No PHP for large projects? Why not? [closed]

Asked 16 years ago Modified 9 years, 3 months ago

Viewed 25k times Part of PHP Collective

64

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 4 years ago.

Improve this question

I've read a few posts where people have stated (not suggested, not discussed, not offered) that PHP should not be used for large projects.

Being a primarily PHP developer, I ask two questions:

1. What defines a "large project"?
2. Why not? What are the pitfalls of using PHP

I run a small development team and I know from experience the quality construction, organization, documentation, commenting and encapsulation are our highest priority. We are able to develop great projects using our own framework and approach but still, I don't want to invest further if I'm wasting my time.

Thoughts?

PHP php scalability projects

Share

Improve this question

Follow

edited Jun 4, 2009 at 14:47

cletus
625k ● 169 ● 917 ● 945

asked Dec 21, 2008 at 23:42

jerebear
6,635 ● 4 ● 33 ● 39

1 me.veekun.com/blog/2012/04/09/php-a-fractal-of-bad-design
– user606723 May 2, 2012 at 15:42

11 Answers

Sorted by: Highest score (default)

97

I really hate it when people say flat out that PHP is a terrible language because you can write code which mixes presentation with logic, or that it lets you allow SQL injection. **That's nothing at all to do with the language, that's the developer.**

PHP has proved itself to be highly scalable: Wikipedia is one of the largest and most popular sites on the Internet and it runs PHP. Enough said?

There are a lot of tools/libraries out there that give you a framework to work in, making it less likely that someone will write poor, less-maintainable code: see CakePHP, Symfony, PDO, Smarty, etc etc etc.

It has received a bad rap because it's a language which has very low barriers to entry: it's free, you can get very cheap PHP hosting, **the documentation is the best there is**, there's plenty of tutorials online, plus it makes a lot of things very easy (eg: open a URL and get the contents of the file: `file('http://www.google.com');`). This means that a lot of newbies picked it up and made a lot of very dodgy sites with it, but that's going to happen with whatever language you choose as your first.

Work with a solid ORM framework (there's about 30 questions on SO about which is best), and it will treat you good.

Share Improve this answer Follow

edited May 23, 2017 at 12:17
Community Bot
1 ●1

answered Dec 22, 2008 at 0:21
nickf
546k ●198 ●658 ●725

12 Agreed. PHP is a very easy language to learn, and is much more flexible than say, Java. This, unfortunately, has the effect of allowing inexperienced coders (I conciously did not use the word "programmers") to write scripts that manage to

run, but are extremely buggy. This is what gives PHP it's (undeserved) bad reputation -- a bunch of amateur coders who write crappy code. – J. Taylor May 29, 2010 at 7:10

1 You can do almost anything in any language, you can write spagetti code or mix logic and presentation on a hybrid OO language in .NET and a good template based design in PHP or a small app in ASP/VB6 that serves your purpose perfectly. A bad design would lead you to costly maintainace. So does too much complexity for a simple project. I would not build a complex ERP system or a graphical game in PHP, there a better tools for that. In a few words, Use the right design and tool for the job. – hagensoft Sep 8, 2012 at 9:30

1 As a both PHP (Laravel) and Ruby (Rails) developer, I find PHP lacks in few areas which matter in enterprise development. I tend to write messy code in PHP whereas Rails gently force me to write clean/maintainable code. – Nilay Anand Dec 8, 2016 at 9:11

I'm okay with core PHP and never feel a need of any framework.I use PHP pdo for database. Which making my tasks very easy. If i need any i would like to develop my own class and call it where ever i need. – mofidul Jul 10, 2017 at 16:17

34

A lot of people who say not use it are really saying don't use PHP 4. It comes down to this

you can write good code in any language

and

you can write bad code in any language

PHP can very often lend itself to becoming tangled spaghetti code libraries and make your 'application' really just a series of scripts (see Moodle for a good example of this...)

I think a lot of the 'Don't use PHP for large stuff' is coming from PHP being hacked up from it's original purpose: a templating language. Which I can understand, but there are lots of projects that prove you can do it (Drupal, mediawiki, Facebook).

Share Improve this answer Follow

edited Sep 25, 2014 at 5:55

answered Dec 21, 2008 at 23:58

jon skulski

2,305 ● 3 ● 18 ● 27

I've heard facebook aern't using PHP any longer. The .php links are now for legacy and are probably rewritten or parsed differently through the server. I can't remember where I read the article, but from what I can it was reputable. – alex Dec 22, 2008 at 0:07

Facebook's frontend is written in php, if anything at all. I know that Myspace uses .NET but has ".cfm" links for legacy compatibility... – navitronic Dec 22, 2008 at 0:10

1 Facebook is still written in PHP – Cal Jan 27, 2009 at 17:40

Yes, CFM is associated mainly with Cold Fusion Markup. – DoctorLouie Feb 4, 2010 at 4:42

6 Facebook is PHP, but they had to write a PHP->C++ translator to overcome performance problems.

developers.facebook.com/news.php?story=358&blog=1
– Matt Garrison Feb 26, 2010 at 15:41

19

Theres no reason you can't use PHP for large projects. After all, Facebook is built on PHP. There will be issues however but there are issues with any large project.

What makes PHP so pervasive is the low barrier to entry and cheap hosting. It runs as an Apache extension and you can pretty much just start coding. If you go to more enterprise platforms such as .Net or Java, they have a much higher barrier to entry but they also come with a lot of infrastructure to help you make applications that scale.

For example, the database abstraction in PHP is (imho) woeful. It's vendor specific. With MySQL, people tend to do things like:

```
function get_users($surname) {
  mysql_query("select * from users where surname = '$s
  ...
}
```

which is bad for several reasons:

- It makes poor use of the query cache;
- It doesn't handle escaping of characters (which, of course, can be done with `mysql_escape_string()` but you'll be surprised how often people don't do this); and

- It's fairly easy to code in such a way as to allow SQL injection attacks.

Personally I prefer mysqli for all the above reasons but it has it's own problems: namely that using LONGTEXT fields crashes mysql and has done since at least 2005 with still no fix (yes I and several others have raised a bug).

Compare this to Java (with which I'm more familiar) and JPA or Ibatis are vastly better ORM solutions with higher startup costs but they will help you on an enterprise scale.

So you're not prohibited from doing large projects on PHP. It's simply harder in that you have to do increasingly more work yourself to replicate what other platforms provide you.

That being said, PHP + memcached/APC + beanstalkd goes a long way.

Oh that's the other problem: PHP doesn't really support background processing or threading. You need something else for that (or standalone scripts). If you're using something else, why not use that for the Web stuff too (eg Java, Ruby, .Net, etc)?

Share Improve this answer Follow

answered Dec 22, 2008 at 0:00

cletus
625k ● 169 ● 917 ● 945

2 Facebook is kind of a tortured example. Facebook isn't a complex business app. It is a fairly simple that is scaled very highly. – BobbyShaftoe Dec 22, 2008 at 0:05

1 how exactly does that example query make poor use of the query cache? – ʞɔıu Dec 22, 2008 at 0:37

5 PHP now has database agnostic libraries for connecting to a database. (PDO). PDO also supports prepare() and execute(), which handles escaping of input. Just because PHP allows for poor coding doesn't make it a bad language. – Ryan Doherty Dec 22, 2008 at 1:01

1 @Ryan: Did anyone suggest PHP was a bad language here? Kneejerk reaction to some not-completely-positive PHP discussion perhaps? ;) – Gareth Dec 22, 2008 at 1:39

That example has a problem you've overlooked: it doesn't abstract database access away from the function request: get_users() shouldn't even be assembling SQL, never mind submitting the query itself. So many people make this mistake it just isn't funny. – staticsan Dec 22, 2008 at 5:04

18

As the question I linked was deleted, I'll place some of it here:

Question

I made a tongue-in-cheek comment in another question thread calling PHP a terrible language and it got down-voted like crazy. Apparently there are lots of people here who love PHP.

So I'm genuinely curious. What am I missing? What makes PHP a good language?

Here are my reasons for disliking it:

- PHP has inconsistent naming of built-in and library functions. Predictable naming patterns are important in any design.
- PHP has inconsistent parameter ordering of built-in functions, eg array_map vs. array_filter which is annoying in the simple cases and raises all sorts of unexpected behaviour or worse.
- The PHP developers constantly deprecate built-in functions and lower-level functionality. A good example is when they deprecated pass-by-reference for functions. This created a nightmare for anyone doing, say, function callbacks.
- A lack of consideration in redesign. The above deprecation eliminated the ability to, in many cases, provide default keyword values for functions. They fixed this in PHP 5, but they deprecated the pass-by-reference in PHP 4!
- Poor execution of name spaces (formerly no name spaces at all). Now that name spaces exist, what do we use as the dereference character? Backslash! The character used universally for escaping, even in PHP!
- Overly-broad implicit type conversion leads to bugs. I have no problem with implicit conversions of, say,

float to integer or back again. But PHP (last I checked) will happily attempt to magically convert an array to an integer.

- Poor recursion performance. Recursion is a fundamentally important tool for writing in any language; it can make complex algorithms far simpler. Poor support is inexcusable.
- Functions are case insensitive. I have no idea what they were thinking on this one. A programming language is a way to specify behavior to both a computer and a reader of the code without ambiguity. Case insensitivity introduces much ambiguity.
- PHP encourages (practically requires) a coupling of processing with presentation. Yes, you can write PHP that doesn't do so, but it's actually easier to write code in the incorrect (from a sound design perspective) manner.
- PHP performance is abysmal without caching. Does anyone sell a commercial caching product for PHP? Oh, look, the designers of PHP do.

Worst of all, PHP convinces people that designing web applications is easy. And it does indeed make much of the effort involved much easier. But the fact is, designing a web application that is both secure and efficient is a very difficult task.

By convincing so many to take up programming, PHP has taught an entire subgroup of programmers bad habits and bad design. It's given them access to capabilities that they lack the understanding to use safely. This has led to PHP's reputation as being insecure.

(However, I will readily admit that PHP is no more or less secure than any other web programming language.)

What is it that I'm missing about PHP? I'm seeing an organically-grown, poorly-managed mess of a language that's spawning poor programmers.

So convince me otherwise!

Top Rated Answer

I'll take a stab at responding to each of your bullet points

> PHP has inconsistent naming of built-in and library functions. Predictable naming patterns are important in any design.

I both love and hate this topic. Because at its core, this issue is correct. Why are some bi-word function split with an underscore, and some aren't? Why do needle and haystack parameters swap positions in the argument signature sometimes? It's ridiculous. But at the end of the day... does this really matter? My IDE with intellisense and php.net just a browser click away, this is just plain not that big of a deal. Is it a negative against PHP as a language? Yes. Does it hinder my ability to be an effective programmer? No.

> The PHP developers constantly deprecate built-in functions and lower-level functionality. A good example is when they deprecated pass-by-reference for functions. This created a nightmare for anyone doing, say, function callbacks.

Personally, I think this is not a good point. Deprecation is necessary to the evolution of a language, especially one that has as much kruft as PHP does. PHP gets a lot of flak for "making it easy to be a bad programmer*" but at the same time, the PHP group also gets in trouble when they try to remove stupid constructs from the language, such as call-time pass-by-reference. Eliminating call-time pass-by-reference was one of the best moves they ever made. There was no easier way for a novice developer to shoot themselves in the foot than with this "feature".

> A lack of consideration in redesign. The above deprecation eliminated the ability to, in many cases, provide default keyword values for functions. They fixed this in PHP 5, but they deprecated the pass-by-reference in PHP 4!

I don't think there's a general lack of consideration at all, I think you just got stung by this particular change and have been left with a sour taste in your mouth. Language changes are often known months if not years ahead of time. A migration guide was provided for the move from 4 to 5, and the version differences are documented in the manual. Call-time pass-by-reference was a horrible "feature" and doesn't give the developer any expressive power they can't get by other means. I'm glad it is gone (along with other crap like magic quotes)

> Poor execution of name spaces (formerly no name spaces at all). Now that name spaces exist, what do we use as the dereference character? Backslash! The character used universally for escaping, even in PHP!

I have mixed feelings about this. Part of me thinks "who cares, character escaping has no meaning outside of a string anyway", and part of me thinks "surely they could use something better". But could they? I don't know, I'm not a developer for the Zend parser. Is it a huge oversight that until 5.3 PHP never had namespaces at all? Yes, absolutely.

> Overly-broad implicit type conversion leads to bugs. I have no problem with implicit conversions of, say, float to integer or back again. But PHP (last I checked) will happily attempt to magically convert an array to an integer.

I think it's ok to disagree with how PHP does this, but disagree that it makes the language "bad". But ask me how much I want to sit in this topic and argue about weak vs strong typing. (P.S. I don't, *at all*) For the record: PHP will issue an E_WARNING level error when the type of an argument matters and cannot by solved by coercion.

> Poor recursion performance. Recursion is a fundamentally important tool for writing in any language; it can make complex algorithms far simpler. Poor support is inexcusable.

PHP is a DSL for the web. I've been doing it full-time for 8 years and have maybe used recursion 4 or 5 times, usually for some type of annoying directory or XML traversal. It's just not a pattern that is needed for web development that often. I'm not excusing the slow performance, but this is an academic issue far more than it is a production issue. If you need really powerful recursive performance, PHP is already the wrong language for you.

> Functions are case insensitive. I have no idea what they were thinking on this one. A programming language is a way to specify behavior to both a computer and a reader of the code without ambiguity. Case insensitivity introduces much ambiguity.

I totally 100% agree with this.

> PHP encourages (practically requires) a coupling of processing with presentation. Yes, you can

> write PHP that doesn't do so, but it's actually easier to write code in the incorrect (from a sound design perspective) manner.

*Hmmm, this topic sounds desperately familiar...

But seriously, I find it remarkable that people will complain about a language that will absolutely 100% let you implement any output system you want (the sheer volume and style of PHP templating systems alone speaks to this) - OR - skip all that overhead and just output directly. This does not make PHP bad at all. It's part of what makes PHP good.

> PHP performance is abysmal without caching. Does anyone sell a commercial caching product for PHP? Oh, look, the designers of PHP do.

Do you mean bytecode caching (like an accelerator), or output caching?

If the former, then I don't really know how much I care about this topic. Accelerators are free and easy to run. We could argue about why it isn't part of the language but in the end, I don't think it matters much.

If you are talking about output caching then I don't know what to say to you. ANY web project with significant traffic needs caching (seed podcast #27, for example). This is not a PHP-specific issue *at all*.

In summary, I think you consider PHP a "bad" language in a very academic fashion. And in your previous post you were probably voted down by people like me who use PHP to "get things done".

Second Top Rated Answer

All your criticisms (and some more) are valid. You are allowed and even expected to hate PHP.

But, then again, it has some benefits:

- Ubiquitous
- Fast (especially using opcode caches)
- Huge community (and great documentation)
- Works

Finally, you can overcome many if not all the downsides by writing good code you'd write in any other language. You can write solid, secure and good smelling code in PHP, which many times will run faster and be easier to host and to scale than many alternatives.

Third Top Rated Answer

> What is it that I'm missing about PHP? I'm seeing an organically-grown, poorly-managed mess of a

> language that's spawning poor programmers.

Simple. The fact that poor programmers get very defensive about their language. ;) PHP is easy to learn, much easier than the alternatives, and once you've learned it, it's not exactly obvious 1) what's wrong with PHP, 2) how the alternatives are better, and 3) how to switch to, and learn, one of the alternatives.

And perhaps the fact that, well, what alternatives do people have? ASP? That has plenty of problems on its own, from being unable to run on the majority of webservers (Apache), to some ridiculous and overengineered design choices on its own (webforms? Viewstate? AJAX where your asynchronous" requests are intercepted and run *sequentially*?) Ruby on Rails? Well, perhaps, except how many webservers support it again? It's not exactly easily approachable at the moment. And it's slow. So perhaps PHP's "strength" is really that no *good* alternative exists. At least this is why I stay away from all web programming when at all possible. PHP sucks, and I'm not too keen on any of the alternatives either.

PHP has so many fundamental problems that it's not even funny. From the lack of unicode support, to the many implicit type conversions which often lead to unexpected security holes, to the complete mixing of presentation and... everything else, or to the default database module which doesn't (last I checked) use parametrized queries. We're talking about a language made for two things, database access and generating HTML, and which is terrible at both.

It's just a nasty mess, a language designed by people who aren't qualified, or able, to design a language. ;)

Share Improve this answer Follow

edited May 23, 2017 at 12:10
Community Bot
1 ● 1

answered Dec 22, 2008 at 0:00
Vinko Vrsalovic
340k ● 55 ● 340 ● 373

Is this true in 2013? – johnny Oct 9, 2013 at 20:24

4

For me the worst PHP sin is coupling of presentation with business logic. It's not that you can't write it in a better way, but it doesn't encourage you to, and if anything it encourages you not to.

A large number of security vulnerabilities are associated with PHP sites, too. I can't prove it is disproportionate (after all a lot of sites are written in PHP), but I suspect it is. If I'm right, then since security vulnerabilities are a class of bug, I suspect PHP sites tend to be more buggy on the whole also.

(I don't think that pointing at a few large sites and saying they managed to do it in PHP is any argument against this, by the way. It's a bit like saying that cigarettes don't cause cancer because your next door neighbour smoked and lived to be 100.)

Share Improve this answer Follow

answered Dec 22, 2008 at 0:07

frankodwyer
14k ● 9 ● 52 ● 70

Smoking and living isn't a matter of choice, it's a matter of circumstance. Building a successful site in PHP is a demonstrated matter of choice that can lead to better industry practices when shared at sites like www.highscalability.com – jerebear Dec 22, 2008 at 6:00

2

Check out this similar question - [Can PHP handle enterprise level sites as well as Java]()

Recapping - Facebook, Wikipedia, Yahoo.com, Digg, Flickr and many other giant sites are running on PHP. If you ever come close to making something of that caliber, you can still rest assured you can get there with PHP.

How maintainable, extendable, reliable, secure and performant your applications will be is entirely up to you and is language-agnostic. In favor of PHP though, it has very convenient tools for building web applications.

Share Improve this answer Follow

edited May 23, 2017 at 12:17

Community Bot
1 ● 1

answered Dec 22, 2008 at 0:36

Eran Galperin
86.8k ● 24 ● 118 ● 132

I think you're mixing scalability versus enterprise class design. Facebook et al are not very complex sites. Sure, they do scale out very well, and that is impressive in some sense. You can also scale out static pages fine but what about designing complex software sstems. Is PHP the best choice?
– BobbyShaftoe Dec 22, 2008 at 1:43

1 First of all, facebook has very complex innards. When you consider all the relationships between information pieces and integration with the public API - and how it works on the scale it does, I would consider it very complex. Second, nobody said anything about enterprise - – Eran Galperin Dec 22, 2008 at 2:01

1 the title is about large PHP projects, and facebook is certainly one. You need to know your target audience - PHP wasn't meant for building hospital monitoring software, but constructing web applications. – Eran Galperin Dec 22, 2008 at 2:01

2

For me, and talking about large or even huge projects, it (primarily) boils down to one word: **Dependencies**.

The problem with a scripting language is like in every thing in the world: The greatest advantage is the greatest disadvantage at the same time.

The greatest advantage is to code free and fast. Just write a script and it will server it's purpose. No verbosity needed, simply code.

The greatest disadvantage is, in a way, to check if this script is not disturbing other scripts. Or better: Change an old script others are relying on. Are you sure that all dependencies do work out as you desired?

This is not true for "normal" web page generation, whatever normal means here. But we have a product relying on some 500k lines of source code, with customizations for clients consisting of an additional 100k lines of code, too. And I am deadly glad that a compiler checks all dependencies and warns/errors me in case I did something wrong (like, speaking lowest level here, misstyping a variable or method call).

I think this and the fact that other languages provide more simple-to-use "enterprisy" features by their nature (i.e. application servers for "bank usages") boils it down why many do not see PHP in large (or better: huge) projects.

Share Improve this answer Follow

answered Dec 22, 2008 at 0:38

Georgi
4,411 ● 4 ● 26 ● 20

1

Our company runs several large-ish web sites using PHP, and have had no problems that were related to the language.

Share Improve this answer Follow

answered Dec 21, 2008 at 23:47

JW.
51.6k ● 39 ● 121 ● 150

1

There's something about the construction of the PHP language that is not good enough for me. For example, the name of the functions. It is a non best practice use a several ways to name a function in one language. The mixture between underscores (function_name), words stick together (functionname), etc.I mean, this is really a mess. There are too many functions that are very similar or do the same things, but their names are so confusing. This is not characteristic of a good programming language.

In large deployments, the language must be enough easy and specific to write. Something that PHP omits like the declaration of variable types, become very difficult to understand and deal with later.

Other point is the constant addition of features and canceling of some other. It supposes that the addition of OOP in PHP 5 gonna make the things easier for programmers, but what about the considerations of back compatibility?

The principal reason that this programming language is like it is, is due to its origins: Personal Home Page. It was not designed for large deployments.

I know that there are great efforts to make this language an enterprise-weight language, and personally, I'm waiting for a good enough open source server-side programming language; but until this day comes, it's gonna run a lot of water under the bridge.

Share Improve this answer Follow

answered Oct 2, 2009 at 1:33

nidorano

PHP Actually stands for PHP Hypertext Preprocessor, a kind of recursive in-joke for geeks :) – Chris Sep 18, 2013 at 15:07

0

These are all good answers.

I was a newbie. I've only been coding for 5 years but I directly support and manage 85 small to large websites and I'll tell you what, the potential to get sued by having a website down for a day will contribute a lot to your desire to learn how to and make better code.

It's nice to hear established developers share their thoughts on this matter. I don't think PHP is the best, but it seems my investment in "best practices" is well served.

Thanks everyone!

Share Improve this answer Follow

answered Dec 22, 2008 at 6:04

jerebear

6,635 ● 4 ● 33 ● 39

See:
http://www.ukuug.org/events/linux2002/papers/html/php/

0

Share Improve this answer Follow

answered May 29, 2010 at 7:10

J. Taylor

4,855 ● 4 ● 38 ● 56