PHP 7

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

PHP 7 is the most awaited and is a major feature release of PHP programming language. PHP 7 was released on 3rd Dec 2015.

This tutorial will teach you the new features of PHP 7 and their usage in a simple and intuitive way.

Audience

This tutorial has been prepared for PHP developers from a beginner’s point of view. After completing this tutorial, you will find yourself at a moderate level of expertise in the knowledge of PHP from where you can take yourself to next levels.

Prerequisites

We assume that you already know about the older versions of PHP and now you can start learning the new features of PHP 7.

Execute PHP-7 Online

For most of the examples given in this tutorial, you will find an option **Try it.** Just use this option to execute your PHP-7 programs at the spot and enjoy your learning.

Try the following example using **Try it** option available at the top right corner of the below sample code box –

```
<html>
   <head>
      <title>Online PHP-7 Script Execution</title>
   </head>
      <body>
      <?php
         echo "<h1>Hello, PHP-7!</h1>";
      ?>
   </body>
</html>
```

Copyright & Disclaimer

Table of Contents

1. PHP 7 – Introduction

What is PHP 7?

PHP 7 is a major release of PHP programming language and is touted to be a revolution in the way web applications can be developed and delivered for mobile to enterprises and the cloud. This release is considered to be the most important change for PHP after the release of PHP 5 in 2004.

New Features

There are dozens of features added to PHP 7, the most significant ones are mentioned below -

- **Improved performance** - Having PHPNG code merged in PHP7, it is twice as fast as PHP 5.
- **Lower Memory Consumption** - Optimized PHP 7 utilizes lesser resource.
- **Scalar type declarations** - Now parameter and return types can be enforced.
- **Consistent 64-bit support** - Consistent support for 64-bit architecture machines.
- **Improved Exception hierarchy** - Exception hierarchy is improved.
- **Many fatal errors converted to Exceptions** - Range of exceptions is increased covering many fatal error converted as exceptions.
- **Secure random number generator** - Addition of new secure random number generator API.
- **Deprecated SAPIs and extensions removed** - Various old and unsupported SAPIs and extensions are removed from the latest version.
- **The null coalescing operator (??) -** New null coalescing operator added.
- **Return and Scalar Type Declarations -** Support for return type and parameter type added.
- **Anonymous Classes -** Support for anonymous added.
- **Zero cost asserts -** Support for zero cost assert added.

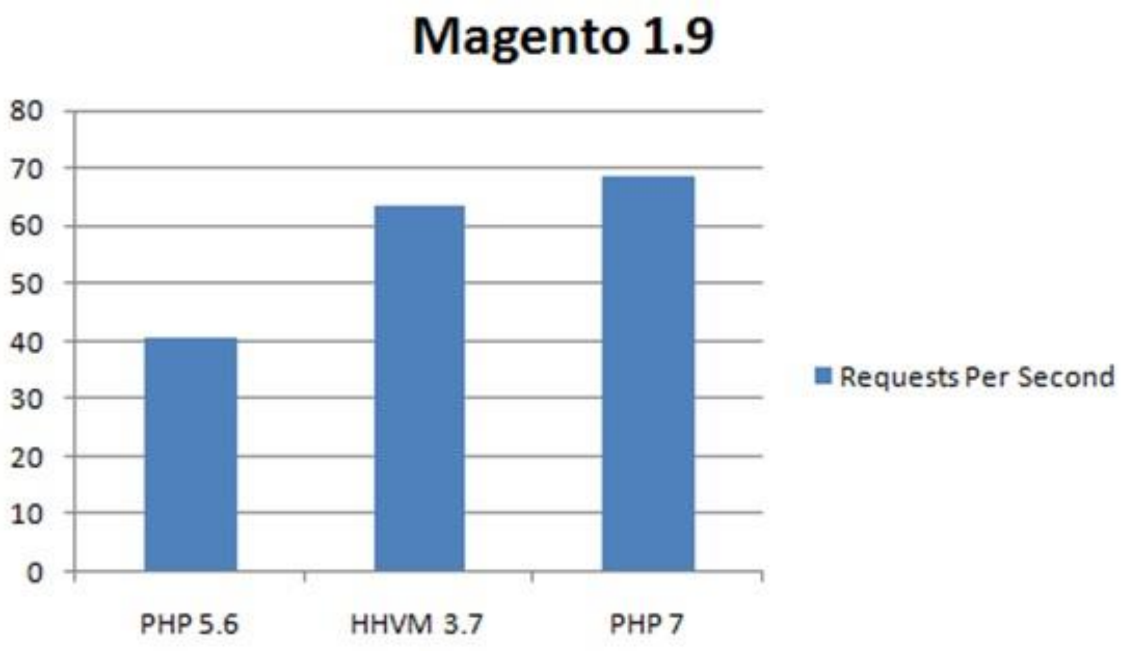
PHP 7 uses new Zend Engine 3.0 to improve application performance almost twice and 50% better memory consumption than PHP 5.6. It allows to serve more concurrent users without requiring any additional hardware. PHP 7 is designed and refactored considering today's workloads.

2. PHP 7 – Performance

As per the Zend team, following illustrations show the performance comparison of PHP 7 vs PHP 5.6 and HHVM 3.7 on popular PHP based applications.

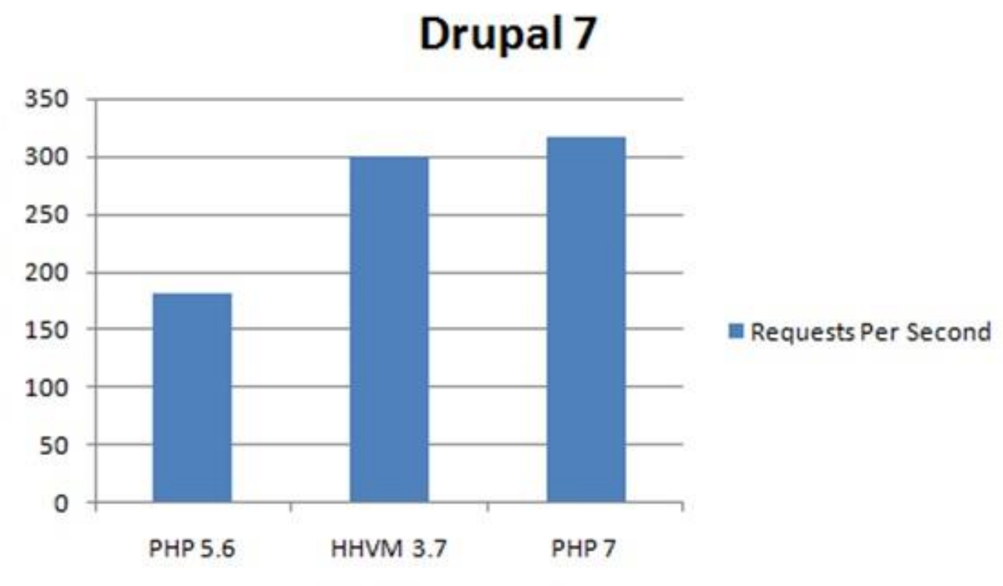
Magento 1.9

PHP 7 proves itself more than twice as faster, as compared to PHP 5.6 while executing Magento transactions.



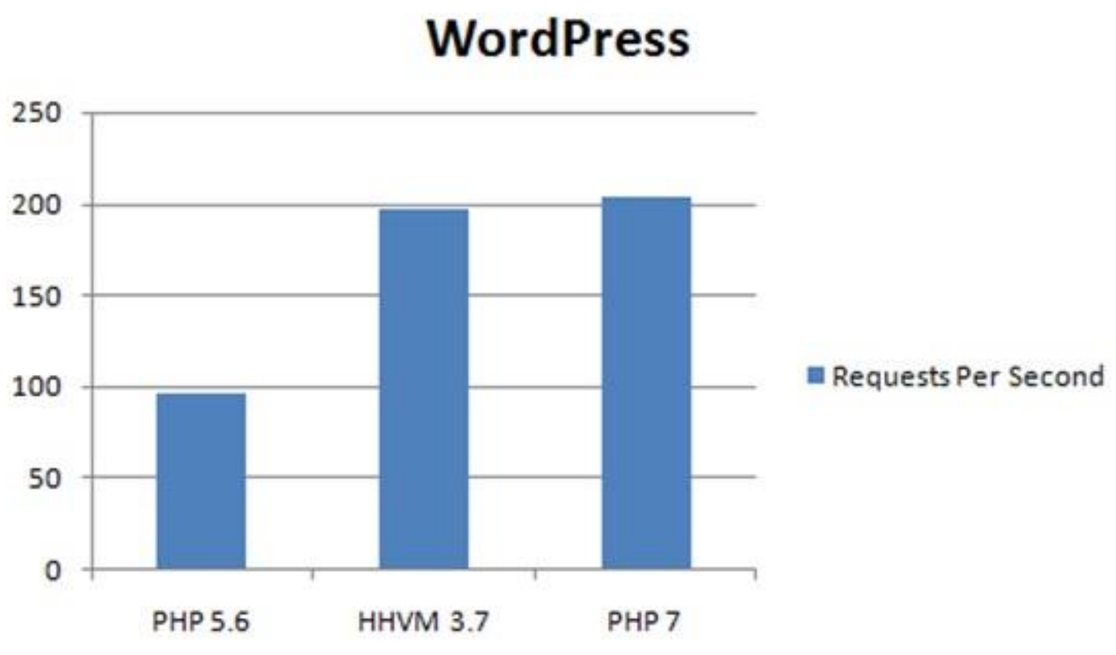
Drupal 7

PHP 7 proves itself more than twice as faster, as compared to PHP 5.6 while executing Drupal transactions.

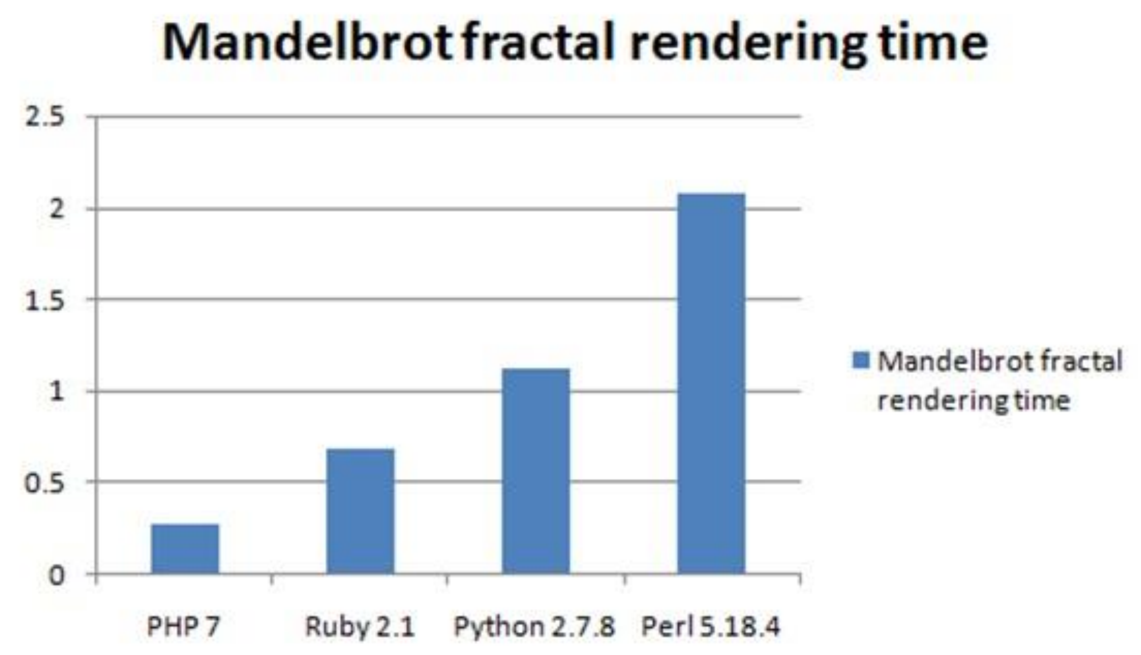


Wordpress 3.6

PHP 7 proves itself more than twice as faster as compared to PHP 5.6 while executing Wordpress transactions.



Comparison of Dynamic Languages



3. PHP 7 – Environment Setup

Try it Option Online

We have set up the PHP Programming environment on-line, so that you can compile and execute all the available examples online. It gives you confidence in what you are reading and enables you to verify the programs with different options. Feel free to modify any example and execute it online.

Try the following example using our online compiler available at CodingGround.

```
<html>
   <head>
      <title>Online PHP Script Execution</title>
   </head>
   <body>
      <?php
         echo "<h1>Hello, PHP!</h1>";
      ?>
   </body>
</html>
```

For most of the examples given in this tutorial, you will find a **Try it** option in our website code sections at the top right corner that will take you to the online compiler. So just use of and enjoy your learning.

In order to develop and run PHP Web pages, three vital components need to be installed on your computer system.

- **Web Server** – PHP works with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but most often used is Apache Server. Download Apache for free here – http://httpd.apache.org/download.cgi

- **Database** – PHP works with virtually all database software, including Oracle and Sybase but most commonly used is MySQL database. Download MySQL for free here –http://www.mysql.com/downloads/

- **PHP Parser** – In order to process PHP script instructions, a parser must be installed to generate HTML output that can be sent to the Web Browser. This tutorial will guide you how to install PHP parser on your computer.

PHP Parser Installation

Before you proceed, it is important to make sure that you have proper environment setup on your machine to develop your web programs using PHP. Store the following php file in Apache's htdocs folder.

phpinfo.php

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

Type the following address into your browser's address box.

```
http://127.0.0.1/phpinfo.php
```

If this displays a page showing your PHP installation related information, then it means you have PHP and Webserver installed properly. Otherwise, you have to follow the given procedure to install PHP on your computer.

This section will guide you to install and configure PHP over the following four platforms –

- PHP Installation on Linux or Unix with Apache
- PHP Installation on Mac OS X with Apache
- PHP Installation on Windows NT/2000/XP with IIS
- PHP Installation on Windows NT/2000/XP with Apache

Installation on Linux / Unix

If you plan to install PHP on Linux or any other variant of Unix, then here is the list of prerequisites –

- The PHP source distribution http://www.php.net/downloads.php
- The latest Apache source distribution http://httpd.apache.org/download.cgi
- A working PHP-supported database, if you plan to use one (For example MySQL, Oracle etc.)
- Any other supported software, to which PHP must connect (mail server, BCMath package, JDK, and so forth)
- An ANSI C compiler.
- Gnu make utility – you can freely download it at http://www.gnu.org/software/make

Now, here are the steps to install Apache and PHP5 on your Linux or Unix machine. If your PHP or Apache versions are different then please take care accordingly.

Step 1

If you have not already done so, unzip and untar your Apache source distribution. Unless you have a reason to do otherwise, /usr/local is the standard place.

```
gunzip -c apache_2.4.x.tar.gz

tar -xvf apache_2.4.x.tar
```

Step 2

Build the apache Server as follows -

```
cd apache_2.4.x

./configure --prefix=/usr/local/apache --enable-so

make

make install
```

Step 3

Unzip and untar your PHP source distribution. Unless you have a reason to do otherwise, /usr/local is the standard place.

```
gunzip -c php-7.x.tar.gz

tar -xvf php-7.x.tar

cd php-7.x
```

Step 4

Configure and Build your PHP, assuming you are using MySQL database.

```
./configure --with-apxs=/usr/sbin/apxs \

            --with-mysql=/usr/bin/mysql

make

make install
```

Step 5

Install the php.ini file. Edit this file to get configuration directives –

```
cd ../../php-7.x

cp php.ini-development /usr/local/lib/php.ini
```

Step 6

- Tell your Apache server where you want to serve files from, and what extension(s) you want to identify PHP files. A **.php** extension is standard, but you can use .html, .phtml, or whatever you want.

    - Go to your HTTP configuration files (/usr/local/apache/conf or whatever your path is).

    - Open httpd.conf with a text editor.

    - Search for the word DocumentRoot (which should appear twice), and change both the paths to the directory you want to serve files out of (in our case, /home/httpd). We recommend a home directory rather than the default **/usr/local/apache/htdocs** because it is more secure, but it does not have to be in a home directory. You will keep all your PHP files in this directory.

- Add at least one PHP extension directive, as shown in the first line of the code that follows. In the second line, we have also added a second handler to have all HTML files parsed as PHP.

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

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

Step 7

Restart your server. Every time you change your HTTP configuration or php.ini files, you must stop and start your server again.

```
cd ../bin

./apachectl start
```

Step 8

Set the documentroot-directory permissions to world-executable. The actual PHP files in the directory need only be world-readable (644). If necessary, replace /home/httpd with your document root below –

```
chmod 755 /home/httpd/html/php
```

Step 9

Open a text editor. Type: **<?php phpinfo(); ?>.** Save this file in your Web server's document root as **info.php**. Start any Web browser and browse the file. You must always use an HTTP request (http://www.testdomain.com/info.php or http://localhost/info.php or http://127.0.0.1/info.php) rather than a filename (/home/httpd/info.php) for the file to be parsed correctly.

You will see a long table of information about your new PHP installation message **Congratulations!**

Installation on Mac OS X

Mac users have a choice of either a binary or a source installation. In fact, your OS X probably came with Apache and PHP preinstalled. This is likely to be quite an old build, and it probably lacks many of the less common extensions.

However, if all you want is a quick Apache + PHP + MySQL/PostgreSQL setup on your laptop, this is certainly the easiest way to fly. All you need to do is edit your Apache configuration file and turn on the Web server.

Just follow the steps given below–

Step 1

Open the Apache config file in a text editor as root.

```
sudo open -a TextEdit /etc/httpd/httpd.conf
```

Step 2

Edit the file. Uncomment the following lines –

```
Load Module php7_module

AddModule mod_php7.c

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

Step 3

You may also want to uncomment the **<Directory /home/*/Sites>** block or otherwise tell Apache which directory to serve. Restart the Web server.

```
sudo apachectl graceful
```

Step 4

Open a text editor. Type: <?php phpinfo(); ?>. Save this file in your Web server's document root as info.php. Start any Web browser and browse the file. You must always use an HTTP request (http://www.testdomain.com/info.php or http://localhost/info.php or http://127.0.0.1/info.php) rather than a filename (/home/httpd/info.php) for the file to be parsed correctly.

You will see a long table of information about your new PHP installation message **Congratulations**!

Installation on Windows with IIS

The Windows server installation of PHP running IIS is much simpler than on Unix, since it involves a precompiled binary rather than a source build.

If you plan to install PHP over Windows, then here is the list of prerequisites –

- A working PHP-supported Web server. Under previous versions of PHP, IIS/PWS was the easiest choice because a module version of PHP was available for it; but PHP now has added a much wider selection of modules for Windows.
- A correctly installed PHP-supported database like MySQL or Oracle etc. (if you plan to use one).
- The PHP Windows binary distribution (download it at www.php.net/downloads.php)
- A utility to unzip files (search http://download.cnet.com for PC file compression utilities)

Now here are the steps to install Apache and PHP5 on your Windows machine. If your PHP version is different, then please take care accordingly.

1. Extract the binary archive using your unzip utility; C:\PHP is a common location.
2. Copy some .dll files from your PHP directory to your systems directory (usually C:\Winnt\System32). You need **php5ts.dll** for every case. You will also probably need to copy the file corresponding to your Web server module - C:\PHP\Sapi\php7isapi.dll. It is possible you will also need other files from the dlls subfolder, but start with the two files mentioned above and add more if you need them.
3. Copy either the php.ini-development or php.ini-recommended (preferably the latter) to your Windows directory (C:\Winnt or C:\Winnt40), and rename it php.ini. Open this file in a text editor (for example, Notepad). Edit this file to get the configuration directives. We highly recommend the new users to set error-reporting to E_ALL on their development machines at this point. For now, the most important thing is the doc_root directive under the Paths and Directories section. Make sure this matches the IIS Inetpub folder (or wherever you plan to serve out of).
4. Stop and restart the WWW service. Go to the **Start menu** → **Settings** → **Control Panel** → **Services.** Scroll down the list to IIS Admin Service. Select it and click Stop. After it stops, select the World Wide Web Publishing Service and click Start. Stopping and restarting the service from within the Internet Service Manager will not suffice. Since this is Windows, you may also wish to reboot.
5. Open a text editor. Type: <?php phpinfo(); ?>. Save this file in your Web server's document root as info.php.
6. Start any Web browser and browse the file. You must always use an HTTP request (http://www.testdomain.com/info.php or http://localhost/info.php or http://127.0.0.1/info.php) rather than a filename (/home/httpd/info.php) for the file to be parsed correctly.

You will see a long table of information about your new PHP installation message **Congratulations**!

Installation on Windows with Apache

To install Apache with PHP 5 on Windows, follow the given steps. If your PHP and Apache versions are different then please take care accordingly.

Step 1

- Download Apache server from www.apache.org/dist/httpd/binaries/win32. You want the current stable release version with the no_src.msi extension. Double-click the installer file to install; C:\Program Files is a common location. The installer will also ask you whether you want to run Apache as a service or from the command line or DOS prompt. We recommend you do not install as a service, as this may cause problems with startup.

- Extract the PHP binary archive using your unzip utility; C:\php7 is a common location.

- Rename php.ini-development to php.ini. Open this file in a text editor (for example, Notepad). Edit this file to get the configuration directives. At this point, we highly recommend that the new users set error reporting to E_ALL on their development machines.

- Tell your Apache server where you want to serve files from and what extension(s) you want to identify the PHP files (.php is the standard, but you can use .html, .phtml, or whatever you want). Go to your HTTP configuration files (C:\Program Files\Apache Group\Apache\conf or whatever your path is), and open httpd.conf with a text editor. Search for the word DocumentRoot (which should appear twice) and change both the paths to the directory you want to serve files out of. (The default is C:\Program Files\Apache Group\Apache\htdocs.). Add at least one PHP extension directive as shown in the first line of the following code –

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

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

LoadModule php7_module "C:/php7/php7apache2_4.dll"

PHPiniDir "c:/php7"
```

Step 2

Open a text editor. Type: <?php phpinfo(); ?>. Save this file in your Web server's document root as info.php. Start any Web browser and browse the file. You must always use an HTTP request (http://www.testdomain.com/info.php or http://localhost/info.php or http://127.0.0.1/info.php) rather than a filename (/home/httpd/info.php) for the file to be parsed correctly.

You will see a long table of information about your new PHP installation message **Congratulations**!

Apache Configuration

If you are using Apache as a Web Server, then this section will guide you to edit Apache Configuration Files.

Check here – PHP Configuration in Apache Server

PHP Configuration in Apache

Apache uses httpd.conf file for global settings, and the .htaccess file for per-directory access settings. Older versions of Apache split up httpd.conf into three files (access.conf, httpd.conf, and srm.conf), and some users still prefer this arrangement.

Apache server has a very powerful, but slightly complex, configuration system of its own. Learn more about it at the Apache Web site – www.apache.org

The following section describes the settings in httpd.conf that affect PHP directly and cannot be set elsewhere. If you have standard installation then httpd.conf will be found at /etc/httpd/conf:

Timeout

This value sets the default number of seconds before any HTTP request will time out. If you set PHP's max_execution_time to longer than this value, PHP will keep grinding away but the user may see a 404 error. In safe mode, this value will be ignored; instead, you must use the timeout value in php.ini.

DocumentRoot

DocumentRoot designates the root directory for all HTTP processes on that server. It looks something like this on Unix –

```
DocumentRoot ./usr/local/apache_2.4.0/htdocs.
```

You can choose any directory as the document root.

AddType

The PHP MIME type needs to be set here for PHP files to be parsed. Remember that you can associate any file extension with PHP like .php3, .php5 or .htm.

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

AddType application/x-httpd-phps .phps

AddType application/x-httpd-php3 .php3 .phtml

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

Action

You must uncomment this line for the Windows apxs module version of Apache with shared object support –

```
LoadModule php7_module modules/php7apache2_4.dll
```

on Unix flavors –

```
LoadModule php7_module modules/mod_php.so
```

AddModule

You must uncomment this line for the static module version of Apache.

```
AddModule mod_php7.c
```

PHP.INI File Configuration

The PHP configuration file, **php.ini**, is the final and immediate way to affect PHP's functionality.

Check here – PHP.INI File Configuration

PHP.INI Configuration

The PHP configuration file, php.ini, is the final and immediate way to affect PHP's functionality. The php.ini file is read each time PHP is initialized. In other words, httpd is restarted for the module version or with each script execution for the CGI version. If your change is not showing up, remember to stop and restart httpd. If it is still not showing up, use phpinfo() to check the path to php.ini.

The configuration file is well commented and thorough. Keys are case sensitive, keyword values are not; whitespace, and lines beginning with semicolons are ignored. Booleans can be represented by 1/0, Yes/No, On/Off, or True/False. The default values in php.ini-dist will result in a reasonable PHP installation that can be tweaked later.

Here we are explaining the important settings in php.ini, which you may need for your PHP Parser.

short_open_tag = Off

Short open tags look like this: <? ?>. This option must be set to **Off**, if you want to use the XML functions.

safe_mode = Off

If this is set to ON, you probably compiled PHP with the --enable-safe-mode flag. The Safe mode is most relevant to CGI use. See the explanation in the section "CGI compile-time options" given earlier in this chapter.

safe_mode_exec_dir = [DIR]

This option is relevant only if the safe mode is ON; it can also be set with the --with-exec-dir flag during the Unix build process. PHP in safe mode only executes external binaries out of this directory. The default is /usr/local/bin. This has nothing to do with serving up a normal PHP/HTML Web page.

safe_mode_allowed_env_vars = [PHP_]

This option sets which environment variables the users can change in safe mode. The default is only those variables prepended with "PHP_". If this directive is empty, most variables are alterable.

safe_mode_protected_env_vars = [LD_LIBRARY_PATH]

This option sets which environment variables the users cannot change in safe mode, even if safe_mode_allowed_env_vars is set permissively.

disable_functions = [function1, function2...]

A welcome addition to the PHP4 configuration and one perpetuated in PHP5 is the ability to disable the selected functions for security reasons. Previously, this necessitated hand-editing the C code from which PHP was made. Filesystem, system, and network functions should probably be the first to go because allowing the capability to write files and alter the system over HTTP is never such a safe idea.

max_execution_time = 30

The function set_time_limit() will not work in safe mode. Therefore, this is the main way to make a script time-out in safe mode. In Windows, you have to abort based on maximum memory consumed rather than the time. You can also use the Apache timeout setting to timeout but that will apply to non-PHP files on the site too.

error_reporting = E_ALL & ~E_NOTICE

The default value is E_ALL & ~E_NOTICE, all errors except notices. The development servers should be set to at least the default; only the production servers should consider a lesser value.

error_prepend_string = [""]

With its bookend, error_append_string, this setting allows you to make error messages a different color than the other text.

warn_plus_overloading = Off

This setting issues a warning if the + operator is used with strings, as in a form value.

variables_order = EGPCS

This configuration setting supersedes gpc_order. Both are now deprecated along with register_globals. It sets the order of the different variables: Environment, GET, POST, COOKIE, and SERVER (aka Built-in). You can change this order around. The variables will be overwritten successively in the left-to-right order, with the rightmost one winning the hand every time. This means, if you left the default setting and happened to use the same name for an environment variable, a POST variable, and a COOKIE variable, the COOKIE variable would own that name at the end of the process. In real life, this does not happen much.

register_globals = Off

This setting allows you to decide whether you wish to register the EGPCS variables as global. This is now deprecated, and as of PHP4.2, this flag is set to **Off,** by default. Use the superglobal arrays instead. All the major code listings in this book use superglobal arrays.

gpc_order = GPC

This setting has been Deprecated.

magic_quotes_gpc = On

This setting escapes the quotes in incoming GET/POST/COOKIE data. If you use a lot of forms which possibly submit to themselves or other forms and display form values, you may need to set this directive to **On** or prepare to use addslashes() on string-type data.

magic_quotes_runtime = Off

This setting escapes the quotes in incoming database and text strings. Remember that SQL adds slashes to single quotes and apostrophes when storing the strings and does not strip them off when returning them. If this setting is Off, you will need to use stripslashes() when outputting any type of string data from an SQL database. If magic_quotes_sybase is set to On, this must be Off.

magic_quotes_sybase = Off

This setting escapes single quotes in incoming database and text strings with Sybase-style single quotes rather than backslashes. If magic_quotes_runtime is set to On, this must be Off.

auto-prepend-file = [path/to/file]

If a path is specified here, PHP must automatically include() it at the beginning of every PHP file. Include path-restrictions do apply.

auto-append-file = [path/to/file]

If a path is specified here, PHP must automatically include() at the end of every PHP file, unless you escape by using the exit() function. Include path-restrictions do apply.

include_path = [DIR]

If you set this value, you will only be allowed to include or require files from these directories. The include directory is generally under your document root. This is mandatory if you are running in safe mode. Set this to **.in**, in order to include the files from the same directory your script is in. Multiple directories are separated by colons: .:/usr/local/apache/htdocs:/usr/local/lib.

doc_root = [DIR]

If you are using Apache, you have already set a document root for this server or virtual host in httpd.conf. Set this value here if you are using safe mode or if you want to enable PHP only on a portion of your site (for example, only in one subdirectory of your Web root).

file_uploads = [on/off]

Turn on this flag if you will upload files using PHP script.

upload_tmp_dir = [DIR]

Do not uncomment this line unless you understand the implications of HTTP uploads!

session.save-handler = files

Except in rare circumstances, you will not want to change this setting. So do not touch it.

ignore_user_abort = [On/Off]

This setting controls what happens if a site visitor clicks the browser's Stop button. The default is On, which means that the script continues to run till completion or timeout. If the setting is changed to Off, the script will abort. This setting only works in module mode, not CGI.

mysql.default_host = hostname

The default server host to use when connecting to the database server if no other host is specified.

mysql.default_user = username

The default user name to use when connecting to the database server if no other name is specified.

mysql.default_password = password

The default password to use when connecting to the database server if no other password is specified.

Windows IIS Configuration

To configure IIS on your Windows machine, you can refer your IIS Reference Manual shipped along with IIS.

4. PHP 7 – Scalar Type Declarations

In PHP 7, a new feature, Scalar type declarations, has been introduced. Scalar type declaration has two options:

- **coercive -** coercive is default mode and need not to be specified.
- **strict -** strict mode has to explicitly hinted.

Following types for function parameters can be enforced using the above modes:

- int
- float
- bool
- string
- interfaces
- array
- callable

Example – Coercive Mode

```
<?php
// Coercive mode
function sum(int ...$ints)
{
   return array_sum($ints);
}
print(sum(2, '3', 4.1));
?>
```

It produces the following browser output –

```
9
```

Example - Strict Mode

```
<?php
// Strict mode
declare(strict_types=1);
function sum(int ...$ints)
{
   return array_sum($ints);
}
print(sum(2, '3', 4.1));
?>
```

It produces the following browser output –

```
Fatal error: Uncaught TypeError: Argument 2 passed to sum() must be of the type
integer, string given, ...
```

5. PHP 7 – Return Type Declarations

In PHP 7, a new feature, **Return type declarations** has been introduced. Return type declaration specifies the type of value that a function should return. Following types for return types can be declared.

- int
- float
- bool
- string
- interfaces
- array
- callable

Example - Valid Return Type

```
<?php
declare(strict_types=1);
function returnIntValue(int $value): int
{
   return $value;
}
print(returnIntValue(5));
?>
```

It produces the following browser output –

```
5
```

Example - Invalid Return Type

```
<?php
declare(strict_types=1);
function returnIntValue(int $value): int
{
   return $value + 1.0;
}
print(returnIntValue(5));
?>
```

It produces the following browser output –

```
Fatal error: Uncaught TypeError: Return value of returnIntValue() must be of the
type integer, float returned...
```

6. PHP 7 – Null Coalescing Operator

In PHP 7, a new feature, **null coalescing operator (??)** has been introduced. It is used to replace the **ternary** operation in conjunction with isset() function. The **Null** coalescing operator returns its first operand if it exists and is not NULL; otherwise it returns its second operand.

Example

```
<?php
// fetch the value of $_GET['user'] and returns 'not passed'
// if username is not passed
$username = $_GET['username'] ?? 'not passed';
print($username);
print("<br/>");

// Equivalent code using ternary operator
$username = isset($_GET['username']) ? $_GET['username'] : 'not passed';
print($username);
print("<br/>");
// Chaining ?? operation
$username = $_GET['username'] ?? $_POST['username'] ?? 'not passed';
print($username);
?>
```

It produces the following browser output –

```
not passed
not passed
not passed
```

7. PHP 7 – Spaceship Operator

In PHP 7, a new feature, spaceship operator has been introduced. It is used to compare two expressions. It returns -1, 0 or 1 when first expression is respectively less than, equal to, or greater than second expression.

Example

```
<?php
//integer comparison
print( 1 <=> 1);print("<br/>");
print( 1 <=> 2);print("<br/>");
print( 2 <=> 1);print("<br/>");
print("<br/>");
//float comparison
print( 1.5 <=> 1.5);print("<br/>");
print( 1.5 <=> 2.5);print("<br/>");
print( 2.5 <=> 1.5);print("<br/>");
print("<br/>");
//string comparison
print( "a" <=> "a");print("<br/>");
print( "a" <=> "b");print("<br/>");
print( "b" <=> "a");print("<br/>");
?>
```

It produces the following browser output –

```
0
-1
1

0
-1
1
```

```
0
-1
1
```

8. PHP 7 – Constant Arrays

Array constants can now be defined using the **define()** function. In PHP 5.6, they could only be defined using **const** keyword.

Example

```
<?php
//define a array using define function
define('animals', [
   'dog',
   'cat',
   'bird'
]);
print(animals[1]);
?>
```

It produces the following browser output –

```
cat
```

9. PHP 7 – Anonymous Classes

Anonymous classes can now be defined using new class. Anonymous class can be used in place of a full class definition.

Example

```
<?php
interface Logger {
   public function log(string $msg);
}

class Application {
   private $logger;

   public function getLogger(): Logger {
      return $this->logger;
   }

   public function setLogger(Logger $logger) {
      $this->logger = $logger;
   }
}

$app = new Application;
$app->setLogger(new class implements Logger {
   public function log(string $msg) {
      print($msg);
   }
});

$app->getLogger()->log("My first Log Message");
?>
```

It produces the following browser output –

```
My first Log Message
```

10. PHP 7 – Closure::call()

Closure::call() method is added as a shorthand way to temporarily bind an object scope to a closure and invoke it. It is much faster in performance as compared to **bindTo** of PHP 5.6.

Example – Pre PHP 7

```
<?php
class A {
   private $x = 1;
}

// Define a closure Pre PHP 7 code
$getValue = function() {
   return $this->x;
};

// Bind a clousure
$value = $getValue->bindTo(new A, 'A');

print($value());
?>
```

It produces the following browser output –

```
1
```

Example – PHP 7+

```
<?php
class A {
   private $x = 1;
}

// PHP 7+ code, Define
$value = function() {
   return $this->x;
};


print($value->call(new A));
?>
```

It produces the following browser output –

```
1
```

11. PHP 7 – Filtered unserialize()

PHP 7 introduces Filtered **unserialize()** function to provide better security when unserializing objects on untrusted data. It prevents possible code injections and enables the developer to whitelist classes that can be unserialized.

Example

```
<?php
class MyClass1 {
   public $obj1prop;
}
class MyClass2 {
   public $obj2prop;
}

$obj1 = new MyClass1();
$obj1->obj1prop = 1;
$obj2 = new MyClass2();
$obj2->obj2prop = 2;

$serializedObj1 = serialize($obj1);
$serializedObj2 = serialize($obj2);

// default behaviour that accepts all classes
// second argument can be ommited.
// if allowed_classes is passed as false, unserialize converts all objects into
__PHP_Incomplete_Class object
$data = unserialize($serializedObj1 , ["allowed_classes" => true]);

// converts all objects into __PHP_Incomplete_Class object except those of MyClass1 and
MyClass2
$data2 = unserialize($serializedObj2 , ["allowed_classes" => ["MyClass1", "MyClass2"]]);

print($data->obj1prop);
print("<br/>");
print($data2->obj2prop);
?>
```

It produces the following browser output –

```
1
2
```

12. PHP 7 – IntlChar

In PHP7, a new **IntlChar** class is added, which seeks to expose additional ICU functionality. This class defines a number of static methods and constants, which can be used to manipulate unicode characters. You need to have **Intl** extension installed prior to using this class.

Example

```
<?php
printf('%x', IntlChar::CODEPOINT_MAX);
print (IntlChar::charName('@'));
print(IntlChar::ispunct('!'));
?>
```

It produces the following browser output –

```
10ffff
COMMERCIAL AT
true
```

13. PHP 7 – CSPRNG

In PHP 7, following two new functions are introduced to generate cryptographically secure integers and strings in a cross platform way.

- **random_bytes()** - Generates cryptographically secure pseudo-random bytes.
- **random_int()** - Generates cryptographically secure pseudo-random integers.

random_bytes()

random_bytes() generates an arbitrary-length string of cryptographic random bytes that are suitable for cryptographic use, such as when generating salts, keys or initialization vectors.

Syntax

```
string random_bytes ( int $length )
```

Parameters

- **length** - The length of the random string that should be returned in bytes.

Return Values

- Returns a string containing the requested number of cryptographically secure random bytes.

Errors/Exceptions

- If an appropriate source of randomness cannot be found, an Exception will be thrown.
- If invalid parameters are given, a **TypeError** will be thrown.
- If an invalid length of bytes is given, an Error will be thrown.

Example

```
<?php
$bytes = random_bytes(5);
print(bin2hex($bytes));
?>
```

It produces the following browser output –

```
54cc305593
```

random_int()

random_int() generates cryptographic random integers that are suitable for use where unbiased results are critical.

Syntax

```
int random_int ( int $min , int $max )
```

Parameters

- **min** - The lowest value to be returned, which must be **PHP_INT_MIN** or higher.
- **max** - The highest value to be returned, which must be less than or equal to **PHP_INT_MAX.**

Return Values

- Returns a cryptographically secure random integer in the range min to max, inclusive.

Errors/Exceptions

- If an appropriate source of randomness cannot be found, an **Exception** will be thrown.
- If invalid parameters are given, a **TypeError** will be thrown.
- If max is less than min, an **Error** will be thrown.

Example

```
<?php
print(random_int(100, 999));
print("
");
print(random_int(-1000, 0));
?>
```

It produces the following browser output –

```
614
-882
```

14. PHP 7 – Expectations

Expectations are a backwards compatible enhancement to the older assert() function. Expectation allows for zero-cost assertions in production code, and provides the ability to throw custom exceptions when the assertion fails. assert() is now a language construct, where the first parameter is an expression as compared to being a string or Boolean to be tested.

Configuration directives for assert()

Directive	Default value	Possible values
zend.assertions	1	**1 -** generate and execute code (development mode) **0 -** generate code but jump around it at runtime **-1** - do not generate code (production mode)
assert.exception	0	**1 –** throw, when the assertion fails, either by throwing the object provided as the exception or by throwing a new **AssertionError** object if exception was not provided. **0 -** use or generate a Throwable as described above, but only generates a warning based on that object rather than throwing it (compatible with PHP 5 behaviour)

Parameters

- **assertion** - The assertion. In PHP 5, this must be either a string to be evaluated or a Boolean to be tested. In PHP 7, this may also be any expression that returns a value, which will be executed and the result is used to indicate whether the assertion succeeded or failed.

- **description** - An optional description that will be included in the failure message, if the assertion fails.

- **exception** - In PHP 7, the second parameter can be a **Throwable** object instead of a descriptive string, in which case this is the object that will be thrown, if the assertion fails and the **assert.exception** configuration directive is enabled.

Return Values

FALSE if the assertion is false, **TRUE** otherwise.

Example

```
<?php
ini_set('assert.exception', 1);

class CustomError extends AssertionError {}

assert(false, new CustomError('Custom Error Message!'));
?>
```

It produces the following browser output –

```
Fatal error: Uncaught CustomError: Custom Error Message! in...
```

15. PHP 7 – use Statement

From PHP7 onwards, a single use statement can be used to import Classes, functions and constants from same namespace instead of multiple use statements.

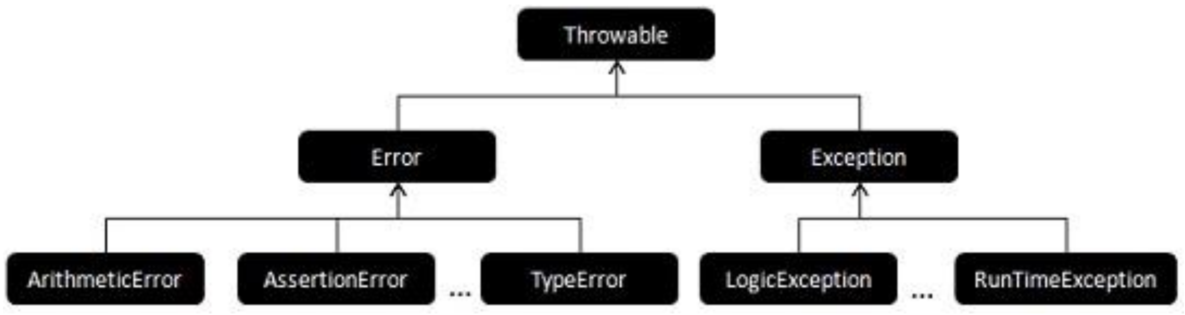
Example

```
<?php
// Before PHP 7
use com\tutorialspoint\ClassA;
use com\tutorialspoint\ClassB;
use com\tutorialspoint\ClassC as C;

use function com\tutorialspoint\fn_a;
use function com\tutorialspoint\fn_b;
use function com\tutorialspoint\fn_c;

use const com\tutorialspoint\ConstA;
use const com\tutorialspoint\ConstB;
use const com\tutorialspoint\ConstC;

// PHP 7+ code
use com\tutorialspoint\{ClassA, ClassB, ClassC as C};
use function com\tutorialspoint\{fn_a, fn_b, fn_c};
use const com\tutorialspoint\{ConstA, ConstB, ConstC};
?>
```

16. PHP 7 – Error Handling

From PHP 7, error handling and reporting has been changed. Instead of reporting errors through the traditional error reporting mechanism used by PHP 5, now most errors are handled by throwing Error exceptions. Similar to exceptions, these Error exceptions bubble up until they reach the first matching catch block. If there are no matching blocks, then a default exception handler installed with **set_exception_handler()** will be called. In case there is no default exception handler, then the exception will be converted to a fatal error and will be handled like a traditional error.

As the Error hierarchy is not extended from Exception, code that uses catch (Exception $e) { ... } blocks to handle uncaught exceptions in PHP 5 will not handle such errors. A catch (Error $e) { ... } block or a **set_exception_handler()** handler is required to handle fatal errors.



Example

```
<?php
class MathOperations
{
   protected $n = 10;
   // Try to get the Division by Zero error object and display as Exception
   public function doOperation(): string
   {
      try {
         $value = $this->n % 0;
         return $value;
      } catch (DivisionByZeroError $e) {
         return $e->getMessage();
      }
   }
}
$mathOperationsbj = new MathOperations();
print($mathOperationsObj->doOperation());
?>
```

It produces the following browser output –

```
Modulo by zero
```

17. PHP 7 – Integer Division

PHP 7 introduces a new function **intdiv(),** which performs integer division of its operands and return the division as int.

Example

```
<?php
$value = intdiv(10,3);
var_dump($value);
print("
");
print($value);
?>
```

It produces the following browser output –

```
int(3)
3
```

18. PHP 7 – Session Options

From PHP7+, **session_start()** function accepts an array of options to override the session configuration directives set in **php.ini**. These options supports **session.lazy_write**, which is by default on and causes PHP to overwrite any session file if the session data has changed.

Another option added is **read_and_close**, which indicates that the session data should be read and then the session should immediately be closed unchanged. For example, Set **session.cache_limiter** to private and set the flag to close the session immediately after reading it, using the following code snippet.

```
<?php
session_start([
   'cache_limiter' => 'private',
   'read_and_close' => true,
]);
?>
```

19. PHP 7 – Deprecated Features

Following features are deprecated and may be removed from future releases of PHP.

PHP 4 Style Constructors

PHP 4 style Constructors are methods having same name as the class they are defined in, are now deprecated, and will be removed in the future. PHP 7 will emit E_DEPRECATED if a PHP 4 constructor is the only constructor defined within a class. Classes implementing a __construct() method are unaffected.

Example

```
<?php
class A {
   function A() {
      print('Style Constructor');
   }
}
?>
```

It produces the following browser output –

```
Deprecated: Methods with the same name as their class will not be constructors
in a future version of PHP; A has a deprecated constructor in...
```

Static Calls to Non-Static Methods

Static calls to non-static methods are deprecated, and may be removed in the future.

Example

```
<?php
class A {
   function b() {
      print('Non-static call');
   }
}
A::b();
?>
```

It produces the following browser output –

```
Deprecated: Non-static method A::b() should not be called statically in...

Non-static call
```

password_hash() salt option

The salt option for the **password_hash()** function has been deprecated so that the developers do not generate their own (usually insecure) salts. The function itself generates a cryptographically secure salt, when no salt is provided by the developer - thus custom salt generation is not required any more.

capture_session_meta SSL context option

The **capture_session_meta** SSL context option has been deprecated. SSL metadata is now used through the **stream_get_meta_data() f**unction.

20. PHP 7 – Removed Extensions & SAPIs

Following Extensions have been removed from PHP 7 onwards-

- ereg
- mssql
- mysql
- sybase_ct

Following SAPIs have been removed from PHP 7 onwards-

- aolserver
- apache
- apache_hooks
- apache2filter
- caudium
- continuity
- isapi
- milter
- nsapi
- phttpd
- pi3web
- roxen
- thttpd
- tux
- webjames