax checking**: This used to be found only in expensive, dedicated programs, but it's now a feature in several free programs. Syntax checkers monitor the code as you type and highlight errors, saving a great deal of time and frustration.
- **PHP syntax coloring**: Code is highlighted in different colors according to the role it plays. If your code is in an unexpected color, it's a sure sign you've made a mistake.
- **PHP code hints**: PHP has so many built-in functions that it can be difficult to remember how to use them, even for an experienced user. Many script editors automatically display tooltips with reminders of how a particular piece of code works.
- **Line numbering**: Finding a specific line quickly makes troubleshooting a lot simpler.
- **A "balance braces" feature**: Parentheses (( )), square brackets ([ ]), and curly braces ({ }) must always be in matching pairs. It's easy to forget to close a pair. All good script editors help find the matching parenthesis, bracket, or brace.

The program you're already using to build web pages might already have some or all of these features. Even if you don't plan to do a lot of PHP development, you should consider using a dedicated script editor if your web development program doesn't support syntax checking. The following dedicated script editors have all the essential features, such as syntax checking and code hints. It's not an exhaustive list, but rather one based on personal experience:

- **PhpStorm** (`www.jetbrains.com/phpstorm/`): Although this is a dedicated PHP editing program, it has excellent support for HTML, CSS, and JavaScript. It's my favorite program for developing with PHP. It's sold on an annual subscription. If you cancel after a minimum of 12 months, you get a perpetual license for an older version.
- **Visual Studio Code** (`https://code.visualstudio.com/`): An excellent code editor from Microsoft that runs not only on Windows but also on macOS and Linux. It's free and has built-in support for PHP.
- **Sublime Text** (`www.sublimetext.com/`): If you're a Sublime Text fan, there are plug-ins for PHP syntax coloring, syntax checking, and documentation. Free for evaluation, but you should buy the relatively inexpensive license for continued use.
- **Zend Studio** (`www.zend.com/products/zend-studio`): Powerful, dedicated PHP editor created by Zend, the company run by leading contributors to the development of PHP. It runs on Windows, macOS, and Linux. Different pricing applies to personal and commercial use.

- **Eclipse PHP Development Tools (PDT)** (https://projects.eclipse.org/projects/tools.pdt): Similar to Zend Studio but with the advantage of being free. It runs on Eclipse, the open source IDE that supports multiple computer languages. If you have used Eclipse for other languages, you should find it relatively easy to use. PDT runs on Windows, macOS, and Linux.

So Let's Get On with It…

This chapter has provided only a brief overview of what PHP can do to add dynamic features to your web sites and what software you need to do so. The first stage in working with PHP is to set up a testing environment. The next chapter covers what you need for both Windows and macOS.

CHAPTER 2

Getting Ready to Work with PHP

Now that 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
- Creating a local testing setup with 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`.

   If you see a three-part number like 8.0.3 displayed onscreen, you're in business: PHP is enabled. The number tells you which version of PHP is running on your server.

D. Powers, *PHP 8 Solutions*, https://doi.org/10.1007/978-1-4842-7141-4_2