NEWS & EDUCATION BLOG

20+ PHP BEST PRACTICES THAT YOU MUST FOLLOW

Posted on August 13, 2013 by Aritra Roy

This article attempts to teach you with some of the best practices of one of the most widely used programming languages in the world, PHP. There are many beginners or even experienced PHP developers who don't bother to follow the best practices of the language, either unintentionally or intentionally.

Quite often, many people loose many high-paying jobs just because they have never put their attention to the best practices of the language. And it's really very difficult for anybody to get hold of the best practices overnight.

So, if you want to be a professional developer then you must know, understand and follow the best practices of the language and should start practicing them from the very next piece of code you write.

(Image Source: prestashop.com)

1) Maintaining a Proper Documentation of your Code

Maintaining a proper documentation of your code is more important than you had ever thought of. It's really very unfortunate seeing some beginners or even seasoned programmers not focussing on writing meaningful comments to their codes just because they are too lazy.

Not only does it help others to understand your code better, but also it helps you to remember what you have written when you review it in the future. Suppose today, you write a critical function and you remember how it works for a day or two or even for a week. But there is an ample scope for you to forget what it does, a month or two later when you review it.

So, it is always a good practice to write proper comments to make your code understandable to everybody (even yourself).

2) Maintaining a Proper Coding Standard

It is very essential to maintain a proper coding standard, as it can become a painful problem if different programmers (working on a same project) starts using different coding standards.

The source code can become completely unmanageable, if everybody working on them starts inventing their own coding standards. There are some programmers who don't even bother to maintain any coding standards at all and their code starts looking like a huge pile of garbage and nothing more than that.

If you maintain a proper coding standard, then it would be easier for others to debug your code and your joint projects will tend to be much more productive.

3) Never Use Short Tags

There are many programmers, I don't know why, who tries to take a shortcut way of declaring PHP. Using "<?" or "<%" is never a good practice and it doesn't make your code look more professional than others.

(Image Source: tomjepson.co.uk)

I really don't understand why people often tend to skip a few characters in declaring PHP. What time does that save for them? Just give me one simple reason why you would use short tags instead of writing the full and official "<?php" tag.

This simple malpractice can cause conflicts with XML parsers and can also make your code incompatible with future versions of PHP.

4) Use Meaningful Variable and Function Names in your Code

It can surely become a painful experience for other programmers to understand your code if you don't stick with a proper naming standard and use generic and un-meaningful names everywhere. It's not a good practice at all, and you must not follow it in any way.

Always try to use meaningful and grammatically sensible names for your variables and functions and try to make the good habit of separating every word with underscores. And also try to be consistent with the standard you are following so that other people can get to understand your convention quickly and easily.

5) Indentation, White Spacing and Line Length

I have already mentioned the importance of maintaining a proper coding standard, but the things that you need to specifically consider are indentation, white spaces and line lengths.

Try to keep an indent of 4 spaces. Never use Tab as different computers can have different Tab settings. Also try to keep the line length less than 80 characters to ensure that your code is easily readable to you and other developers.

The main idea here is to make your code look clean and easily readable and debuggable by you and other fellow programmers.

6) Single Quoted vs. Doubles Quoted Strings

You have to understand the difference between the single quoted strings and the double quoted ones. If you just have a simple string to display, then always go for single quotes.

But if you are going to have variables and special characters like "n", "t" in your string, then you must use double quotes which will get your string parsed by the PHP interpreter and will take more execution time than single quoted strings.

So, try to understand the difference in their working nature and use them appropriately, whenever necessary.

7) Never Use Functions inside Loops

I have seen many programmers who tend to make the mistake of using functions inside of loops. If you are doing this intentionally and is ready to compromise performance just for the sake of saving a line of code, then you certainly need to think once again.

Bad Practice:

```
for ($i  = 0, $i <= count($array);  $i++)

{

//statements

}
```

Good Practice:

```
$count = count($array);

for ($i = 0; $i < $count;  $i++)

{

//statements

}
```

If you take the burden of storing the value returned by the function in a separate variable before the loop, then you would be saving enough execution time as in the first case the function will be called and executed every time the loop runs which can increase the time complexity of the program for large loops.

8) Using Single Quotes around Array Indexes

There is a difference between $array[‘quotes’] and $array[quotes] and you have to understand this difference properly.

A feature of PHP is that, it considers unquoted strings used as array indexes as constants, and if the constant have not been defined before, then it will be “self-defined” and a warning will be raised. This will not stop your code from working, but the bug will remain in the code.

9) Understanding Strings in a Better Way

Take the code snippet as an example and try to guess which statement will run the fastest and which one the slowest.

Code Snippet:

```
$a = ‘PHP’;

print “This is my first $a program.”;

echo “This is my first $a program.”;
```

echo "This is my first ".$a." program.";

echo "This is my first ",$a," program.";

Guess what, the last and the most uncommon statement of all wins the speed test. The first one obviously loses as the "print" statement is slower than "echo" statement. The third statement wins over the second one as it uses concatenation operation rather than using the variables inline.

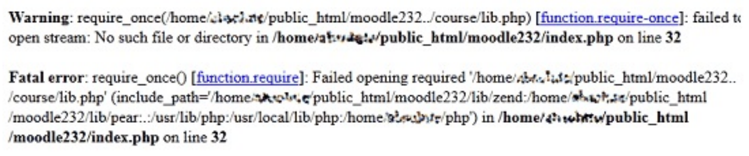
The lesser-known last statement wins as there are no string operations being performed and is nothing other than a list of comma-separated strings.

10) Turning on Error Reporting for Development Purposes

PHP has got a very useful function, error_reporting() which can be used to spot various problems or errors in your PHP application during development.

Quite often there are some errors, which will certainly not stop your entire application from working, but they are potential bugs which will reside in your code and as a professional developer you will always want to write codes which are error free in every way.

There are various levels of error reporting available in PHP, like E_NOTICE, E_WARNING, E_PARSE, etc. but you can use E_ALL to report all kinds of errors. Remember to disable error reporting when you are done with the development process of your code, so that your visitors don't get scared with various unintelligible messages.



```
Warning: require_once(/home/[redacted]/public_html/moodle232../course/lib.php) [function.require-once]: failed t
open stream: No such file or directory in /home/[redacted]/public_html/moodle232/index.php on line 32

Fatal error: require_once() [function.require]: Failed opening required '/home/[redacted]/public_html/moodle232..
/course/lib.php' (include_path='/home/[redacted]/public_html/moodle232/lib/zend:/home/[redacted]/public_html
/moodle232/lib/pear:.:/usr/lib/php:/usr/local/lib/php:/home/[redacted]/php') in /home/[redacted]/public_html
/moodle232/index.php on line 32
```

(Image Source: inmotionhosting.com)

11) Using the DRY Approach

DRY or Don't Repeat Yourself is a programming concept in software engineering, which tries to cut down redundancy from your code.

It's not a PHP-specific concept and can be applied to any programming language like JAVA, C++, etc. An example code will help you understand the DRY approach in a much better way.

$mysql = mysql_connect('localhost', 'admin', 'admin_pass');

mysql_select_db('wordpress' or die('Cannot Select Database.');

Now after using the DRY approach, the code will look something like this:

```
$db_host  = 'localhost';

$db_user = 'admin';

$db_pass = 'admin_pass';

$db_name = 'wordpress';

$mysql = mysql($db_host, $db_user, $db_pass);

mysql_select_db($db_name);
```

12) Try to Prevent Deep Nesting

Try to prevent deep nesting levels in your code, as much as possible. It can make things really difficult for you when you need to debug your code and can take the heck out of people who will try to review your code.

Try to use conditions as logically as possible to avoid unnecessary deep nesting. It is not only a very poor programming practice, but also can make your code look ugly enough to your fellow developers.

13) Do Not Ever Put phpinfo() in the Root Directory

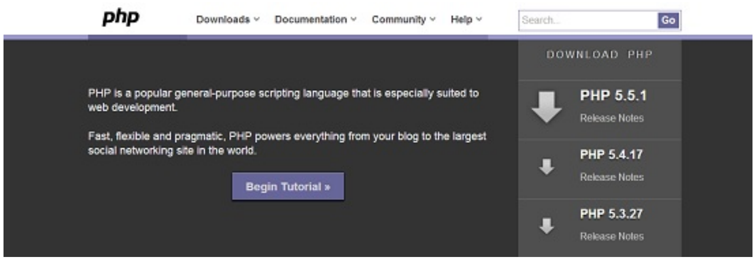
If you want a detailed information about the server environment you are working on, then phpinfo() is a very handy function for you. You can simply create a PHP file with:

```
<?php

phpinfo();

?>
```

You can simply put this file in your server and can get detailed information about your server. But a lot of developers make the mistake of placing this file in the root directory of their server as it can open many security loopholes for hackers prying on their site.

The best idea is to delete this file whenever you are done with it. It can hardly take a few seconds for you to create this super-small file again when you need.

14) Make the PHP Manual Your Best Friend



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

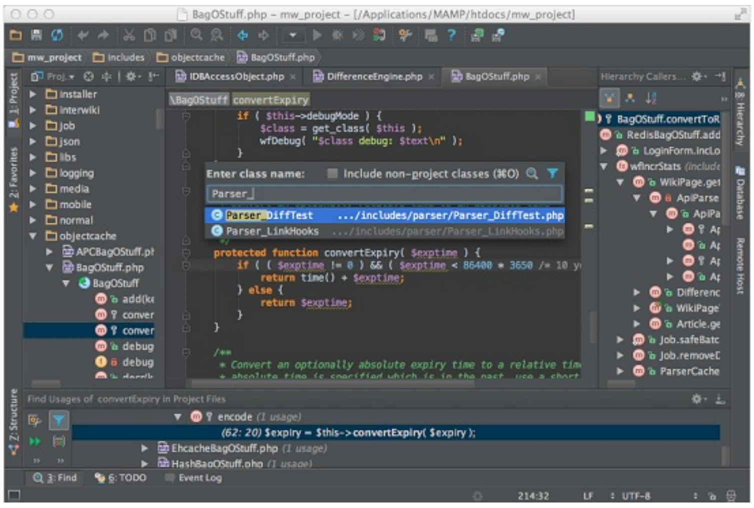
Fast, flexible and pragmatic, PHP powers everything from your blog to the largest social networking site in the world.

(Image Source: php.net)

The best thing about PHP is that it has got a very well-documented manual for you to use anytime you need. Just head over to*http://php.net* and you will get almost every possible information you may need, in a very well-organised manner.

They have also got a new look to their site (which is still in beta stage), for you to try out. Another great thing about the PHP manual is that almost every article is filled with some truly useful comments.

15) Time to Use an IDE


BagOStuff.php – mw_project – [/Applications/MAMP/htdocs/mw_project]
mw_project
includes
objectcache
BagOStuff.php
IDBAccessObject.php
DifferenceEngine.php
BagOStuff.php
Hierarchy Callers...
Enter class name:
Include non-project classes (⌘O)
Parser_
Parser_DiffTest .../includes/parser/Parser_DiffTest.php
Parser_LinkHooks .../includes/parser/Parser_LinkHooks.php
Find Usages of convertExpiry in Project Files
3: Find
6: TODO
Event Log
214:32
LF
UTF-8

```
\BagOStuff convertExpiry
        if ( $this->debugMode ) {
            $class = get_class( $this );
            wfDebug( "$class debug: $text\n" );
        }

    protected function convertExpiry( $exptime ) {
        if ( ( $exptime != 0 ) && ( $exptime < 86400 * 3650 /* 10 y
            return time() + $exptime;
        } else {
            return $exptime;
        }
    }

    /**
     * Convert an optionally absolute expiry time to a relative tim
```

```
encode (1 usage)
    (62: 20) $expiry = $this->convertExpiry( $expiry );
EhcacheBagOStuff.php (1 usage)
HashBagOStuff.php (1 usage)
```

(Image Source: jetbrains.com)

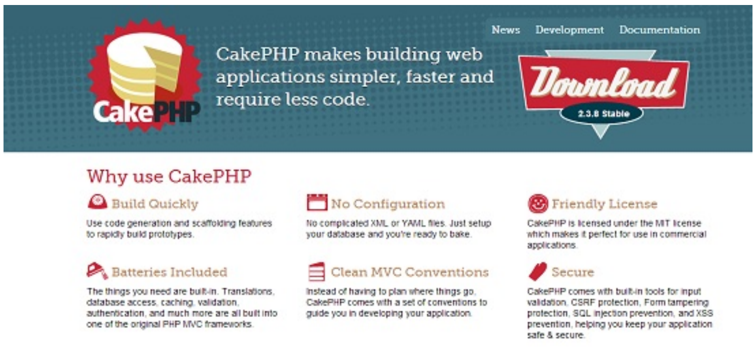
IDE stands for Integrated Development Environment which can make PHP development a lot easier and interesting for developers. IDE's offer a great range of features like *Syntax Highlighting, Code Completion, Navigation, Documentation, Debugging*, etc. which can make you much more productive and can help you to write good code with very less mistakes.

Some of the best IDE's for PHP are NetBeans, phpDesigner, phpStorm, phpEdit, etc. Choose the one which suits you the best.

16) Try a PHP Framework

If you have learned the fundamentals of PHP then it's time for you to try some PHP frameworks. There are tons of PHP frameworks available out there, most of which are designed on the basis of Model-View Controller (MVC) software architecture.

There are lots and lots of new and interesting things for you to learn using a PHP framework. Frameworks like CakePHP, CodeIgniter, Zend, Symfony can help you create some awesome PHP applications with ease.



CakePHP makes building web applications simpler, faster and require less code.

Why use CakePHP

Build Quickly
Use code generation and scaffolding features to rapidly build prototypes.

No Configuration
No complicated XML or YAML files. Just setup your database and you're ready to bake.

Friendly License
CakePHP is licensed under the MIT license which makes it perfect for use in commercial applications.

Batteries Included
The things you need are built-in. Translations, database access, caching, validation, authentication, and much more are all built into one of the original PHP MVC frameworks.

Clean MVC Conventions
Instead of having to plan where things go, CakePHP comes with a set of conventions to guide you in developing your application.

Secure
CakePHP comes with built-in tools for input validation, CSRF protection, Form tampering protection, SQL injection prevention, and XSS prevention, helping you keep your application safe & secure.

(Image Source: CakePHP.org)

17) Running PHP Locally

If you are developing PHP applications, then it's quite likely that you would want to carry out the entire development process locally in your computer.

You just need to have a web server like Apache and PHP for this. If you need to carry out database related operations like registrations, login validation, etc., then you would need MySQL for that purpose.

You can obviously install all of them separately, but there are better solutions for you to use like, XAMPP, WAMP or MAMP which are very simple and easy to configure and can help you setup a local server within minutes.

(Image Source: mamp.info)

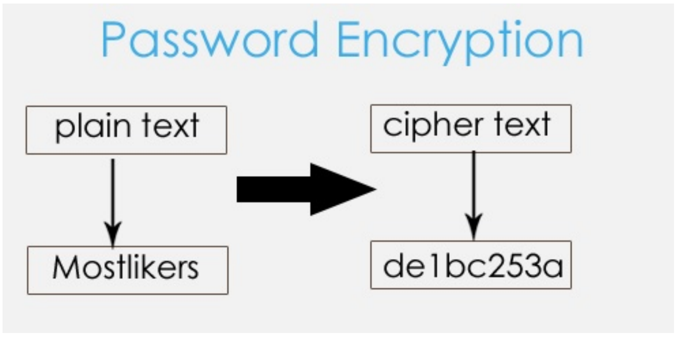
18) Start Playing With Objects

The concept of object-oriented programming was first introduced in PHP 4, but the major improvement in OOP was introduced from PHP 5.

There are generally two types of programming languages – Procedure Oriented Programming Language and Object Oriented Programming Language, and the first programming approach you have ever used must be the Procedural one.

But now it's time for you to get into object-oriented programming with PHP as it offers a higher level of flexibility and can reduce the code length immensely. Your code will look much more organised and easily debuggable with the object oriented approach.

19) Encrypting Passwords is Necessary



(Image Source: karthickinfotech.blospot.com)

Many beginners in PHP make the mistake of storing sensitive data like passwords directly in the database without using any kind of encryption. But it is never a good practice to store passwords like "bare" strings in the database. There are various ways of storing password safely, like MD5, SHA1, MD4, MD2, Whirlpool, Salsa20, etc.

You may use:

$password = md5($password);

$password = sha1($password);

But I personally think that no public encryption algorithm is fully safe and you must create your own algorithm to make things really difficult for hackers.

20) Upgrade PHP

An obvious statement it is, but there are many developers who are reluctant enough to upgrade their PHP software in time. There are various performance improvements and feature additions in the newer versions of PHP. So there is absolutely no reason for you to avoid upgrading your PHP.

These new features can make your programming life a lot interesting and easier. There are some bug fixes and security improvements with almost every new release too.

21) Interact With Other Developers

One of the easiest ways to learn new things is to interact more with people. There are many people who are reluctant enough to admit that they don't know much about a certain topic and need to ask others for help.

But if you think like this, then you will put an end to a great way of learning new and interesting things.

Ask you friends or other seasoned PHP developers on various programming related forums like Stack Overflow, Dream in Code, etc. There are many people who are always ready to lend their hand to help others who really need it.

ABOUT ARITRA ROY

Aritra Roy, is a Blogger, Freelance Writer, Designer and Online Entrepreneur who believes in the power of written words to educate, influence and inspire people.
View all posts by Aritra Roy →

This entry was posted in PHP and tagged php, PHP Best Practices, Programming, Web Development. Bookmark the permalink.

JOIN OUR MAILING LIST

First Name	
Last Name	
Email	

Subscribe

LOGIN

Email	Email
Password	Password

Remember Me ☐

Log in

Create an account
Forgot your password?

MAKE MONEY FROM US. BECOME AN AFFILIATE!