tuts+

CODE > PHP

30+ PHP Best Practices for Beginners

by Glen Stansberry 13 Aug 2009

Difficulty: Intermediate Languages: English

PHP Web Development

PHP is *the* most widely-used language for programming on the web. Here are thirty best practices for beginners wanting to gain a firmer grasp of the fundamentals.

Editor's Note: The "Best Practices" series has been my baby for three articles now. However, due to my focus on the CI video series, I've decided to hand off this next entry to Glen. Having said that, I'm not very good at keeping my mouth shut! I thought it might be fun to sporadically add a few rebuttals to his tips. I hope he doesn't mind!

1. Befriend the PHP Manual

If you're new to PHP, then it's time to get acquainted with the awesomeness that is the PHP manual. The PHP manual is incredibly thorough and has truly helpful comments following each article. Before asking questions or trying to figure out an issue on your own, save some time and just head straight to the manual. Odds are the answer to your question is already nestled in a helpful article at the PHP.net site.

2. Turn on Error Reporting

Error reporting in PHP is very helpful. You'll find bugs in your code that you might not

have spotted earlier, as not all bugs keep the application from working. There are different levels of strictness in the reporting that you can use, but E_ALL will show you the most errors, critical and warnings alike.

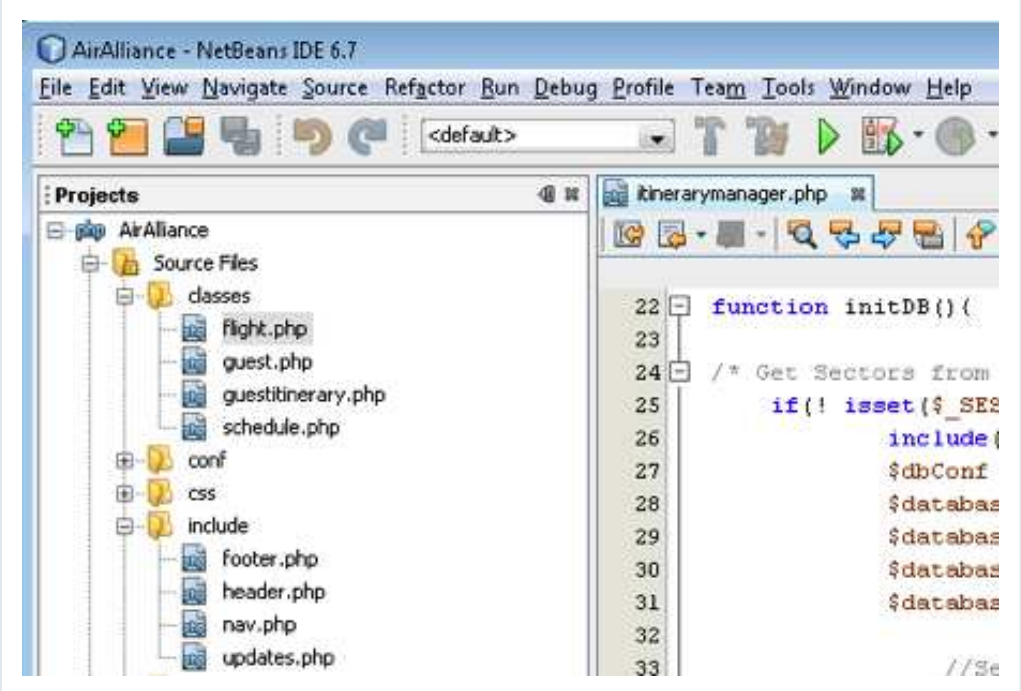
Once you've gotten your application ready for production, you'll want to turn off error reporting, or your visitors will see strange errors that they don't understand.

3. Try an IDE

IDE's (Integrated Development Environments) are helpful tools for any developer. While they're not for everyone, an IDE definitely has its place. IDE's provide tools like

- syntax highlighting
- code completion
- error warnings
- refactoring (reworking)

And many other features. There are plenty of great IDEs out there that support PHP.



4. Try a PHP Framework

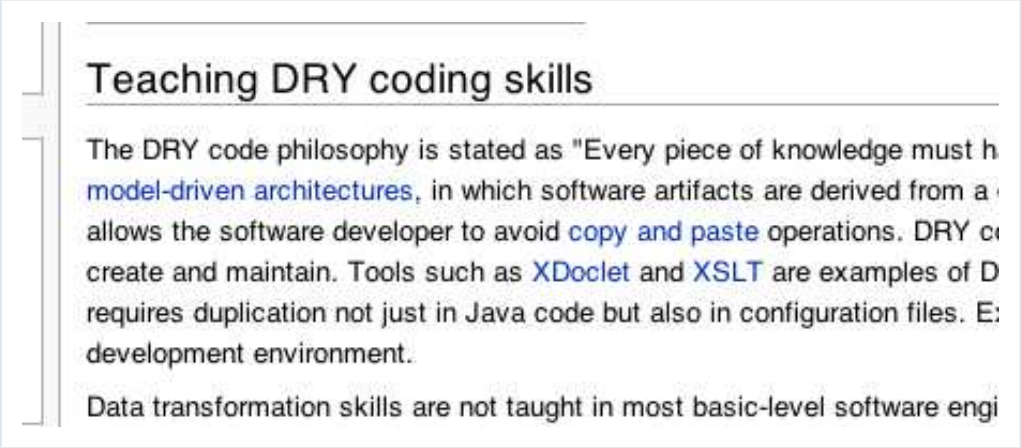
You can learn a lot about PHP just by experimenting with PHP frameworks.

Frameworks like CakePHP or CodeIgniter allow you to quickly create PHP applications, without having to be an expert with PHP. In a sense, they're almost like PHP training wheels that show you what a PHP application should look like, and show you valuable programming concepts (like separating the logic from the design, etc.).

> *Rebuttal: I personally wouldn't recommend that beginners use a framework. Learn the fundamentals first. :)*

5. Learn the DRY Approach

DRY stands for Don't Repeat Yourself, and it's a valuable programming concept, no matter what the language. DRY programming, as the name implies, is ensuring that you don't write redundant code. Here's an example from Reinhold Weber:



> Teaching DRY coding skills
>
> The DRY code philosophy is stated as "Every piece of knowledge must h
> model-driven architectures, in which software artifacts are derived from a
> allows the software developer to avoid copy and paste operations. DRY c
> create and maintain. Tools such as XDoclet and XSLT are examples of D
> requires duplication not just in Java code but also in configuration files. E
> development environment.
>
> Data transformation skills are not taught in most basic-level software engi

This code...

```
1  $mysql = mysql_connect('localhost', 'reinhold', 'secret_hash');
2  mysql_select_db('wordpress') or die("cannot select DB");
```

now with the DRY approach:

```
1  $db_host = 'localhost';
2  $db_user = 'reinhold';
3  $db_password = 'secret_hash';
4  $db_database = 'wordpress';
5
6  $mysql = mysql_connect($db_host, $db_user, $db_password);
7  mysql_select_db($db_database);
```

You can read more about the DRY programming principle here and here.

6. Indent Code and Use White Space for Readability

If you don't use indentations and white space in your code, the result looks like a Jackson Pollack painting. Ensure that your code is readable and easy to search because you'll most definitely be making changes in the future. IDEs and advanced text editors can add indentation automatically.

7. "Tier" your Code

Tiering your applications is nothing more than separating the different components of the code into different parts. This allows you to easily change your code in the future. NETTUTS writer Jason Lengstorf has written an excellent article on how to tier your PHP applications for easier maintenance.

8. Always Use <?php ?>

Often times programmers try to take shortcuts when declaring PHP. Here are a few common ones:

```
<?
    echo "Hello world";
?>

<?="Hello world"; ?>

<% echo "Hello world"; %>
```

While these do save a few characters, all of these methods are depreciated and unofficial. Stick with the standard <?php ?> as it will be guaranteed to be supported in all future versions.

9. Use Meaningful, Consistent Naming Conventions

Naming this isn't just for your own good. There's nothing worse than trying to find your way through some other programmer's nonsensical naming conventions. Help yourself *and* others by using names that make sense for your classes and functions.

10. Comment, Comment, Comment

Aside from using white space and indentations to separate the code, you'll also want to use inline comments to annotate your code. You'll thank yourself later when you're needing to go back and find something in the code, or if you just can't remember what a certain function did. It's also useful for anyone else who needs to look over your code.

11. Install MAMP/WAMP

MySQL is the most popular type of database to use with PHP (though it's not the only one). If you're wanting to set up a local environment to develop and test your PHP applications on your computer, look into installing MAMP (Mac) or WAMP (Windows). Installing MySQL on your own computer can be a tedious process, and both of these software packages are drop-in installs of MySQL. Clean and simple.

12. Give your Scripts Limits

Putting a time limit on your PHP scripts is a very critical thing. There are times when your scripts will fail, and when they do, you'll want to use the set_time_limit function to avoid infinite loops and database connection timeouts. The set_time_limit puts a time limit on the maximum number of seconds a script will run (the default is 30). After that time period, a fatal error is thrown.

13. Use Objects (or OOP)

Object-oriented programming (OOP) uses objects to represent parts of the

application. Not only is OOP a way to break your code into separate, logical sections, it also reduces code repetition and makes it much easier to modify in the future. If you're wanting to learn more, DevArticles has a great write-up on object-oriented programming with PHP.

14. Know the Difference Between Single and Double Quotes

It is more efficient to use *single* quotes in strings as the parser doesn't have to sift through the code to look for escaped characters and other things that double quotes allow. Always try to use single quotes whenever possible.

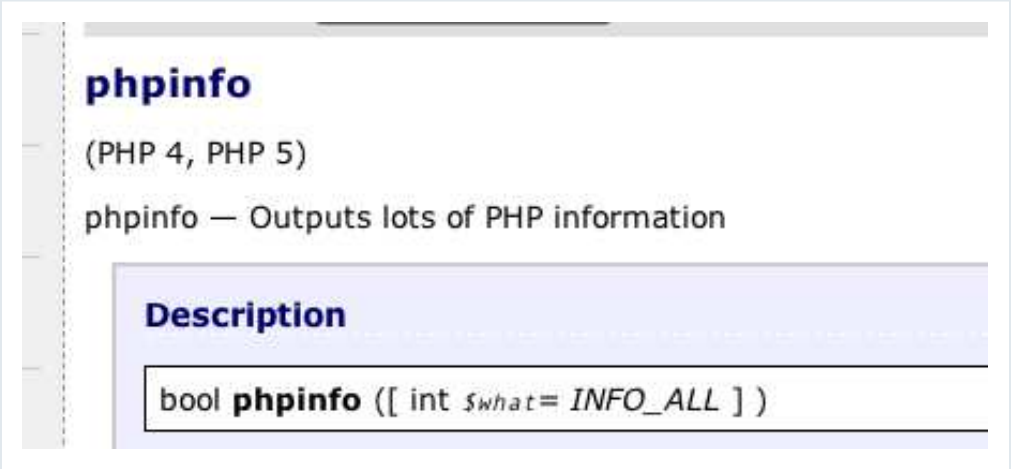
> *Rebuttal: Actually, that's not necessarily true. Benchmark tests show that, when testing strings without variables, there are definite performance benefits to using **double quotes**.*

15. Don't Put phpinfo() in your Webroot

Phpinfo is a beautiful thing. By simply creating a PHP file that has

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

and dropping it onto the sever somewhere, you can instantly learn everything about your server environment. However, a lot of beginners will place a file containing phpinfo() in the webroot of the server. This is a really insecure practice, and if prying eyes gain access, it could potentially spell doom for your server. Make sure phpinfo() is in a secure spot, and as an extra measure, delete it once you're done.



16. Never, Ever Trust Your Users

If your application has places for user input, you should always assume that they're going to try to input naughty code. (We're not implying that your users are bad people. It's just a good mindset.) A great way to keep your site hacker-free is to always initialize your variables to safeguard your site from XSS attacks. PHP.net has an example of a properly secured form with initialized variables:

```
<?php
if (correct_user($_POST['user'], $_POST['password']) {
    $login = true;
}

if ($login) {
    forward_to_secure_environment();
}
?>
```

17. Store Passwords with Encryption

Many PHP beginners often plunk sensitive data like passwords into the database without applying any encryption. Consider using MD5 to encrypt passwords before you put them into the database.

```
echo md5('myPassword'); // renders - deb1536f480475f7d593219aa1afd74c
```

Rebuttal: Keep in mind, however, that MD5 hashes have long since been compromised. They're absolutely more secure than

not, but, with the use of an enormous "rainbow table," hackers can cross reference your hash. To add even more security, consider adding a salt as well. A salt is basically an additional set of characters that you append to the user's string.

18. Use Database Visualization Design Tools

If you're finding it difficult to plan and modify databases for your PHP applications, you might look into using a database visualization tool. MySQL users can work with DBDesigner and MySQL Workbench to visually design your databases.

19. Use Output Buffering

Output buffering is a simple way to greatly improve the performance and speed of your PHP script. Without output buffering, your script will show the HTML on the page as it's processed - in pieces. Adding output buffering allows the PHP to store the HTML as a variable and send it to the browser in one chunk.

To enable output buffering, simply add ob_start() like so at the top of the file.

Rebuttal: Though not required, it's generally considered to be a good practice to go ahead and append the "ob_end_flush();" function as well to the bottom of the document. P.S. Want to compress the HTML as well? Simply replace "ob_start();" with

"ob_start('ob_gzhandler')";

Refer to this Dev-tips article for more information.

```
<!DOCTYPE html>
<?php ob_start('ob_gzhandler'); ?>
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>untitled</title>
</head>
<body>

</body>
</html>
<?php ob_end_flush(); ?>
```

20. Protect your Script From SQL Injection

If you don't escape your characters used in SQL strings, your code is vulnerable to SQL injections. You can avoid this by either using the mysql_real_escape_string, or by using prepared statements.

Here's an example of mysql_real_escape_string in action:

```
$username = mysql_real_escape_string( $GET['username'] );
```

and a prepared statement:

```
$id = $_GET['id'];
$statement = $connection->prepare( "SELECT * FROM tbl_members WHERE id = ?" );
$statement->bind_param( "i", $id );
$statement->execute();
```

By using prepared statements, we never embed the user's inputted data directly into our query. Instead, we use the "bind_param" method to bind the values (and escaping) to the query. Much safer, and, notably, faster when executing multiple CRUD statements at once.

Read more on creating secure PHP applications at Nettuts.

21. Try ORM

If you're writing object-oriented PHP, then you can use the nifty object relational mapping (ORM). ORM allows you to convert data between relational databases and object-oriented programming languages. In short: ORM allows you to work with databases the same way that you work with classes and objects in PHP.

There are plenty of ORM libraries for PHP like Propel, and ORM is built into PHP frameworks like CakePHP.

22. Cache Database-Driven Pages

Caching database-driven PHP pages is an excellent idea to improve the load and performance of your script. It's really not all that difficult to create and retrieve static files of content with the help of our good friend ob_start(). Here's an example taken from Snipe.net:

```
01 // TOP of your script
02 $cachefile = 'cache/'.basename($_SERVER['SCRIPT_URI']);
03 $cachetime = 120 * 60; // 2 hours
04 // Serve from the cache if it is younger than $cachetime
05 if (file_exists($cachefile) && (time() - $cachetime < filemtime($cachefile))) {
06 include($cachefile);
07 echo "<!-- Cached ".date('jS F Y H:i', filemtime($cachefile))." -->";
08 exit;
09 }
10 ob_start(); // start the output buffer
11 // Your normal PHP script and HTML content here
12 // BOTTOM of your script
13 $fp = fopen($cachefile, 'w'); // open the cache file for writing
14 fwrite($fp, ob_get_contents()); // save the contents of output buffer to the file
15 fclose($fp); // close the file
16 ob_end_flush(); // Send the output to the browser
```

This bit of code will use a cached version of a page that is less than 2 hours old.

23. Use a Caching System

If you're wanting a more robust caching system, there are a few caching scripts for

PHP that might be more complete than the above example.

- Memcached
- APC
- XCache
- Zend Cache
- eAccelerator

24. Validate Cookie Data

Cookie data, like any data passed on the Web, can be harmful. You can validate cookie data with either the htmlspecialchars() or mysql_real_escape_string().

25. Use Static File Caching Systems

Aside from using database caching systems like Memcached, you might also want to try a templating system to increase performance in your PHP applications. Smarty is a robust templating system has caching built into it.

26. Profile your Code

Profiling your code with a tool like xdebug can help you to quickly spot bottlenecks and other potential problems in your PHP code. Some IDEs like Netbeans have PHP profiling capabilities as well.

27. Code to a Standard

Once you've gotten the ropes of PHP down, you can start learning about coding to a standard. There are differences between standards out there (say Zend and Pear),

and finding one and sticking with it will help with the consistency of your coding in the long run.

28. Keep Functions Outside of Loops

You take a hit of performance when you include functions inside of loops. The larger the loop that you have, the longer the execution time will take. Take the extra time and line of code and place the function outside of the loop.

> *Editor's Note: Think of it this way. Try to remove as many operations from the loop as possible. Do you really need to create that variable for every iteration of the loop? Do you really need to create the function each time? Of course not. :)*

29. Don't Copy Extra Variables

Some people like to try and make their code more appealing by copying predefined variables to smaller-named variables. This is redundant and could potentially double the memory of your script. Google Code has bad and good examples of variable usage:

Bad

```
1  $description = strip_tags($_POST['description']);
2  echo $description;
```

Good

```
1  echo strip_tags($_POST['description']);
```

> *Rebuttal: In reference to the comment about "doubling the memory," this actually is a common misconception. PHP implements "copy-on-write" memory management. This*

basically means that you can assign a value to as many variables as you like without having to worry about the data actually being copied. While it's arguable that the "Good" example exemplified above might make for cleaner code, I highly doubt that it's any quicker.

30. Upgrade to the Latest Version of PHP

While it seems like a common sense thing, many people don't upgrade PHP as often as they should. There are lots of performance increases between PHP 4 and PHP 5. Check your server to make sure you're up to date.

31. Reduce the Number of Database Queries

Any way that you can cut back on the number of database queries, the better your PHP script will perform. There are tools like Stace (Unix) and Process Explorer (Windows) that allow you to find redundant processes and how you might combine them.

32. Don't be Afraid to Ask for Help

It's only human nature to want to hide the fact that we don't know much about a certain topic. Nobody likes being a n00b! But how are we going to learn without asking? Feel free to use forums, IRC, StackOverflow to ask more seasoned PHP developers questions. The PHP website has a page on getting PHP help.

Have any rebuttals of your own? I'm sure you do! Let's start the debate.

Follow us on Twitter, or subscribe to the NETTUTS RSS Feed for more daily web development tuts and articles.

Glen Stansberry

Glen Stansberry is a web developer and blogger. You can read more tips on web development at his blog Web Jackalope or follow him on Twitter.

Paul M. • 4 years ago

Neither MD5 nor SHA1 actually encrypt anything. They simply create hashes, which you then store so you don't have to store the plaintext password. You then compare the hashes; you never 'decrypt' anything.

I split this hair because I have to explain it every single freakin' day, and it would help if people got it right in 'blog posts like this one, which is otherwise quite excellent. :-)

20 ^ v • Reply • Share ›

Saurav Shrestha ↗ Paul M. • 3 years ago

Thanks. I hadn't really thought about it as much. Whenever the term 'MD5' or 'SHA1' come up, I usually think of encryption! Gotta rewire the brain now.

2 ^ v • Reply • Share ›

mar tuico ↗ Saurav Shrestha • 2 years ago

Better use Salted Password Hashing.

5 ^ v • Reply • Share ›

Johannes • 4 years ago

Best Practice 32:
Omit the end tag ?>

If you are for example working with header functions of php it can be very annoying when there is a whitespace or line after the ?> tag in one of your of your files

13 ^ v • Reply • Share ›

Fill • 4 years ago

I'm surprised code versioning, such as with Subversion or Git, wasn't included. Makes it easier to have multiple programmers working on the same project at the same time, have multiple checkouts of the project, recover from an accidentally deleted or corrupted file, see who/what/when/why histories, etc.

9 ^ v • Reply • Share ›

chadwithuhc ↗ Fill • 3 years ago

That's a good point fill. They might have left it out because of it not being PHP specific. That could be a more general "Best Programming Practices" instead of just in PHP. Nonetheless, keep versions of your code, kids!

4 ^ v • Reply • Share ›

Tetra ↗ chadwithuhc • 3 years ago

10. is not PHP specific too

1 ^ v • Reply • Share ›

Adam Riddick • 4 years ago

In your article, you say both 'Consider using MD5 to encrypt passwords before you put them into the database.' and then 'MD5 Hashes' .. So I would like to make some corrections.

As Paul M. stated, MD5, and sha1, are hashing algorithms, which is NOT encryption, far from it. Encryption and be reversed, using the method of decryption. Hashing, cannot be reversed, and this is it's biggest use when it comes to storing passwords.

At no point do you, or anyone else, need to be able to gain access to your user's passwords, and so they should not be saved in a retrievable form.

Your website should use a salt value when it comes to authentication, an overall website salt that is added to a users password before it is hashed (MD5 and SHA1 aren't the safest, but other alternatives exist.).

I like to take this a step further by adding a second, user based salt, or random length, which nobody will ever see. With two salt values added to a password before it is hashed, or so some other intricate design, you are massively increasing the security of your user's passwords, and thus their trust.

I would also like to add that database queries should use prepared statements, either through PDO or MySQLI (I am a PDO man myself, because they are better.). This negates the need for having to hard-code injection protection.

4 ^ | v • Reply • Share ›

Ixalmida ➚ Adam Riddick • 4 years ago

Note: Oracle strongly discourages using PDO with MySQL. PDO does not fully support MySQL's capabilities and does not perform as well as MySQLi.

2 ^ | v • Reply • Share ›

J • 4 years ago

Good article. By the way, I found a typo in number 31: "There are tools like Stace (Unix) "

Stace is missing an 'r' (Strace).

4 ^ | v • Reply • Share ›

Kirk Hansen • 3 years ago

Would a better practice for database connection be using PDO?

3 ^ | v • Reply • Share ›

Guest ➚ Kirk Hansen • 3 years ago

Link

^ | v • Reply • Share ›

Kirk Hansen Guest • 3 years ago

http://net.tutsplus.com/tutori...

Reply • Share ›

d33k4y • 3 years ago

Passwords should not be "encrypted" but hashed and salted. MD5 is not an encryption algorithm since it shouldn't be possible to "decrypt" a hash. It's just a matter of naming, but still important.

3 Reply • Share ›

jack d33k4y • 2 years ago

Well no, unless you are a company like LastPass, then there has to be encryption going on there.

Reply • Share ›

Sarath Kumar • 3 years ago

jghjhkjhk

1 Reply • Share ›

Usman Ghani • 3 years ago

Amazing article. This is helpful for both intermediate & experts. But I have one question in mind always that is it better to use framework. As I have no problem in doing coding manually. Then why should I use frameworks like CakePHP or CodeIgniter etc.

http://www.grasphub.com

1 Reply • Share ›

Adrian • 3 years ago

OK, but the first:

http://adrian-stolarski.pl/art...

Ps. dry and kiss and multitiers are the best !

1 Reply • Share ›

JJ • 4 years ago

Single quotes are nice so you can place double quotes in them but double quotes are nice as well, you can do this "some string {$somevar}" or "some string ${somevar}" or "some string {${$somevar}}" Handy, took me a longtime to realize this and I have no clue why. I always just used single quotes or double depending on what i needed in them. Now I just use them both "single or double" freely with no care because I finally understand them:) Just remember to be

a critical thinker when anyone tells you that you should do something, there is likely a better way no matter what!

1 • Reply • Share ›

bernard ➚ JJ • 3 years ago

anyone can tell me what the advantage of using ${somevar} instead of $somevar is? or how this notation is called....

• Reply • Share ›

Saurav Shrestha ➚ bernard • 3 years ago

http://stackoverflow.com/quest...

• Reply • Share ›

Brian • 5 years ago

"validate" isn't really the right word for #24. I would write it as "You can make cookie data safe by using mysql_real_escape_string() when storing it to the database, and htmlspecialchars() when displaying it on an HTML page".

1 • Reply • Share ›

Haxxan • 2 months ago

you have explained it very nicely but i prefer to learn from video more than text and found this tutorial course helpfull.

php beginner course

• Reply • Share ›

Timo • 10 months ago

Really cool collection of best practices. There are some more good sources for PHP Best pratices.
Take a look at:
http://www.phptherightway.com/
https://phpbestpractices.org/

I wrote a blog post about these sites (in german): http://www.joocom.de/blog/php-...

• Reply • Share ›

IIstarscreamII • 2 years ago

I tried the "19.Use Output Buffering" example and got "Content Encoding Error" on browser, I moved ob_start('ob_gzhandler') to first line and the error is gone...

Thanks!! Very nice!! :D

• Reply • Share ›

IIstarscreamII • 2 years ago

Hi!! The "CI video series" link is down... :)

• Reply • Share ›

Scopey • 2 years ago

The "best practice" about short tags is not completely correct. Firstly, ASP style (<%) is the only form which is deprecated. Also, in PHP5.4, the "" tags are enabled regardless of the short tag setting you have in the php.ini. Short tags in that form are fine for most developers (I would encourage it) as long as you are using PHP5.4.

• Reply • Share ›

Scopey ➚ Scopey • 2 years ago

Disqus mangled my message. Refer to http://php.net/manual/en/ini.c...

• Reply • Share ›

ilesh Raval • 2 years ago

Thanks for the real help... I'm sure this will help to one who is freshers

• Reply • Share ›

Steve • 2 years ago

Thank you for the list. I'm actually using this kind of algorithm in http://moo.pw

• Reply • Share ›

GAM Software Solutions • 2 years ago

Well, nice post sharing here about PHP with good useful examples.

• Reply • Share ›

Eric Gillette • 2 years ago

Solid!! This is an excellent article!

• Reply • Share ›

jack • 2 years ago

MD5 is not an encryption function. It's a hash function. Also, you probably shouldn't be using MD5 nowadays.

• Reply • Share ›

jack ➚ jack • 2 years ago

PS. I'm not blind. I did notice other people mentioned this too. But like others said it's confused FAR too often that it won't hurt to repeat this a few times.

• Reply • Share ›

I • 2 years ago

IIII

• Reply • Share ›

andrewCarlson • 2 years ago

One thing I don't like about frameworks is they focus more on running cli commands to generate required code without explaining what or why that code is needed or how it's used. They become a crutch, most frameworks rely on static route tables which, for simple sites is ok, but large complex sites is silly. Having to read a text file (yml, php, xml, whatever) prior to passing off to a controller will slow your app down, for every user, and every action. Some frameworks claim MVC, but skip the M altogether. There is no model, only controllers and views.

As far as database access, choose PDO or ORM (which will probably use PDO for you). PDO can really help with escaping and protecting data inserted into the database. ORM is really the next step.

I'm surprised Unit Testing wasn't here. Learn test driven development and if possible, behavior driven development practices. It really does streamline the whole development process. As part of unit testing, Continuous Integration can be a big help.

• Reply • Share ›

Olaf • 2 years ago

If you have php 5.5+ please start using password_hash http://us3.php.net/manual/en/f...

• Reply • Share ›

tetra • 3 years ago

+1 thing: never use "==" operator. Use "===" instead.

• Reply • Share ›

functionN ➚ tetra • 3 years ago

wrong, on certain instances you need ==

3 • Reply • Share ›

bjb • 3 years ago

dcffcdcd

• Reply • Share ›

fyrye • 3 years ago

One function not mentioned is the *printf line of functions. It concatenates and formats variables with with a static formatting rule.

Other best practices that weren't discussed are validating argument data types and exception handling.

Some examples (don't use in production):
echo 'SELECT ' . $table . '.' . $columnA . 'FROM ' . $table . ' WHERE ' $table . '.' . $columnB . ' = ' $value;

vs

printf('SELECT %1$s.%2$s FROM %1$s WHERE %1$s.%3$s = %4$d', $table, $columnA, $columnB, $value);

or

vprintf('SELECT %1$s.%2$s FROM %1$s WHERE %1$s.%3$s = %4$04d', array($table, $columnA, $columnB, $value));

---

see more

• Reply • Share ›

Mohamed Shimran • 3 years ago

Really good practices , thank you :3

Ultimate programming tutorials

• Reply • Share ›

Renu • 3 years ago

if ($abc) { // some code // }
OR
($abc)? 'hello':'hi';

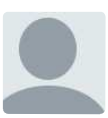
which one is better? "IF" or "?:"

• Reply • Share ›

Vinu ↗ Renu • 3 years ago

if your code is short,and you have only few conditions to check conditional statements are better & efficient.

• Reply • Share ›



Prajwol Atno • 3 years ago

Great tips.

• Reply • Share ›

KASIBANTE KEN • 3 years ago

HI please help me aim teaching myself how to develope a web and i have been using DREAMWAVER but someone toled me its not good as such so i want to know how i can use

DREAMWAVER but someone toled me its not good as such so i want to know how i can use php.thanks

• Reply • Share ›

Prajwol Atno ➚ KASIBANTE KEN • 3 years ago

Dreamweaver has nothing to do with programming or coding, its just an text editor with many features. Alternatively u can use sublime, netbeans, notepad++ etc. If you want to learn php codeacademy.com would be a nice place to start..

4 • Reply • Share ›

henry58 • 3 years ago

Thanks for this i do find that the lessons available here worked really well for me. http://learnphpquick.com/blog/

• Reply • Share ›