Contents

How to Use This Book

The examples in this book demonstrate features of the PHP scripting language, and the screenshots illustrate the actual results produced by the listed code examples. Certain colorization conventions are used to clarify the code listed in the steps...

Components of the PHP language are colored blue, programmer-specified names are colored red, literal text and numeric values are black, and code comments are green, like this:

```
<?php

        # Write the traditional greeting.
        $greeting = 'Hello World!' ;
        echo "<h1>$greeting</h1>" ;
?>
```

Additionally, in order to identify each source code file described in the steps, a file icon and file name appears in the margin alongside the steps:

script.php   page.html   style.css   data.xml   text.txt   image.png

Grab the Source Code

For convenience, the source code files from all examples featured in this book are available in a single ZIP archive. You can obtain this archive by following these easy steps:

1. Browse to **www.ineasysteps.com** then navigate to Free Resources and choose the Downloads section

2. Next, find PHP in easy steps, 4th edition in the list, then click on the hyperlink entitled All Code Examples to download the ZIP archive file

3. Now, extract the archive contents to the web server's **/htdocs** directory on your PC

If you don't achieve the result illustrated in any example, simply compare your code to that in the original example files you have downloaded to discover where you went wrong.

1 Getting started

Welcome to the exciting world of the interactive web with PHP. This chapter demonstrates how to create a dynamic development environment with a web server and the PHP engine.

Introducing PHP

The most appealing modern websites provide a customized user experience by dynamically responding to some current conditions – user name, time of day, latest blog, shopping cart contents, etc. Many of these dynamic websites are created using PHP.

This is the official logo of the PHP project – the official online home of PHP can be found at **php.net**

What is PHP?

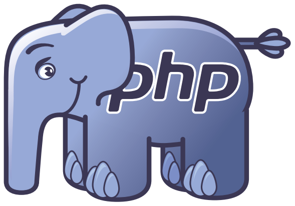
PHP is a widely-used general purpose scripting language that is especially suited for web development and can be embedded into HTML. It was created by programmer Rasmus Lerdorf, as a set of scripts to maintain his website that he released as "Personal Home Page Tools (PHP Tools) version 1.0" on June 8, 1995.

The tools were extended in the version 2 release of 1997, and the name changed to become a recursive acronym "PHP: Hypertext Preprocessor" in version 3 the following year. Performance, reliability and extensibility were improved in 2000 with the release of PHP4, which was powered by the Zend engine virtual machine.

Subsequently, PHP5 was released in 2004 powered by the new Zend II engine and produced as free software by the PHP group. A planned experimental version PHP6, which intended to introduce native Unicode support throughout PHP, was abandoned but PHP7 was released in 2015. PHP8 was released in 2020 and offers "Just In Time" (JIT) compilation for improved performance. Today, PHP is installed on over 20 million websites and 1 million web servers.

Why is PHP popular?

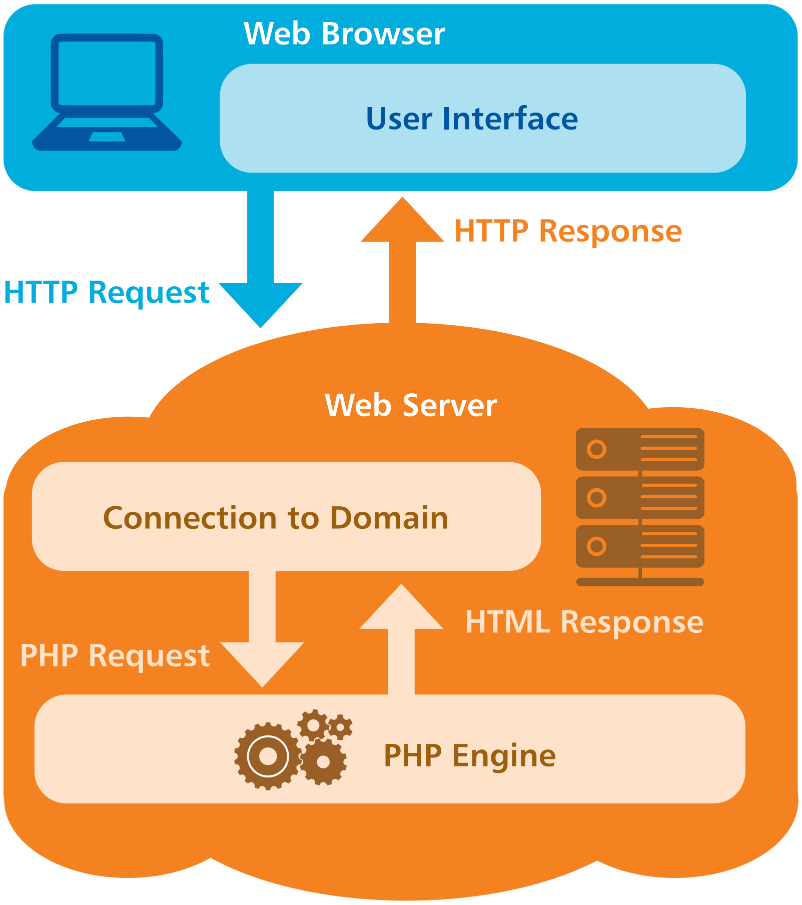
- PHP is extremely simple for a newcomer, but offers many advanced features for a professional programmer.
- PHP code is enclosed in special start and end processing tags that allow you to jump into and out of "PHP mode", to implement instructions within an HTML document.
- PHP code is executed on the server ("server-side"), unlike JavaScript code that is executed in the browser ("client-side"). The client receives the results of running the script without knowing what the underlying code was. Recently, server-side has become known as "The Cloud".



This is the "elePHPant" – the mascot of the PHP project, designed by Vincent Pontier.

Understanding The Cloud

Whenever a user asks to view a web page in their browser, it requests the page from the web server and receives the page in response, via the HTTP protocol. Where a web page contains PHP script, the web server will first call upon the PHP engine to process the code before sending the response to the web browser:



Hot tip

HTTP (HyperText Transfer Protocol) is the common communication standard that allows any computer connected to any web server to access files across the web.

The ensuing pages describe how to create a development environment for interactive websites by installing the following server-side technologies on your own computer:

- **Web Server** – Abyss Web Server X1
- **PHP Engine** – PHP 8 (8.0.1)

Don't forget

The examples in this book are created and tested with these software versions but may require modification for other versions.

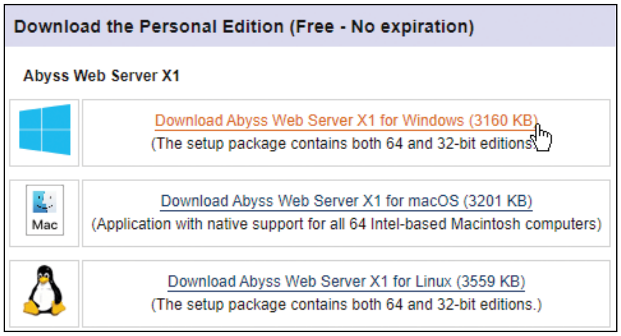
Installing the Abyss server

Abyss X1 is a free compact web server available for Windows, macOS, and Linux operating systems available for download at **aprelium.com**
Despite its small footprint, Abyss supports many powerful features including dynamic content generation with server-side scripts – so is an ideal companion for PHP.

The Abyss Web Server can be installed on your own computer to provide an environment for interactive PHP website development:

Further guidance on installation of the Abyss Web Server is available at **aprelium.com/abyssws/start.html**

1. Download the Abyss X1 Web Server setup package for your system from **aprelium.com/abyssws/download.php**

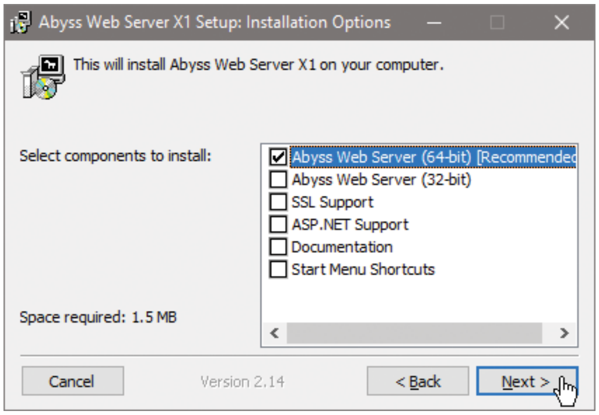


Download the Personal Edition (Free - No expiration)

Abyss Web Server X1

	Download Abyss Web Server X1 for Windows (3160 KB) (The setup package contains both 64 and 32-bit editions.)
Mac	Download Abyss Web Server X1 for macOS (3201 KB) (Application with native support for all 64 Intel-based Macintosh computers)
	Download Abyss Web Server X1 for Linux (3559 KB) (The setup package contains both 64 and 32-bit editions.)

2. Run the setup installer and **Agree** the License terms, then select the Abyss Web Server component and click **Next**



Abyss Web Server X1 Setup: Installation Options

This will install Abyss Web Server X1 on your computer.

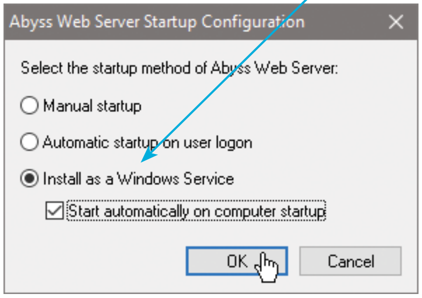
Select components to install:

- [✓] Abyss Web Server (64-bit) [Recommended
- [ ] Abyss Web Server (32-bit)
- [ ] SSL Support
- [ ] ASP.NET Support
- [ ] Documentation
- [ ] Start Menu Shortcuts

Space required: 1.5 MB

The Abyss setup package for Windows is an executable file named **abwsx1.exe** that you run to install the web server.

3 Accept the suggested location of **C:\Abyss Web Server** then choose to **Install as a Windows Service**


Abyss Web Server Startup Configuration

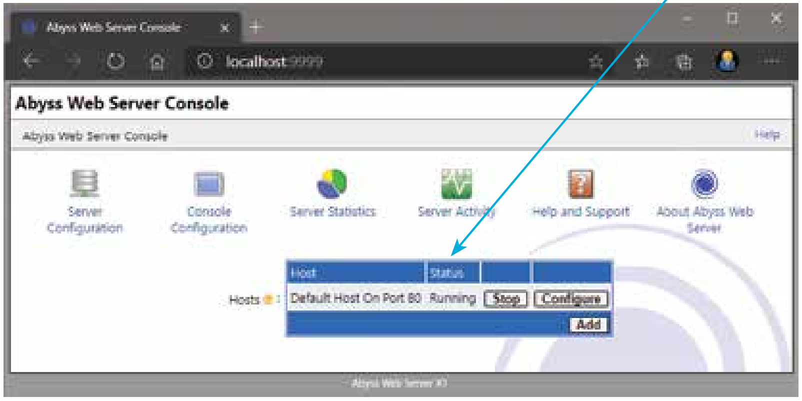
Select the startup method of Abyss Web Server:

- ○ Manual startup
- ○ Automatic startup on user logon
- ● Install as a Windows Service
  - ☑ Start automatically on computer startup

After the installation process completes, your system's default web browser will open, displaying the Abyss Web Server Console.

4 Select your preferred language, then enter a name and password for future access to the Abyss Console

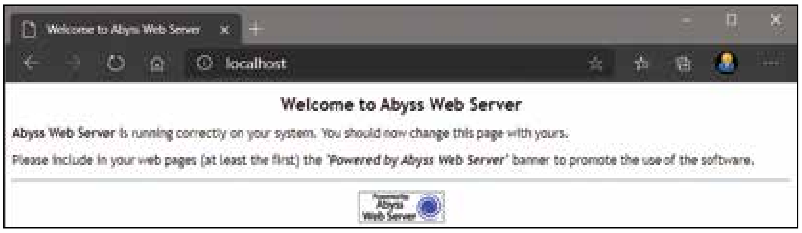
5 Now, log in using your chosen name and password to see the Abyss Console confirm the server Status as "Running"



Abyss Web Server Console

Host	Status		
Default Host On Port 80	Running	Stop	Configure

6 Type **http://localhost** into your browser address field, then hit **Enter** – to see the default Abyss "Welcome" page



Welcome to Abyss Web Server

Abyss Web Server is running correctly on your system. You should now change this page with yours.

Please include in your web pages (at least the first) the 'Powered by Abyss Web Server' banner to promote the use of the software.

Beware

If you choose the "Manual startup" option, the Abyss logo will not appear in your system tray for easy start/stop control and access to the server console. Instead, the console can be found with your browser at **http://localhost:9999** or numerically at **http://127.0.0.1:9999** ("localhost" is an alias for the IP address 127.0.0.1).

Hot tip

In the Abyss console, click the "Configure" button then the General icon to see the default HTTP Port is 80 and the default Documents Path (where your web pages will reside) is **/htdocs**.

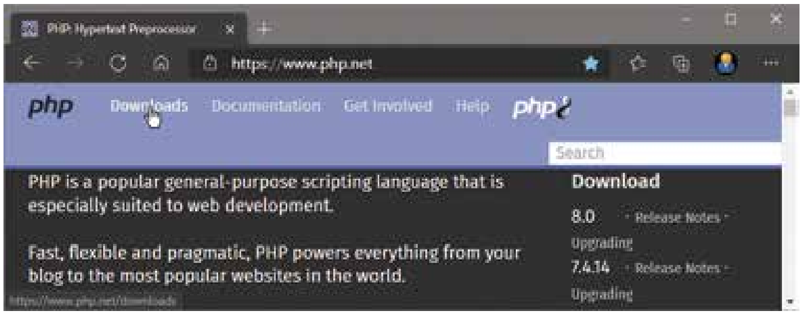
Installing the PHP engine

The PHP interpreter "engine", which implements PHP scripts within web pages, is available for Windows, macOS and Linux operating systems as a free download at **php.net**

Further guidance on installation of PHP is available at **php.net/manual/en/install.php**

1. Open a web browser and navigate to **php.net**

2. Next, choose the **Downloads** item on the page menu – to open a "Downloads" page



PHP is a popular general-purpose scripting language that is especially suited to web development.

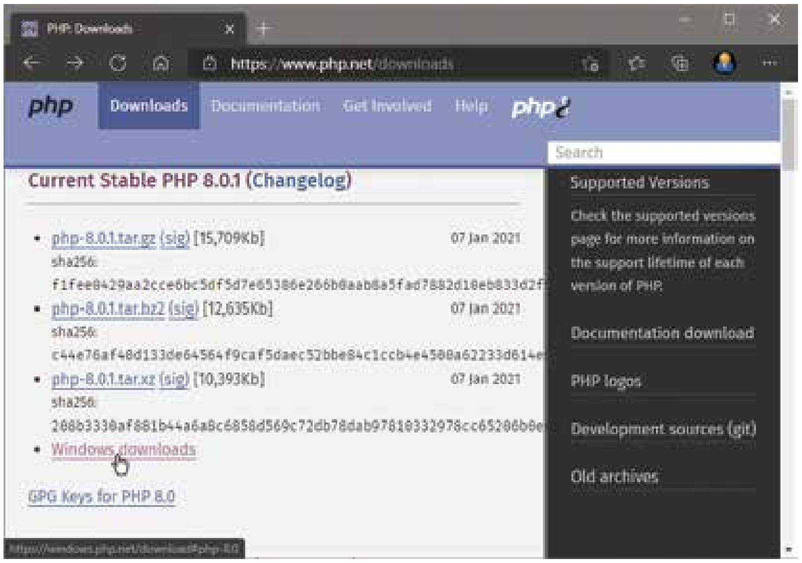
Fast, flexible and pragmatic, PHP powers everything from your blog to the most popular websites in the world.

Download

8.0 · Release Notes ·
Upgrading
7.4.14 · Release Notes ·
Upgrading

https://www.php.net/downloads

3. Now, select a package to download or select the **Windows downloads** link – to open a "Windows binaries" page



Current Stable PHP 8.0.1 (Changelog)

- php-8.0.1.tar.gz (sig) [15,709Kb] 07 Jan 2021
  sha256:
  f1fee8429aa2cce6bc5df5d7e65386e266b0aab8a5fad7882d10eb833d2f
- php-8.0.1.tar.bz2 (sig) [12,635Kb] 07 Jan 2021
  sha256:
  c44e76af48d133de64564f9caf5daec52bbe84c1ccb4e4500a62233d614e
- php-8.0.1.tar.xz (sig) [10,393Kb] 07 Jan 2021
  sha256:
  208b3330af881b44a6a8c6858d569c72db78dab97810332978cc65206b0e
- Windows downloads

GPG Keys for PHP 8.0

Supported Versions

Check the supported versions page for more information on the support lifetime of each version of PHP.

Documentation download

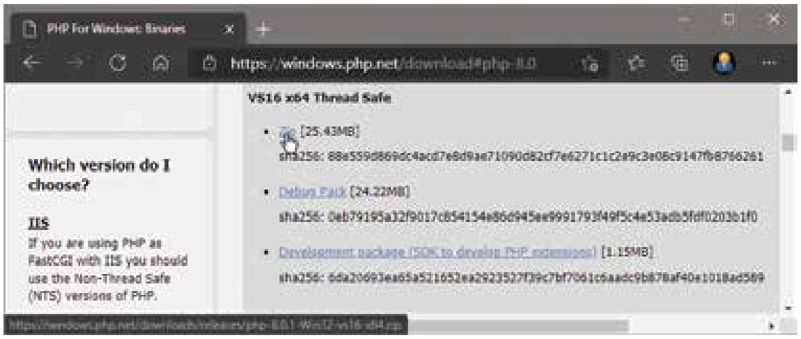
PHP logos

Development sources (git)

Old archives

https://windows.php.net/download#php-8.0

4 Then, click the **Zip** link to download a Thread Safe archive of PHP for 64-bit systems


PHP For Windows: Binaries
https://windows.php.net/download#php-8.0

VS16 x64 Thread Safe

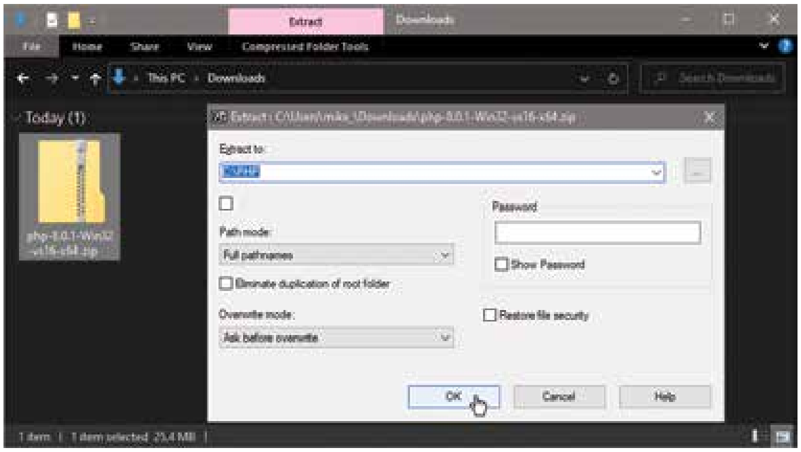
- Zip [25.43MB]
  sha256: 88e559d869dc4acd7e8d9ae71090d82cf7e6271c1c2e9c3e08c9147fb8766261
- Debug Pack [24.22MB]
  sha256: 0eb79195a32f9017c854154e86d945ee9991793f49f5c4e53adb5fdf0203b1f0
- Development package (SDK to develop PHP extensions) [1.15MB]
  sha256: 6da20d93ea65a521652ea2923527f39c7bf7061c6aadc9b878af40e1018ad589

Which version do I choose?

IIS
If you are using PHP as FastCGI with IIS you should use the Non-Thread Safe (NTS) versions of PHP.

https://windows.php.net/downloads/releases/php-8.0.1-Win32-vs16-x64.zip

5 When the download completes, extract its contents to a convenient location on your system – such as **C:\PHP**



Extract : C:\Users\mike_\Downloads\php-8.0.1-Win32-vs16-x64.zip

Extract to: C:\PHP

Path mode: Full pathnames

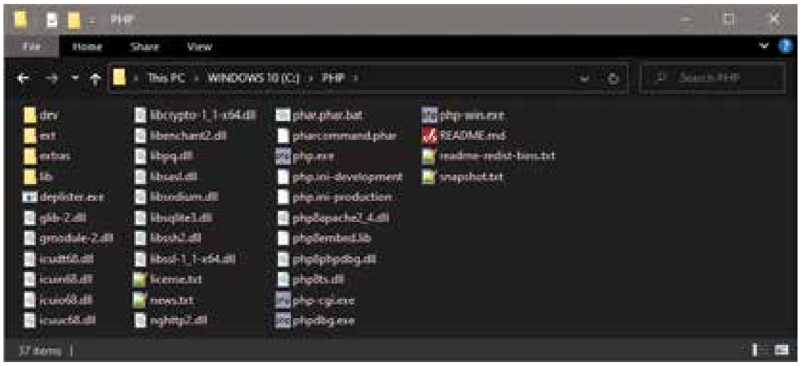
☐ Eliminate duplication of root folder

Overwrite mode: Ask before overwrite

Password:

☐ Show Password

☐ Restore file security



```
dev              libcrypto-1_1-x64.dll  phar.phar.bat            php-win.exe
ext              libenchant2.dll        pharcommand.phar         README.md
extras           libpq.dll              php.exe                  readme-redist-bins.txt
lib              libsasl.dll            php.ini-development      snapshot.txt
deplister.exe    libsodium.dll          php.ini-production
glib-2.dll       libsqlite3.dll         php8apache2_4.dll
gmodule-2.dll    libssh2.dll            php8embed.lib
icudt68.dll      libssl-1_1-x64.dll     php8phpdbg.dll
icuin68.dll      license.txt            php8ts.dll
icuio68.dll      news.txt               php-cgi.exe
icuuc68.dll      nghttp2.dll            phpdbg.exe
```

Hot tip

The PHP installation location will be required when configuring the Abyss Web Server to integrate with PHP – make a note of the Destination Folder.

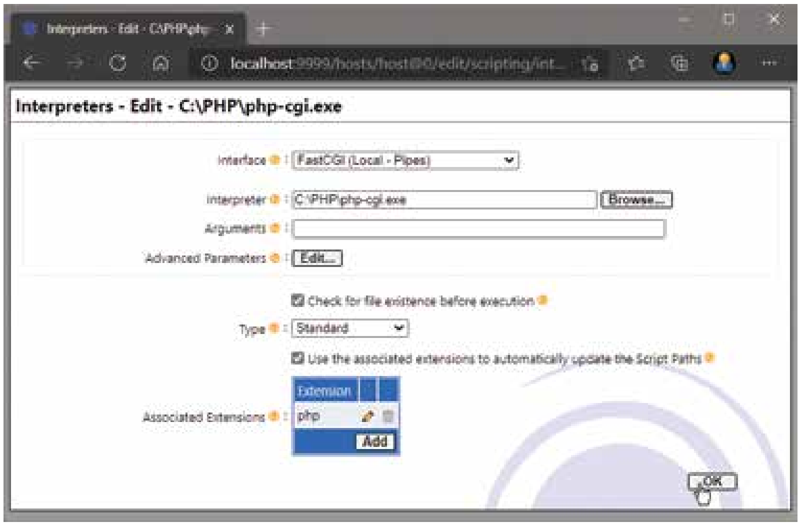
Don't forget

Following installation of PHP, the web server cannot yet execute PHP scripts until it is configured to recognize them and to find the PHP interpreter engine – all as described on pages 14-15.

Integrating Abyss and PHP

The Abyss Web Server must be configured to recognize PHP scripts and employ the PHP interpreter when it encounters them. This is achieved in the Abyss Console by associating the file extension ".php" as being PHP scripts, and by specifying the location of the PHP engine on your system to interpret them:

1. Enter **http://localhost:9999** into your browser address field to launch the Abyss Web Server Console, then click the **Configure** button – to open the Configuration page
2. Click on the **Scripting Parameters** icon – to open the Scripting Parameters page for editing
3. Ensure that the **Enable Scripts Execution** box is checked, then click the **Add** button in the Interpreters table – to open the Interpreters-Add page
4. Set the **Interface** parameter to "FastCGI (Local - Pipes)"
5. Set the **Interpreter** parameter to the PHP path location on your system of the **php-cgi.exe** file
6. Set the **Associated Extensions** parameter to "php", so your configuration should look like that shown below:


Interpreters - Edit - C:\PHP\php
localhost:9999/hosts/host@0/edit/scripting/int...

Interpreters - Edit - C:\PHP\php-cgi.exe

Interface : FastCGI (Local - Pipes)
Interpreter : C:\PHP\php-cgi.exe
Arguments :
Advanced Parameters : Edit...
[✓] Check for file existence before execution
Type : Standard
[✓] Use the associated extensions to automatically update the Script Paths
Associated Extensions :

Extension
php

Further guidance on configuration of the Abyss Web Server is available online at **aprelium.com/abyssws/start.html**

Don't forget

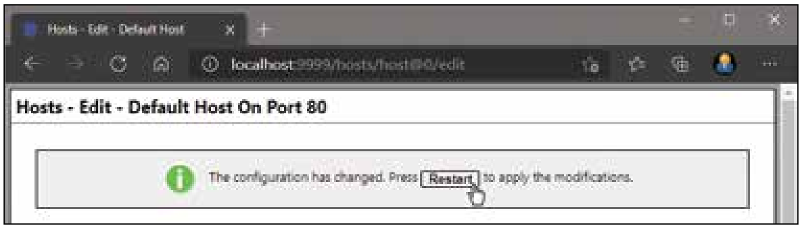
To clarify the code examples in this book, components of the PHP language are colored **blue**, programmer-specified names are **red**, numeric and string data is **black**, and comments are **green**.

Don't forget

The **localhost** domain name is an alias for the domain IP address of **127.0.0.1** – so the Abyss Web Server Console can alternatively be addressed as **http://127.0.0.1:9999**.

7. Click the **OK** button to validate your new configuration

8. Click the **Restart** button that next appears, to apply the changes you have made to the Abyss configuration



Hosts - Edit - Default Host On Port 80

The configuration has changed. Press Restart to apply the modifications.

Documents can only be interpreted by the PHP engine if served up by the web server using the HTTP protocol. You cannot simply open a PHP file in your browser directly. Always use the location **http://localhost/** to serve the examples in this book.

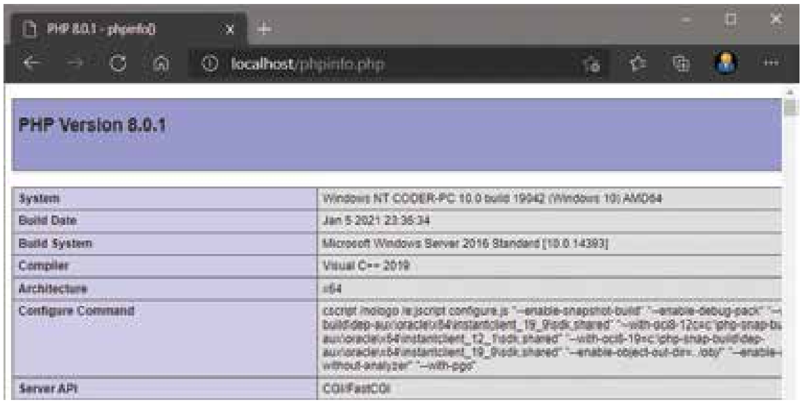
The Abyss Web Server should now be running on your system, correctly configured to recognize that documents having the **.php** file extension should be interpreted by the PHP engine. Configuration can now be tested by creating a simple PHP script for service to your web browser by Abyss:

1. Open a plain text editor and exactly type the script below
   `<?php  phpinfo() ;  ?>`

2. Save the script as **phpinfo.php** in the Abyss document path directory, typically at **C:\Abyss Web Server\htdocs**

3. Exactly enter the location **http://localhost/phpinfo.php** into your web browser's address field to see Abyss serve up a web page containing your PHP version information

phpinfo.php



PHP Version 8.0.1

System	Windows NT CODER-PC 10.0 build 19042 (Windows 10) AMD64
Build Date	Jan 5 2021 23:36:34
Build System	Microsoft Windows Server 2016 Standard [10.0.14393]
Compiler	Visual C++ 2019
Architecture	x64
Configure Command	cscript /nologo /e:jscript configure.js "--enable-snapshot-build" "--enable-debug-pack" "--build-dep-aux\oracle\x64\instantclient_19_9\sdk,shared" "--with-oci8-12c=c:\php-snap-bu aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-19=c:\php-snap-build\dep-aux\oracle\x64\instantclient_19_9\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-without-analyzer" "--with-pgo"
Server API	CGI/FastCGI

PHP scripts are case-sensitive so you must copy the listed script using lowercase characters only.

Embedding PHP script code

PHP script may be embedded within HTML documents – meaning PHP and HTML code can both happily co-exist in the same file. All embedded PHP code must be contained within **<?php** and **?>** tags so it can be readily recognized by the PHP engine for interpretation. Typically, the PHP code will write content into the body section of the HTML document, which is then sent to the web browser:

hello.php

1. Launch a plain text editor and create this valid barebones HTML document with an empty body section

```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Getting Started In PHP</title>
</head>
<body>

</body>
</html>
```

2. Insert tags into the body section to contain PHP code

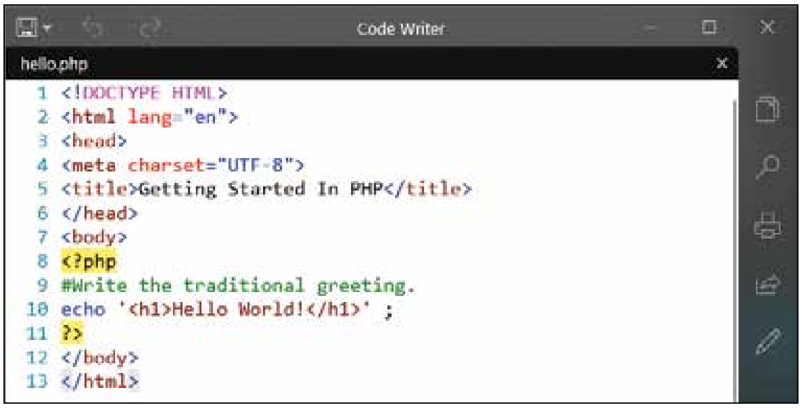
```
<?php

?>
```

3. Next, insert between the PHP tags a descriptive comment and a line of code to write content into the body section

```
# Write the traditional greeting.
echo '<h1>Hello World!</h1>' ;
```

Hot tip

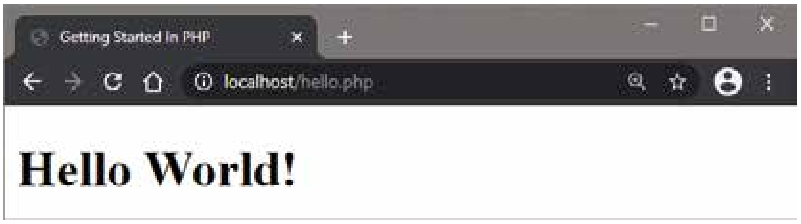
Notice that the descriptive comment is enclosed between the PHP tags for information purposes only.



```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Getting Started In PHP</title>
</head>
<body>
<?php
#Write the traditional greeting.
echo '<h1>Hello World!</h1>' ;
?>
</body>
</html>
```

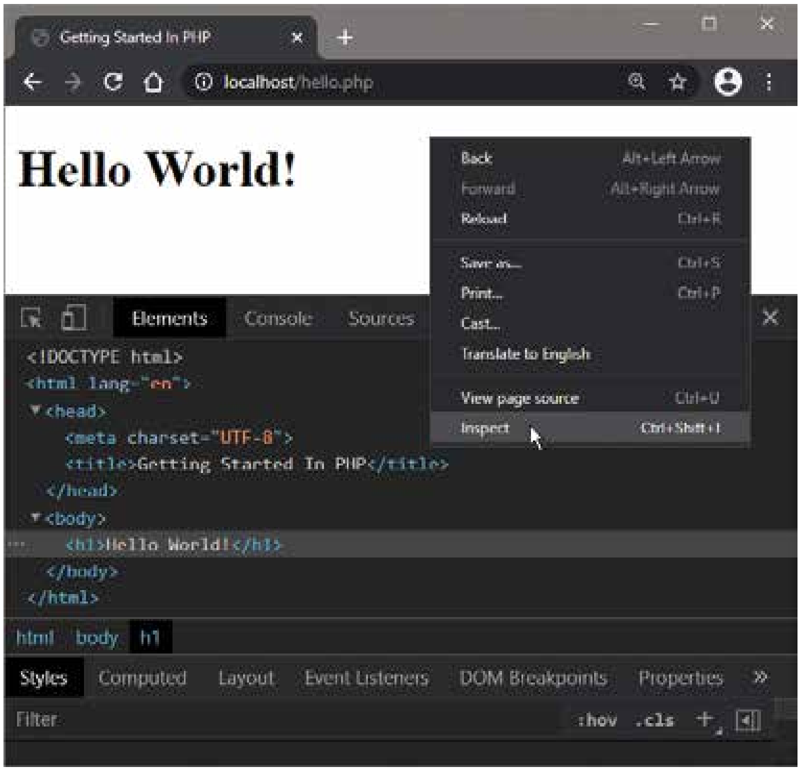
4 Save the document as **hello.php** in the Abyss web server's **/htdocs** folder

5 Now, enter the location **http://localhost/hello.php** into your web browser's address field – to see Abyss serve up a web page containing content written by embedded PHP code



Hello World!

6 Use your web browser's tools to inspect the page's source code – to see that PHP has written the content into the body section, including the HTML **<h1> </h1>** tags



Hello World!

```
<!DOCTYPE html>
<html lang="en">
▼<head>
    <meta charset="UTF-8">
    <title>Getting Started In PHP</title>
  </head>
▼<body>
    <h1>Hello World!</h1>
  </body>
</html>
```

Note that the PHP **echo** instruction literally writes the entire content contained within the pair of **' '** single quote marks. The examples in this book demonstrate embedded PHP script but do not repeatedly list the HTML code.

Scripting by the rules

Tag rules

When the PHP engine receives input from the web server it reads the input, from top to bottom, in a process called "parsing". During the parsing process, the PHP engine (a.k.a. parser) looks for the opening and closing **<?php** and **?>** tags and understands that the content between those tags is script code that it must interpret. Everything outside the **<?php** and **?>** tags is completely ignored, which allows PHP files to have mixed content and allows PHP code to be embedded within HTML, like this:

```
<p>Ignored by PHP and displayed by the browser</p>

<?php echo 'Script code that will be parsed' ; ?>

<p>Also ignored by PHP and displayed by the browser</p>
```

Hot tip

Typically, your embedded PHP code will always have opening and closing **<?php** and **?>** tags around script.

In order for this to work properly it is therefore important that all your PHP script code is enclosed between opening and closing **<?php** and **?>** tags, when embedded in an HTML file.

The only exception to this rule is when PHP script is written in a pure PHP file, which contains only code. In this case, it is preferable to omit the closing **?>** tag, like this:

```
<?php  echo 'Print this First' ;
       echo 'Print this Last' ;
```

Where your PHP script code intends only to insert a single string of text into an HTML document, you may optionally use the PHP short echo **<?=** and **?>** tags:

<?= 'Hello!' ; ?> is equivalent to **<?php echo 'Hello!' ; ?>**

Advanced PHP script code can also insert text only when a tested condition is met, like this:

```
<?php if ( $expression == true ) : ?>

      Insert this text only if the expression is true.

<?php else : ?>

      Otherwise insert this text.

<?php endif ; ?>
```

Don't forget

Conditional testing is fully explained and demonstrated later in this book, in Chapter 3.

Statement rules

Each statement within the PHP language must be terminated by a **;** semicolon character – just as each sentence in the English language must be terminated by a **.** period character. The semicolon is recognized by the parser as marking the end of an individual instruction that it must interpret. So a PHP code block containing two statements could look like this:

```
<?php echo 'First statement' ; echo 'Second statement' ; ?>
```

The closing **?>** tag of a block of PHP code automatically implies a semicolon, however, so you can optionally omit the semicolon terminating the last statement of a PHP block, like this:

```
<?php echo 'First statement' ; echo 'Second statement' ?>
```

Hot tip

It does no harm to terminate the last statement of a PHP block if you wish to do so.

Comment rules

It is often worthwhile adding comments to your PHP script code so it can be more easily understood by others, or by yourself when revisiting your code later. All whitespace and comments are completely ignored by the PHP parser so you can add as many comments as you like, without any adverse effect on performance.

Single-line comments may begin with a **#** hash character, or alternatively they may begin with a **//** double-slash sequence.

Multi-line (block) comments must be enclosed within **/*** and ***/** character sequences, as used in the C programming language:

```
<?php
        echo 'First statement' ; // A single-line comment.

        /* This is a multi-line comment
        containing two lines of comment. */

        echo 'Second statement' ;

   echo 'Final statement' ; # Another single-line comment.

?>
```

Hot tip

The // single-line comment style is also used in C++ programming and the # single-line comment style is also used in Unix/Linux BASH shell scripting.

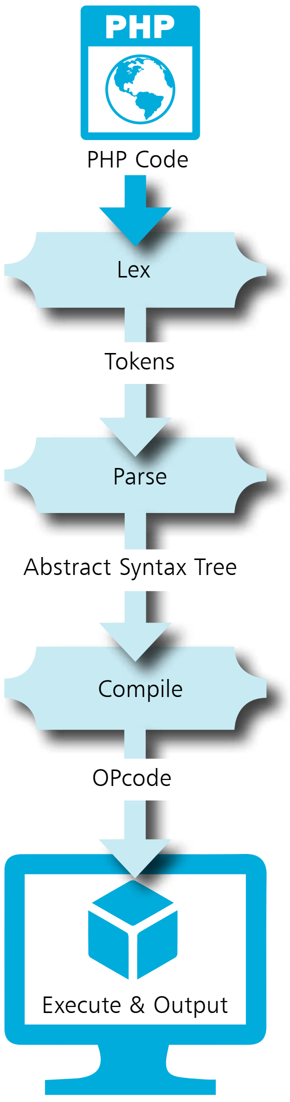
Improving performance

It is useful to understand that the PHP engine performs a sequence of operations in order to execute your PHP code:

- **Lexing** – analyzes the code and generates a series of "tokens". For example, the tag **<?php** generates the token **T_OPEN_TAG**, the keyword **echo** generates the token **T_ECHO** and the string **"Hello World!"** generates **T_CONSTANT_ENCAPSED_STRING**.
- **Parsing** – arranges the tokens into an "Abstract Syntax Tree" (AST). This represents what operations should be performed. For example, the AST to print the result of adding two numbers might look something like this:

```
    operation => ECHO,
    operand => expression (
                operation => ADD,
                operand1 => 4,
                operand2 => 8
    )
```

- **Compiling** – translates the Abstract Syntax Tree into instructional operation code ("OPcode"). This is an Intermediate Representation (IR) in machine language.
- **Executing** – runs the OPcode on the "Zend Virtual Machine" (VM). This interprets the OPcode to produce output.



Performing all these steps each time to execute a PHP script is not very efficient, so PHP includes an extension named "OPcache" that can be enabled to improve performance. OPcache stores OPcodes in memory so that lexing, parsing and compiling will only be performed when the PHP script is first executed. Subsequently, the low-level Intermediate Representation stored in memory can be executed on the Zend VM immediately.

To enable OPcache you will need to edit PHP's initialization file. The PHP installation folder, such as **C:\PHP**, contains files named **php.ini-development** and **php.ini-production**, which are initialization template files. When starting, PHP tries to find and load an initialization file named **php.ini**. A copy of either of the template files can be used for this purpose but the template named **php.ini-production** is recommended as the default settings in that file are optimized for performance and security. The OPcache extension is a file named **php_opcache.dll** located in an "ext" folder within the PHP installation folder.

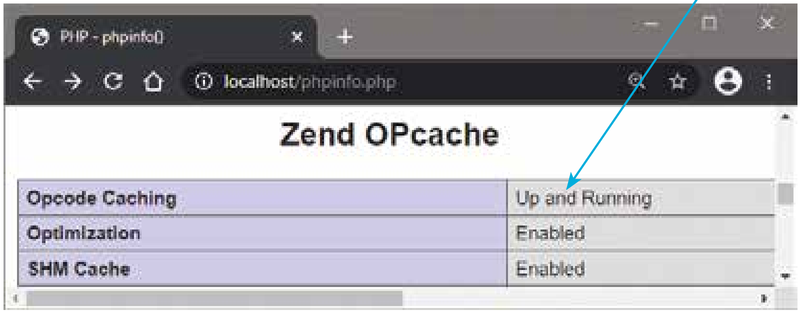
1. Navigate to your PHP installation folder then save a copy of the **php.ini-production** file precisely as "php.ini"

2. Next, open the **php.ini** file in a text editor such as Windows' Notepad app

3. Now, at the end of the **php.ini** file, add a line to specify the location of OPcache
zend_extension=C:\PHP\ext\php_opcache.dll

4. Then, add a line to enable the OPcache extension
opcache.enable=1

5. Save the file then restart the web server and examine the **phpinfo.php** output to confirm OPcache is now Enabled

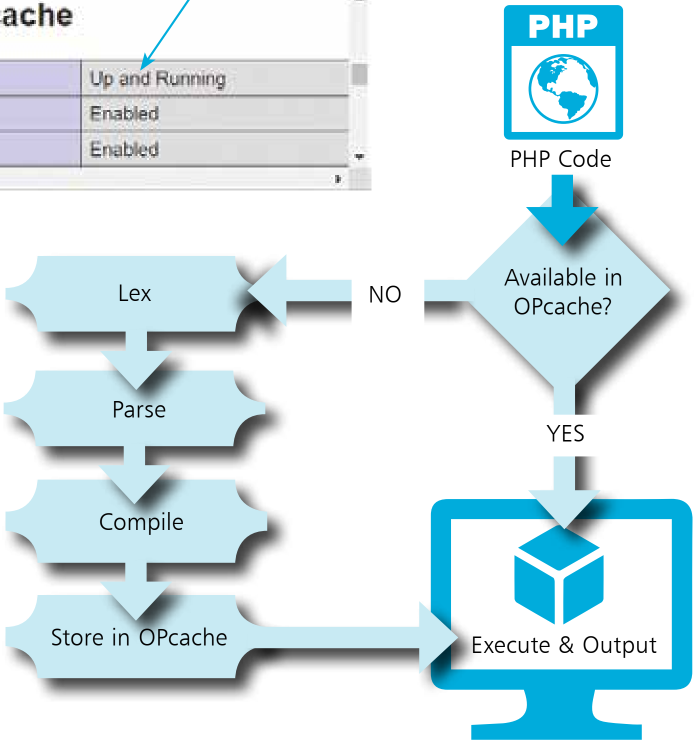


Zend OPcache

Opcode Caching	Up and Running
Optimization	Enabled
SHM Cache	Enabled

6. Scroll down the Zend OPcache section to see that "Cache hits" and "Cache misses" are both initially zero

7. Load the **hello.php** file and re-examine the **phpinfo.php** output repeatedly to see that "Cache misses" first increases to 1 then "Cache hits" increments on successive occasions



Enabling JIT compilation

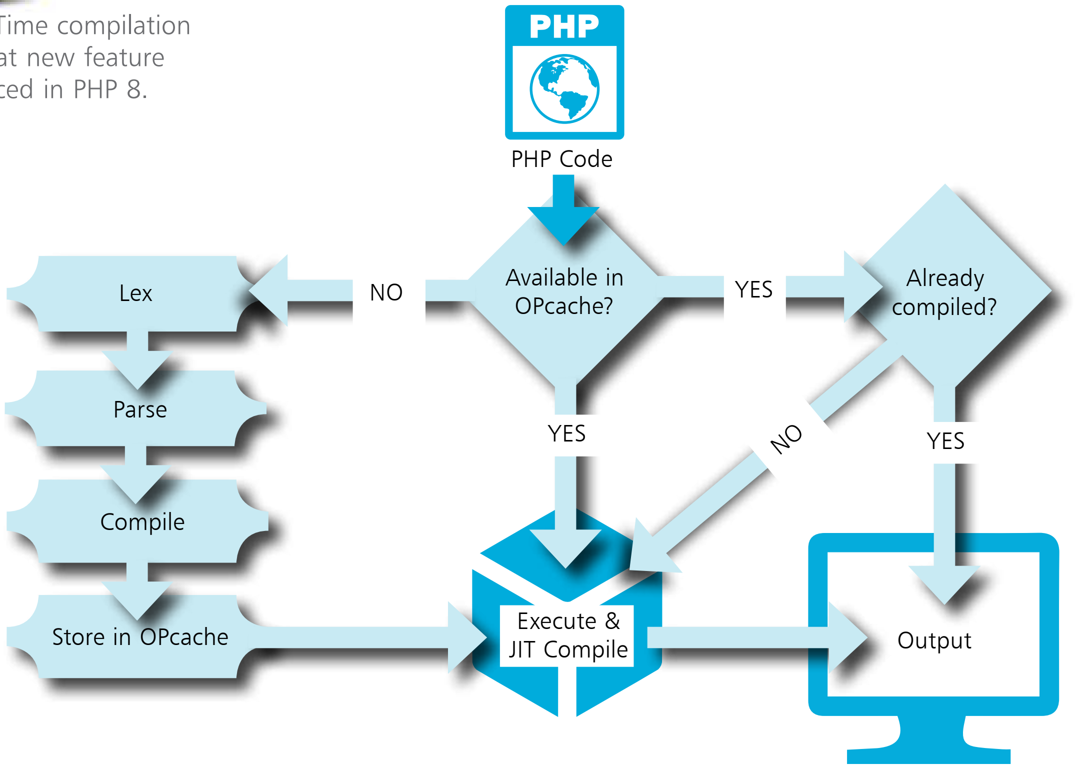
The OPcache extension, described on pages 20-21, allows PHP to avoid repeated lexing, parsing, and compiling, but a further performance improvement can be made using “Just In Time” (JIT) compilation.

PHP’s JIT compilation feature generates low-level CPU instructions in Assembly language code that can be executed by many different CPU types. It transforms the OPcode into architecture-specific machine code that can be executed directly by the system’s CPU.

When PHP code is first run, the OPcode being executed by the Zend VM is examined by PHP to decide what would benefit from JIT compilation. Subsequent execution is performed directly by the system’s CPU – bypassing the Zend VM.

Hot tip

Just In Time compilation is a great new feature introduced in PHP 8.



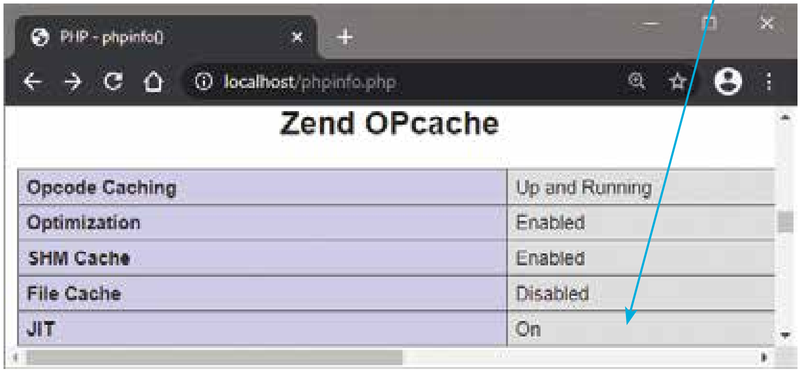
The OPcache extension must first be enabled to use JIT compilation, and you must specify in the PHP initialization file **php.ini** how much memory to allocate for compiled code.

1. Open the **php.ini** file in a text editor such as Windows' Notepad app

2. Now, at the end of the **php.ini** file, add a line to specify memory allocation of 100 megabytes for compiled code
opcache.jit_buffer_size=100M

3. Save the file then restart the web server and examine the **phpinfo.php** output to confirm JIT compilation is **On**

php.ini

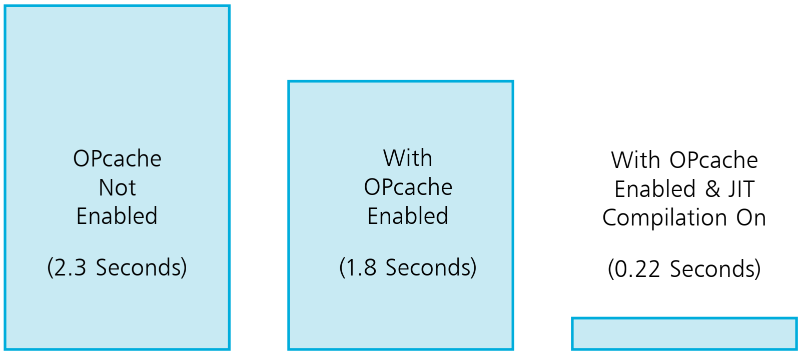


Zend OPcache

Opcode Caching	Up and Running
Optimization	Enabled
SHM Cache	Enabled
File Cache	Disabled
JIT	On

It is anticipated that performance will become more important in the future for CPU-intensive tasks such as Artificial Intelligence (AI). To compare the performance improvements provided by OPcache and JIT compilation, a benchmark PHP script might contain a variety of CPU-intensive tasks. Not listed here but included in the source code archive for this book is a PHP benchmark script that produced the results below when executed with various PHP configurations:

bench.php



Summary

- PHP is a scripting language that is especially suited for web development as it can be embedded in HTML.
- PHP code is executed server-side on The Cloud, unlike JavaScript code that is executed client-side in the browser.
- A web server will first call upon the PHP engine to process PHP code before sending the response to the web browser.
- A local development environment can be created by installing a web server and the PHP engine on your own computer.
- The **http://localhost** URL is an alias for the numerical IP address of **http://127.0.0.1**.
- Access to the Abyss Web Server Console requires your user name and password, and is found at **http://localhost:9999**.
- The Web Server must be configured to recognize scripts so it will direct them to the PHP engine, by setting Interface, Interpreter, and Associated Extension parameters.
- The PHP code instruction **phpinfo()** can be used to serve up a web page containing your PHP version information.
- The PHP **echo** instruction literally writes the text content contained within following quote marks.
- All embedded PHP code must be enclosed within **<?php** and **?>** tags so it can be recognized by the PHP engine.
- Each statement within the PHP language must be terminated by a **;** semicolon character.
- Single-line comments may begin with a **#** hash character or with a **//** double-slash sequence.
- Multi-line block comments must be enclosed within **/*** and ***/** character sequences.
- OPcache stores OPcodes in memory for execution by the Zend Virtual Machine.
- Just In Time compilation generates low-level instructions for execution directly by the system’s CPU.