Um. You don't need to spend nearly that much, actually. Just a fraction.

NOTE: B1gw0rm was a sockpuppet, and has been deleted.

NOTE: B1gw0rm was a sockpuppet, and has been deleted.

NOTE: B1gw0rm was a sockpuppet, and has been deleted.

Oh my god! Of COURSE creating 100,000 new constants is going to use up memory! # EVEN C++ HAS TO STORE MEMORY FOR CONSTANTS DURING THE COMPILATION STAGE!! PHP \*\*does\*\* inline substitute constants \*\*in the compiled opcodes\*\*. Constants are \*always\* better memory/CPU time \*\*in executed apps when used with an opscode cacher\*\*.

NOTE: B1gw0rm was a sockpuppet; that account has been deleted.

You didn't prove anything. \*\*PHP treats constants much like C++ does.\*\* You are just thinking about it in a flawed way: PHP does much more than just "interpret code". You seem so head-strong, you can't even accept that.

It's only been a month. I always suspected it'd take years, maybe even a few lifetimes, for you to come to your senses.

\*sigh\* This is wrong on a few levels. First, the pre-processor replaces macros with their values. Those values have already been replaced before the complication process begins. Second, the final program is not using memory to store the value of the constants, because the constants no longer exist. PHP on the other hand must keep the values of the constants in memory during the entire execution cycle. Now, please go back and read my original statement where I said it would be great if PHP used \*\*no\*\* memory for defines the way C/C++ does. And by C/C++, I mean a program written in C/C++.

I have never much cared for MVC. Instead, I use a simplified rough model I call "PBD", "Process, Build, Display". MVC has its use... but I don't care for it much myself.

Are you laughing at an e-dick swinging content, and then throwing your own into the ring?

[Others disagree with you.](http://en.wikipedia.org/wiki/Enumerated\_type) &gt; Enumerated types can also prevent a programmer from writing illogical code such as performing mathematical operations on the values of the enumerators.

&gt; Your disdain for testing is something that is systemic in the PHP community When did I say I have a "distain" for unit testing? You can look at my projects on GitHub. They all have unit tests. All of my projects get unit tested. Nice try on the generalization though.

My exact quote was "In which case no memory is used to store them." I said that's what PHP does. I proved him wrong. Simple as that.

This is why the Python and Ruby guys don't take us seriously... :)

Clone? Probably not... though [Redmine](http://www.redmine.org/) is flexible enough for what I've done (however, it's done in Ruby).

Why does it need to be written in PHP? Most PHP bugtrackers suck.

i regret i have but one upvote to give.

newInstanceWithoutConstructor is mostly good for those rare times when you actually need to instantiate an object without actually calling the constructor (in case the constructor has some weird logic in it that isn't exactly portable). As discussed in [the thread](http://comments.gmane.org/gmane.comp.php.devel/67850) about the patch, it's mostly ORMs and stubbing/mocking for unit tests. The person that submitted the patch was Sebastian Bergmann, the guy behind PHPUnit. It's immediately useful in unit testing when you want to test a method of an object without ever calling the constructor.

I think he understands that very clearly, which is why he noted &gt; (however, it's done in Ruby) No need to be a smartass.

He actually suggested a good bug tracker. You commented on a basic feature of a programming language. He wins, you're still being a smartass.

I know you're joking, but I should mention that PHP was designed to integrate with HTML somewhat seamlessly, while Python and Ruby weren't. I happen to love MVC, but it's easier to ignore it when you can throw PHP directly into an HTML page. Py and ruby don't have it quite as easy, so they rely more on frameworks. I guess. ;)

Hey, I find writing complex SQL joins to be fun… it's the mundane stuff I repeat all the time in every project I find boring. But I'll still let a framework do it all for me when I can.

even so, how is that a problem?

+1 for Mantis, we used it at my old job and I was very happy with it - I found it to be very userfriendly compared to Bugzilla.

Indeed, sort() returns boolean while sorting the argument variable. Fell into that trap so many times before ...

&gt; He may have added something to some discussion somewhere, but it was completely useless to the actual discussion. ...at least, \*in your opinion\* ... I think I clearly noted that 1) there's not a clone (as far as I've ever seen) and 2) there are probably good alternatives, albeit not in PHP. I think I made it fairly clear, made a \*reasonable\* contribution to the discussion ... if it's not your favorite, no biggie - just move on and upvote something you like more... /shrug I mean, for what it's worth, it takes \*very little\* google-fu to find [a comparison page](http://en.wikipedia.org/wiki/Comparison\_of\_issue-tracking\_systems) that gives you plenty of comparison criteria, including language (were that really the primary goal).

&gt; I do not think you understand the meaning of "written in PHP" I do not think you understand the point of enumerating "however, it's done in Ruby" (Aka, sorry, not PHP).

But referrers can trivially be faked, no? Better than nothing I suppose.

[Candycane](https://github.com/yandod/candycane) is a Redmine clone in CakePHP

Well, you sounded very cocky and pretentious about automated testing's utility in finding bugs your enum class could help with. And then you were very pretentious to StoneCypher about how he must be \*very\* busy (e.g. wasting time, by your tone) writing unit tests. You \*\*were\*\* misinterpreted, and you should just admit that you could have said things differently, more clearly and concisely and with less vitriol. Hey! If you had done that in the beginning, you wouldn't have wasted the better part of a day attacking StoneCypher.

Excuse me, sir. A number of things: 1. OK, others disagree with me. I have known to be wrong, I also know other people are frequently wrong. That others have differing opinions, or even if I happen to be in the minority, has no determination of if I'm wrong. 2. You quoted Wikipedia for this? Seriously? It's widely known that Wikipedia suffers from anti-professional bias: \*Certified\* pros have had their own professional (and correct) opinions edited/removed/twisted by lay people ad nauseum to the point that most do not bother with the site any more. That's why Wikipedia cannot/should not be cited to prop up one's arguments. 3. \*\*Your rebuke has in no way touched on the reason for including enums in the languages.\*\* It may or may not be true that enums help programmers in certain instances, but if so \*\*the benefit is an epiphenomenon and has nothing to do with the reasons why enums exist in static-typed languages.\*\*

You're right. I was frustrated at this BS and wanted to try to show you that if you want to be pretentious and start waving around figures to seem all high and mighty, that you really ought to pick more respectable statistics. Years of coding mean nothing. I've taught people for a year and they're better than most of the people I've worked with over the years who have been coding for 8 or 9 years. Great coding requires many things, the most important being a non-cocky attitude when it comes to believing we know it all. StoneCypher has a cocky attitude. This is readily apparent. He doesn't, however, have a "I know everythign completely and everyone else is a noob automatically" attitude about coding, as best as I can tell. He is what you can call a Crusader, and like the crusaders in the Middle Ages, he goes a &lt;sarcasm&gt; "little" &lt;/sarcasm&gt; too extreme in his parlays. But I would bet he constantly improves his skills and is willing to admit he's wrong when decisively proven in a calm fashion. But I have a total lack of experimental data to work with towards this hypothesis, so who knows?

Sir, we both know that when I mention the "compilation stage" I am also stating that preprocessing is a part. You're just being overtly pedagogical to the point that rational discussion is impossible. Throw me a bone, come on! # OK check it: "PHP" is basically a JIT virtual machine that doesn't save the opcodes to disk but keeps them constantly in memory. Please see http://stackoverflow.com/questions/3380581/does-php-have-a-virtual-machine-like-java Once you realize this, you will realize you are dancing around the memory issue. If you store the opcodes outside of Zend Core's memory set into APC, you are using \*zero\* memory for hodling constants during execution. See this article: http://imrannazar.com/Memory-Usage-of-Constants-in-PHP

The referer can be faked by individual users wishing to download, but not by other webmasters who hotlink to your content. It's not clear which one he is worried about.

Enumeration: to assign numbers to.

&gt; ...at least, in your opinion ... Tyler Menezes is correct. If I ask you for a red object, and you hand me a blue object, you're not helping.

Please be wary of Flyspray. I used to use it, until twice in a row it imploded and ate its own data. These days I use a tracker which isn't written in PHP, and does not satisfy your requirement (Redmine.) However, trackers that do include GLPI, MantisBT, ZenTrack, BugGenie, ~~Flyspray~~ and phpBugTracker. Mantis appears to be a dead project.

A word to the wise: I used Flyspray until it took a dump on its own database twice. (The DB itself was fine; the app just stored bad data.) After I found and reported the bug, the devs repeatedly closed it without a fix. Consider alternatives.

Yes, you would need to maintain the HTTP\_BASE const (one line of code in the bootstrap file), but as part of the template, I use the the &lt;base href="&lt;?php echo HTTP\_BASE; ?&gt;" /&gt; in the head, then just use relative linking for image, css, js, etc. Works fine in subfolders... BUT, I do routing different than what you are doing in htaccess. I do not send everything to index and let php route to a class method, I just have a few simple "catch all" rewrite rules for nice urls. I do not really like mapping a class method to a URL, doesn't really fit for the type of projects we do at work. The micro-framework does have code separation, but it isn't a true/strict mvc. Basically we have db table wrapper (~ model classes), business logic / display rules (~ controller classes), and views (~ very simple php views with only foreach, if, etc).

I know codeigniter pretty well, then I looked into a symfony app to hack on it and i was like wat. the. fuck. I didn't have time to learn it though.

What was this deleted comment linking to? :(

I wish sort() would return a sorted array rather than returning a boolean.

Mantis [looks very alive to me](https://github.com/mantisbt/mantisbt/commits/master). The latest stable is version is only a week old.

If you think that's bad you should check out the Ruby community. They seem to compete to make their code as compact and cryptic as humanly possible. Cramming several lines of code into 1 complex soup of hieroglyphics. One of the reasons I like PHP &amp; Python is they have less tendency to do this.

\*most\* programming questions involve someone asking for a red object, when they actually need a blue object.

&gt; He may have added something to some discussion somewhere, but it was completely useless to the actual discussion. That sentence is self-contradictory.

Yep. Instead of everyone here argument about the mention of some other language, why not go ask the OP how fixed the idea of PHP is, and why? It may reveal some information that was not presented in the original question that changes everything. Just a case in point - \*I\* was looking for a PHP ticket tracking system a month ago. I wanted PHP because it would run on our servers, and I could make fixes to it easily. In the end we went for a Java solution - JIRA. It still ran on our servers, but is so well-featured and supported, I don't \*need\* to make any code tweaks. Yeah, I wanted red, but the blue turned out to be a better fit in the end.

right backatcha.

And one sure way to induce eye strain.

Dunno, but I just read this: &gt; However, remember that the job of your model layer is not to represent objects but to answer questions. My brain wants models to be objects somehow. You know, if you have a Person table, I instinctively want the Person model to be a representation of a specific instance of a Person. But, it's not; it's stuff about the Person database table.

LOL, PHP is my first and only programing language I learned.

This wasn't a programming question.

I was in error. Thank you for the correction.

&gt; You're just being overtly pedagogical to the point that rational discussion is impossible. Throw me a bone, come on! I feel like I need to say the same to you and that other guy. Listen, I'm sure we've all been using PHP for a very long time now. I've been using it since 2001. I know how PHP defines work, and I know how they're used. Which also why you can skip the lessons on the how PHP and opcodes work. I've read the two books available on writing PHP extensions (And spent plenty of time talking to the author of the one, who goes by "pollita" in ##PHP), and I've written extensions. I fully understand how PHP works under the hood. Now if you re-read my original comment without the "this guy must be a noob" bias, it takes on a different meaning. We both know PHP and the C/C++ pre-processor use a glorified find &amp; replace with defines \\*. Once that's established we can stop arguing about how defines are used, and start focusing on the entire point of this thread: The difference in memory usage of PHP defines and C/C++ macros. The conversation has finally taken that turn with you, and I thank you for that, but I haven't been buying your argument about the C/C++ compiler needing memory for the macros, the same way the PHP run-time uses memory for them, because it's not the same. All that matters is the final product. The final product of a compiled C/C++ is a binary file written in native machine code, and the final product of a PHP application is a script that runs inside the PHP run-time. One uses memory for every macro defined, and the other doesn't. If PHP had a pre-processor just like C/C++ it too could use zero memory for its defines. And believe me, I've tried writing one. Until that day comes, PHP defines are \*not\* the same as C/C++ macros. There is a performance penalty to using them, and I proved that point with my benchmark. Rather than having a productive counter-point to my benchmark, which I would have been more than happy to read, and which you're providing now, he decided to attack my grammar? wtf. Troll. Also like a good programmer I plan for the worse case scenario, not the best. The best case being the end user has an opcode cacher installed. I typically dismiss arguments centered around opcode cachers, like "Well, PHP \*is\* just as fast as X scripting language.... If you have an opcode cacher installed", because planning for the best case scenario is just poor planning, and I think most people only bring out the opcode cach argument when they're backed up against a wall. I also suspect most shared hosting accounts don't have opcode cachers installed. &gt; See this article: [2] http://imrannazar.com/Memory-Usage-of-Constants-in-PHP That was an interesting read. There are a couple things to note: \* Constants in PHP use zvals the same as variables, which if I haven't said it yet, supports my argument that PHP defines are glorified variables that aren't meant to be changed once they're set. \* The author suggests using a class of constants instead of defines, which is \*exactly\* what my enum class does (And so does SplEnum). The only difference is my class (And SplEnum) allow you to create instances of the value. \\* It could be argued that C/C++ macros are much more powerful than PHP defines, because a C/C++ macro can be \*anything\*. Not just scalar values.

&gt; He doesn't, however, have a "I know everythign completely and everyone else is a noob automatically" attitude about coding Haha That's really not true at all. Read some of his other replies. Every other sentence is "If you listen you just might learn something from me". He certainly does seem to think he's the bee's knees. The be-all and end-all of programming. Regardless, I didn't mention my experience to be cocky. I mentioned it so we could move past the noob lectures, and bring the conversation to some common ground. Like your JIT example from a previous thread. We can skip all that, and get right to the heart of the matter. It's like if Joe is a particle physicist, and he meets Sally at a party. At first he may feel the need to explain what a particle is, but once Sally says "I'm a particle physicist too", Joe can drop all that, and the two of them can get right to talking shop. That's all I was trying to do. &gt; But I would bet he constantly improves his skills and is willing to admit he's wrong when decisively proven in a calm fashion. I doubt that, but I may be biased. Like I said in a previous thread, I see a lot of his kind in PHP forums, and in ##php. They come along to prey on noobs, and make them feel stupid. They spend most of their time telling people how little they know, instead of helping them. Again, I may be generalizing.

\*\*Please\*\* make sure you \*\*read the source\*\* before denying anything so venomously. Authors of comments tend to put them there for a very good reason. If the source is wrong (I'm not denying it might be wrong), then it's best to notify the author \*nicely\* (\*\*not\*\* stop perpetuating this nonsense), then place a comment at the source so others more knowledgeable can determine who's right and wrong. If you are indeed right, then authors of comments like me will probably stop linking to that source. From my above cited [source](http://stackoverflow.com/questions/13620/speed-difference-in-using-inline-strings-vs-concatenation-in-php5): function timeFunc($function, $runs){ $times = array(); for ($i = 0; $i &lt; $runs; $i++){ $time = microtime(); call\_user\_func($function); $times[$i] = microtime() - $time; } return array\_sum($times) / $runs; } function Method1(){ $foo = 'some words'; for ($i = 0; $i &lt; 10000; $i++) $t = "these are $foo"; } function Method2(){ $foo = 'some words'; for ($i = 0; $i &lt; 10000; $i++) $t = "these are {$foo}"; } function Method3(){ $foo = 'some words'; for ($i = 0; $i &lt; 10000; $i++) $t = "these are " . $foo; } print timeFunc('Method1', 10) . "\n"; print timeFunc('Method2', 10) . "\n"; print timeFunc('Method3', 10) . "\n"; Results: 0.0035568 0.0035388 0.0025394 &gt; Have a look at phpbench.com which disproves it. It appears to use them only in the context of arrays and doesn't come to a conclusion (it finds both are right): "In today's versions of PHP it looks like this argument has been satisfied on both sides of the line. Lets all join together in harmony in this one!". That certainly isn't a mark of disapproval as you stated. Not to try to pick a fight, but did you even read your own source!? TL;DR - read citied sources completely (including your own), please stop perpetuating undue hate on reddit: there are enough trolls here already.

1. That's fine. 2. I don't really see a problem quoting Wikipedia in this instance. My quote leaves little room for profession bias. The wiki quote is factually correct if even one person believes it's true, and I am one person. 3. The other guy is the one that latched on to the "helps prevent errors" argument. That was one of three reasons I gave for using enums, but it's all he chose to focus on. I pointed out the other reasons enums are used, and showed how their use in PHP is equally valid, but he continued to focus on the error argument. The way I use my enum class in my code is for boxing and un-boxing (I'm using those terms loosely, so lets not start an argument about that) of enum values from other sources that support enums, such as from a database, or another language. I like my data structures in PHP to closely match the structure sent from another source, such as a Java app, or an API. If it's an enum in Java, I want it to be an enum in PHP land. That's my personal preference. I'm sure there are those who would argue "Don't try to make PHP like other langauges. Use it like PHP. Use an array.", but I personally feel it makes my code more organized, and more self documenting.

To be fair, I did try to bring the conversation back to a "normal" level, so we could actually discuss the issue like gentleman. He wasn't interested. Now you could argue, "Why \*would\* he be interested? It's not someone's fault for being defensive when you're yelling at them.", and that's true, but as I pointed out to him, he was the first to be rude and sarcastic, and he set the tone for the rest of the discussion. I will admit I didn't have to fall for it. I could have remained a gentleman, or simply discontinued having the discussion early on. I broke internet etiquette rule #1... Don't feed the trolls. I've learned my lesson.

Less readable: I'll agree with a yes (although it's more pov). Slower with them out: Check your sources, I'd be interested to see any articles comparing this. My [sources](http://stackoverflow.com/questions/13620/speed-difference-in-using-inline-strings-vs-concatenation-in-php5) say the exact opposite.

isn't that... against your religion?

Care to explain?

&gt; Bad habit: Expose public fields Absolutely agreed. And to expand on this, I'm not a fan of when ActiveRecord-style ORMs expose column values through magic \\_\\_get and \\_\\_set. This verily easily leads to a codebase filled with dependencies on the database tables' column names this way. Not only that, but one \*hugely\* common thing that comes up is needing to add a little bit of logic to the process of storing or retrieving a particular value. The magic \\_\\_get &amp; \\_\\_set approach leads to very ugly overridden \\_\\_get and \_\_set methods with screenfulls of switch-statements, when what was needed all along was accessor methods. Edit: added backslashes to stop reddit from interpreting two underscores as formatting

Ironically, I write ruby for a living. It doesn't have to be that bad.

&gt; Bad habit: Not handling errors Today I was messing around with some PHP-based blog aggregation code. While I was setting it up and getting its various dependencies installed, one of the error messages I came up against was this: Fatal error: Call to undefined method MDB2\_Error::setFetchMode() in /usr/lib/notarealpath/something.php on line 1 For the longest time I thought I had the wrong version of MDB2 installed. Turns out the code just completely assumes you'll provide the correct database login details and doesn't even bother checking if you connected successfully. [MFW](http://files.sharenator.com/son\_i\_am\_disappoint\_RE\_OMG\_MUST\_SEE\_WELL\_SEXY-s480x600-47994-580.gif)

Please for the love of god look into Zend Framework / Zend\_Db .

This "article" was straight up stolen from the IBM tech site. Boo. http://www.ibm.com/developerworks/opensource/library/os-php-7oohabits/

Another reason to use encapsulation is when you are doing calculations that have many variables but only one variable is meant to be accessed. An example is calculating physics thrust. You might easily have the following variables. thrustScale thrustIntegrator thrustDecay thrustNoiseGate thrustAmplitude When you first write the class it will remember which variable is one to interface to, however one week later....

He's stuck in the world of not even acknowledging the difference between `public` and `private` and using neither in the process.

yet another amazingly php-centric answer. you can't make this shit up. learn another language, odds are you will end up hating php and use it less.

I tried Ruby on rails for a few months and hated it.

may i ask what, specifically, you hated about it? i ask because i've been a php dev by trade for the past 7 years, have built numerous big apps in cakePHP, and recently made the switch to rails and im never looking back.

And IBM even used a more reasonable type size.

&gt; You're right. &gt; Years of coding mean nothing. &gt; StoneCypher has a cocky attitude. This is readily apparent. He doesn't, however, have a "I know everythign completely and everyone else is a noob automatically" attitude about coding, as best as I can tell. These quotes are the reasons why this comment received an upvote from me.

&gt; Read some of his other replies. Every other sentence is "If you listen you just might learn something from me" Despite my reply to the grandparent comment (the one you're replying to), this is also \*very\* fucking true.

Looking at the other articles on the site, it seems that this is all they do. They copy and paste the content of other sites onto their own. As such, I've removed the post. I'll do some sleuthing on the OP and see if this is their site, and if so, it may result in a ban.

It only becomes true once you've established the credentials of the person making the statement.

Unless you use some apache fu to cover it up: &lt;FilesMatch "\.inc$"&gt; Order allow,deny Deny from all &lt;/FilesMatch&gt; Of course, other web servers have ways of denying access as well.

Example yes, but it's still perpetuation a bad habit. 95% of the time I see "or" used is when it's used in "or die()" To be honest, I've only seen it used otherwise maybe once or twice.

I would argue that it's more readable with them out. It's much easier to spot even w/o syntax highlighting. Add some highlighting to the equation and now your editor is going to show that variable in a different colour when most likely the variables inside a string will not be. But like you said, that's POV. And yes, he was completely wrong about it being faster when in quotes. It can be easily confirmed to be slower.

It's more important to do something right than to fit it onto a single line. if (file\_exists('path/to/file') { // Do something } else { // Do something else } is significantly better than simply suppressing an error because it lets you handle the error. If you just suppress it, you don't know whether it worked or not.

"I don't consider myself by any means a web developer" "Currently I'm building a website" My professional response is to hire a professional. My r/php response is to start by clearly defining your business logic (which ads to show to whom, how often to repeat ads, etc), then building the app around that definition. You'll likely need to use some sort of storage and a db would probably be the best choice for speed and scalability. If you use a table as you describe you will wind up with a very large amount of rows pretty quickly. While this isn't a terrible problem, it is one that you will need to address when designing your app.

Check the year on your source. This has not been true for a VERY long time - in fact the opposite is true nowadays.

&gt; To be honest, I've only seen it used otherwise maybe once or twice. Which is exactly my point. It needs to be used more often because it's a very useful feature.

Does the &amp;$datum live past the scope of the foreach loop? Also how would one go about cleaning up leftover refrences? unset()? Ive probably completely overlooked this in the past.

Yes, it will live past the foreach loop. Unset() will remove the reference. For example: &lt;?php $array = array(1, 2, 3); foreach ($array as &amp;$value) { echo "$value&lt;br&gt;\n"; } $value = 5; var\_dump($array); unset($value); $value = 6; var\_dump($array); ?&gt; Output: 1 2 3 array(3) { [0]=&gt; int(1) [1]=&gt; int(2) [2]=&gt; &amp;int(5) } array(3) { [0]=&gt; int(1) [1]=&gt; int(2) [2]=&gt; int(5) } Unset() only completely removes a value if the variable being unset contains the last available reference to that value. In this case, there are two references to the value, so removing the one attached to the variable $value doesn't hurt the array. Note the &amp; in the first var\_dump() output. This means that there are other references attached to the same value.

This is common, but also very, very terrible. You should never handle errors in this way.

In the docs, it says: float round ( float $val // some other crap) Keep an eye out in the docs for functions that do this (watch for the &amp; symbol): float round ( float &amp;$val) That means it's passing the value by reference. The only reason some PHP functions do that is for performance. Let's say you want to sort a gigantic array... unless you pass it by reference into PHP's super fast array sort functions, it will make a copy of the array then overwrite it rather than simply sorting the array. Doing stuff like this: sort($array); Is much faster than: $array = sort($array); But for silly little functions like float(), passing by reference needlessly complicates things for a questionable performance benefits. IMHO, passing by reference is always a bad idea unless you need SPEED.

After watching the wire recently, the first thing I thought about when I saw the title of this post was about the drug game haha

The only reason they do this is because: sort($array); is faster than: $array = sort($array); Because the later would have to sort the array and then reassign $array... the former only needs to sort the array, and the reassignment doesn't need to happen because it was passed by reference. I agree, I hate it.. but it's a performance gain that can't really otherwise be achieved in any other way.

The junior devs at my work usually start out doing "bits and pieces" type tasks. Like, adding a widget to a sidebar, plugging in an API here and there, or throwing together a simple theme. Something I would recommend you do is to start mastering some frameworks like CodeIgniter, CMSs like Wordpress, and APIs like Facebook and Google. These are good things to have on an application that really mean something, and should be pretty easy for you given you already have PHP skills. I would also recommend brushing up on your HTML/CSS; it's not absolutely necessary, but will definitely help a great deal. It's also a good idea to learn some basic photoshop techniques for resizing, compressing and slicing images. Having said all that, the #1 ability employers want in a developer is someone who is willing and able to learn. So long as you don't outright lie in your application, and mention that you always seek to use best practices etc etc. that's a good start. The biggest fear of an employer (or fellow developer coworker for that matter) is someone who doesn't learn from mistakes. THey don't care if you're not a gun dev (unless you sell yourself as such), but they do want you to be the sort of developer that asks themselves "is this the best way to do this?" every step of the way, rather than cobbling together something with duct tape to kick it out the door. EDIT: good luck by the way! :)

well in the time i've taught myself i've learned quite a lot but I've never went to school professionally for it. I'm not a "chump" when it comes to db management and php though and this website is something I cofounded so hiring a professional is something I'm not doing. I appreciate your feedback but perhaps I didn't make the question clear enough

Now rereading this I can see how I did sound like I didn't know what I was doing. To put it another way, I've been learning PHP over the last 6 years but never dug deep into OOP and other things like that until recently, that's why I don't consider myself a professional web designer. Not because I'm like 12

No, you do not create individual tables for each user. You attach a user ID line to each row, and query based on that. Personally, I'd just take a completely random ad every time, though. It'll average out - if it's not, you're doing something wrong.

thank you. yeah i was thinking that one large table would be better than one large db. it was just a thought. my advertisers do have an option though. they can choose to advertise to a selected target (tenants/landlords/agents) and select a target area (nationwide/statewide/city within a certain radius) I guess my problem lies within making sure my users and advertisers are happy

You could trying doing some freelancing... You can build a portfolio like this while learning more front-end development which would eventually help you in getting a job.

User not logged in: pick a random ad User logged in and has state set: pick a random ad from that state or nationwide (something like SELECT \* FROM ads WHERE (type='statewide' AND state='state') OR type='nationwide' ORDER BY RAND() LIMIT 1) User logged in and has state and city set: pick a random ad from that city or state or nationwide User logged in and doesn't have either set: pick a random ad There isn't really a need for anything more complicated than that, IMHO.

i'm going to give this a shot. seems simple enough. thank you again

&lt;?php if ($omar-&gt;isComing()) { $ghetto-&gt;population = array\_map('runAway', $cornerKids); sssshhhhheeeeeeeiiiiiiiiittt(); } ?&gt;

If I could upvote that more than once I would haha

&gt; \*\*in my opinion\*\* cakePHP is a bad choice FTFY

Also, I am interested in what would be the best way to find companies hiring devs? There doesn't seem to be many advertising... How might I be able to find companies in my area to whom I can perhaps send letters of interest? Thank you for the assistance so far !

Bad PHP is easy. Good PHP can be easy if you know good practices, but if someone has been self-taught it's likely they've got a number of bad practices ingrained (I know I do) and it can be a struggle to unlearn them and learn the correct way of doing things.

Within reddit, try /r/forhire. There's also Stack Overflow's [Careers](http://careers.stackoverflow.com/jobs/tag/php) site.

Log which ads get clicked on in a manner that lends itself to building an expected click through rate model to refine the "randomness" you're still using.

Never ORDER BY RAND with large tables

True. Unfortunately, I haven't touched a database in a while, so any hints on how to do this properly?

Try this - [code sample](http://media.pragprog.com/titles/bksqla/random.pdf) taken from Bill Karwin's excellent book SQL Antipatterns: &lt;?php $rand = "SELECT ROUND(RAND() \* (SELECT COUNT(\*) FROM Bugs))"; $offset = $pdo-&gt;query($rand)-&gt;fetch(PDO::FETCH\_ASSOC); $sql = "SELECT \* FROM Bugs LIMIT 1 OFFSET :offset"; $stmt = $pdo-&gt;prepare($sql); $stmt-&gt;execute( $offset ); $rand\_bug = $stmt-&gt;fetch();

Getting something happening in PHP is easy / fast / excellent. Maintenance can be a bit of a chore, but that's part of the trade off

That seems like an interesting book, I might get a copy of it. Thanks!

great idea thank you. i have an amateur style stats db right now that's only tracking things like logins/logouts, which header links are being used and on what pages, redirects to off page links, and some ad stats. I'll be sure that add that one in

/sighs Enumerate: [to ascertain, as-if to count in a list](http://dictionary.reference.com/browse/enumerate).

This is a long shot but if you happen to live in the south of the Netherlands you might be just what my company is looking for.

such bad advise

It's not clear to me how you think that by that definition you would be any less incorrect.

Sorry if I came across like an asshole, but please note the key weasel words in my response: \*\*tangible, immeasurably small difference\*\* Firstly, your "faster", "fastest" examples are wrong, the performance of the two is identical (did you try it?). The source doesn't make this actual comparison, it fails to compare single quotes vs double quotes for strings that don't contain variables. Benchmark it, the results are exactly the same. Secondly, let's focus on the actual performance increases you'd get from say, refactoring a method that echos 10000 variable-encrusted-double-quoted-strings to use single quotes. echoing the double quotes takes: 0.0025880336761475 seconds echoing the single quotes takes: 0.0020289421081543 seconds Wow, 28% faster I hear you scream, that must be worth the refactoring effort! Except in real terms that performance difference is absolutely completely utterly unnoticeable to anyone, it won't make the slightest difference to performance of your application, there will be a million other things that are worth optimizing before this is. Optimising your database, caching layer, templating system, css, js and business logic will produce performance gains orders of magnitude more significant than this, it is a total waste of time to change existing code to fit this pattern. TLDR. If you are in a situation where this is making a \*\*tangible\*\* difference to site performance or server load, congratulations, you might be facebook, I believe you have a different solution: https://github.com/facebook/hiphop-php/wiki/

My advice would be to just jump in! The only prerequisite I can think of is understanding namespaces, and that only comes into play if you're dealing with modules or packages.

Where are you? The company I work for (Virginia) has positions open for Jr Level PHP developers.

Sort of relevant to your comment and the OP's question: I'm kind of transitioning from frontend work to development as the months go by. The other day, I asked the devs at work what I was doing right and what I was doing wrong, why I'm good and why I'm bad. The good things are basically what you mentioned. My familiarity with the frontend results in some choices made earlier on that save a bit of time, I like a challenge rather than avoid it, learn from mistakes, ask myself what the right/best way to do something is... Being willing to fight through a problem and really invest your energy in solving difficult things seems really desirable. It shows there is more happening than just trying to get the job done - You have a personal interest in learning and doing it well. What I'm bad at is due to my lack of experience, primarily. Sometimes I take too long on things that are okay to be half-assed for the first iteration, and I have to spend a while in the docs at times. I ned to realize when and when not to dwell on certain bits of code. Anyone can work through that though, and OP already has a leg up on people like me due to a history with php and some probably still relevant schooling. I'd say keep at it, focus on strengths outside of just development during interviews, and continue coding whenever you have the time. It's the best way to get better.

Taking a casual glance at this: 1) Switch to a DB abstraction, or at least upgrade to mysqli and use prepared statements. Using the MySQL php library now is fairly looked down upon. 2) Right at the index.php: $con = mysql\_connect($db\_host, $db\_user, $db\_pass) or die('Could not connect to the database: '.mysql\_error()); No! That's terrible error handling -- who knows what mysql\_error() is going to output, and whatever it is, it's not going to be something I, as a user, want showing up on my site. 3) Alignment tricks: $title = mysql\_result(mysql\_query("SELECT \* FROM dnp\_settings WHERE name='title'"), 0, 'value'); $description = mysql\_result(mysql\_query("SELECT \* FROM dnp\_settings WHERE name='description'"), 0, 'value'); $analytics = mysql\_result(mysql\_query("SELECT \* FROM dnp\_settings WHERE name='analytics'"), 0, 'value'); $email = mysql\_result(mysql\_query("SELECT \* FROM dnp\_settings WHERE name='email'"), 0, 'value'); There's two issues here: A) You are running 4 separate queries to fetch what could be pulled out in a single query B) You are using tabs to get those = signs to line up. I, personally, like aligning stuff like this, but you should use spaces, not tabs. Most \*sane\* editors are 4 spaces to a tab, but you just never know. 4) You're going to get some people bitching about not using MVC -- ignore it. For a small project like this, it'd probably be overkill. It's nice to separate logic and view, but for a script this small and simple, it's just going to be overhead you don't really need. That said, if you're interested in learning MVC, a small script like this would be pretty quick to patch together. 5) &lt;script&gt; tags should go right before the closing &lt;/body&gt; tag for performance. Google that up for more information why. 6) Mailto links giving out email addresses is just evil for people using this. Create a proper "Contact me about this domain" page which has PHP send an email instead. Much more secure. Also, trying to obfuscate an email address by change @ to [AT] is going to fail pretty quickly. Just don't put the email on the page. \*\*Feature Wise\*\* In the admin you make people enter the registrar. With some parsing and nslookup action, you ought to be able to figure that out. More complex, but would be a neat feature. More problematic, however, is the "asking price". If I have a domain, I want people to make me offers, not set a fixed rate. Sure, setting a fixed price is a neat idea, but I'd much rather leave the asking price field empty and have it say "Make me an offer" as opposed to "Not for sale".

// Start the session session\_start(); Is this line commented out or just looks that way on Reddit?

nope not commented out, apologies that reddit alters some of the formatting - that's not commented out, nor are any of the other lines that have a comment then a command. for some reason the carriage returns didn't come across.

I finally implemented it, it was very easy and just what i needed, thanks again!

Not PHP specific, but bit fields/flags really helped me out with procedural programming, ie. for Drupal modules. I should probably feel bad for not realizing their possible usages earlier. db\_query("SELECT uid FROM users WHERE gender &amp; $user-&gt;gender\_preference"); // male = 1; female = 2; any = 3 $status = file\_prepare\_directory($location, FILE\_MODIFY\_PERMISSIONS | FILE\_CREATE\_DIRECTORY); if($user-&gt;fb\_connect\_status &amp; FB\_CONNECT\_PERMISSION\_CHECK\_PASSED) { //...

Make sure your not forwarding to a different subdomain. For example if sign into http://www.mydomain.com/login.php but your header is forwarding to http://mydomain.com/home.php the session is no longer valid.

[php.net](php.net) is going to become your best friend. [http://php.net/manual/en/function.set-error-handler.php](http://php.net/manual/en/function.set-error-handler.php) [http://php.net/manual/en/ref.errorfunc.php](http://php.net/manual/en/ref.errorfunc.php) Error handling is pretty easy in PHP. Also, tabs vs spaces is just personal preference. I, for example, prefer tabs while chuy likes spaces, so just use whatever you want. All in all though, good start and you're on your way. Chuy also did you a good service by providing some ridiculously thorough feedback.

Scotland :)

I am not entirely sure, but if I’m correct, this: $dbc = mysqli\_connect(DB\_HOST, DB\_USER, DB\_PASSWORD, DB\_NAME) or die('Error Connecting to Database.'); will cause \_$dbc\_ to have the value \_true\_ so everything database related will fail. Otherwise, when you say it doesn’t work, what does it do \*\*exactly\*\*?

Would also like to put [this (php exceptions) here](http://php.net/manual/en/language.exceptions.php) here

When I hit the submit button, it calls login.php, but it just goes to login.php - doesn't appear to have executed any of the database related code, or checked against the users inputted email/password. When I inspect the page source to see what it's echo-ing, it's echoing a blank error message - so it's basically just ignoring all of the code above that last bit: // If the session var is empty, show any error message and the log-in form; otherwise confirm the log-in if (empty($\_SESSION['user\_id'])) { echo '&lt;p class="error"&gt;' . $error\_msg . '&lt;/p&gt;'; } else { // Confirm the successful log-in echo('&lt;p class="login"&gt;You are logged in as ' . $\_SESSION['email'] . '.&lt;/p&gt;'); } And clearly we weren't successfully logged in - but it's not throwing any error message. Thoughts?

Have you tried removing the or die() like I suggested? also, add this at the beginning of the login.php error\_reporting(E\_ALL); ini\_set("display\_errors", 1);

Indent each line with four spaces to make it a code block.

Lol, don't be ashamed of how much you reference a site. I use it on a daily basis just because across all the languages I have to use for my job, it's sometimes difficult remembering what functions I need.

pfft, you \*totally\* stole my karma. :P

It doesn't actually tell me what it does :S An "About"/"What does it do?" section would be much appreciated.

Really cool contest, I hope that I'm considered for the prize. I've submitted an entry; openSpree (an adaptation of the board game Spree!). It was incomplete at the first deadline, but when the extension was announced I got to polish it off. Looking at the other entries, I think mine is the most comprehensive application that was developed (not adapted) explicitly for the contest. Even if it doesn't win, I'm proud of the progress I made in five days and it got me in touch with James Ernest (the author of Spree) - a cool guy. \* Source - https://github.com/fluxsauce/OpenSpree \* Demo - http://openspree.orchestra.io/

Or remembering whether the needle or the haystack comes first :-)

RIGHT? lol

I use it everyday at my job and I'm a full time php developer. If anyone gives you shit, they're dumb.

I see, I have some family in Glasgow. But none in the tech industry.

In can be played with via [github](https://github.com/php/php-src) and/or [git.php.net](http://git.php.net/) and/or [hg.php.net](http://hg.php.net/) today. And let's not forget about the current [svn](http://svn.php.net/) and old [cvs](http://cvsold.php.net/). Actually, the [DVCS RFC at php.net](https://wiki.php.net/rfc/dvcs) is most relavent to this topic.

my guess is that its checking your 1st if statement and failing i would recommend putting name="signin\_submit" inside your input tag in your navigation .php when i cant figure out a problem i do ghetto debugging and put echo '1'; echo '2' echo '3'; in different areas to see where its failing.. anyways let me know if that helped

No, the assignment operator has higher precedence than or, unlike ||.

I do pretty good with that except for in\_array()... for some reason, I can never remember the order for in\_array().

I think the people downvoting you either didn't look at your username, or don't get sarcasm.

The only way is to run the amazon s3 sample php code provided. make sure you put your keys correctly in the sample and run with the same filename and try. 100% sure that there is an issue with the signature request.

Most of your arguement has to do with this comment of yours, so I'm going to quickly address that out of order: &gt; it is a total waste of time to change existing code to fit this pattern. I agree with you and I don't remember me \*ever\* advocating re-writing whole webapps. You probably confused what I said and included my words with some of the other child threads to this comment. It was mentioned [here](http://www.reddit.com/r/PHP/comments/kgzm3/whats\_your\_favorite\_php\_shortcutsecret/c2kemhk) and [here](http://www.reddit.com/r/PHP/comments/kgzm3/whats\_your\_favorite\_php\_shortcutsecret/c2kb8de) that some people explicitly strip variables out while others put them back in. I personally agree with this [comment](http://www.reddit.com/r/PHP/comments/kgzm3/whats\_your\_favorite\_php\_shortcutsecret/c2kf1bj) rather than those two. &gt; The source doesn't make this actual comparison, it fails to compare single quotes vs double quotes for strings that don't contain variables. True. Here's a Method4() function Method4(){ $foo = 'some words'; for ($i = 0; $i &lt; 10000; $i++) $t = 'these are ' . $foo; } Here are the results of the code in my previous comment and the above additional function: 0.0019338 0.0019618 0.0013836 0.0013816 Just so everyone's on the same page: Method 1 -&gt; \*Slow\* $foo = "This is $bar"; Method 2 -&gt; N/A Method 3 -&gt; \*Faster\* $foo = "This is " . $bar; Method 4 -&gt; \*Fastest\* $foo = 'This is ' . $bar; I'd be happy to benchmark any of your code. Maybe there's a reason that double quoted and single quoted strings are being handled differently on your end? &gt; Optimising your database, caching layer, templating system, css, js I get what you're trying to say by referencing PHP accelerators, that most of my arguement is moot (which it is in that circumstance). I think they are all great methods \*that should be used\* in a website environment. However, in my eyes I always view PHP as a full fledged programming language and not just a "hypertext preprocessor". For instance, it is just as capable in the command line as a replacement for shell scripts. Checkout [m0n0wall](http://m0n0.ch/wall/) where PHP is soley used in its init system instead of shell scripts. In this case maybe we can agree that small differences in speed are very important? ---- As a final note, it's just easier to use single quotes: " -&gt; Shift + ' key ' -&gt; ' key :-)

&gt; Check the year on your source. 2008, so 3 years old. PHP5 was released in 2004, since then major revisions have mostly just been bug and security fixes. Is there a version number you're referencing? &gt; This has not been true for a VERY long time - in fact the opposite is true nowadays. ORLY? [Benchmark](http://www.reddit.com/r/PHP/comments/kgzm3/whats\_your\_favorite\_php\_shortcutsecret/c2kz950) on my system, PHP 5.3.8

I don't want to reveal too much personal/work info. But I do work for small technology company (only 15 people all the way up to the owner). I've helped create a SaaS product, created a DRM server for locking down our software (its $8,000 per license, this is a different product from the SaaS product). I run all of our linux infrastructure, help design new project architecture, extend/build API's, build/maintain product websites. I don't want to give away my product/industry, but I'm really fortunate that I have digital high def cable run to my cubical. I get to play with new computers/tablets/smart phones daily. Also, I don't have a bachelor degree. I worked my ass off to get where I am. I would suggest that you learn linux if you want to be more valuable to your employer. Being the only linux guy in a windows shop is job security :)

To begin with, make sure you have error\_reporting at a suitable level (during development only!\*) to catch all errors. Add error\_reporting(E\_ALL | E\_STRICT); to the top of your document and see if any errors appear then. \\* If `display\_errors` is off and you're logging errors somehow, you still want `E\_ALL | E\_STRICT` in production.

What's wrong with sha1($password . $salt)? Why is that so hard?

So many upvotes and nary a comment is suspicious...

Not sure where you are based but at work for clients we use: http://www.onesoon.co.uk Then it's just a simple CURL request with the username, password, sender (you can make it appear as anyone on their phone! (e.g. "Business Name")), message and recipient

It depends on your provider, I'd say. I had read a neat article a few years ago (or at least a long time ago to me) about how someone had set up a cron job that checked for emails from a certain address, and then parsed it for commands. At the time, I had Verizon, which assigns an email to each number [(###)###-####@vzwsomethingorother.com]. So what I'd do is send an email to the server I ended up setting this cron job on from the number I had and it would execute, say, 'uptime' and email it back. It would go to Verizon, and then my cell phone as a text. In other words, try sending a text to your email, and if it comes out with an address, use that and set up some kind of cron job on your server.

hm receiving is tough... for me at least, I've never needed/tried it. This is a little hacky, but it works great for sending... I made a list of all of the service provider email addresses I could find, and I blast out an email to all of them at once ;D function sendSMS($number,$m){ $to = "$number@message.alltel.com,"; $to .= "$number@vtext.com,"; $to .= "$number@mms.att.net,"; $to .= "$number@messaging.nextel.com,"; $to .= "$number@messaging.sprintpcs.com,"; $to .= "$number@tms.suncom.com,"; $to .= "$number@tmomail.net,"; $to .= "$number@voicestream.net,"; $to .= "$number@mms.mycricket.com,"; $to .= "$number@mymetropcs.com,"; $to .= "$number@vmobl.com,"; $to .= "$number@cwemail.com,"; $to .= "$number@csouth1.com,"; $to .= "$number@gocbw.com,"; $to .= "$number@myboostmobile.com,"; $to .= "$number@qwestmp.com'; $subject = 'Whatever'; mail($to, $subject, $m); } I haven't used that in a while.. but it worked like a charm when I needed it. Better than paying someone else to figure out who the service provider is anyway...

&gt; I had read a neat article a few years ago (or at least a long time ago to me) about how someone had set up a cron job that checked for emails from a certain address, and then parsed it for commands. I remember this too! God that was scary.

At work we use http://www.sms2email.com/ You send an email &lt;number&gt;@sms2email.com from an authorised email address, and they send the sms for you. You can even customise the "From" field of the sms, so you can make it look like its from anything.. including plaintext: From: YourCompanyName

im actually researching sms gateways for work and i found that a couple of them have free services. http://www.zeepmobile.com/ http://www.freesmsapi.com/ https://www.tropo.com/

Twilio is usually the default internet-to-phone service to try, since their API is so quick to learn and they have easy [PHP tutorials](http://www.twilio.com/docs/howto/sms-notifications-and-alerts), but [Tropo](https://www.tropo.com/pricing/) has a free developer account if you want to try them first.

Checkout Amazon SNS/SQS, it's cheap and might work for you.

I've been playing around with www.txtlocal.co.uk. They give you ten SMS messages for free when you register which allows you to set up your pages properly before you need to start forking out the cash. 100 messages is £4.90.

Just by glancing over the code I agree, this will never be true: if (isset($\_POST['signin\_submit'])) {

Agreed. MVC has limitations on the web and wasn't originally intended for web-style requests, esp. in languages like PHP that execute top-down and end at the end of a request. I don't think MVC needs to be taken so literally, and that we can simplify things greatly for the developer by abstracting some of it away. MVC can be expressed more succinctly in Ruby and other languages, but it gets hairy pretty quickly in PHP's less elegant syntax. That's why I created a new framework that does this by trying to be as transparent as possible: http://www.elefantcms.com/ (includes an admin UI for end users, but that's just a couple "apps" built on top of the framework) Some posts on MVC's challenges: \* http://wardley.org/computers/web/mvc.html \* http://www.sitepoint.com/mvc-and-web-apps-oil-and-water/ \* http://www.phpwact.org/pattern/model\_view\_controller#drawbacks\_of\_mvc

Is this useful at all? [Send Text Messages with PHP](http://davidwalsh.name/send-text-message), David Walsh Blog. edit: Apologies, I forgot this doesn't cover receiving at all.

Problem with Twilio is that they don't do international. At least they stopped doing it once my application depended on it; maybe they've started again recently.

Where abouts? We advertised a php job (edinburgh) last year and it took us a long time to fill due to lack of candidates, so there's definitely some demand and not much supply. I hear glasgow companies have a harder time hiring good devs. If you're in edin there's a few digital agencies you could fire off your cv to: Line Digital / Whitespace / Blonde / Evolution website design. They'd be a good starting point, maybe not the best place to work, but they have a reasonably high turnaround so more likely to get a job. If you're not in edin, glasgow or dundee, you're probably not going to have much luck I'm afraid. Protips: Learn Javascript/jquery/etc, this will make you much more valuable. A lot of companies use MVC frameworks, learn one of them too. Learn a CMS, such as drupal. Maybe learn an ecommerce package also. Don't put too much faith into your degree, they mean virtually nothing when it comes to hiring php devs. Get a github account and publish some fun side projects, this will make up for lack of portfolio. good luck :)

Doesn't look like your typical MVC, but that's what I'm going for here: https://github.com/jbroadway/elefant

The company I work for (based in the UK, develop PHP-based e-commerce platforms) use [MediaBurst](http://www.mediaburst.co.uk/api/) which you have to pay for ~5p per SMS.

I hope you're not sending sensitive information over these SMS messages. Every one of those companies could potentially accept and view those messages if they want.

Nope, just texts that say "You've been invited to play some (location) disc golf at (time)" - I don't use email for sensitive info, I like face-to-face.

yeah i'm playing with tropo now. but isn't it just a fancy frontend for google voice on the backend? don't need international. and don't need this to be something for multiple users. this is just a personal project. and yes i could do the sms by email address, but that's ghey.

Unless their website is out of date, [Amazon SNS](http://aws.amazon.com/sns/#details) doesn't support SMS yet: &gt; additional delivery transport protocols such as SMS are expected to be added in the future.

Why do you think it has anything to do with Google Voice? Seems like an odd assumption to make.

Sorry, didn't verify before posting, check this out: http://nexmo.com/

Alternatively, you can use your providers sms/email gateway to send information from your program to your phone. I made a script that allowed me to search torrent sites (and begin downloads remotely), get answers from answers.com, and howto stuff from wikihow all through text messages. Just set up an hmailserver, or another email server that uses SQL to store email information. It makes it very easy to tap into with your program, so it can parse the message and reply accordingly.

Is there something wrong with requesting the carrier from the user and using "##########@carrier.com"?

A reddit formatting note: if you indent the code lines with 4 spaces, it formats them properly as code and doesn't join lines together as is currently occurring.

well the user is just me. and that's sooooo 2001.

http://www.php.net/manual/en/dateperiod.construct.php#100730 Basically, the last day of the period is excluded.

I think I recall the signatures being time based, correct? Have you changed anything on your server regarding time? Have you added any PHP scripts that change the default time zone? Just a shot in the dark here, but worth looking into.

You are correct. The range will not include the last day as the range is seen as "from this start date, up to but not including this end date" The easiest way to make this do what you want is to do: $end-&gt;add(new DateInterval('P1D')); directly after you define $end. Do not just add one to the total count, as the day could possibly land on a weekend.

I'm actually asking b/c I have an online application that sends out texts to multiple users. I'd be interested if there's an actual reason why I shouldn't be doing this...

Awesome package but it's dependent on FuelPHP and not everyone uses that framework.

The same applies to sending any SMS/email to any provider.

well if all you want to do is broadcast to multiple phone numbers, then how you have it is fine. but, what happens when an end user changes carriers? they'd have to update their profile, or they stop getting messages. who wants to rely on clueless users to do that? i'm looking for 2-way communication. my app i can actually send my system commands and have it execute on the server then send a text response back. while i could do that thru an email address and a parser, i just don't want to.

&gt; Alternatively, you can use your providers sms/email gateway to send information from your program to your phone This is something that largely exists only in the USA and a handful of other countries, where the recipient pays for the SMS.

I use git with multiple branches and tags along the lines of this model: [A successful git branching model](http://nvie.com/posts/a-successful-git-branching-model/). I do freelance work, and I tend to follow an iterative development cycle, with sequential numbers. Every time I send a new iteration to the client for review I merge to master and tag it with the number of the iteration. Works for me, and makes it easy for me to refer back to the different iterations to see what was done in each.

Lol

git or hg

Personally I like Mercurial. Checkout Bitbucket.

Just because you don't understand it, doesn't make it incorrect... it only means \*you're\* not familiar with the particular context or application. Put bluntly, enumeration (in this context) is a process of discovery or, really, of "listing issues." It's often most generally used in the network and/or security field.

git is made for what your needs in this case.

I wasn't sure if OP was asking about version control or versioning strategy, but I liked that you covered both. Nice link by the way. And OP, once you get the hang of using version control you will never look back, that's a promise.

Git is the popular choice. I don't care for it though. For me, Mercurial is much more user-friendly, other than that there's really very little difference. I also like Bazaar, but it might not be the best. I'd suggest that you install Git and Mercurial, use both of them for a bit and decide which one you like better.

Generally only providers in the USA provide those types of email addresses. We don't get them in Australia for instance, there's no free "email to SMS" gateways here :( If your site is restricted to the USA then it might work well enough. But please remember the Internet is global! I get so frustrated every time I see a "US-only" site. &lt;\_&lt;

git - it works great, especially for collaborative development (hg is pretty similar, so consider it as well). I've been using the "git flow" model on one of my recent projects and it seems to be working well: http://nvie.com/posts/a-successful-git-branching-model/

Svn, moving to git in the near future. No such thing as consensus on versioning tools, hell we can't even agree on what notepad replacement to use.

vim! &lt;flamewar&gt;

The awesome book "Multitool Linux" has a chapter on doing that.

Okay I'll play along. EMACS YOU HEATHEN! RAAAAR

&gt; Just because you don't understand it, doesn't make it incorrect... No. What makes it incorrect is that the definition you quoted doesn't fill into the sentence you're claiming. &gt; I do not think you understand the point of [ascertaining] "however, it's done in Ruby" (Aka, sorry, not PHP). Le cough. &gt; Put bluntly, enumeration (in this context) is a process of discovery Er. No, it isn't. Just because you've found a half-sentence in a dictionary doesn't mean you're suddenly using a word correctly. Nobody has ascertained anything here by counting it as in a list. You just stated that you were answering a question for a PHP tool with a Ruby tool. No enumeration. No numbers. No ascertaining. No counting as if in a list. It's just you using words you don't quite understand. &gt; or, really, of "listing issues." But you didn't do that either. &gt; It's often most generally used in the network and/or security field. Given that that's where I work, I can tell you quite certainly that no, it and/or is not.

They came out with 1.3 which seems to include everything I need now. Yes!

If you're doing anything with volume, you'll quickly get banned by most of those carriers. Source: experience.

git

Everyone here seems to be talking about source control. I'm not sure if that's what you meant by versioning, so I'll offer two answers. For source control, I use git. For version numbering, I use [semantic versioning](http://semver.org/).

interesting.. thanks for that heads up! I didn't realize that was a possibility.

I like nano...

Here's a quick solution: function count\_days\_between\_dates($start,$end,$format='Y-m-d',$skip\_weekends=true){ # Populate an array of days between any two given dates $dt = array($start,$end); $count = 0; while(strtotime($dt[1]) &gt;= strtotime("+".$count." day",strtotime($dt[0]))){ $relativedate = strtotime("+".$count++." day",strtotime($dt[0])); if($skip\_weekends &amp;&amp; (date('N',$relativedate) == 6 or date('N',$relativedate) == 7)) continue; $dates[] = date($format,$relativedate); } return $dates; } Usage: count\_days\_between\_dates('2011-09-01','2011-09-30');

"Hey everyone, check out the noob!" ;)

Feedback very welcome and appreciated! Very much a first draft and plenty of room for adding additional options / functionality / error handling. I couldn't find anything halfway decent that I could just drop in and would work for google checkout, so I decided to roll something for my own use. I'm not even sure what else needs to be done to complete a checkout, but I thought I'd commit a rough draft to see if I got any helpful feedback sooner rather than later. It would be really awesome if someone more familiar with the google checkout process would chime in and give me some pointers. The example code they give seems even more verbose than their documentation for how to do this stuff, so I've just been walking through this without much else to go by.

An optional `$skip\_weekends` can be set to false if you want to include weekends. Also you can use `count()` to simply count the amount of days between specified dates.

I'm so 1337 I don't even need a text editor. I just think and the PHP writes itself.

This fake argument is awkwardly sincere.

Looks good :) documentation is good, everything looks logical and well thought-out coding wise.

Twilio is great, we've had good luck with it and the API is really easy. Though they don't have a free developer account, they do give you a $30 account credit to start with. At $0.01 per SMS that'll last quite a while...

It's a very specific scenario but Google Voice does not have a emil-to-SMS gateway for an SMS email address. Clarification: They sort of do but only for having it show up in your GMail box if you have the option selected. Even then it's like &lt;your number&gt;+&lt;the other persons number&gt;+&lt;random code&gt;@txt.voice.google.com

for the solution pile: function iamthewhiteman(DateTime $start, DateTime $end) { $weekdays = 0; $tmp = clone($start); while ($tmp &lt;= $end) { if (!in\_array($tmp-&gt;format('w'), array('0', '6'))) { $weekdays++; } $tmp-&gt;modify('1 day'); } return $weekdays; }

&gt; I just think and the Ruby writes itself. FTFY

Thanks, I will try to remember that.

This seems to have the effect I needed. I was worried about this missing day because the total amount is used in other calculations.

The time for version control was long ago - like, when you created the first file. I used to just edit files live using FTP, and that gets really messy. A VCS provides backups, history, statistics, better collaboration, an easy way to ditch changes, an easy way to find what introduced a bug, and some more things that I can't even think of right now. So, moral of the story: I'm glad that you're doing it now (yay!), but for the future, start much, \*much\* earlier.

Folders.

git

That's perfect clarification, thanks. Right now it's US based jut because we're starting out. I'll keep this in mind as we grow though.

I think asm and I multiply the fish and bread

I like git. Its far better than svn in many aspects. Also I have taken a look into mercurial, simply because [Bitbucket.org](http://bitbucket.org) gives five private repositories for free.

Personally, I'd load up jquery ui tabs http://jqueryui.com/demos/tabs/ It's kind of heavy just for tabs, but it'll give you what you're asking for. If you want to skip the whole ui thing, you could have something like a list of tabs ( a ul list) and make them show up in a row (display inline), and then a set of divs all set to display:none, then when you click on one of the ul-&gt;li items it triggers a toggle() on the right div. I'd use jquery for that meself (as shown by the toggle()) so I didn't have to track which browser got which sorts of elementById() reference. Huh, that dissolved into gobbledegook.

Clickatell seems to be one of the most popular SMS gateways, I've never used it myself though.

&lt;? if(isset($\_GET['contacts'])) { ?&gt; &lt;div&gt; E-mail and some text &lt;/div&gt; &lt;? } elseif(isset($\_GET['about'])) { ?&gt; &lt;div&gt; John Doe, 28, graduated from Harvard &lt;/div&gt; &lt;? } else { ?&gt; &lt;div&gt; Main page stuff &lt;/div&gt; &lt;? } ?&gt;

I hate to do this, but could I get a little more elaboration? Still a bit unsure of how to implement it.

CloudFlare works by doing DNS redirection if I remember correctly (quite brilliantly) so it's very possible your authorizations may have changed in some way.

Thank you so much! All of these APIs for financial services are typically overly complicated and annoying, but yours is clean as a fiddle. Love it and will use it!

Nooooooooooooo! My eyes!

The turtle has flatlined.

The best way to do this would be to create a switch to load an HTML file. switch($\_GET['section']){ default : $file = 'home.html'; break; case 'contact' : $file = 'contact.html'; break; case 'about' : $file = 'about.html'; break; // et cetera } # Path name $path = './pages/'.$file; if(file\_exists($path)){ include\_once $path; }else{ header('Location: index.php'); }

Way overcomplicated and unreliable.

Yep, best easy to do it.

The filename for this code should be `index.php`. In the same directory where `index.php` is, you should have a folder called "`pages`" where all of your `.html` files should be. You can changes pages using `http://mysite.com/?section=contact`

Nah, it made sense, personally, I'd go for server side for something like that, but it would not be terrible done in JS.

No problem at all. Place it in the &lt;body&gt; of your HTML file. If you don't put ?contacts=anything or ?about=anything at the end of the URL, you'll see "Main page stuff". If you do have ?contacts=anything you'll see "E-mail and some text" and if you have ?about=anything you'll see "John Doe, 28, graduated from Harvard"

It worked! Thank you!

He doesn't seem to know too much about PHP yet so your example might be a little overwhelming. It's a good clean example though.

Do you mean in this particular context? Or in general?

No problemo...

In this context. Building a while website using Jquery? I just can't see it working very well.

Not bad. Some minor criticisms. \* Never use is\_numeric(). It is too broad in what it accepts as numeric. Use the [filter library](http://php.net/filter). \* Hard coded endpoint url? \* \_validItem() doesn't belong in class Order unless the order knows what a valid item is (it doesn't) \* End user documentation is very sparse. A sinlge example is not good end user documentation. Fortunately the clsses are small and simple enough that most should get it.

http://el-momento.com/new/index.php Works like a charm.

Awesome. I was thinking today about if my product should use Google Checkout or Paypal - guess I'll add an option for both. If I see anything that can be improved upon in my testing I'll add a PR to GitHub.

Me, I have an old lady come and copy my code off the screen with an old typewriter.

I'll be switching to git soon.

More than likely it's not working 1 out of 20 times because Google has a spam check. They're probably throwing up a captcha. Are you checking the string output when it fails? One way to get around this is to have like 50 proxy server IPs. Whenever you get one with a captcha coming back, switch to another IP until you get one without a captcha. Eventually the captcha thing times out and Google will let you through.

Exactly! Does anyone know what it is/does?

I take Polaroids of the screen every 30 mins

This is a very basic trick but it helps a lot with facepalm mistakes late at night. instead of: if($var == "blah") use: if("blah" == $var) Why? Because if you're like me, sooner or later you'll type one "=" instead of 2 and spend a long time trying to spot it (the latter approach will cause a PHP error immediately). :)

[Pimple](http://pimple.sensiolabs.org/) uses anonymous functions to great effect.

Bazaar \*\*EDIT\*\* If you're goal with version control is to save all your changes and exchange them with others quite easily, use Bazaar. Very few cryptic commands and the help/interface is excellent. We've trained people to use it that had never used any VCS before.

&gt; &gt; It's often most generally used in the network and/or security field. &gt; &gt; Given that that's where I work, I can tell you quite certainly that no, it and/or is not. You know, when you first started this tirade, I was mostly willing to listen to you, figuring it may just be a misunderstanding. But really, it's become all-too-clear that you're really nothing more than [a troll](http://lmgtfy.com/?q=network+enumeration).

Euhm, if I were just starting with PHP and had no background on arrays, this tutorial would only confuse me. First they introduce arrays in pseudocode like this: pricesLA ("hamburger"=1.30, "fries"=.50) and then they say that the syntax in PHP is quite different: $pricesNYC = array("hamburger" =&gt; 1, "fries" =&gt; .70);

Cheers for that

If you mean version control then I recommend git, if you mean application version numbering then the system we use is: vA.B.C Where we increment 'A' when adding, 'B' when changing and 'C' when fixing. For instance. \* calculator v1.0.0 \* v1.0.1 - fixes bug where pressing + actually divides \* v1.1.0 - changed 'c' function to clear all history \* v2.0.0 - now with memory! \* v2.0.1 - fixed bug where adding to memory actually cancels your credit card. and so on.

and free private repositories for up to 5 users ;-)

Unlimited free private repos :), and it suits me just fine for internal stuff with a few clients.

The very very basics are covered in the NetTuts tutorial linked below and I'll be posting some more fun advice in a "Part 2" later, but if you are used to CI then Fuel will make a lot of sense to you. It's essentially the same pattern with some command line work and a lot more 5.3 usage. No more $this-&gt;load :D http://net.tutsplus.com/tutorials/php/getting-started-with-the-fuel-php-framework/

I use Mercurial, with all my public and private repos hosted for free on bitbucket.org.

It performs DLTK indexing.

No point using a switch statement unless you're looking to add specific logic for each page: $pages = array( 'about' =&gt; 'about.html', 'contact' =&gt; 'contact.html', 'home' =&gt; 'home.html' ); $file = isset($\_GET['section'], $pages[$\_GET['section']]) ? $pages[$\_GET['section']] : $pages['home']; require "./pages/$file"; And you even get the `$pages` array to play with, for something like generating a navigation menu with the available pages.

Explanation? What am I looking at?

It builds the search indexes that are used by things like code suggestion, linking to declarations, etc.

The [Dynamic Languages Toolkit](http://wiki.eclipse.org/DLTK) is a toolkit for the Eclipse IDE (&amp; its derivatives like the Zend IDE) that provides a framework for working with dynamic languages. It provides a search engine that lets Eclipse autocomplete code snippets, locate the original declaration of an object or method, do automatic refactoring, and lots of other useful stuff. Unfortunately creating the indexes for this search engine sometimes takes longer than we'd like. In the lower right of the picture, you can see the text "DLTK indexing in progress".

DLTK stands for Dynamic Languages Took Kit.

Something that helped me was removing jquery from the index. That shit will break parsers.

Another optimization would be to only fetch what you need: SELECT one,two,three FROM table WHERE id in (1234,1235);

Two people working in a project for a year without any source control :-S it should be one of the first things you do after creating the folder. most of the popular ones have been mentioned already. If you're going to be opening it I'd recommend github. If you want to keep it closed and have your own server/repos then git can need a bit more work to get setup so you might want to check out svn since it's a bit easier to get working. Also, check what tools are available for your OS of choice. There are many GUI clients for linux/windows/mac but yiu might find that some don't fit into your work flow. In the end it doesn't really matter, just get it under something so your changes are tracked.

I do both. When I only need the one object, it loads all its data when initialized. Sometimes I delay loading of big data chunks or seldom used data until it's needed though. I also have one or two other ways of getting the same data. One is a helper class that can build multiple objects from one query. Another is a method for getting very simple representations, like id-name pairs for selections.

Pippin? http://en.wikipedia.org/wiki/Peregrin\_Took

For small personal projects I prefer svn to git, but either are good. To be honest, even older or less useful tools like cvs and hg are better than nothing.

Thank you! I've been staring at that thing for a while and even googled it a while ago, but I could never find a satisfactory explanation. I figured it had something to do with this (given the name and all), but it's exact meaning was never clear to me. So thanks again :-)

Exactly.

FWIW, the change to support replicated database connections without changing the API or rethinking everything :) https://github.com/jbroadway/elefant/commit/a6fe63a84b61e2fecf093aead18e8d329842260b

&gt; Never use is\_numeric(). It is too broad in what it accepts as numeric. Use the [filter library](http://php.net/filter). Everyone keeps talking about that filter library around here, I'll have to give it a look. TBH, I almost just left the is\_numeric check out, since the only reason I am being anal about the value being a string there is to discourage storing currency as a floating point.. the is\_numeric check wasn't really for security as much as to provide a real quick sanity check before trying to send something to checkout that will just fail anyways. &gt; \_validItem() doesn't belong in class Order unless the order knows what a valid item is (it doesn't) I thought that felt a little weird, and I think the reason I ended up leaving it in the Order class was because I was thinking an item might be valid for some other not-yet-conceived operation, but not for an Order. That and I was just working quickly :-P I take it I should perform validation in the Item class instead, then? &gt; End user documentation is very sparse. A sinlge example is not good end user documentation. Fortunately the clsses are small and simple enough that most should get it. Yeah, it needs lots more of a lot of things still, ha. I'm just trying to make sure I have a structure that is forward-compatible before really taking the time to document (mostly non-existant) features.

&gt; If I see anything that can be improved upon in my testing I'll add a PR to GitHub. That would be awesome. It's not done yet (by default it uses the sandbox - i.e. not a real transaction), but it's certainly usable up to the point of getting a redirect URL!

My thoughts exactly. I have an item, I need it invoiced, and I don't want to hack around some crappy web store framework that's for some reason included in the example code to do it. I just want, order-&gt;additems(items)-&gt;invoice.

Why create a Google Checkout namespaced InvalidArgumentException?

So I can throw an InvalidArgumentException that is capable of being handled differently than a SomeOtherLib\InvalidArgumentException if it happens to exist in the same block of code?

To expand a little: Doing type checks is a good thing. You don't do a bad job of it, just the one check doesn't properly check for a float as it probably should. validItem() should probably be a part of the Item class. The reason for this is that you may want to extend the Item class and it should already be aware of what it takes for an Item to be valid. This check could conceivably change depending on how you want to extend the class. Typically you'd want to run validators when saving a record to persistent storage. However since your class does not use persistent storage it may be logical to fire the validations from addItem(). One final thought, and this is more for interest than anything since you coud get away without doing this in your class. When working with money, it is often desirable to store it as an integer/long rather than a float. The reason for this is that it is easier to perform math operations on. Money operations are best done as integer math. For example: if I have 10 cents, and I want to split it 3 ways, what happens is I wind up having 3, 3, 3, with 1 left over. A penny is not divisible since there is no denomination of cash smaller than one cent. Your code should decide what to do with remainders. You could round, but do you round up, or round down? 10/3 is 3.333... And the problem becomes this: Rounding up: 4 + 4 + 4 = 12 (you've given out 2 cents more than you have) Rounding down: 3 + 3 + 3 = 9 (one cent is missing) A good money object can do this for you. Your money object can also be aware of how to format based on locale, rate conversions, and other items specifically related to money amounts.

How would you handle an InvalidArgumentException in a checkout library? Give the client a predefined product :) ? Exceptions like InvalidArgumentException point out errors in code that are caused by wrongly passing parameters. It's not like a database exception from which you can recover.

Thanks Bulldada. I can get to Glasgow easily enough so I am currently looking there. Fingers crossed :D

Yep this image sums up my day... The worst is when you go to save and eclipse feels the need to index everything BEFORE the save. I then figure it's a good time to go to reddit... boom another hour wasted.

\\*\\*\*\*COUGH\*\*\\*\\*[Aptana](http://www.aptana.com/)\\*\\*\*\*COUGH\*\*\\*\\*

How about [this](http://code.google.com/p/php-mime-mail-parser/)? It requires a PECL extension, but is fairly up-to-date, it appears.

This is a great workflow, but overly complicated in my opinion. I like the [Github Flow](http://scottchacon.com/2011/08/31/github-flow.html) mentioned by Scott Chacon. It's much simpler to follow and (IMO) just as effective.

It's prehistoric and throws all kinds of E\_STRICT errors, but PEAR's [Mail\_mimeDecode](http://pear.php.net/package/Mail\_mimeDecode) does the job nicely and doesn't care were the MIME message comes from. It's pure-PHP and BSD licensed, so you can just bundle it with your code.

&gt; It's not like a database exception from which you can recover. That's true, and you shouldn't use exceptions to handle invalid user input either.. but that said, ZF2 still throws an InvalidArgumentException in situations where you can't really recover [(example)](https://github.com/zendframework/zf2/blob/master/library/Zend/Log/Logger.php). I suppose the same "what's the point?" question could be asked of the Zend guys, no? Maybe they're just trying to be verbose, and it's not as though it's extra overhead to do so (Edit: unless you expect the exception to be thrown on a regular basis in production, in which case you have bigger problems :-P ). Maybe it's so you can implement logging of exceptions by their type if you want in your application? In a larger application in production, you might have lots of exceptions that get logged and it might be nice to organize them by their type / namespace. That's just a guess, I'd be interested to know more about that.. but in my case, I just like being verbose.. Generally speaking, in the event of a failure, I'd rather see something that says: "invalid argument error" rather than "error". I guess I've just seen enough legacy code out there filled with unhelpful / generic error messages and exception swallowing to the point where failure to set a foobar\_flag on a transaction results in an error message like "307 error could not send", which in that type of a scenario is completely unhelpful.

bzr totally.

&gt; validItem() should probably be a part of the Item class. The reason for this is that you may want to extend the Item class and it should already be aware of what it takes for an Item to be valid. This check could conceivably change depending on how you want to extend the class. &gt; Typically you'd want to run validators when saving a record to persistent storage. However since your class does not use persistent storage it may be logical to fire the validations from addItem(). That makes plenty of sense, thanks. &gt; you coud get away without doing [money stuff] in your class. yeah, that's definitely my hope.. luckily I think I should be able to completely avoid having to perform monetary arithmetic, and let google handle all that nonsense. It's just one more place for something to go wrong.

JetBrains PHPStorm all the way. Couldn't take Eclipse's bullshit anymore.

Thoughts with absolutely no scientific data to back it up. You're mileage may vary: \* Might be due to the number of projects you have. I have 2 large projects that I primarily work on and indexing doesn't take that long. Before I closed the inactive projects it would take forever. \* In preferences, I have everything in the workspace (General-&gt;Workspace) unchecked (ie Build automatically, refresh on access) Not sure if related but I have autocomplete HTML turned off. Through trial and error, I've got Eclipse running pretty snappily.

A second vote for jQuery UI Tabs. There is a way to get them to be horizontal like that as well, but the current implementation is kind of nasty - 1.9 will be better. The tabs could really just pull in the HTML from another page, too; you don't have to have all the markup on that page, just in their own respective places if they needed to be re-used elsewhere.

I totally disagree, although I'd be interested to see if you could say anything to convince me. Magic get and set are a far worse readability issue in my opinion. One of the really common things that comes up maintaining a large, complex codebase is to add a little extra bit of logic to the storage or retrieval of an attribute. If you're using accessors, you just add your extra code there and you're done. If you're using magic get and set, then you have to add code to a growing mess of a switch statement in your \\_\\_get or \\_\\_set method. It's messy and it violates the principle of separation of concerns. Plus, [performance is also an issue](http://stackoverflow.com/questions/3330852/get-set-call-performance-questions-with-php).

Have they added full PHP support (ie click-through into classes, source hovers, etc...) in 3 yet? They hadn't as of a couple of months ago. And 2 was way to slow/buggy for me to use as a PHP IDE. IMHO.

Yes. I hated Aptana 2, but 3 is leaps and bounds better. It even seems less buggy than Eclipse PDT, though I'll leave that for you to decide. EDIT: The only thing I don't care for with Aptana is the default theme, but I just use the [Tomorrow Night](https://github.com/ChrisKempson/Tomorrow-Theme) theme I found and everything is beautiful.

PHP n00b here. Instead of the switch statement why not just append ".html" to $\_GET['section']? Wouldn't that be easier?

[fMailbox::parseMessage()](http://flourishlib.com/docs/fMailbox#Parsing) does what you are looking for. It accepts a string and parses it via pure PHP. You get back an array of structured information about the message, including attachments, headers, etc.

Sorry, you misread me. I was more so asking as of why namespacing the InvalidArgumentException is a useful thing. Given it is unhandlable it doesn't help being namespaced. An example of handlable namespaced InvalidArgumentException would be on a method that has sensible defaults. For example: namespace Posts; class InvalidArgumentException extends Exception {} // in another file try { $posts = Posts::getPostsByQuery($user\_query); // may throw \InvalidArgumentException $posts-&gt;sortBy($user\_sort); // throws \Posts\InvalidArgumentException } catch(\Posts\InvalidArgumentException $e) { $posts-&gt;sortBy(DEFAULT\_POSTS\_SORT); } And maybe let the `\InvalidArgumentException` bubble up to my error handler that sends the user a friendly error message. The point being that in this case a namespaced exception makes sense because in the exceptional scenario I can still fix things up. But then again that could be easily moved into the `sortBy` method itself, by a defensive programmer. P.S. Hope it makes some sense at least. As for the Zend guys, yeah they like to namespace everything.

I like this, thanks for the tip!!

I also found that using different Workspaces to group related projects helps a lot as well. Closing inactive projects, grouping with workspaces, and properly setting up the project build and include paths has helped quite a bit. I rarely see this anymore.

3 still seems buggy to me, but agreed, it's much better than 2. I don't know why, but every time I decide to give Aptana another shot, I wind up back on PDT.

I hate Eclipse with a passion. [Architecture Astronauts](http://www.joelonsoftware.com/articles/fog0000000018.html) abound at Eclipse.

No, because that would let people poke around your hard drive to look for other HTML files of interest, perhaps including some you weren't interested in sharing.

What if you wanted search engines to find your content?

It's an offense to humankind and nature alike.

Make a queue. Keep track of the number in the queue. Users can be added the the end and as long as the number of users in the queue is at least 2 keep popping off two people at a time and run the function from there. The only thing you have to worry about is if you want to make this thread safe you have to make sure you lock the queue before popping, otherwise you might get multiple people without a partner waiting instead of just sets of partners.

So, someone asking you to defend your claims with more than vague assertions is a troll, based on that you googled a new phrase that \*also\* doesn't defend you, and want to walk away. Network enumeration is when you find devices on a network. Maybe you should read the thing you just googled; that has nothing to do with it (again). Be sure to get more insults in with your next last word, though.

That's been my experience too. Plus now that I finally have PDT configured to my liking, the incentive to switch is a lot lower.

Oh, I see what you mean. Yeah, I guess it's not really useful in this situation.. that's not to say it might not be useful elsewhere as the library grows, but I see your point, ha

that would cause a mess. how i would do it to simply have a table structured like [this](http://i.imgur.com/qscnT.png). once a user gets added to this table you you query the same table again and finds out what user\_id he has if the number is odd the this user gets to search for a partner which are the users with even numbers. a way to output this to the users would be to have them both query the database and check if their entries "partner\_id, partner\_IP" have been filled out.

You'd use the mail() function. Look it up on php.net and follow one of those examples. Really though, you should be doing a lot of extra work to validate the form inputs before just passing it all to mail().

the [mail\(\) function](http://php.net/manual/en/function.mail.php) is the way to go. If you have Zend installed I would try out [Zend Mail](http://framework.zend.com/manual/en/zend.mail.html). OOP makes it so much easier and Zend does all the heavy lifting for you.

use fucking google

Thanks for information. Will look into that.

Sorry, but when it becomes obvious you're arguing simply "to argue," we've crossed the point at which this is a viable conversation (though it's entertaining watching you try to argue against yourself). [Do Not Feed The Trolls](http://www.reddit.com/r/PHP/comments/iu7xw/stonecypher\_can\_you\_tell\_us\_about\_your/). Edit: terse explanation added

Disregarding your rudeness, an issue with PHP tutorials on Google, especially for a beginner, is that there are \*so many\* tutorials written by \*other\* beginners who, having just figured out how to do something, apparently decide they're now qualified to teach other people how to do it, and don't realize that the way they're doing/teaching it is wrong in some way. Not that there's any guarantee an answer here will be better, but it seems more likely that a wider variety of answers will be encountered, at least.

http://php.net/manual/en/function.mail.php well what do you know, google + php direct source + examples at the bottom, AMAZING Edit: r/phphelp then i probably wouldn't be as rude.

Just [more troll fodder](http://www.reddit.com/r/PHP/comments/ki1sr/is\_there\_any\_bugzilla\_clone\_written\_in\_php/c2kj9nv?context=3) from StoneCypher ... though I'm guessing that just checking out [his profile](http://www.reddit.com/user/StoneCypher) is probably decent history, too (I've not yet bothered).

I use mercurial tortoise HG. I use odd numbered version for new features (and some bug fixes) and even number versions purely for fixes. These are very often being developed side by side so that fixes can be issued while development of new features is happening. I closely manage branches, which are always clones of the main branch, which in and of itself does not have outstanding branches, only closed branches and stubs for when clones were made. The branch clones are patch queues are main branches for whatever version its governing. Clones are created off these for individual features. Once an odd numbered branch has been made feature complete, its queue stack is duplicated on the main repository branch representing that version. Almost always this goes off without a hitch because if there were a problem it would have shown up on the clone. Once all the patches are set, a merge or transplant occurs so that the main trunk comes into sync with the new version. Even numbered (bug fix) version act much the same way, but merging and closing of outstanding clones happens much more often so that so that fixes get out quicker and main development has non-broken stuff to work from. It can be overwhelming trying to reconcile conflicts in this development style especially since you have two actively developed branches open at any given time, but I think its worth it. Whatever you use, be it git, mercurial, or etc, I recommend doing so via patches. I've never been comfortable with people pushing changes onto the main repository, even the manager of the project. With patches no one needs to FTP stuff or copy stuff to the repository or directory. You just email the patch to the manager and they can apply the patch and finish patch or rollback as necessary.

I'm going to have to agree with cerealbh. This subreddit is quickly becoming like the dregs of Stackoverflow. "Hey guys, I downloaded this 3rd party script, I've never done any programming, how do I computer?"

agreed :( If someone is trying like "Hey, the subject on my mail() isn't outputting properly? what gives?" ill jump all over to help with that, but "TELL ME HOW, NOW" yah fuck that.

I would purpose something different. First, sanitize the input. $page = preg\_replace('([^a-zA-Z0-9\_])','',$\_GET['page']); $page = strtolower($page); Check if the input leads to an existing page. In this example, all you'r page files should be in one folder, called pages. if(file\_exists('pages/'.$page.'.php')) require('pages/'.$page.'.php'); else require('pages/default.php'); This should be a good way to do it on small applications/webpages. If you want 'logic' url's, you can do it in htaccess with the following line: RewriteRule ^([A-Za-z0-9\_-]+)/?$ /index.php?page=$1 [L] This way, the url for loading pages/contact.php in this example would look like this: http://www.yourwebpage.com/contact

"Hey kid, I'm a computer!"

Try [PHPMailer](http://phpmailer.worxware.com/index.php?pg=phpmailer). It wraps the basic mail sender (and a bunch of others) with an easy interface.

It's hard to take a man (spolsky) seriously who's greatest achievement is a bug tracker programmers hate to use.

nano is one of my favorite editors for basic text files. PHP not so much, but it does work! I'm so used to using it for config files/etc that when I'm in vim I find myself trying to CTRL-O/CTRL-X.

It's called menus. In your specific example, it's a vertical menu. The page doesn't change because everything is loaded on the page request and clicks are handled by javascript, which basically hides/displays the relevant content.

\* co-founder and CEO of the most successful Q&amp;A platform on the net \* Program Manager on the Microsoft Excel team, where he designed Excel Basic and drove Microsoft's Visual Basic for Applications strategy \* author of five books (ironically, one is titled \_[Smart and Gets Things Done](http://www.joelonsoftware.com/items/2007/06/05.html)\_) \* founder and CEO of his own company that has survived 11 years in a tumultuous industry You're right though, let's take a completely subjective opinion about one of his products and use that to sum up \_all that is Joel Spolsky\_. Your logic is flawless. I'm curious though, since his accomplishments evidently don't impress you, what your resume looks like; by all indications, it should be quite impressive! Do you have a site that gets millions of unique pageviews a day? Are you the CEO of your own company that has survived over a decade? Have you written \_any\_ published books? Have you held a technical management position within a tier-1 tech company (Google, MS, Y!, etc.)? Aside from your opinion of Spolsky, that doesn't negate the fact that Eclipse blows goats.

on first glance, it doesn't look like this supports any sort of IPN functionality - is that correct? Would be fantastic if it did / will in the future... What I'd love to find but haven't gotten around to digging for would be a library like this that handles Google Checkout, Paypal, and maybe even Amazon payments, too...

Normally I would't use this response, but in this instance I agree. Googling "how to send email with php" yields thousands of examples on using the mail() function. I would expect anyone to take \*at least\* 5 minutes to answer their own question.

&gt; on first glance, it doesn't look like this supports any sort of IPN functionality - is that correct? Yeah, I haven't even got past getting the redirect url yet for processing an order. I'll probably end up building that over the next few weeks / months as I get the time to. &gt; What I'd love to find but haven't gotten around to digging for would be a library like this that handles Google Checkout, Paypal, and maybe even Amazon payments, too... Yeah, that'd be a dream, and probably not even all that difficult to accomplish. I'm kinda shocked something like that doesn't seem to exist in PHP.. you'd think PHP shopping cart devs would be dying to have a centralized payment library that they could all share and implement in their payment modules. Lots of reinventing the wheel out there these days.

Your code is actually more complicated, I think. A switch is so simple, but this... This could get extremely ugly way too easily. This is also not very nice to read, meaning someone new to php might be just as (Or even more) daunted by it as they would by a switch.

&gt; Yeah, that'd be a dream, and probably not even all that difficult to accomplish. I'm kinda shocked something like that doesn't seem to exist in PHP.. you'd think PHP shopping cart devs would be dying to have a centralized payment library that they could all share and implement in their payment modules. Lots of reinventing the wheel out there these days. I'm shocked too. Every time I think of building it, I'm just like "this is stupid.. someone has to have built this"... and I search... and I find nothing.

I'd say you could/should associated a location with all records. Then, when a controller request is made, check the location with the level of the user. This should alleviate any potential address-bar exploits.

Can you clarify/expand on that?

Well, if you have a single entry for each employee regardless of status (director, manager, employee) you should have a field for the status as well as their location (say an id based off of a location reference table). This way when you pass the location id (or employee id) to edit, the controller (or an auth/acl controller) checks the location against the active user's credentials to see if they are a manager or employee and, if so, can they edit this record. You can do this with all editable records, just include a location code. Make sense?

Sort of. When you say location, are you referring to the controller path? Say a table with records such as: Edit: user\_id | status | location 1 | director | users/edit/10 2 | manager | users/edit/10 10 | employee | users/edit/10

&gt; Sorry, but when it becomes obvious you're arguing simply "to argue," You're just making excuses.

Look at you opening new comments on two month old posts to complain that someone else is a troll.

Somewhat. The controller path, that you referred to originally, contained at least a record id for the record to edit. With that in mind, the easiest way to do what you want is to have a company table and probably an office table with the id of the company. With this in mind, you can place the company id and office id in the user authentication table so you can check the record to edit (also with a company and office) to see if the person has sufficient permissions to perform the edits. Companies id | company 1 | apple 2 | microsoft Offices id | company | office 1 | 2 | R&amp;D 2 | 2 | marketing 3 | 1 | iTunes office 3 | 1 | housekeeping status id | status 1 | director 2 | manager 3 | user users id | office | status | user 1 | 1 | 2 | some r&amp;d dude 2 | 3 | 3 | some janitor 3 | 2 | 1 | some director now you would check the record to be edited against the office location. Then, check the user's status then office. This way you can see if they are a director (at which point you could skip other checks) then check office to make sure they can edit the record. Sorry but this is getting more complicated than a reddit post should be IMHO! :) This should lead you on your way. EDIT: sorry for the formatting, I'm not up on the proper way to make tables here!

In regard to formatting, put 4 spaces in the front of whatever lines you want to format.

Reddit doesn't have tables, but you can get monospace with either four lead spaces or framing backticks.

This is starting to make more sense. I'll get back to it when I get home. Thanks.

It's simpler than you're making it. You're trying to do what amounts to rudimentary heirarchal role-based permissioning, but you're trying to face the whole thing in a single blow. This is complex. You have to do it in pieces. The \*first\* thing you do, which you haven't done, is write out formally what every decision actually is. You got started with your four bullet list, but I almost guarantee there are more cases you haven't thought of. You need to write every single criterion down the second you discover it; no exceptions. Next, you write some stuff that'll let you ask questions. function is\_at\_least\_manager(actorId) { ... SQL ... } function is\_at\_least\_director(actorId) { ... SQL ... } function is\_super\_admin(actorId) { ... SQL ... } function is\_own\_profile(actorId, profileId) { ... SQL ... } function is\_office\_subordinate\_profile(actorId, profileId) { ... } function is\_company\_subordinate\_profile(actorId, profileId) { ... } None of those individually are particularly hard, and with them, the rest is actually relatively simple. function may\_edit(actorId, profileId) { if (is\_own\_profile(actorId, profileId)) { return true; } if (is\_at\_least\_manager(actorId) &amp;&amp; is\_office\_subordinate\_profile(actorId, profileId)) { return true; } if (is\_at\_least\_director(actorId) &amp;&amp; is\_company\_subordinate\_profile(actorId, profileId)) { return true; } if (is\_super\_admin(actorId)) { return true; } return false; } I mean, can you do this table-driven? Yeah. Can you do it all in one query in the SQL to keep business logic out of the PHP? Sure. Is this easier to reason about, and easier for you to keep secure when it changes? \*\*\*\*\*Yes\*\*\*\*\*.

I just hit the "Code" option above the box to format it. Works nicely! thanks for the spacing notes, though!

This may or may not be applicable to your situation but you may take something out of it... One app I maintain involves a situation where users of multiple companies can access cases. Permissions are based around the company who 'owns' the record, the company who submitted the record, and any other participants whom the case is shared (you can add arbitrary participants to the case and choose to share with them). The way we handle this is with a view which associates each case id with company and user id's. Essentially this view flattens the list of who can access the case. On the software side, the case object has a pre-query filter which automatically does an inner join on the view (we're using Doctrine) using the current user's user id and company id. So essentially if I do something such as: $obj = Doctrine::getTable('Object')-&gt;find('123'); this will always return null if the user does not have access, it's automatic everywhere in the code so there is no risk of slipping up and accidentally showing someone something they should not have access to. It works reasonably well, though I would not say it's ideal (views in MySQL aren't exactly speedy, and this would be painful to implement if you needed access control to multiple, unrelated objects).

&gt;co-founder and CEO of the most successful Q&amp;A platform on the net Says little about him and lots about those who contribute answers to questions. Nobody goes there to read about spolsky. &gt;Program Manager on the Microsoft Excel team, where he designed Excel Basic and drove Microsoft's Visual Basic for Applications strategy Because programmers in general have good things to say about visual basic. &gt;author of five books (ironically, one is titled Smart and Gets Things Done) This doesn't make him right on many points. In fact, many times he's just blatantly wrong. &gt;founder and CEO of his own company that has survived 11 years in a tumultuous industry That makes products only liked by pointy haired bosses. Trying to troll me won't work.

I'll agree with this one. I recently had to process umpteen zillion emails to extract attachments and other goodies, and Mail\_mimeDecode handled them all. I tried a bunch of things but this ended up being the best combination of reliability and ease-of-use.

But you agree that Eclipse is a POS? And still, if you're going to attack the guy, what have you to show for yourself? I'm curious what \_your\_ opinion is worth.

I'm not sure how far you want to go or what your threat is exactly. Also, while I believe these are good practices, but please anyone critique these as needed. You could grab GACL and use that, which is a tried and true library. It's not pretty, but it will work and many projects have used it. This is the easiest/most basic solution, although not the best. Most of the time you're going to have a single point of failure: the database. You can also use Linux file system permissions, groups, and users to create an additional layer of security. If you use suexec or suphp this works nicely because everyone only can see what they need and nothing above them. For example Managers would be able to see their files AND Employees, but not Directors. This has higher maintenance as the number of user types goes up. Another method that I toyed around with, but have not yet used in production is private/public key cryptography and signing of rows into the database. Not to encrypt any data, but to sign the data in a way that an attackers inserts can easily be detected. Say you pull a record, check it's hash with the public key and it's a mismatch - you can safely ignore the inserted data. This is an idea I had for very limited use on immutable tables like order data.

Wouldn't Aptana also have this problem, since it is Eclipse based? I don't use it, so I honestly don't know.

Thanks for this. You're right, I am making it harder than it is because I'm trying to deal with it all at once. I was trying to keep my function(s) small, and I could see implementing your solution, the may\_edit may get a bit big. But thinking about it, this is probably the best route. I think I'm to implement this, but utilize MySQL Views for the SQL statements as some of them are long and messy. I understand there's a performance decrease when using Views, but I am incredibly anal about my code being clean and would rather have that then a ton of SQL statements or CI Active Records confusing things.

Look into sessions and saving the product of the two random numbers into a session variable you can access across pages. Official [PHP session reference guide](http://www.php.net/manual/en/book.session.php).

you can put the random numbers into hidden inputs on the form: &lt;form&gt; &lt;input type='text' name='answer'&gt; &lt;input type='hidden' name='number1' value='$number1'&gt; &lt;input type='hidden' name='number2' value='$number2'&gt; &lt;/form&gt; (note: code not complete. add php code, submit button, etc.)

If page 1 generates two random variables, A and B, then you need to tell page 2 what those are. You can set them as session variables in page A, and read them in page B, or you can set them as hidden fields in your form and read them from the GET or POST variables in page B. If you want to get creative, you can do it in one page basically like this (I'm pseudocoding): if (POST[submit] is empty) { generate varA generate varB show the form } else { grab varA, varB and userAnswer from POST check if they got it right if (answer is correct) { print "correct" } else { print "wrong!" }

Interesting article, but he seems to conflate software architects with the marketing hype machine. These days, I'm pretty sure that samples of Tide come in the mail with claims that they will help end users take control of their technological lives. That said, I'm still glad to see that I'm not the only one with a fuming hatred for Eclipse and its horrid excuse for PHP support. I am saddened by the fact that it apparently hasn't improved all that much since I last used it in 2007/2008, though.

I don't know if I misunderstood what you are trying to do, but here is a 2 page solution: firstpage.php &lt;form action="check\_answer.php" method="post"&gt; &lt;input type="text" name="number1" value="$number1" readonly&gt; &lt;input type="text" name="number2" value="$number2" readonly&gt; &lt;input type="text" name="answer"&gt; &lt;input type="submit"&gt; &lt;/form&gt; ------ check\_answer.php if(($\_POST["number1"] \* $\_POST["number2"]) == $\_POST["answer"]) { echo "You're a goddamned mathematical genius!"; } else { echo "Dumbass. The answer is ".($\_POST["number1"] \* $\_POST["number2"]); } PS: Style the inputs on the first page to remove their borders. EDIT: As per calmcamel notes below. Thanks!

Use a [Nested Set](http://en.wikipedia.org/wiki/Nested\_set\_model), one query for any logged in user to see what they have access to edit.

if the above example, you would either have to specify method='post' on the form, or change all the POSTs to GETs

you could make the ID hashable, generates simple md5 hashes or use mysql UUID() and pass the UUID to the browser, no user would try to touch something like that as it will come off as highly obfuscated to them. They will figure its not worth their time.

Yeah! I saw that! They keep making it better and better. They also added a bunch more to the CSS side.

Don't bother with views (for that matter, don't bother with CodeIgniter.) This isn't what views are for, and CodeIgniter in the long run will cost you more than it gains you.

Find some interesting projects on Github and submit some patches to open-source projects. It will get your name out there, and you'll have something tangible to show an employer. Since it's open-source, they can see what you wrote and that a community liked it enough to accept your contribution. You'll learn the ins-and-outs of version control very quickly, and you'll learn write software that works for everybody, not just the particular instance you're writing it for.

This is the comment you want, except you can save a step by not making those inputs hidden and make them read only instead.. Once you get a little more comfortable with programming, look into using sessions.

/r/php is an ocean of piss lately Perhaps the rest of us ought to make more of an effort to make it a place for proper discussion and news, because it's pretty crap at the moment. "Help me", "how do I do this really basic thing", "my code is giving me this error". Fucking boring.

Well, it depends on tons of things such as quality, experience, the market, speed, etc. Do you think you'd code things fast or learn as you go? I generally would expect a PHP developer to start off around $25-30/hr. I've seen developers charging up to $200/hr though, and their work was worth it.

If you are starting out I would probably aim at $20-$30 an hour. It also depends on where you're located.

The reason that I think the link is appropriate is that Eclipse has gone to so much trouble to make their "platform" as generic and reusable as possible, but in doing so, have made it a nightmare to use.

You say you havent worked full time, and if you feel inexperienced to handle a LARGE scale project or website, then you should charge lower, say 18-20$. Once you gain great personal experience, build a good portfolio and clientele base, then start charging more. If you charge high, and fail at a project, its going to kick you in the ass. You need to get the experience and work your way up. When you feel like you can say YES to anything a client may need - start charging the big dough. Personally I charge $50-$60 and hr to most of my cleints and feel its worth it based on my experience, clients and knowledge.. and also charge for any phone calls and e-mails as well.

Saul Goodman.

Keep in mind that you're not selling a product so much as you're selling your time. What your time is worth is something you need to decide for yourself. The market will let you know if it agrees with you or not. There is no polite way to say this, and for that I apologize in advance. Most programmers who ask this question with only a few years of personal experience aren't worth much. In most cases they still have no concept of what good code is. The problem is that because you're self taught, there are probably gaps in your education that you could park a bus in. Very few people self train and have a solid grasp of everything they need in two years.

A programmer that can get done in 2 hours what 2 coders could have crammed into 8.

oh he's worth more than that.

Please, don't do that.

Self taught with practical experience is a different animal, though. I know plenty of really talented developers that are "self taught" (really, aren't we all to a huge degree?), but their experience has more than made up for any lack in formal education. I don't mean to imply that you were saying otherwise; I just wanted to add some potential clarification.

High quality development of software that is meant to support medium to large scale, mission-critical functionality is worth every penny (and then some). How much money do you think high quality consultancies charge? Depending on the area, $200/h \_per developer\_ is not at all unreasonable.

I use both lunarpages and 1and1. They are both pretty good.

Thank You, I will check them out

I've had good success with hostgator over the years.

Think nothing of it. Clarification is good.

I've been using Dreamhost for 5+ years with great success. A number of people I know have also been happy with them, but a few have reported problems as well.

I've had the opportunity to work with a guy like that. Speed was slightly more of his value than quality, but both were stellar. He was picked up last year by a company willing to nearly triple his salary. He inspired me quite a bit. Edit: the guy was also talented with electronics. Really interesting to talk to! I was bummed to see him go.

My company has had awful luck with 1and1. Just another opinion.

I have nothing but good things to say about Linode, however they are a VPS and not just a dedicated PHP host, so it's about $20 a month and may be way more than they need.

You could also check out a place like [this](http://hosting-review.com/).

I've been working with [rackspot](http://rackspot.com) and no complains so far. Good latency for Europe and USA.

I'm going to chime in here as well and say 1and1 isn't good. DB queries are atrociously slow, and I have scripts that need to make API calls that run FASTER if I have the client/visitor's machine do the look-up and return data to the server instead of just having the SERVER do the call. It is unacceptable.

My two cents, not a big fan of their domain management either. Personally use Rackspace yet that's a little more involved than a simple hosting project like this. In the long run it can save you money but you're the server admin and on the hook for downtime\misconfigurations\etc. Probably can't go wrong with Lunarpages.

Over the years, I have racked up over 500 clients on [TotalChoice](http://www.totalchoicehosting.com), and have had great success with them. Small sites on virtual hosting are their bread and butter. They offer dedicated servers, but you don't need that. They're also cheaper than anyone else. $4/mo. if I remember correctly. Oh, and they do 12-hour incremental backups, provided with the service.

Most phones will tell you how long youve been on a call, support or not, I charge for the call, rounded to an hour. Emails, is more tricky, if your finding yourself back-and-fourth with a client al day with email, charge them for an hour.

I use and have had good experience with Bluehost

Bloat?

I've also been using Dreamhost for over five years and their services are \*almost\* problem-free. If you look at their TOS, technically they can delete your account if you upload anything using a PHP file (like form uploads). Although you can store and stream audio or video files to users, it's also technically a violation of their TOS. Also, their "unlimited" bandwidth and disk space is obviously not true. tl;dr: Don't use Dreamhost if you plan on building a music or video-based website. Otherwise, they're great.

I'm not seeing anything that suggests anything of that nature in their TOS or Acceptable Use Policy. They do forbid the use of their servers for personal backup (except within an optionally configured specially designated space for that purpose), but I see nothing about uploading files or storing/streaming audio and video files unless, of course, they're in violation of copyright.

How bout no. I like thinner MVCs

Most "PHP programmers" that I've worked with have been self-taught and don't really understand CS like a more traditionally trained programmer. You can probably get a job for $15-20/hour. Experienced PHP programmers can charge $40-60/hour. Really amazing PHP programmers can charge $80-120/hour.

It depends on the project. We do both Drupal and WordPress work. For WordPress a lot of clients balk at $50/hour. For Drupal, when we say $80/hour they wonder if we have the experience to do the job because we're not billing at $100/hour.

i18n seems more complete on ZF, but I'm just starting that on S2. I miss how quickly you could create view helpers. Maybe even Zend\_Application.

In addition to that, many of the top-tier consultancy firms charge by the day or half day. You can't even get their programmers for less than $800, and that's for four hours of work.

I honestly don't do any of the Drupal dev. We have a few contractors that love it. My partner doesn't mind it too much, but I've patently refused to do any Drupal dev.

Then you haven't worked in an emergency mode with a hired gun. It happens. The guys that command that kind of hourly rate are there for emergencies and command incredible pricing because of the fact that they can hammer out fixes/features quickly. For example, we ran into some severe mysql issues this summer, we worked with several great DBAs that didn't have solutions for the problems and everything we tried was just leading to new issues. We ended up hiring a Percona consultant that fixed the issues and then some in ~1.5 hours. Now, they charged $400 an hour to get him in without scheduling it out 6 weeks in an advance, but it was cheaper to do that than to keep running in circles. Not only did the initial issues we were trying to solve get solved quickly, but we got an almost 30% performance increase across the board that has been sustained for more than 4 months. Our guys could have done the same thing he did, but he didn't have to search for solutions etc. He got access, did what he knew by heart - and got out. We got billed in 15 min increments with a 1 hour minimum. He then went back about his normal day of riding the rescue in similar situations. Our guys would have been able to do it but it would have cost more in the long run because they would have taken more hours at a lower rate to achieve the same thing. Anyone that has worked in this business on large projects for any amount of time has similar stories.

“The higher we soar, the smaller we appear to those who cannot fly.” --Friedrich Nietzsche

Well, I think [this place](http://tlfhosting.com/hosting) is keen. Of course, I'm ~highly~ biased because I work there! Ask me any questions you may have though.

I'm of the opinion that when you begin dealing with SSL and credit card orders, it is time to go to a VPS or dedicated server. $20/month is not a lot of money for a revenue generating service.

um. we charge more then 200/hr in my company all the time. It would be difficult for a lone person to get that but if you are with a company the sales people can pull that off pretty easily.

I charge €50/hr

Depends on where you live. I know of reception jobs in NYC that pay more than $20 an hour.

tell them that project administration is billable time up front, that way you get paid to think about/manage the project, and they aren't surprised.

I charge €49.99/hr

I'm 20 and I just got 2 offers paying $40 an hour. I have no degree. Needless to say this is a great industry.

Could be you're not charging enough. Your confidence and your abilities comes through, to an extent, in what you charge. There are people who know PHP and can make it do shit - they work for $20-40/hr. Then there are programmers who know how to program and PHP just happens to be the language they are using. For these people, $60+/hr is expected and the amount of work available is much greater. I charge $80 and stay booked.

If I ever have to explain it I usually just say that turning my attention from to one client, no matter how small the distraction, is taking time that could be spent on another. My invoice for this comment is in the mail.

Linode is amazing. I've been using them for 3yrs and have several Linodes in various data centers and never have any problems. They had to do emergency surgery on one of my servers a short while back, but they were on top of it like crazy and had it fixed and up and going again in minutes. Their technical support is fantastic.

BTW, That "may\_edit" function is essentially what [Zend\_Acl Assertions](http://framework.zend.com/manual/en/zend.acl.advanced.html#zend.acl.advanced.assertions) are for. You can hide the complex logic away in an assert class (that implements Zend\_Acl\_Assert\_Interface), and then use the regular ACL isAllowed() call to check whether an actorId (Role) has access to a profileId (Resource). It's easier if you're working with model objects that implement Zend\_Acl\_Resource\_Interface, though, instead of working with scalar database IDs.

You can pay me $40 per hour. Once I'm done I'll go ahead and multiply my hours by 5. You'll never know the difference ;).

You state that you are so talented with a view that no one could possibly be paid N because you only get paid Y. Now you are stating that you can't figure out Drupal. If you are so talented, why would you have a hard time with it?

&gt; generic function (or functions) that will test to see if a certain user has permissions to edit specific records in the database. One possible solution would be to implement some sort of Row Level Security. If you're using Zend\_Db or something similar, you can modify your selects by doing something like this: abstract class My\_DB\_Security\_Invoices() { static public can\_update(Zend\_Db\_Select $select) { if (//user has global permissions to update in this table) { return $select; } if (//user can't edit certain rows in this table) { $select-&gt;where("office\_id IN ?", My\_User::getUser()-&gt;getUserOffices()); } if (// user can't update \*anything\*) { $select-&gt;where("1=0"); } return $select; } }

I've never used Zend Framework or even looked into it. You might want to mark it as a requirement because you use the Zend\_Http\_Client class and include whatever versions are compatible.

You have a very limited understanding of scale. The first response nailed it so well I'm not going to attempt to top it. Feel free to continue to make your fashionable blogs and rss feeds - I'll agree all day long that you don't need $200 p/hour to do that. However, you have numerous actual humans with no potential benefit beyond winning an argument on reddit while looking incredibly foolish in their business sense that HAVE paid or worked with people paid that much. I find it downright amusing that you think, even faced with overwhelming dispute, that we are all trolling you and lying.

Lets not and say we did.

Use an hourly calculator (like [this](http://freelanceswitch.com/rates/)) to estimate your rate.

In LA apparently the money I spend in Ohio renting a 3 bedroom house with a privacy fenced back yard will only net me half the rent for a decent apartment. At least that's the way they make it out to be. I'd get a feel for what to charge based on where your clients are located if you do a lot of across-country jobs. Midwest - $20/hr; East/West Coast - $30/hr; doesn't sound so unreasonable to start with.

I inform them upfront when starting projects with any of them. Theres no point to spend an hour or so on the phone with a client and not get paid for it - its common in a lot of industries... get on the phone with your lawyer, they will charge you for the calls.

I know a guy who has so much PHP work coming in, he can't help but turn it down. His problem is that everyone on his list of competent go-to guys are in the same situation and can't help him. And all of the kids out of college are completely useless in terms of having to babysit them and teach them everything they don't know within 2 seconds of looking at something. He's recently come to me, and although I'm not the most rounded developer, he knows that if I don't know something, I will figure it out myself and go from there. I met him ~7 years ago when I started attending our local 2600 group when I was 16 before it fizzed out. I owe a lot of my tendency to self-teach from those days, and apparently it's invaluable.

http://www.edis.at/ won't find better, won't find cheaper. I got VRS Basic SSD, and during my benchmarks I can get from 4000 to 9000 requests per second from simple hello world class script &lt;/end-marketing-pitch-i-dont-even-work-for-them&gt;

Speaks volumes to your skill level. Continue to believe that the tens of thousands of developers that don't have your confusion with a year after year "Best CMS" award winner know less than you.

€50 to €70 an hour when I'm on the clock, but I have 10 years of experience and do more than PHP development.

€40 to €70 an hour when I'm on the clock, but I have 10 years of experience and do more than PHP development.

The views are due to the amount of joins in the SQL statement, and I can't necessarily back out on CodeIgniter, I'm one of three developers on this project, and we all have the most experience with it. FWIW, what would you recommend?

One lawyer I know charges for "reciept of voicemail"...

My opinion is that the work you save up front with a framework does not compare to the work you have to spend hammering one-size-fits-all to fit you. As a result, for things other than throwaway projects, I do not use frameworks. Your mileage may vary. Void in Delaware.

Annual salary / 1000. You're entry level, so I'd say $35-45k depending on your location, thus $35-$45/hr. Good luck!

Haha I wouldnt go that far, thats pretty funny.

So basically, he is [this guy](http://2.bp.blogspot.com/\_MGkSWvWF0hA/TTW502xBfbI/AAAAAAAAAUo/Fi6NQu0oYAo/s1600/Mr-Wolf.jpg)

Believe me, the bigger the crowd, the less sense comes out. Politicians thrive and build their careers on this senselessness! There are good comments, here and there but the voting aspect of this site should not to be taken at face value. High upvotes in no way guarantees a good post, and downvotes don't mean it's crappy, it`s just the way the crowd shifted on that particular day. The same repost 6 months down the line could get a totally different result.

Wow. That calculator throws an ALERT if you enter a non-numeric value, rather than silently ignoring it.

Yes, we even jokingly called him the Wolf.

I also charged $100 per hour and had no shortage of clients. You're just not charging them enough

I was charging $100 per hour as a freelancer in the US and had no shortage of clients. Hell, I was turning them down before I moved overseas

Don't forget the Rule of Scotty: at least double your time/cost estimates.

&gt; I don't think there's a PHP coder in this world who's worth more than, max, $70/hr, if that. There are plenty of clients out there that think differently. As a freelance developer, if you're offering me $70 an hour and I have three other clients offering me $100 on different projects, I'm turning you down.

Last time I did freelance I charged $80/hr in West Michigan. Seems like a pretty standard rate here, but I have been doing this a while now. When I was just starting out, I charged around $30/hr and lowball estimated the hours because I figured I would spend a lot of time learning on the job and I didn't feel that I could charge for that.

ding ding ding. time is always short - if you can take any project and manage it from beginning to end quickly and adeptly i think your value will go up immensely.

While I don't enjoy that ZF is stuck behind many of its peer applications in PHP 5.2-centric mode at the moment, there is a new version in development that has emphasized performance enhancements and getting the most out of PHP 5.3's new features. Until then, ZF1 still works quite well with PHP 5.3 and its specialized libraries, like Doctrine 2. Long story short, much like any of its peer frameworks, it handles much of the otherwise inconvenient work of building out boilerplate, while offering modular utilities that can be loaded (but don't need to be) to provide common functionality. The entire library can be bulky, for sure, but each running instance of the application doesn't bear this full burden, and it can be streamlined and optimized in all the standard ways. To be fair, I've also considered Symfony, and it looks like an excellent option. I'm planning on spinning up a project using it at some point in the near future, but haven't made it around to it yet, short of the initial configuration, so I can't comment. I just wanted to counterbalance anyone who would stare down their noses at you for even considering it.

Which country are you living in? If you're in Germany or the UK i'd say around 30-45 GBP / 35 - 55 EUR an hour.

Does it estimate rates correctly?

A developer with highly specialised skills.

interesting. currently using geshi for wiki, wonder if this will be easy to switch to.

Curious, why is hyperlight a better syntax highlighter than GeSHi? Side note, I knew Nigel (Oracle) back when GeSHi was first released. Back then it only supported QBasic and PHP. I also remember when qbnz used to host many QBasic projects. (Including my own.) My how times have changed...

I... never did any of that, and don't know what most of it is. I just write code for money, there's nothing terribly complicated about it! I was just doing it in my spare time for some extra cash, fly-by-night, nothing too serious. I received all of my freelance work from previous clients who recommended me to anybody looking for a website. Most of them lived in areas with a high cost of living (while I did not), so I could get away with charging a higher wage as long as I could communicate effectively. I knew these people could afford it, and all of them were happy with my work. Perhaps the American economy is lowering the price of what people are able to afford. I left the United States last year and haven't tried freelancing since I left, it's too complicated

No downloads?

I too am curious as to why Hyperlight is a better syntax highlighter (better on what basis?). I am using GeSHi for my blog and it's by far the slowest part of page generation, so anything that's faster/lighter is interesting to me :)

It depends what you want and what works for you. I personally have no experience with the US market, but what is common here in NL is to have intermediate companies. There's lots of companies that only do one thing, get a network of clients that need work, and get a network of small companies and freelancers that need clients. Connect the two, and take a slice of the cake. Other option is finding a freelance sales person, who effectively does the same as the intermediate. Personally I don't use jobs websites. The rates are too low, I can't compete with those people, and the clients there aren't exactly looking for quality, they are looking for price.

No download and SVN doesn't work, but you can still view the code here: http://code.google.com/p/hyperlight/source/browse/#svn%2Ftrunk%2Fhyperlight%253Fstate%253Dclosed

Define "better".

Just to add a different solution (I wouldn't use this myself, but someone might), you could do something similar to unix permissions: Give each record a user-owner and group-owner field, and write a simple assertion to verify that the editing user is either the user-owner, or a member of the group-owner group. Bonus points if you implement varying levels of access for each (i.e. read, update, delete instead of read, write, execute)

Yes it would be easier, and it wont be a problem if the input is sanitized first. Using a function like this, $section = preg\_replace('([^a-zA-Z0-9\_])','',$\_GET['section']); if(file\_exists('pages/'.$section.'.html')) require('pages/'.$section.'.html'); else require('pages/default.html');

yeah, I have seen this done a few times in the code I have looked at. Usually you would add a wrapper around the object to either: a. add more functionality to PDO or b. make PDO fit more directly with your code (more example having a persistent connection and using static DB calls). Hope that helps!

Why not just use github gists? Aside from not showing in feeds.

Well firstly, there's the feeds, secondly, I feel like it's a waste of GitHub's capacity hosting random code snippets in full git storage. The most important to me though is the ability to change the colors. Probably the most silly reason but I love me some eye candy.

You are joking, right?

Ha. Sounds familiar. Good luck with that.

Our legacy cms must be about 10 years old. Modules are called pages and pages are called sites. Just to start with :).

The answer to this is going to have more to do with the people involved than anything, so making sweeping "this is how you handle it" proclamations isn't going to do. What I can tell you is how I've dealt with this in the past with people I've known, and hope that some of these things might help you too. Remember, though, what works for one person is often poison for the next, so think over very carefully any of these before trying them. Right off the bat, in a few cases, I just never found a way. Sometimes people are too emotionally invested in control, or too afraid of change; there are cases where I was not able to act. But it can help a lot to understand where the "no" is actually coming from. Remember, a lot of people protest for the first reason that comes to mind, but \*\*\*\*\*because\*\*\*\*\* of the first reason that comes to their heart; just because they say their concern is X doesn't mean that's actually what they're stuck on, just what they're thinking about at that time. It sounds like this system was largely made by people who were making it work and don't really understand what they did, and with their apparently spectacular bug record, who could blame them for being gun shy? In a case like that, I wouldn't approach it as a total rewrite up front. First, I'd ask to be allowed to change the low hanging fruit, such as the one-account SVN thing. That's a pretty easy fix, but it's a big deal, so, "hey Tony, listen, I was wondering, could I take the afternoon on Friday to fix the SVN account thing? This is kind of dangerous, what we're doing here, and I don't want us to learn the hard way." Do several of those, one each week; the purpose isn't so much to fix anything as to change his mind about the safety of fixing things. Follow that with a \*medium\* fix; say, "hey Tony, I'm worried about that our database structures aren't the same between machines; I'd like to spend three days making a tool which checks the structure of one machine, then run it against our various machines, to reduce problems." Again, no automatic changes, just a tool to find problems, which you'll fix the same way you always have - by hand, the horrible way. Then it's an automated backup tool. Use low-hanging fruit to reduce the cost of a failure. Finally, a benchmark. Once that's done, say "hey Tony, we need to start unit testing. Could I start spending my Fridays building tests? I know, it seems like I'm losing 20% of my work time, but think about it: every one of those days I spend we can re-do instantly by pushing a button. We'll make each one of those days back during development days in about a month, just by having those checks in place while we're working, I promise." If you just had a string of small, then medium, successes, then he should say yes. Unit testing is a big change: you can use it as an excuse to start pull-up refactorings - so that you can test the pieces, of course - and from there, you've got your system-wide refactoring over time, budgetted into a crystal clear time frame. You don't actually need to do it all up front. Go get Fowler's refactoring. You can do this piecewise, unless you feel the need to go running to some framework. I mean, if the problem is that your bosses don't understand the stuff that needs to be done, you basically have two legitimate options within the company, right? Teach them the code, which seems to not be working, or teach them in decreasingly small steps that they can trust you to fix these things even when they don't get it themselves. &gt; Also the fact it may take 2 devs the best part of a year to get to refactor the monster of an app we have now isn't good in a business sense. This would probably kill the business, and don't forget the prospect that a big lift can fail. Just maneuver yourself towards unit testing, then use that and markers in comments to perform an amortized full refactor. &gt; So Reddit, what the hell do I say? "I'd like to fix SVN this Friday." You can't come out of the blue like "we're starting over," at least IMO. Earn your way up the trust ladder, and do this piece-wise.

There could be instances where this is desirable. One such example would be so your application isn't tied in to using the PDO library (i.e. if you think there's a chance that your application might get moved to a server without these extensions installed). Another example would be if your application connects to more than one type of data source, and one of those data sources is not supported by PDO, but you'd essentially like the application code to remain the same for all. I found this when needing to connect to SQL Server, but the version of PHP I was working with did not have the PDO SQL Server drivers installed.

Excellent advice! I now feel a lot better about my own situation. I've got a limited time frame to build a very complex application, but on the bright side, I am able to make decisions about using frameworks, and how our code is maintained. I feel for the OP, this sounds like a nightmare, but with advice like yours, I suspect he'll have a better go of things.

Speaking from my own experience, I bet the existing code will prevent them from adding new stuff because it'll keep falling on it's arse every time you change something. Sounds like a complete re-write is necessary. Also sounds like they'll never agree to it. Sounds like it's time to look for a better job.

Personally I'd rather use a [JS library](http://alexgorbatchev.com/SyntaxHighlighter/manual/demo/) to do this. As much presentation stuff as possible should be done client side. Having said that, thumbs up for giving people an alternative to GeSHi.

Start small, do it in pieces. If you just started the job you need to be there a little while to build some trust and get that power to make those calls. But no company can afford to shut down for a year to redo things that are working. You have to do it in pieces between adding new things.

- Get integration tests set up covering everything - use separate SVN accounts for each developer - once you have tests covering everything, refactor slowly and deliberately, don't try to go for a full rewrite of a working codebase. you might find this book interesting, also (free PDF): http://scg.unibe.ch/download/oorp/

&gt; Speaking from my own experience, I bet the existing code will prevent them from adding new stuff because it'll keep falling on it's arse every time you change something. This is exactly what happens, some obscure edge-case will cause something to fail. Thank god the support staff here know the system so well that if 1 tiny thing goes wrong they notice it and tell me.

Great advice.

I was reading this post and was like "oh god, that guy is from my old work place for sure ..." but then I've got to the SVN part and was "ok, he's not, all they're using is a shared FTP serwer". Sorry to say but it can be hard to force a complete redesign. We tried for 5 years and every time it was "ok, when we'll have free time we'll look more into the issue". But that time never came.

Having said all that I just said above, I've got to admit that the best advice on here has come from StoneCypher. Did you read his advice? If the advice he's given to you doesn't help, then seriously, just look for a new job...

Posted removed as this is a request for a spam script.

They wouldn't shut down, it would just be a feature-freeze until the new system is up to the same level of the current. Been here for nearly a year now

Remember your bosses are looking out for the bottom line and that's really what matters in business. By the sounds of it there are some critical things you can fix, like the SVN accounts, that wouldn't take long to fix up surely. Get on top of those. Don't go in with a "Lets rewrite everything!" attitude. So many developers are like this and in a business situation it's usually not viable (yes I understand the benefits over time, but a year of coding is pretty epic amount of work). I worked in a company with pretty dodgy code, thing is, every developer that came in wanted to rewrite it, so they all tried and it turned into a mash of different code because each developer had a different vision of how it should be changed. The bosses won't want to rewrite the code. They don't see a benefit because it's a year of coding to get what you already have. Yes you see the code and it stinks, but the customers don't! The company isn't going to get more customers by you saying "hey look we rewrote everything!". The advice I can give is that with companies like these is to live with the code and make small changes over time. i.e. \*\*Evolution not revolution\*\*. You can make the code much better than it is with a bit of rewriting here and bit there. You might not be able to use a Framework but you can definitely make the code manageable. Start with a database class and start using it for new code only. When you get spare time, go back through and change existing code to use the class as an ongoing project. Then do this for other parts of the code. It'll never be perfect, but you can make it \*much\* better than it currently is and you'll feel better for that too. Edit: On the "what do I say" stuff, just ask if it's OK for you to make code improvements every so often to improve code efficiency. :) They'll probably be more likely to be OK with a bunch of smaller changes over time than a complete rewrite.

You could could try doing a \*transition\* instead of a \*replacement\*. All I mean by this, is start using Framework X to do some simple, tiny, things, and then slowly progress. Example, say they want a live chat feature. You could say "Oh, I know how to do that \*easily\* using X! I've done it before!" (who's to know if you're exaggerating slightly) And hopefully they let you. And then you've got one small, tiny, part of the site's functionality working on system X, instead of the monolithic code. Then the idea is to transition all possible modular parts of the system to the new one, slowly. Hopefully after a while of this, you can start showing them real examples of how much better the new system is for writing new things, how much safer, easier to modify, etc it is. It makes you more productive! You get things done quicker! "Do you \*really\* want me to write things three times as slowly using the old code?" Also, as a well-trained developer, you should consider the fact that he \*should be\* paying you to write code he might not initially understand. Because you know code better than he does. If you weren't doing so, you wouldn't be doing your job to your highest degree. Other than that, I'd peak around at other jobs...

Any modern PHP ORM is basically that.

You haven't finished reading that book. Finish reading it.

1. You probably don't need to re-write everything. That's a very common perception of someone who doesn't understand the risks and business costs of a platform rewrite. 2. Find a bunch of important security flaws in the system, and explain that they are endemic within the system and can't be simply patched out.

We have recruiters and firms like that in the US as well. Most of the work coming from them is 6 month contracts at various corporations. If I were to lose my job or was looking for more money that's one good avenue to go. Contracts, while not as stable usually pay a much higher salary and many have the option to hire by the end. Only problem is I'd have to either commute over an hour or move and I'm currently comfortable where I am.

&gt; The bosses won't want to rewrite the code. They don't see a benefit because it's a year of coding to get what you already have. Yes you see the code and it stinks, but the customers don't! The company isn't going to get more customers by you saying "hey look we rewrote everything!". While this is obviously true, the company might suddenly get a whole lot less customers by having the entire system crashing down on them or by leaking all their user data. In cases like this, it's not just about maximizing the bottom line in the short term, but also about ensuring that your foundations are stable. Otherwise, it's more likely than not that the company will fail spectacularly at some indefinite point in the future. I've been there, I was once hired (on an hourly basis) to change some of the core functionality in an existing and horribly programmed system. The change was to rewrite the entire user system in order to support N roles per user, and I was not allowed to redo any part of the existing code. I was the third developer to try, and I failed. There was SQL, PHP, javascript and HTML everywhere, with no kind of distinction between them, and every single page was self contained (there was no includes). Obviously, the company is gone now, without ever getting their upgrade.

Two suggestions: 1) Don't tell them the current code sucks. The people who worked very hard on it are going to be emotionally attached to it and hurting someones feelings is not going to help you. 2) Read Code Complete (http://www.amazon.com/gp/product/0735619670). It speaks a lot about rationalizing a situation and when it is appropriate to do what.

delete everything

This isn't specifically you, but I've seen a lot of people mention this and I really don't understand that attitude. The argument of not wanting a long commute or to relocate. If it's six months and a long way, instead of commuting just get a hotel or motel, negotiate a discount for the extended stay and it's a business cost so you can either put that in your contract as a direct cost the client has to pay, or just anticipate it in your hourly rate. Now that's quite easy for me to say because I'm not tied down, and people may have kids a spouse or whatever. But that's just how it works with freelance jobs, at least in my experience. In a more recent project which way quite annoying to commute and hotel prices where high I basically negotiated that I'd be there on location for the planning stage of the project and tone that down after 2 weeks, to only 2 times a week and tele-work the rest, and later 1 times a week, in the grunt work phase of the project. For planning or the start of any new project I feel it's invaluable to have the face time, but when you are just dishing out tasks and working on the project, skype/chat/phone etc.. works just fine really.

This. There are fundamental flaws in the logic in important features (Signing up for example saves the user data from each page, but if they navigate away there is no way to recover - then the user has to call us). I see a code rewrite will not just speed up our service, but it will reduce support calls tenfold because a lot of phonecalls are from users who can't login as they've forgotten their login number (yes, \*number\*) or they left the registration form prematurely and now it requires human interaction and so on.

You need to throw lots of free time or money at them.

This is what I came to say, put way more eloquently than I would have done. This is EXACTLY what you do.

Looks like GeSHi reparses code every time it generates a page. I'd consider creating a database table to store parsed code, then pulling it out when called any other time. It would massively speed up page generation time.

Yeah, if I wasn't getting married in February I would be more in the same boat as you. Without the commitment I'd be game for working in various cities for contract jobs. However since I am getting married and need the stability for the next year or so that will have to wait. After that's out of the way we both agreed that moving is not out of the question. Also, another side note about my comfort. I train Brazilian Jiu Jitsu and the gym is ~5 minutes from work. Really like the gym I train at and pay a great price there so reluctant to give that up as well. Anyways, good advice for some people with freedom and mobility.

I didn't really read you post, so I'm approaching this as "tldr; the codebase I use sucks and I want to fix it by rewriting everything" Rewriting everything is almost universally wrong. Take steps to make code better over time. Trying to do it all at once will - Waste time - Halt feature development - Halt $$$-making productivity - Introduce more bugs as you fix things, move them around, and forget implementation details. Even worse, you might lose entire features. I'd recommend reading something like "Working Effectively With Legacy Code" by Michael Feathers, which outlines a series of steps and techniques for working with large monstrosity codebases and turning them into good unit testable codebases over a period of time. You don't have to do it all at once, just make it a point that whenever you \_DO\_ do something, make it a point to leave it in a better place than it was. Over time, you'll see a marked improvement in the codebase. It takes discipline, but being able to do something like this effectively is was separates bad coders from good coders, and good coders from great ones. If you don't have the people around you to support the coding ability and discipline necessary, then this will never work. But guess what, a complete rewrite won't work either. So it might be time to bail.

do it modularly... it's what you're going to have to do. Analyze the system piece by piece and slowly break it into functional groups / modules. Start recoding by not making the modules, the cleaning up the spaghetti code so when you start to rewrite everything you can replace things piece by piece, block by block. It'll take time, but management will appreciate this approach. They'll have their system, but a slow improvement where features can later be added. The hard part will be convincing them you can do thing piece by piece without breaking the whole thing. (odds are you're going to have to write some temporary wrappers) If you work in Philly... this sounds eerily familiar to a company I've interviewed with. They might actually be planning a rewrite and you don't even know it.

I don't really know how to respond to that other than to say that I feel you've been sorely misinformed.

It is not a request for a spam script. It's request for an account register-er. It doesn't send spam at all or message anyone. When you say shit like that you remind me of my grandma thinking google chrome is a virus. Educate yourself.

StoneCypher FTW (again).

Get out a pen and lots of paper, document the code as best you can. Then, rewrite and refactor parts of it as you can. Rewriting the whole thing from scratch rarely works well, and is usually the mark of an inexperienced developer. Source: [Things that you should never do](http://www.joelonsoftware.com/articles/fog0000000069.html)

I understand your point about not rewriting, but it's as if the code was never properly developed in the first place, I am stilling finding basic lack of SQL escaping and large loop holes in logic (which I am fixing as I go obviously). That being said, it does still float for now... Also I am not talking about rewriting it my way because I don't like it, I want to put it into a proper, supported framework (Cake) so we can keep things stable and getting help years along the line will be easy there are a tonne of CakePHP/Zend developers. I don't think this sort of investment for established internet-based company is unwise. Yes, there will be a lengthy feature freeze - but after it will be easy to maintain and add features because there will be conventions and all logic will be properly grouped and extendable. Right now if I want to update something there can be a multitude of places I have to look - grep is my friend obviously - but even then functions are spread out so much it's worse than trying to follow the Wordpress core. edit - Great link btw

Both SQL escaping and logical bugs can be fixed one at a time. It's excruciating and frustrating, but it's still much better than rewriting. I highly recommend that you read the article I linked.

Please take a minute and put some [more thought](http://blog.rebeccamurphey.com/lessons-from-a-rewrite-9179) into what [you're proposing](http://www.joelonsoftware.com/articles/fog0000000069.html). There are [numerous articles](http://onstartups.com/tabid/3339/bid/2596/Why-You-Should-Almost-Never-Rewrite-Your-Software.aspx) pointing out the [myriad dangers](http://timross.wordpress.com/2010/03/15/if-you-must-rewrite/) that come with a [full code rewrite](http://www.laputan.org/mud/mud.html#Reconstruction). While your intentions may be good and your desire earnest, that does not inherently result in a successful rewrite.

I've been in the same situation, involving software applications. A particular piece of software stuck out. It was a java-based GUI, developed at low cost in India. Complete garbage, utterly unmaintainable pile of shit. The smallest change requests would take weeks to be implemented, often breaking other features. It was a text book example of spaghetti code created by a too big number of incompetent developers with no concept of object oriented design (almost every method static). Of course management put in place "quality processes", spreadsheets, and 3+ hour long meetings at ungodly hours in order to "fix" things. All requests to rewrite everything from scratch were dismissed without a thought. Too expensive. Too risky. My solution was to rewrite most of the software in my spare time and at home (and consequently learned many good things while doing so) and demoed a prototype a few weeks later. That was it. The GUI looked better, looked more professional, and was noticeably faster (JRE CPU usage dropped from 50+% to 0). I was given time and resources to finish it, and we eventually replaced the original garbage. Maintenance became minimal and a non issue. I guess the key thing here is that you must \*prove\* that you can do better. And for that buzzwords don't work, demos do.

I've seen this too, and it defeats much of the intention of PDO. \*\*PDO IS ALREADY AN ABSTRACTION.\*\* While it's common to build abstractions upon abstractions, it also becomes exponentially harder to avoid [leaky abstractions](http://www.joelonsoftware.com/articles/LeakyAbstractions.html), the effects of which are [non-trivial](http://en.wikipedia.org/wiki/Leaky\_abstraction). Some hint that [all abstractions are leaky abstractions](http://www.codinghorror.com/blog/2009/06/all-abstractions-are-failed-abstractions.html), but that doesn't take away from the fact that layered abstractions introduce even more opportunities for leakiness.

why? for fun?

Thanks for the links, hey raise valid points

If you mean forums, then yes. For now, I'm making a concept of it and I hope I will continue it. Coders on that site will get a special place, they will be "the guardians of the code". I know there are php forum pre-sets on the internet, but I think they don't have what I need.

About what, exactly? Have you somehow missed the volumes written in recent years about how difficult it is to validate an address with a regex?

^ right here, pal.

gack, I hate working with xml in php. use json instead :) I'd offer my services but you haven't mentioned what the budget is (a low budget is usually the problem for programmers suck stories...)

You're right, I didn't post enough information. The service we wanted to integrate with only supports XML, so that's what we went with, although our API supports JSON and XML-RPC. I'll reply once we have a better idea of the scope and budget, as they will probably change now.

It might be bullocks yet that can be why you're making barely $40/h. Working with Magento based sites nets me a hell of a lot more than working on WordPress sites. To a point a WordPress project usually gets a flat fee instead of an hourly rate (hourly rate then applied to work beyond the initial project). You're going to make more money working on larger scaled projects above WordPress. Sucks that you'll need to work in more complex\less entry level code but it'll be worth it come pay day. Side note, personally find WordPress to be very primitive\ugly in the coding department, it's just good because it's quick to get up and running and for a lot of mom &amp; pop sites that's all they need. Not trying to be condescending if it comes off that way, just trying to say there are different pay grades for PHP and WordPress is a lower tiered level.

A better job doesn't always equal a better-managed company. Sometimes it does, sometimes it doesn't. If you are the smartest guy there, there are some advantages to that.

Magento has a good plugin architect called Magento Connect. Another great thing, you can hook into everything (events). For instance today I'm hooking into a method getFinalPrice for products to add customer based discounts on the fly. It's very flexible. The biggest problem with Magento is the lack of good documentation so it makes for a pretty steep initial learning curve. After getting your feet wet it works well. So, it seems it has both things you're looking for, just with bad documentation. Also, most plugins worth anything will cost you. Only free one I found that really stands out is Owebia Shipping 1 which is great (and not written in English...). The lack of documentation\free plugins is another reason you can charge a lot more. Edit: Also, it was not made for cheap webhosting. You'll need a decent amount of memory to run it as it's pretty bloated. Heavy usage of XML for configuration.

Sadly, PDO isn't a very good abstraction. Even a thin wrapper around it can provide more functionality and consistency.

What I usually like to do, when I find such a mess (made by others or my former self): propose a gradual rewrite. When you have to work on a certain area, rewrite that area. Keep MVC in mind (for the future) while doing it. Refactor this specific code into OOP libraries. But first, get a grasp on the structure of the site. Think about the right way to organize this structure into meaningful objects. Write documentation for this structure. Then do every refactoring according to this structure. This is hard, longtime work. But if you want to stay in this company for long, it will repay itself. Once you organized most of the code, it'll be much easier to make a transition to MVC... Complete refactoring of a large codebase is \*really\* dangerous! So many possible ways you could start with the best intentions, but then fuck it all up, because you overlooked some weird specifics...

couldn't have said it better.

Mandatory security update to 5.3, and then watch the fun.

I agree about not running directly on the production site. At this point, I'm desperate to get where I can get started on making changes to the code that's not working properly. Can you set breakpoints in Notepad++? I've used it before, but never for that sort of action. Also, I'm still left with the problem of the site not actually working locally on my machine.

rewriting from scratch is the "single worst strategic mistake that any software company can make" http://www.joelonsoftware.com/articles/fog0000000069.html don't fucking do it.

good luck getting people to work for free

How I did it (latest XAMPP, Netbeans 7.whatever): \* [download the 32-bit thread-safe VC6 PHP5.3 version of XDebug](http://www.xdebug.org/files/php\_xdebug-2.1.2-5.3-vc6.dll) and save it to XAMPP's php/ext directory \* edit your php.ini file to replace any existing "`zend\_extension=blah/blah/php\_xdebug.dll`" with "`zend\_extension=C:/Users/me/xampp/php/ext/php\_xdebug-2.1.2-5.3-vc6.dll`". Make sure you get the name and path of the dll right \* also in php.ini add (or uncomment) the line `xdebug.profiler\_enable = 1` \* make sure Netbeans' PHP options have the Debugger Port set to 9000 and the Session ID set to netbeans-xdebug \* restart XAMPP That's about it.

I'm not sure what you mean by "use an IDE". I use an IDE to write my code, and I either develop locally, push to a shared test server, or deploy to a production server. It is the same code, all written in an IDE. I use the IDE to control Xdebug on my local machine. We do not run Xdebug on any production machines.

Otter, Thanks for the .dll location. I had downloaded one before, but had no idea where it needed to go. I've set all the properties as you mentioned, including the zend\_extension in the .ini file, but I believe I've still got a problem with my Apache setup. Did you have to do anything for the Netbeans project to run under Apache on your setup?

Kelroy, I would do just that (I've got an old Ubuntu box sitting underneath my desk), but I need to be mobile on a laptop with this setup. I am a grad student, so I need the ability to take my work with me from home to campus. Which is why I opted to use Netbeans or some IDE for what I need.

Considered, but not valid. I'm under my parent's ISP, on this box, which my father already has used the dynDNS on his own machine. Under our ISP and router settings, we can only utilize one machine at a time to be reached outside our LAN. Which brings be back to wanting to keep it all central on one machine for testing.

vmware server on your windows machine.

I set the Project's Run Configuration to "Run as" "Local Web Site" and set the project URL to "`http://localhost/whatever\_my\_app\_is\_named/`" and use the "Browse" button to find the index file. If you're trying to debug from an entry point other than the index file, make sure you launch the debug session using the "Debug File" command and not "Debug Project". In my experience, Xdebug problems are usually server-related rather than client-related. Looking back at my php.ini, my relevant settings differ from what I said (and the article): \* `zend\_extension = C:\Users\florinb\xampp\php\ext\php\_xdebug-2.1.0-5.3-vc6.dll` \* `xdebug.profiler\_enable = 1` \* `xdebug.remote\_enable = 1` \* `xdebug.remote\_host = "localhost"` \* `xdebug.remote\_handler = "dbgp"` \* `xdebug.remote\_port = 9000` I've got to warn you--if you are used to Microsoft's debuggers, prepare to be underwhelmed by Xdebug + Netbeans.

I would say CodeIgniter, simply because of it's clear, easy to understand documentation with examples. That's one of the reasons I bounced over to it from the newest Kohana. I don't like documentation that makes me say stuff like "Hey guys, how do I internet?" That being said, the only "new features" in CI that I know of is its abandonment of PHP4 support, but that's been that way for about a year. If Kohana would get it's act together with documentation, I might switch back. I really liked their Template Controller.

http://www.elefantcms.com/ After coming back to PHP from a couple years of Python, JS and Ruby, it became obvious that the popular frameworks in PHP are just too verbose. So I started a new one with conciseness in mind, and playing on PHP's strengths. Controllers appear as ordinary scripts and routing to them is logical and automatic. Secure defaults (SQL injection, XSS prevention etc), concise + compiled templates, loads of documentation, unit tests, etc. And a built-in simple CMS, because that's the first app most sites need on top of a framework. It's in RC for 1.0 now with no known bugs remaining, so a 1.0 will be out any day now (prob after I record a few screencasts for the site first).

Funny, Reddit did that themselves when they moved from Lisp (I believe) to Python. It worked out alright for them. But yeah I still agree with you.

I'm someone who started out in much the same way this throwaway's boss did, but i've been learning and updating my code as i've learned but i always feel like there's something else that could be improved upon, and various things that could be updated. is there a website or resource i can use to learn about various things i can do to improve my code? right now i still use mysql\_query, i'll be switching that over to pdo once i figure out why it's not working, but things like function files. i've nearly stripped all my function files down and placed them in appropriate classes, but there are still some generic files that i don't know where they belong, and i've put some in a base class but im not sure if it belongs there either. i'm the only one working on it, but it'd like to setup an svn to track changes. i'd also like to use test cases but i don't know anything about them, and i don't know how it could benefit me to use them. anyways, i'm not interested in rewriting it all, i just switched it over to object oriented code a while back and i've done my best to keep view code out of the classes, but i would like to improve it in whatever various ways i can. thanks.

In terms of the big existing players, the "latest" (and I'm sure some are bound to argue "greatest") is no doubt Symfony 2. They launched this major overhaul as stable in late July, and it has no shortage of fanfare. Symfony 2 is provided as drop-in components or as a "standard edition" application framework that provides the means to do very rapid application development at the expense of flexibility. I don't have any experience using symfony components outside of the standard edition, but I can say that their claim to be a rapid development framework is absolutely correct. The forms component, in particular, is really impressive. By "binding" your models to types of forms, you can have your form objects built on the fly; symfony does its best to figure out which field types and validations would be appropriate based on the way you have your model setup. For simple and common usages, this works extremely well. For more advanced stuff, you always have the option to configure the fields manually. As with many components in symfony, if your models are doctrine 2 entities, then your forms can make even more impressive decisions. For example, if you have an association defined in your doctrine 2 schema, the form component will identify that and the corresponding form element will be a select field that is pre-populated and validated against all available options from the database. I used symfony a lot recently, and I've been happy enough with it. The only significant problem in my eyes is that once you decide not to do something the "symfony way", then all of that "magic" that makes symfony so appealing suddenly falls away. At that point, I don't see any big reason to use symfony 2 over other frameworks that are specifically designed not to have any one "correct" way to do things. Zend Framework 1 is still used all over the place, and there are very valid reason for that. It is flexible, its documentation is extensive, and it has an unbelievable amount of field testing. It is a first generation framework though, and it is still plagued by some poor decisions that were made. Its performance is pretty crappy out of the box as well. You definitely need to tweak a lot (like not using Zend\_Application or the ViewRenderer) in order to use it in a high-traffic situation. That said, there is little that you outright cannot do in ZF1. ZF2 development is very active right now, and the work the developers are doing is really impressive. They've obviously learned a lot from their past experiences with ZF1. While they've also adapted some of the best features of other modern frameworks (like symfony 2), the developers are not just repeating what they see elsewhere. ZF2 is turning out to be much more flexible than ZF1 while simultaneously considering the requirements of a "standard" ZF2 MVC application. The MVC system itself in ZF2 is totally changed from the ZF1, but it is flexible enough to conform to a lot of different styles. Just a few hours ago, Matthew Weier O'Phinney demonstrated how the flexibility of ZF2 could be used to very closely mimic a ZF1 application. These are the types of experiments early on that the devs seem to be doing in order to not only make the framework fast and flexible but also to help developers make the transition from existing ZF1 projects into ZF2. Link to Matthew's ZF2-ZF1 demonstration: http://zend-framework-community.634137.n4.nabble.com/ZF1-style-application-built-using-the-ZF2-prototype-td3833997.html \*\*tl,dr: Overall, I think this next generation of frameworks that were recently released or will be released in the near future is awesome. As in the past, each framework offers something unique, but much more emphasis is being placed on best practices and performance.\*\*

This is actually how I currently deal with it, I parse my blog posts, which includes generating the syntax highlighting, and then store it in Memcache for speedy retrievals :)

And it plays well with other frameworks for added functionality: \* https://github.com/jbroadway/elefant/wiki/Elefant-with-other-frameworks \* https://github.com/jbroadway/elefant/wiki/Elefant-with-Backbone.js

How could you name something so concise and lightweight 'Elefant'?

The name was derived from a code name from a private project it came out of, and I just kept it... At least the Elephant is PHP's mascot :)

Cake 2.0 is in RC2 right now, I've build an internal asset management system off of that; Lithium is also interesting. I'm using SlimPHP for smaller stuff, and I like it a lot. Simple bootstrap, file organization, takes about 5 minutes to get it going for what I'd need. Good enough for me. Those two are my go-to frameworks.

I heard codeigniter development has slowed and kohana was to take it's place? I could be wrong on that.

If the wrapper class doesn't add anything, that is to say if the class is completely transparent, then such practice is completely useless.

I'm not particular interested in developing forums at this point, but I have been working on a project recently that may help you get started: https://github.com/epixa/Talkfest It's a pretty basic discussion board that may be a little bit more like reddit/hacker news than a traditional forum, but I don't know how "traditional" you're interested in staying for your project. There isn't any fancy functionality going on, but given that my interest has waned on the project, perhaps you can find some use in it.

Zend Server Community Edition + Zend Studio 8 = pretty robust debug environment with breakpoints and line by line code execution with a visual display of variables. It takes a little effort to get it set up and working but it's pretty amazing stuff.

I've been working on the documentation for Emergence. http://emr.ge Ruby-like AR implementation, Virtual File System (no other framework has a VFS)

I had not heard that. I do know that until Kohana gets clearer documentation, they won't have the same user base as CI.

I'm really enjoying Lithium, myself. The documentation is a bit out of date in places and is lacking entire sections occasionally, but overall I've managed to get it working for what I need and I like it.

Yeah, if the docs were better, I would have used it over Cake2.0, but since I was in a time crunch, better to go with what you know

so your proposed solution to that problem is to rewrite the entire codebase?

I just a standard base of reusable code I have built up over 7 years of php projects.

Listen to Otterfan - xdebug allows you to place breakpoints anywhere in your code. Don't listen to kelroy. His suggestions seem to be kind of... wrong. Use an IDE because it has autocomplete. notepad++ is not an IDE, it's a text editor.

If you don't need it to be in netbeans and aren't able to get those methods to work, I found the [Komodo IDE](http://www.activestate.com/komodo-ide) debugging very easy to set up (even places the DLL files/edits php.ini for you), it's not free but does have a free trial that is full featured (and an alpha version that is free until the next release)

We've thought about it, and it's hard to give a budget without a timeline. I think ideally we'd like to pay about 3000 for this first bit of the project, however much time that takes.

It's 2011, a Ruby-like AR implementation is not exactly a recommendation any more... And one look at the docs shows that the so-called virtual filesystem is just the filesystem inside the web-based IDE, not an actual part of the framework. Too bad, because I could really use something to abstract away the filesystem, and allow for easily porting apps to different storage solutions. And testing of course.

As a ZF1 user, I've been worried about the progress of ZF2. Symfony2 is currently miles ahead. For a new project that should have a lifespan of +3 years, I would definitely consider switching to Symfony.

Well most frameworks fail at AR. They give you a model where you have to fill in SQL anyway. That's not AR. It may look like it in usage but the point of AR is to not write any SQL at all. Also our VFS isn't web-based. It has a PHP hosted WebDAV connection which any client can connect to. It doesn't have to be a web browser. The cool thing about abstracting away the file system is multi-site inheritance. In essence the site hosts itself. We're working on a NodeJS implementation to connect to the VFS so that it doesn't have to go through the site.

I've seen this crap here before. It's really, really badly written spaghetti code with just a faint hint of OO-design. This is exactly the kind of coding that gives PHP a bad name. WTF have you been doing with Python and Ruby? Most Python and Ruby coders I know would see this as a validation for all their PHP jokes...

reddit was \*tiny\* when they made that switch, and they did it mostly in a weekend. If you can rewrite your software in a weekend, you don't need to ask permission. Just do it, and announce the launch of a new improved version on Monday.

You're almost certainly wrong, and aren't experienced enough to know it. Others have linked the relevant Joel on Software post, so I'll just add [jwz's resignation letter](http://www.jwz.org/gruntle/nomo.html) which also covers the dangers of rewrites. (Note that while one could view Mozilla's rewrite as an eventual win...you have to really be ignorant of the history to realize that the rewrite nearly cost Mozilla everything. It was a mistake, despite the fact that they eventually powered through a half dozen years later. At a time when Mozilla needed to be innovating and competing at its most aggressive, it was conceding the web to IE. It's a miracle of history and dedication and stubbornness that it eventually caught up to where it had been before the rewrite and began to regain the web after many false starts.) Rewriting is a mistake. Refactoring is awesome. Do less of the former and more of the latter. In a year or two, you'll have code as beautiful as you are capable of producing, and you'll never be in a situation where your "new and improved" software is demonstrably worse than the old software, which is guaranteed if you rewrite from scratch.

[FuelPHP](http://www.fuelphp.com)

I've been starting to tinker with [Yii](http://www.yiiframework.com/) a bit. I'm curious what others think of it.

Are there any specific areas of concern for you regarding ZF2? Until very recently, I was also very worried about ZF2 development. In fact, I lectured Matthew at PHP Tek in May (harassed may be a better term, as we were all well into a night of conference-sponsored drinking) about how the ZF2 development seemed completely stagnated at best and going in the entirely wrong direction at worst. At the same time, I was getting my first real taste of Symfony 2, and I was incredibly impressed. Coming from a long background working with ZF1, I was blown away by how Symfony took care of so much functionality that I otherwise developed and maintained myself in my legacy projects. I also loved how Symfony implemented their bundles (modules in zf terms); it is infinitely more flexible, powerful, and extensible than the ZF1 equivalent. But I hit a wall with symfony the moment I started getting into more complex stuff. This is partially because the documentation is non-existent once you get passed the most straight forward, common parts of the framework. I also felt like I was fighting to implement certain pieces of functionality (e.g. acl based domain object querying) and that symfony was constantly fighting back. It got to the point that the only advice I was given was basically "this isn't supported, so you'll have to break out of the system entirely to make it happen". Generally I like this sort of advice as I feel a framework shouldn't restrict you too much, but when I really considered the functionality I'd be losing by breaking out of the system, I couldn't really think of a good reason to keep using symfony. The "magic" is why it is impressive; and it just isn't that impressive once you can't use the magic. As for ZF2, I've been watching the development over the past few months (albeit not as closely as I should), and I've really been happy with it. A lot of the issues I have with ZF1 (e.g. a totally worthless module system, a bloated/inflexible MVC system, poor extensibility, and crappy base performance) are finally being addressed. I'm hoping that they carry this momentum through release.

Could you elaborate? I haven't looked at elephant yet

In my very brief look through their documentation, I'd say this is more along the lines of well-written spaghetti code. I still wouldn't consider that a "plus", but I do think it is a fair analysis. It looks to be a good attempt at mimicking object oriented frameworks procedurally, which while an odd thought to me, is at least interesting. I can't imagine I would ever consider elefant as the right tool for a job, but that is more due to what it is trying to accomplish rather than how it accomplishes it. If you're dead set on writing procedural code but want a framework to help keep things structured or organized, then I can't think of a better option off the top of my head, I guess.

get nusphere's phpED

This sounds like a horrible idea for anything with traffic and also a massive way of opening yourself up to security holes. Perhaps I'm paranoid though?

I doubt this will ever happen as they don't want to have lots of newbies. It's not aimed at that. However, if you can get over that and read some code or ask on IRC, it's awesome and has a really active community.

Looks like you may have had a bad experience with an IDE planting odd save or config files throughout your website tree. Was that it? Or is there another reason you discourage the use of IDE's for PHP scripting in general?

They've recently moved over to Github for code hosting and it's pretty active (make sure you look at the dev branches).

What would their reasoning be for not wanting newbies? I think a nice medium could be found between newbies and the hardcore developers. It makes sense to me that to generate a good user base, you should appeal to those of many skill levels. Then again, I don't know what goes on in their heads, so maybe I'm thinking about it the wrong way.

SilverStripe are working pretty hard on their next release, they are pushing the framework that underpins their CMS into the limelight a bit more. Its quite a nice framework, but they are implementing some of the better features seen in other frameworks for their 3.0 release, plus it is backed by a successful company and has some pretty smart developers. So its not a bad choice, worth checking it out.

The guy has serious issues.

Horde has a VFS class system. (They share it with PEAR\_VFS though.) - Strangely it does not integrate with WebDAV or anything per default (last time I checked).

Actually regarding this question, we have one developer in our company (I'm a linux admin) and we are planning to do ecommerce, our developer is recommending that we use Zencart, I tried to recommend Magento but he shot it down saying its so complicated to setup. Which framework would work best for an ecommerce shopping cart type app?

This post probably belongs in some sort of PHP jobs subreddit rather than here. But regardless... I'm starting to look for part-time, solo contractual work. I'd be happy to at least review the requirements and provide you with a timeline for delivery or at least comment on feasibility. Feel free to message me. To be honest, though you were vague on details, $3000 seems a bit much to pay for something like this. Do you really expect this part alone to take ~100-150 hours?

I'll offer my services. Detailed specs are something that is so rare. I'd love to have detailed specifications, it would make life so easy.

I expected it to take somewhere around 60 hours and pay around $50/hr. This is the first project I've managed, so I'm not sure if these are realistic numbers. That's why I was asking :) I actually hadn't planned on finding developers through Reddit, but this is a welcome surprise.

[Recess](https://github.com/recess/recess) - RESTful PHP Framework. Just found this for a friend of mine. It's not bad

WebDAV over HTTP is inherently insecure but over HTTPS it's fine. The files are stored in the 'real' file system so as long as you resolved which file is which you can cache that. From then on the file loads as quickly as any other static file.

They do. This is true. But they don't use it to host the site's assets which is where site inheritance feature comes into play.

Code you wrote yourself. You just can't go wrong.

I'm actually in a same situation, but on the "employer" side of your story. The system I am using was an off the shelf system that contains coding conventions and styles from the 90s. It suffers from a lot of the same issues you've described, yet I am in no rush to start a new system from scratch? Why? Because of the following reasons: 1. Rewriting from scratch is guarantee to introduce bugs and unexpected behavior. 2. A lot of the process flow, use cases, and fine details will be left out or assumed because of lack of documentation. 3. Using a modern framework and coding conventions and styles is tempting, but that would require discipline from developers to code completely from scratch and not copy and paste from old code. 4. There's no guarantee that the new system will be any more organized, loosely coupled, or modular than the old system when it's all said and done. The way I am approaching the issue is by refactoring and slowly improving the system over time. That way, changes to the system can be tested without affecting (hopefully) other sections of the system.

Zencart is pretty dead. I was dealing with some zencart stuff a while back and their site/documentation was down most of the time I was trying to work with it.

Go with opencart. It uses MVC and seems pretty straight forward to customize/extend.

I know what you mean man...as a ZF1 user I just switched to Django...not for that reason though...I just wanted to learn python...so far so good. ZF2 is supposed to come out with Doctrine which is kind of cool, but having used ZF1 + Doctrine 2 myself,...it's still hard to get things working.

Eclipse with the PDT package lets you set breakpoints and single step PHP programs.

I've been playing around with the [Fat-Free Framework](http://fatfree.sourceforge.net/) lately, and have been quite impressed with it. It's MVC, but doesn't force you into any particular directory structure or file layout. Despite the "fat-free" monkier, it throws everything but the kitchen sink in there: ORMs for PDO/MongoDB/SQLite, routing, logging, template engine, captcha, internationalization, wrappers around popular Google APIs... the trick is that it's all modular, so you only include the classes that you need. It's very well documented and very easy to learn, especially if you're new to frameworks or MVC concepts.

I second OpenCart. Zencart custom development is dying off, and Magento really is quite complicated to get set-up right.

You don't. Go and read "Refactoring" by Martin Fowler instead.

Sure, but realistically how much do those support calls cost the company, compared to 2 full time developers for a year? You have to look at it from a business standpoint, and from a cost/benefit analysis it rarely ever makes sense to rewrite the entire codebase. The whole login issue can be fixed without rewriting the codebase, much quicker and with much less cost. Honestly you need to take baby steps and tackle one issue at a time and gradually improve the codebase. It's the only way you're going to get anywhere with it, and the only way that will make some sort of sense. And if I were you, I'd start by thinking of a way to do a graudual MVC rewrite in place. By that I mean, you start up a new MVC app using the framework of your choice, but leave all the legacy PHP files around. You're going to need either a gnarly .htaccess file or a really interesting front controller, but you can get to the point where you can rewrite functionality bit by bit. Maybe you start with the login/logout/user system and you have some stupid bridge code that makes it so existing .php files can access the new user data or whatever. It'll be unique to the app but hopefully you get the idea. Once you get a foundation in place like that, so that you can run legacy code and new proper code side by side, you're on the path to a nice rewrite without interruping the business for a year.

This is another excellent point. Often times the best way to sell the company on something is to just do it. I have on several occasions undertaken partial rewrites or experimental architectures in my spare time, and if you do a good job and show what it will be like, you now have a strong case to show the boss. As they used to say in college english classes, show, don't tell.

I'm going to have nightmares now.... thanks.

There's really no difference between creating a web site, and creating an API. A website receive requests, performs some amount of business logic, which includes calls to a database and other back end services, and sends HTML to the client. An API does all the same things, but sends XML/json to the client instead. Any framework you would use to build a website is suitable for building an API.

You'll want to be careful. remember the saying "before you tear down a fence, find out why it was put up in the first place". Now this could mean all kinds of things, but most likely it's your boss' business style.. to make iterations and improvements to his idea rather than starting with a complete vision-- this is common and often the natural progression of things. Or maybe your boss didnt have the money to invest before and now wants to. More likely the former. For instance, your new system may be much better but it will be different. Change will be required. People hate change. \*Where is the buttony thing that used to be here? Why doesn't this site do this? This site is a wreck, lets bring the old one back. arguing. arguing. you're fired.\* Beware of the hero mentality, wanting to drastically change everything. Chances are you will create something that in your boss' eyes will be very similar to what he has now (i.e. you wont get a lot of accolades, and could get fired if you fuck up). Lastly, you may try a fishing approach. When you're talking about some obstacles you have now and will continue to have you may explain it like this: What we have now is sort of like a shack that was built and then many rooms and improvements were added on, what they used to call a guinea mansion. To put water or electricity over here we have to do all kinds of janky wiring and questionable stuff. It may be smarter now to stop investing in the guinea mansion and start building a mansion (within a framework or something) where there will always be room and forethought to build in whatever comes along. And lastly if you build something perfect quickly you may build yourself out of a job.

This is true, but some make it easier than others. For example, if the framework gives you a request/response context so that your /widget/:id action can return json or html depending on the request (say widget/15.js) this does simplify the task. Of course this is irrelevant if your API is not essentially just a mapping of existing URL's to json format.

I don't know what sort of functionality your code provides, but perhaps the year-long re-write into a framework could be a much shorter time to port into a CMS like Drupal.

my brain hurts

Not without some sort of additional parameter somewhere in the URL indicating it should be serving an image rather than HTML. The point of URLs is to indicate the location of specific resources. Having the same URL point to two different resources depending on context is not a great idea. A better idea would be to (for example) have /images/blah be the HTML page and /images/blah.jpg the image itself.

For simplicity as a feature, lately I've been all about [Silex](http://silex.sensiolabs.org/).

Thanks for the comment, and I have considered this but the idea is for the HTML page and the image to have the exact same URL. No practical reason for this, just doing it for the challenge. I was able to get it working before posting this but it doesn't work in older browsers sadly. Was fishing to see if anyone else had any ideas. Still though, I appreciate the support.

I certainly don't disagree with that, and I think the ability to extract the view from the business logic -- so the same business code can be used regardless of output -- is an important feature on which to judge a framework.

Thanks.

I used [this REST API library for codeigniter](http://net.tutsplus.com/tutorials/php/working-with-restful-services-in-codeigniter-2/) and absolutely love it.

You could check $\_SERVER['HTTP\_REFERER'] and if it's the script name then output the image. People could eff with the referrer but in general it could work.

Get yourself Virtualbox and set-up a linux distro to run inside of windows. It's seriously going to save you a lot of frustration.

You can work with the HTTP\_REFERER header. So if the HTTP\_REFERER is www.example.com/images/blah.jpg, then you serve up the image properly (as an image/jpef) but if it is anything else, you serve up the HTML wrapped version of it. Typically, sites do this to check if the HTTP\_REFERER is from the correct domain, otherwise serving up an HTML based, usually ad-ified version of the image.

I would not recommend doing this. These are all affiliate links. Those hosts pay to be on the "top ten" lists. Check out [WebHostingTalk](http://webhostingtalk.com) for less biased, and unpaid, reviews.

At a glance, I don't see much logic in your views that doesn't belong there. I think one of the hangups you may be having is that you're using PHP as your templating language but you're not really using it LIKE a templating language. For example, you could do this: http://pastie.org/2577472 ...rather than doing the loop that you use in admin/pages/dashboard.php. Other than a bunch of unnecessary/redundant code in your template, you don't seem to be doing too much business logic (unless I'm totally missing a particularly bad template). In my example, you would pass the result of getContentList(5) to the view as the variable $posts rather than calling it from the template itself. This bit of separation would allow you to do things such as defining the method's parameter based on a GET variable (restricted to appropriate limits, obviously). I recommend trying out a templating engine such as twig. I'm not generally a huge fan of them, but they will definitely help you better understand both how and what code you should be accessing in your templates.

I would love to have that as well. I've actually started a project on Github for exactly that, but I have not gotten far at all.

I've used it for a couple of years - once you get into the mindset of it it's quite quick and easy. That said, I'm not very experienced in other frameworks - I chose it because at the time it was getting good reviews.

Thank you very much. This makes quite a bit of sense. If you'd like your name in the source for that snippet, please leave it as a reply, and as long as the code is in there I'll credit you. :)

Per-say?

&gt; The forms component, in particular, is really impressive. By "binding" your models to types of forms, you can have your form objects built on the fly; symfony does its best to figure out which field types and validations would be appropriate based on the way you have your model setup. For simple and common usages, this works extremely well. For more advanced stuff, you always have the option to configure the fields manually. As with many components in symfony, if your models are doctrine 2 entities, then your forms can make even more impressive decisions. For example, if you have an association defined in your doctrine 2 schema, the form component will identify that and the corresponding form element will be a select field that is pre-populated and validated against all available options from the database. Cake's being doing that since v1... (v2 coming out just now). Glad to see I've got some flak to fire back at all the cake-haters :) (btw, I was hoping to go and learn symf 2, but my host only supports php 5.2, which is what Cake 2 decided to support. And yes, it's too much of a hassle to find a better deal than what I currently have atm)

I'm curious, what is so wrong with this post that it needs 5 downvotes?

man, 100%. I know this post was on frameworks so the whole "roll your own" thing might be unwelcome, but if you slowly build your own framework over a couple projects- you'll have a better result and be a better coder because of it

[http://wiki.netbeans.org/HowToConfigureXDebug](http://wiki.netbeans.org/HowToConfigureXDebug)

I also recently picked up Django (I've used Python a bit for GIS scripting) - very impressed so far.

How'd it happen?

I really like Yii. I'm not the best with PHP, and it has helped me complete tasks I otherwise would have taken ALOT longer to do.

More detail please? Or a possible blog post about this, OP?

magento or shopify

Also, not saying this was your problem, but, for the Flying Spaghetti Monster's sake, \*\*stop using the original MySQL functions.\*\* If you're not using MySQLi or PDO, you're doing it wrong. The only reason the official functions still work is for backwards compatibility -- they're technically deprecated.

I'd like to know, so I see if it's something that my PHP based sites would be vulnerable to as well.

You were doing fine until you mentioned twig. If a template engine has to "compile" in to php, it's not worth using.

Re: Form generation in ZF This generates a from from the doctrine model at run time: https://github.com/jhartikainen/zf-modelform Documentation: http://codeutopia.net/blog/?s=ModelForm Or this one generates the forms using Zend\_Tool: http://framework.zend.com/wiki/display/ZFPROP/Doctrine+1+and+Zend\_Tool+Integration+-+Benjamin+Eberlei

From what I vaguely understand most of them use it every day, some are paid to work on it and don't really need to grow the user base beyond what it is. It's a great framework; the API is (fully, I believe) documented - you can learn from the basic normal docs there are, but if you can't read code or ask questions you're in trouble. This covers a large amount of what you need; http://kohanaframework.org/3.2/guide/kohana/install

They're also really active on #kohana on freenode

I believe that might be true occasionally in the end, but what happens to the older projects where you've started with a half developed system? How do you maintain them? "Oh, sorry, that didn't exist back then..."

The fact WebDAV is implemented in PHP worries me. I mean, what's wrong with SCP? It's just another place something could have a security hole.

Would you please elaborate more on the intrusion, and the said "good logs" stuff for a newcomer?

This is what is known as a house of cards.

Can you send me your requirement?

Nice try, secret representative for the PHP security suite software cartel marketing department.

Thats what i thought. i have been using PDO/sqlite3. my database choice aside, i really like pdo, seems a hell of alot cleaner.

Free? No.

This is a ridiculous response, PHP can be used securely. You just have to know wtf you are doing, like any other language. Stop expecting everything to hold your hand...if you want to code, learn to code.

Most of the posts here are being downvoted. I don't think it's the bots...

Python websites can also get hacked. Same goes for Ruby, C#, Perl and Klingon. Stop being stupid.

Name one Klingon website that was ever hacked. I doubt you can. Clearly Klingon technology is the way to go.

Nothing wrong with using sqlite! I've found it handy for Android apps myself. I just was assuming a large majority of PHP devs are doing so on an AMP stack.

Django = framework written in python. Compare to something like symfony - a framework written in php which does exactly the same kind of query syntax (and stored queries) as you are a fan of in Django. If anything, php's insecurity stems solely from its ease of use - any inexperienced kid can write "functioning" applications in it without first learning basic security principles.

What's wrong with a template engine compiling to PHP? (both Twig and Smarty do this) Cached, compiled templates let you take advantage of opcode caching and PHP's native speed. PS. what would you recommend? something implemented as a C-based extension to PHP?

Could you give some examples of your current work? it might be easier to suggest something new then.

http://ci.wikiop.in/ and http://nevuxab.co.cc/ are the two main ones Also working on several PHP-Fusion mods on and off.

1) If you are using mysql\_query instead of PDO or some other abstraction layer you do NOT have a clue about SQL injection and you need to go back and do your research. 2) If you do not know the basics of how CSRF/XSRF, XSS, and other exploits work you should go back and do your research. A web developer \*needs\* to have a general understanding of all of this and much more. It does NOT matter what language you are talking about. There are common, well known, and well documented methods of \*\*easily\*\* countering these risks within PHP. So once again, if you cannot be bothered to do your research you should just stop now. You are a C developer...fine, that doesn't make you an expert web developer. You have to know about these security issues no matter what you write...its like knowing the basics of not running random binaries you download without any sort of virus/troject/etc detection. To do so would be insanity.

The reason you can't name a hacked klingon site isn't that they don't get hacked. It's that you don't survive long enough to brag about it. The klingon method of defending against SQL injection involves Bat-leth injection.

Can I get something about not using MySQL functions so I can show people I colab with?

SQLite is not meant for concurrent access, and will fall apart under load. It's meant as an embeddable database for local apps (eg firefox uses it for bookmarks and your browsing history.) Unless your case is very special, SQLite is probably not the right choice for a web backend.

I'll definitely try to explore OpenCart, I've had previous experience with zend framework before and I read somewhere that magento was made using it, so I might also give this one a try.

If you figure out what happened, two good things can come of you telling us about it: 1) other people can learn from what just hurt you, and 2) we can look for variations on a theme to help make it more difficult to repeat, on your behalf. If you haven't figured it out yet, please remember to be open to client-side approaches. If you don't consider those, a lot of attack vectors will be missed.

The MySQLi manual page explains it nicely, but it's really quite simple: parameter binding immediately invalidates string injection, perfectly, with zero programmer effort. This means it cannot be forgotten or botched. That's not every attack, but that's a surprisingly huge class of them.

Guys, remember, he got hacked six hours ago. It's gonna be a while before he responds - he's got stuff to clean up.

Exactly. A stupidly high number of exploits over the years have been based around holes allowing SQL injections.

I've used it for about a year after trying out various other frameworks (codeigniter/cakephp etc). While it doesn't have the largest developer base, it's really quite well thought out. With other frameworks I usually seem to end up hitting a wall where the framework makes something harder rather than easier, thus far no such barrier has appeared in yii :)

What host were you using?

I just find it hilarious how wide open the various sanctimonious blog posts about discovering SQL injections are to XSS, CSRF, JS prototype injection, hosting injection, et cetera.

You made a PHP block for a comment?

What he gave you: &lt;?php foreach ($posts as $post): ?&gt; &lt;tr&gt; &lt;td&gt;&lt;a class="recent\_links" href=""&gt;&lt;?php echo $post-&gt;getTitle(); ?&gt;&lt;/a&gt;&lt;/td&gt; &lt;td&gt;&lt;?php echo $post-&gt;getAuthor()-&gt;getUsername(); ?&gt;&lt;/td&gt; &lt;td&gt;&lt;?php echo $post-&gt;getDescription(); ?&gt;&lt;/td&gt; &lt;td&gt;&lt;?php echo $post-&gt;getTagsAsString(); ?&gt;&lt;/td&gt; &lt;?php /\* &lt;--- that may be a bit nicer than calling another model(?) \*/ ?&gt; &lt;td&gt;&lt;?php echo $post-&gt;getDate()-&gt;format('Y-m-d'); ?&gt;&lt;/td&gt; &lt;/tr&gt; &lt;?php endforeach; ?&gt; Rather less silly #1: &lt;?php foreach ($posts as $post) { echo '&lt;tr&gt;', '&lt;td&gt;&lt;a class="recent\_links" href=""&gt;', $post-&gt;getTitle(), '&lt;/a&gt;&lt;/td&gt;', '&lt;td&gt;', $post-&gt;getAuthor()-&gt;getUsername(), '&lt;/td&gt;', '&lt;td&gt;', $post-&gt;getDescription(), '&lt;/td&gt;', '&lt;td&gt;', $post-&gt;getTagsAsString(), '&lt;/td&gt;', // &lt;--- that may be a bit nicer than ten million context changes '&lt;td&gt;', $post-&gt;getDate()-&gt;format('Y-m-d'), '&lt;/td&gt;', '&lt;/tr&gt;'; } ?&gt; Rather less silly #2, with the benefit of having less repetition, being easier to extend and easier to change: &lt;?php foreach ($posts as $post) { echo '&lt;tr&gt;'; foreach (array('&lt;a class="recent\_links" href=""&gt;' . $post-&gt;getTitle() . '&lt;/a&gt;', $post-&gt;getAuthor()-&gt;getUsername(), $post-&gt;getDescription(), $post-&gt;getTagsAsString(), $post-&gt;getDate-&gt;format('Y-m-d')) as $item) { echo "&lt;td&gt;$item&lt;/td&gt;"; } echo '&lt;/tr&gt;'; } ?&gt; Which, in turn, is actually much better written as &lt;?php function echoRow($items) { echo '&lt;tr&gt;'; foreach ($items as $item) { echo "&lt;td&gt;$item&lt;/td&gt;"; } echo '&lt;/tr&gt;'; } foreach ($posts as $post) { echoRow(array( '&lt;a class="recent\_links" href=""&gt;' . $post-&gt;getTitle() . '&lt;/a&gt;', $post-&gt;getAuthor()-&gt;getUsername(), $post-&gt;getDescription(), $post-&gt;getTagsAsString(), $post-&gt;getDate-&gt;format('Y-m-d') )); } ?&gt; Honestly, trying to treat PHP as a templating language, despite that being what it was for 15 years ago, is a novice approach. Notice how easy it is to change that last one without concern of getting it wrong, and how clear it is what's actually being done.

[Frapi](http://getfrapi.com/)

I've been trying this out. The current 'framework' is not well thought out at all. I believe they started with the CMS then built the framework around it after. You never want to do that. The documentation is pretty poor they don't even have any diagrams about how the thing is constructed. How do you design something like that without a clear UML or class diagrams? Basically you get into the forms/template processing and there's no way to decipher what's going on behind the scenes. The code is going round in circles. It's not easy to pick up and definitely only suited to extremely basic blog type sites. I have seen a few small sites hacked together with it but the developers had been using it for a few years so they were somewhat familiar with it. If you want to spend 2years learning it then you'll still only be about intermediate level. End of the day I think the SilverStripe guys had a generic CMS they rolled out for their NZ government clients or whatever then one day they decided to make it open source as it was the trendy thing to do. Basically their only interest is making it work for their needs and their clients. If they can get some clueless devs to help contribute to it via open source then that's a bonus for them. It's not really helpful for any other company that comes along and tries to use it. I think there are much better frameworks out there like Symfony 2. There are also much better CMSs out there that you could link in with Symfony. Trying to do anything moderately challenging in Silverstripe is a real chore. It doesn't make life easy for you, it's counter-intuitive and far too restricting. I wish they'd just unopen source it and go back to using it for their internal sites. That way I won't be forced to use it at my work anymore and we can switch to a better framework.

Your examples are just highlighting what got posted before: &gt; You just have to know wtf you are doing, like any other language. You describe how \*not\* to program securely, claiming it is that which makes the language insecure, which is kind of a straw-man argument.

\*This\* is the main reason. It is easy for people who don't really know what the are doing to pick it up.

Save this thread and come back tomorrow. Give the guy a chance to fix things!

Try writing a simple MVC CMS by yourself, from the ground up.

Actually... the mysql\_\* set of functions are a direct layer over mySQL's C library. They're about the lowest level of functions you can get, and they're exactly what you'd get if you were programming in C or another low-level language. Strangely, you can access these in Python if you want to. The mysql\_ set of functions are agreed to be generally insecure if you don't know what you're doing, and the PHP team are considering removing them for the next major version of PHP. If you read even a little on the subject, you'll find that people suggest using mysqli or PDO. Otoh, if you learn by blog tutorials... well, there's no hope for you, tbh, whatever language you learn.

In cases where you really need speed, Blitz is a great template engine written in C. However, tbh, Twig is one of my favourite template systems for PHP. Allowing raw PHP templates might work for something complex like Wordpress, but for the most part, most sites will get away quite happily with a template engine - and it makes your templates look cleaner, and reminds you to separate your view logic from the rest of your code.

$50 is low for my area, $75 is what a w-2 contractor would make, like $150 for an independent one. Make sure your quote includes time for qa, and a few iterations with room in the budget for changes without having to go out of scope, usually 5-10%

Sure did. I have no reasonable explanation other than "late night makes me make silly mistakes". Would you take "I didn't want the comment to actually show up in the... pastie source?"

Twig compiling into PHP isn't relevant though. As the OP said, he's doing this to learn, and any half-decent templating engine will help him learn what should and should not go into the view script.

While abstracting out templating functionality is an awesome practice (see something like view helpers in ZF1), this is not a particularly good approach to that end. Functions defined in templates? Are you mad, man?!

That's not at all necessary, but thanks for offering. Best of luck.

But I'm not a Cake hater... :P I'm actually really surprised to hear that Cake 2 is going with 5.2, though. I don't think a second generation framework should be actively supporting end-of-life technology.

I prefer to wrap PDO with some custom "Database" class for two reasons: \* If your schema changes, you only have to update queries in one location \* Makes unit testing easier

Way to flame. Your hyperbole-filled attack doesn't add anything to the conversation. My code does not have "just a faint hint of OO". The only non-OO portion is the core database layer (Model objects provide a full OO layer above this), which is way more succinct this way and does the same thing as any direct-to-database code you can produce, including replicated database support: https://github.com/jbroadway/elefant/blob/db-replication/lib/Database.php There's nothing wrong with ordinary functions when they make sense. Hell, that's all you have in C, yet they managed to write Linux in it. My db layer saves so much typing over other PHP db layers it's not even funny. When you just need to write a quick SQL statement, this is so much faster: $res = db\_fetch\_array ('select \* from foo where bar = ?', $bar); When you need more power, or have additional logic going on, create a new model and put your logic in there. The only other thing that's really "out there" for textbook MVC folks is that I abstract away the controller boilerplate so this: &lt;?php namespace Myapp\HelloBundle\Controller; use Symfony\Component\HttpFoundation\Response; class HelloController { public function indexAction ($name) { return $this-&gt;render ('MyappHelloBundle:Hello:index.html.twig', array ( 'name' =&gt; $name )); } } ?&gt; Becomes this: &lt;?php echo $tpl-&gt;render ('myapp/index', array ( 'name' =&gt; $this-&gt;params[0] )); ?&gt; That's not spaghetti, it's just eliminating repetitive typing and the use of OOP for its own sake when it's not needed and adds nothing. Some PHP developers really need to learn to think differently. Spend a couple years in Perl (to the point when you're disciplined enough to write \*clean\* Perl code) and you'll learn why they say "there's more than one way to do it". OOP isn't the only way, or even always the right way, and MVC has plenty of issues in the web context as well.

Both are interesting approaches. I don't know how fond I am of either approach, but I haven't given them a decent shot either. That said, this outlines the difference in philosophies between ZF and Symfony pretty well: ZF is all about providing the flexibility to make stuff, and Symfony is all about providing the stuff.

You've created a one-time use function that takes a few more seconds to read and understand than: &lt;?php foreach ($posts as $post): ?&gt; &lt;tr&gt; &lt;td&gt;&lt;a class="recent\_links" href=""&gt;&lt;?=$post-&gt;getTitle()?&gt;&lt;/a&gt;&lt;/td&gt; &lt;td&gt;&lt;?=$post-&gt;getAuthor()-&gt;getUsername()?&gt;&lt;/td&gt; &lt;td&gt;&lt;?=$post-&gt;getDescription()?&gt;&lt;/td&gt; &lt;td&gt;&lt;?=$post-&gt;getTagsAsString()?&gt;&lt;/td&gt; &lt;?php /\* &lt;--- that may be a bit nicer than calling another model(?) \*/ ?&gt; &lt;td&gt;&lt;?=$post-&gt;getDate()-&gt;format('Y-m-d')?&gt;&lt;/td&gt; &lt;/tr&gt; &lt;?php endforeach; ?&gt; What happens when he wants to break it up into more TR's? The code starts growing, and becoming less legible, at least when compared to mixing in with HTML as above. Of course you usually have a good reason for what you say - would love to hear the one on this one?

&gt; You've created a one-time use function So have they. They just didn't write function in front of it. What I wrote has less logic, is cleaner, easier to understand, easier to change, and is appropriately packaged. Suggesting that making functions for single cases is somehow inherently bad is silly. &gt; that takes a few more seconds to read and understand No, it does not. &gt; What happens when he wants to break it up into more TR's? Then ... he does the exact same change he'd do the other way. &gt; The code starts growing In both cases. Null hypothesis, please. "If you grow it, it gets bigger!" Yes, that's how this works. &gt; becoming less legible It's sort of bizarre that you think the clean one is harder to read. &gt; Of course you usually have a good reason for what you say - would love to hear the one on this one? I already gave it. You just don't agree, and traditionally, that leads you to believe that you haven't heard something. Frankly, the idea that code blocks scattered throughout markup are somehow easier to read than a clear presentation is ... bizarre. Try doing this with the more complicated table you're talking about, both ways, and see if you can really continue to claim that the run-together one is cleaner with a straight face.

&gt; While abstracting out templating functionality Templating does not mean "putting values into markup." This is not templating. &gt; this is not a particularly good approach to that end. Any particular reason for this claim? &gt; Functions defined in templates? There is no template here, and even if there was, the obvious response would be "that's how PHP works, man." Clearly, if you'd actually read, my functions generate the markup. There is no template here. &gt; Are you mad, man?! No, I just have experience with the end result of the other way taken to a large extreme, and it's god-awful.

&gt; Sure did. I have no reasonable explanation other than "late night makes me make silly mistakes". You mean like the other ones you made this morning, where you just accused someone else's much cleaner than your own code of being "a bad approach" when there were actually several, and without explaining what was bad about it? Have you actually written any large PHP applications on your own? I ask because if your advice gets followed at scale, you end up with a nightmarish mess of run together code and markup. Anyone who tried to hack on MediaWiki several years back knows exactly what a nightmare you're advocating. It's essentially the worst available way to write PHP, and I've never met a PHP programmer with experience who's disagreed. Maybe you could be the first. I'd like to see some of your code, to see why you believe this ends up something other than horrible.

Note that they explicitly bound by size and by largely read nature. Note also that the number 99.9% is made up, and incorrectly includes the vast majoritty of websites which do not use a database at all. This is like suggesting that shoes are appropriate for 99.9% of all travel in a discussion about interstate commerce.

Essentially my app does simple file sharing. It's a lot like dropbox, but internal and proprietary. Users can download and upload files, create folders etc. This app has been running continuously for about 5 years, getting continuous upgrades and one major rewrite of the backend. Mid-day yesterday I started getting complaints that files had vanished. The folder structures were all intact. I looked into it immediately and was shocked to find that nearly the entire filestore had been wiped. Most bizzare is that the folders were not touched. I thought that someone must have gained admin access and done a systematic delete using the app's delete file functions. This turns out to not be the case based on what we know. The database was untouched and the app has no record of the deletions. However these files were removed, it was not by using any of the built-in functions of the app itself. Im recovering the files as priority one and then going to continue to look for points of intrusion. Our logs are fairly good, but of course when something like this happens I wish I had logged every possible bit of data. I haven't ruled out the possibility of accident, possibly a function to delete a file went rouge and deleted the entire directory tree. It seems too clean to be a totally malicious attack, since they did not delete any components of the application itself or any other services on the server. I have to figure out how someone could delete all the files without touching the folders, that might leave some clues about how this was done. We are proceeding with caution, locking down what we can until we know more.

Get involved in the open-source projects which you use! Not only would you be gaining additional hands-on experience, but you'd also be giving back to the community. You could do something as simple as cleaning up documentation or verifying bugs, or you could get involved to the point of fixing bugs and submitting patches, or maybe even introducing &amp; coding a new concept for the project. Plus, it looks good on the resume.

\_/me wishes we had a /r/php-frameworks or similar...\_

Yeah, I was up 3, then down 4? Not sure...

Most clients only care if you're in similar time zones to them (ie, American). I don't charge less than a Californian developer just because I'm in Texas.

They're going with 5.2, but they're "sorted" things so they can "quickly" jump to 5.3 only - the way they've reorganized. I think majority of servers now are 5.2.x (from what I've seen), so they still have to support older clients

I got sick of copy / pasting and all the other messy work involved with setting up Doctrine2.1 with PHP, so I'm putting it here so others can use it too. Plenty of improvements to be made, but I've tested it and unit tests work too. I'll try and keep this updated until ZF2 is out, when hopefully someone will come up with a more sane way of doing this.

You're comparing LDAP filters to a complete programming language? How does that even work? You can just as easily shoot yourself in the foot in ANY language, be it C#, Python, Ruby, Java or COBOL. It's NOT about the language, it's only and exclusively about the programmer. EDIT: also, why are you even subscribed to the PHP subreddit, when you're bashing it so hard?

[/r/python is this way.](http://www.reddit.com/r/python)

Might check server level logs and not just your vhost / apache / php logs. Could have been a root / user / os vulnerability and nothing related to your app itself.

http://www.amazon.com/PHP-5-Objects-Patterns-Practice/dp/1590593804 http://www.amazon.com/PHP-Object-Oriented-Solutions-David-Powers/dp/1430210117 &gt; Easy to understand. I'm not stupid, but I get bored when it's a bunch of big words I don't understand. If you come across big words you don't understand, look them up and understand them. &gt;Pictures help. I was born in the 70s and raised in the 80s on TV and Video Arcades. Pictures help a lot. I'm not sure what pictures would be doing in a programming book, but if you really require pictures to keep your interest, Head First books are pretty well reviewed: http://www.amazon.com/Head-First-MySQL-Lynn-Beighley/dp/0596006306 (if the code itself doesn't keep your interest without pictures alongside it, maybe you're in the wrong field?) &gt;Code examples. Books with projects that you work along with would be nice, and preferably in PHP (although I do understand C# somewhat, I'm on a Mac and don't want to use Mono). Once you read through some books and have a better grasp of the language, look into frameworks. They'll start making you organize your applications.

Outside of the PHP Manual, I've found this page to be an excellent primer on the different methods and approaches to using MySQL, PDO. and prepared statements. http://www.kitebird.com/articles/php-pdo.html There's a bit of learning hump, but once I started transitioning my projects to using PDO and prepared statements, I found I was putting together the SQL parts of a script much faster, and it was much more powerful. That said, PDO is lacking in some areas. Namely, and I would love to be proven wrong about this and shown the light, I've found it difficult to do things like pagination, or sorting by different columns with a prepared statement in PDO.

Oh so you're just trolling, silly me.

This might sound dumb, but are you certain it's not possible for someone to upload a php script and have it execute?

That documentation looks different than what I've previously seen. I'll look into it. Thanks,

I don't have much experience with C#/Visual Studio so I'm genuinely interested, what's the retarded stuff that Visual Studio is stopping you from doing?

One thing that always confuses me about these sort of frameworks and their API libraries, is why they presume I want everything in the db returned to the client. Is there any mechanism to support the filtering/removal of fields before presentation to the client in CI?

OK, so you're basically describing the standard features of any even remotely good IDE. I just pasted the snippet you provided into my IDE, and it marked the line red and told me exactly what's wrong. I don't believe any IDE has a feature to make you write more secure code.

I use [Komodo IDE](http://www.activestate.com/komodo-ide) which is not free, but they also make a free, lightweight version, [Komodo Edit](http://www.activestate.com/komodo-edit), which doesn't have all the IDE features (like debugging, DB explorer, source control integration and some fancy features like that), which has syntax checking, autocomplete, and shows you method declarations, variables and things like that.

the filestore is located outside of the webroot

Recess had a bright future, but I don't think it's actively developed anymore. I'm currently using Lithium and Cake 2.0 for most of my development.

Very handy. Haven't done more than just take a quick peek, but it looks to be essentially the same setup I typically use. I'll clone it later and kick some tires. Thanks!

Personally I'd prefer to generate all the base forms and extend them, but I think for smaller projects (or rapid prototypes) just passing a doctrine table definition to the form seems nice. Especially with pre and post render hooks. Part of ZFs problem is that there is so much code there but no coherent instructions on what to investigate next. The documents need help - some links to the auto form generators would be great on the Model section of the documentation.

So...your problem is that you can't write PHP worth shit. Got it.

The means to insert your data into a structure for presentation is the fundamental purpose of any templating system. It can be accomplished in any number of ways, and some of those ways are no doubt better than others. There are few reasons that your solution is less than adequate, one of which I alluded to in the next sentence (which you quoted): Declaring functions inside a template file is not an appropriate way to abstract templating functionality. If the function is declared along side the loop in which it is called, you're just creating extra boilerplate code for no obvious benefit. Declaring the function in a separate context from where you're calling it in that example would be more appropriate (and perhaps this is what you were trying to demonstrate), but then you are concatenating html as strings outside of your template. That kind of stuff used to be really common (in the PHP community anyway), and fortunately we've been moving on from that in recent years. You keep saying "there is no template here", which leaves me scratching my head. Where would you be calling code that outputs your presentation markup if not in a template? Are you instead coupling your response to some means to generate HTML markup? Do you know of any/have any real-world examples of what you're trying to describe that you can share?

You quoted me, so clearly you didn't just misclick the "reply" button on the wrong comment. Kudos to you, though. I'm sincerely impressed that you managed to get defensive about a self-deprecating comment that I made. I'm not under a pseudonym, so if you're seriously interested in seeing anything about myself or my code, I'm confident you will have no trouble doing so.

As headinthesky said, the reality in the world is that most shared hosted solutions are still running 5.2, and cake has always gone with the practical, it-will-run-right-now technology that doesn't need your own dedicated server. I'm not that big a fan of namespaces (at least PHP's implementation..), I'm sure some people are, but cake does implicitly support it now in their file structure. That said, closures are pretty cool. Good to see you're not a 'hater', but from the amount of general downvotes posts can get around here, it seems there are a few that run rampant around here...

The only logic I have in my template was a single control structure and some calls to accessors. You have an identical control structure and calls to the exact same accessors, but you've also created an additional control structure, a function, make use of an additional data structure, and you use concatenation as well. And you've done all of this to accomplish the task of rendering a single table row. I'm all for abstracting logic so people don't have to rebuild redundant markup all over the place, but that's not the case here. You just added more logic to an otherwise straightforward and concise task.

Even so, if your legit PHP has access to read/write/delete the files, so would rogue PHP. Not that I am backing the rogue PHP angle. I don't have enough data to even theorize.

HEREDOCs, anyone?

Random points: \* Don't you usually want to call parent::\_\_construct at the start of your constructor, rather than the end? This is so you have a baseline of 'parent' functionality, which you then modify; as opposed to inserting custom functionality first, which may or may not be accidentally or purposely modified by the parent. \* this can be abstracted away: $this-&gt;templateDir = APP\_DIR . 'admin/pages'; A) the img/ folder contains public images, and so should be under a public directory B) the pages/ folder is still just templates; I personally don't see a reason why 'admin' content should be regarded any different from other controllers. \* Sessions: session\_start(); Many frameworks have an abstracted session wrapper; this is so you can use it where you want, and when you want, without having to worry about calling these functions manually.

I will admit that my knowledge of shared hosting is flaky at best these days, but in a very brief search I found that Dreamhost, Hostgator, and Lunarpages all offer 5.3 shared hosting. Are you aware of any shared hosts in particular that are still outdated?

It's mostly just a fork of [Ralph Schindler's integration](https://github.com/ralphschindler/NOLASnowball/tree/doctrine2-managed-crud), but that broke after some changes were made in Doctrine 2.1 (and Doctrine 2.0 is already EOL). This has that plus a couple bells and whistles like jQuery provided through ZendX and stuff. I'm still trying to come up with a better way to deploy web applications in PHP for myself, but this will have to do for now.

...pick any PHP framework, like django, and you also won't be able to write SQL injectable code, as long as you follow the framework.

If the DB has parameter binding (which both Oracle and MySQL have), then it will likely let you write queries which can be optimised using it.

its mostly for small mom and pop local business sites, its just alot easier for me to use sqlite in those cases.

Well, mysql\_\* functions have only officially become 'the wrong way' over the last few years. PHP has been in use for over a decade. So unfortunately, there is now a hell of a lot more literature about the wrong way, for noobies to pick up.

If you haven't, learn object-orientated programming. Then, learn a web framework or two. CakePHP, Symfony, Yii, FuelPHP, Kohana.. there are many, all of them 'good enough' to give you a shittone of ideas. If you want something more esoteric, learn a functional language like Haskell or Lisp. That'll really get you thinking.

Yea, as far as i understand it, sqlite vs mysql can be debated quite a bit. i use it mostly for mom and pop business websites. I do use mysql, but its rare.

I've found the easiest way to launch a Netbeans debug session is using [Chrome Xdebug Helper](https://chrome.google.com/webstore/detail/eadndfjplgieldjbigjakmdgkmoaaaoc). In Netbeans, go to Project Properties &gt; Run Configuration &gt; Advanced. Choose "Do not open web browser". This way, you can click "Debug Project Ctrl+F5", and Netbeans starts listening. In your browser, click the debug icon. Your next request, whether GET or POST, to any URL, will be captured by Netbeans.

Namecheap, webhostgiant, ixwebhosting, BlueHost, JustHost, FatCow... I'm sure I could find others

As long as you don't expect it to scale. `:)`

&gt; The means to insert your data into a structure for presentation is the fundamental purpose of any templating system. Yes, and the fundamental purpose of a relational database is to store things for retrieval; nonetheless, there are things which are not databases and store things for retrieval. That they share goals does not mean this and a templating engine are one and the same thing. I mean, even if we disregard the issue of what templating is, where's the engine here? &gt; Declaring functions inside a template file is not an appropriate way to abstract templating functionality. That's an assertion, not an explanation. &gt; If the function is declared along side the loop in which it is called, you're just creating extra boilerplate code for no obvious benefit There's also no boilerplate here. &gt; Declaring the function in a separate context There ... is no context here either. &gt; (and perhaps this is what you were trying to demonstrate) Nope. &gt; but then you are concatenating html as strings outside of your template. There's \*also\* no concatenation here, though not knowing how operator comma works in the context of echo in PHP is an understandable mistake, since it's so different than most other languages, and so different than what operator comma does throughout the rest of this language. &gt; That kind of stuff used to be really common (in the PHP community anyway), and fortunately we've been moving on from that in recent years. I don't think you're really empowered to speak for the community, sir, and to be blunt, you've just got the history completely wrong here. I very strongly get the impression that you haven't been doing PHP for much more than a couple years, and that your notion of what the professional community does is largely steered by things like MVC frameworks. Perhaps I'm wrong. (shrugs) &gt; You keep saying "there is no template here", which leaves me scratching my head. Indeed. &gt; Where would you be calling code that outputs your presentation markup if not in a template? I get the impression that you don't actually know what a template is. If you did, you'd point to where it was in the code I presented, instead of asking where I would do things. &gt; Are you instead coupling your response \*Coupling my response\* ? &gt; Are you instead coupling your response to some means to generate HTML markup? Is it that you just don't see the markup that's there? I don't understand why you're asking this question when the answer is right there in front of you. Why would anyone generate HTML, then put it into something that isn't generated HTML? "Did you make this car so you could drive it around a boat?" No, I made the car because I expect a system based on driving. &gt; Do you know of any/have any real-world examples of what you're trying to describe that you can share? Wordpress. Kayako SupportSuite. MediaWiki after 1.7. I don't understand why you're spending so much time saying there are reasons this is technically inadequate, then justifying that in terms of work you imagine wasn't necessary, which has nothing to do with solution adequacy, or asking me to go find examples of things in industry for you because you don't want to find them yourself and imagine that if you just don't know what's out there that a technique is thus somehow made inadequate. For someone who speaks to what the community is doing, I don't get the impression that you actually know what the community is doing. Furthermore, what the community is doing frankly has nothing to do with what's a good approach. Following the herd generally gets you a poorly understood version of an average design, which equates to sub-average software. If you have a specific criticism that justifies the claim that this is technically inadequate, which you have now said several times, then great, have at it. Observing that you don't think a function needs to be there has nothing to do with a solution being inadequate. Observing that you can't find the template in something that you insist is a template engine has nothing to do with a solution being inadequate. My strong impression is that you are arguing against this for no legitimate reason, and that your only goal is to suggest that your approach is better; I believe that the reason that the most you can come up with is "making a one-use function to separate logic from content is somehow generating unnecessary boilerplate" (when boilerplate is the exact opposite of what you previously called it, a single-use function) is that you don't actually have any legitimate criticisms. Basically, all you've said here is "I disagree with you and I don't understand why you made several of these choices." That's nice. That's no basis for technical inadequacy. Technical inadequacy means it can't do the job, not that you personally don't like it. If you want to show me something more ... significant, please, do. These criticisms are largely opinions and explicit "this is bad because I don't know why it's here." When you got to the point of guessing about what I meant, you made it clear how valid a position you are in, to criticise what I said. How can you criticise something when you don't even understand its goals? C'mon. I didn't argue with you after your opinion. Argue with me \*once\* you get me, not before, and please have more important things to discuss than that you don't think having logic in a function is a good idea as a way of claiming technical inadequacy. This is like saying you shouldn't buy that truck, because it can't do the job, and why? Because blue is an ugly color. Your opinion is noted. Let me know if you get anything stronger than "I prefer something else."

Do it right, you're fine. PHP makes it easier to do it, therefore it's easier to do it wrong.

I'm working on a Neural Networking setup in PHP. More likely PHP isn't restricting you at all.

&gt; I'm sincerely impressed that you managed to get defensive That's nice. I haven't, but whatever makes you feel superior. Let me know if you get to a technical criticism. &gt; I'm not under a pseudonym, so if you're seriously interested in seeing anything about myself or my code, I'm confident you will have no trouble doing so. Ah. I hope you realize that it would be uncommon to recognize that nick as a name. Unfortunately Reddit's rules disallow me from pointing to a domain and asking if it's you, so is the fourth letter X? I want to make sure I haven't jumped to conclusions. Of course, the reason I wanted to ask you for code is that that way I'd get something you found to be well written by your own standards; lord knows there's some code I did on the job out there with my name on it where a boss told me to do something a stupid way, or whatever, and I wouldn't want to take a look at code where you had been PHB forced into a bad decision, or didn't have time, or whatever. It is, after all, a courtesy to ask someone to choose their own work by which to be considered, but I guess it has come off somehow as an insult. So yeah, I found this blog, but it suffers the same problem that mine does - it spends more time talking about code than actually showing non-trivial code, and its author is too busy to put in more than a half dozen posts over the last year or two. I think it's yours - it belongs to someone whose name matches your handle, and that is an uncommon first name in my personal experience. (Indeed, I didn't know it was a first name. I've never met a "Court" before, nor have seen one in media.) Still. The only code in the first two pages is a list of prototypes with extensive commenting, and if I go back further than that, it's years ago, and that's not fair. Even very experienced programmers, if they aren't stagnant, are much better than they were three years ago. It is the nature of our trade. And in doing some basic digging, I found a neighborhood in New Hampshire that has your name, and a Hilton in New Jersey that has your name. I've found some tiny pieces of sort-of-code. I turned up a legal decision that has nothing to do with you, but coincidentally shares your last name and talks about a court. I turned up a discussion of the brokenness of PHP, with which I tend generally to agree, and a dozen pages referring to it from various differing polar responses, most of with which I disagree. I found your twitter and a twitpic, a light tutorial about Symfony (and c'mon, there's no judging a person by tutorial code they wrote,) a four-function registry replacement, a diatribe against concatenation in favor of printf, your youtube channel, your hacker news account, several creepy twitter aggregators pretending you're their user (topsy is downright weird,) your digg, google+ and convore accounts, discussion of echolibre, integration with salesforce, a slightly less small example of usage of Doctrine, etc. Maybe I'm being dense. Maybe I'm looking in the wrong places. I don't actually see any large code written by you. I'm sure it's out there; I just don't know where to look. I mean, I'm trying to figure out who I'm talking to; it's not clear to me why that makes you angry, or why you find it inappropriate to say "this is the work of the guy who's telling you you're wrong." That's how software people judge whether they're barking at their superior. How am I to take you seriously if your claims are directly against my experience, your observations of community trends are the exact opposite of what I see, and you won't show me any code? I was trying to give you a platform to demonstrate your experience. I genuinely thought that was a kindness and being polite. If it's offensive (for reasons I really don't understand,) then I apologize. I'm happy to do it myself. Here's one of my Erlang libraries, for example. http://scutil.com/docs/sc.html#index Go down to the bare domain if you want to download the code; .zip and SVN in the upper right hand corner. I'm not challenging you. I'm trying to figure out who I'm talking to. I don't get why that's an offense. Anyway, look, I've tried to talk to you in several threads now, and I'm getting nowhere. Maybe there's a beef I don't understand. Maybe I'm being a dick and don't get it. Whatever. I thought I was giving you a chance. My bad. Do what you like. Just, do me a favor: the next time you use the phrase "technically inadequate," have a justification that isn't about your personal preferences or things you say you don't get. I mean, really, man. I can't call you an amateur, because I saw this thing that's probably your blog, and you make a lot of good points there. What I can say is "you don't explain your opinion in a way which other people can actually understand." Or, at least me. I still have no fucking clue why you think my suggestion is "technically inadequate," and I kind of think you just dropped that phrase to sound final. And frankly, I'm old enough to not really care when someone argues primarily on a basis of ostentatious yarping. I can see, from this blog, that you know your shit. What I cannot see is what you think the problem is. I listened, I offered you chances to explain yourself, and you treated those like defensiveness and attacks. Okay, guy. If you have the goal of explaining yourself, here's another chance. If your goal is to feel correct, you probably already do; leave it there. I don't really care, either way. But if there's a legitimate basis for the criticisms you made other than that it goes against your personal preferences, I'm still listening. Lemme know if you feel like doing this in a way anyone would ever, in the history of humanity, respond positively towards.

Upvote for learning a functional language - it will help you see things in such a different (and useful!) light. I think I'm going to buy a dead-tree copy of [Learn You A Haskell](http://learnyouahaskell.com/).

Honestly, the thing that's kept me going over the years is choosing larger projects. However, if you're looking to be stretched, there are things like the Prolog 99 problem list, Project Euler, contributing to open source projects, or learning new languages. Remember, other languages can be radically different. Picking up Mozart-Oz, Fortress, FormulaONE, prolog or Erlang could really change how you see software. Or, maybe it's time to write a video game. (shrugs)

I would advocate this only if it fell into the realm of "micro-framework". /s

Could.

Yeah it looked like the updates to the blog stopped in early 2010 but the code is still being updated in github so I have no idea?

nah, most just want a quick easy way of posting the occasional news update, and thats about it

&gt;I'm still trying to come up with a better way to deploy web applications in PHP for myself Have you tried [Capistrano](https://github.com/capistrano)? http://en.wikipedia.org/wiki/Capistrano &gt;\*Capistrano is an open source tool for running scripts on multiple servers; its main use is deploying web applications. It automates the process of making a new version of an application available on one or more web servers, including supporting tasks such as changing databases.\* W.J. Gilmore reccomends it highly in his ZF book: ["Easy PHP Websites with the Zend Framework"](http://www.wjgilmore.com/books/read/easy\_php\_websites\_with\_the\_zend\_framework) He also uses Doctrine 2 instead of Zend\_Db, which is why this post stuck out for me.

This could be genuinely useful. Thanks.

Well shit, Capistrano looks awesome. I'm definitely going to be playing around with it this weekend....

I wrote an article about going from PDO from a mysql\_\* user's perspective: http://wiki.hashphp.org/PDO\_Tutorial\_for\_MySQL\_Developers also mysqli developers wrote a migration guide as well: http://forge.mysql.com/wiki/Converting\_to\_MySQLi

StoneCypher is a [troll](http://www.reddit.com/r/PHP/comments/iu7xw/stonecypher\_can\_you\_tell\_us\_about\_your/c26swxn). By now I'm sure he has made fun of your grammar, told you that he is the best programmer in the universe, and introduced you to the awesomeness that is "SCUtil". Pretty much all he has to his name. It's a deflection technique he uses to stop you from pointing out that he is wrong. Pretty cowardly, but standard troll behavior. Best to ignore him!

I can't speak for CI's implementation of activerecord, but I use [PHPActiveRecord](http://php-activerecord.org) with CI. It makes that presumption if you want to do some basic coding to get your app up and going, but there are definitely ways to override those assumptions. PAR allows you to send custom queries if you have to. Basically, if you follow all the conventions everything will work super, but as with most things, following convention in legacy code isn't always possible.

I think the reason SQL injections get a lot of focus is they're both easy to prevent and easy to exploit. It's a beginning programmer's mistake, and it's a beginning hacker's best target.

Skynet has become self-aware.

don't listen to this guy and start echoing html, it's ugly, less readable and the rookie way to do it.

People are paid salaries to do this for a living. Many of them/us have degrees in this field and have proficiency in several languages. "Rapid application development" (as much as i hate that idea) presupposes a developer. As a hobbyist, you can't expect to throw together a project over a weekend, no matter what the PHP detractors might claim. You might want to investigate a framework - there are a dozen decent ones out there, but you still need to know what you're using and what you're doing. If you find a nice MVC framework you like, it'll save you some development time. PHP has the best documentation of any language I've ever used. PHP.net should be the first place you go for anything - explore the manual. And have fun!

I dunno what your issue with WebDAV is. It's built ontop of HTTP. HTTP is being used by everyone. PHP is designed for HTTP. Yes, there is an issue if you use HTTP instead of HTTPS but that's about it. I would prefer to use SSH FTP so later we could maybe build a custom SSH server that connects to the VFS. Regardless WebDAV is just a transport layer decision on top of the VFS. Once there's a better alternative we'll be glad to switch. Everything is a potential security hole. As for SCP. No directory listings = fail.

Zencart is by definition NOT a framework. At most it's an e-commerce CMS.

It's not taking an age and a half because I'm having trouble coding features but for the volume of features that I have to code. Maybe 'newb hobbyist' was cutting myself a bit short; I'm a fairly proficient programmer in general (C/++, C#, Java), I've been into webdev casually for about a year and I have a very complete understanding of all things involved with it. Nothing is \*hard\* about php development apart from the drudgery of actually writing some of the more mundane features. I guess a better way to phrase my question is 'are there any places to get prepackaged modules for common features in php.' Or will I just have to make my own little library, adding little bits as I move forward, and reuse those for every new project I have?

So, are CMSs not appealing? Drupal has so many modules to prevent reinventing the wheel.

I've been putting it off... I guess that'd solve most of my problems though. I was against it for this project because I wanted a full php experience to help me learn the language better. I hadn't really considered it moving forward...

The key to super fast development is the same as the key to super fast speaking: Understand the language like it's second nature. If you're looking in your little "English to Spanish dictionary" all of the time mid sentence, you're never going to have a properly fast conversation like someone who speaks it fluently. Also, try to stay DRY (Don't Repeat Yourself). That, and having experience with past failure cuts bugfixing time in half, which is unfortunately impossible to get out of a book or tutorial. Learn how to write plain-old PHP/SQL, understand how that works on a basic level. Then start incorporating more OOP methodology into your coding style. At which point learning a framework would benefit you much greater than starting with one. You'll also want to streamline your workflow. If you're editing a file and just drag/dropping it from a file explorer into some FTP app, you need to re-think your strategy.

So you wanted to write everything from scratch to learn PHP, but are now complaining about having to write everything from scratch?

CI, and other frameworks, are nice because they do cut out a lot. For authentication, for example, many have written plugin systems that drop into place and with a little db stuff, work quickly. As with anything, don't forget what you put together and begin to build a library of components. Sure, you'll have to tweak them each time but that is much better than "from scratch," hence the niceties of a framework.

I've been doing this project for about three months. I'm past the wanting to learn it part and now I'm just tying up loose ends.

I've built a number of applications, both large and small, in several different languages. The thing that made the biggest difference? Having specifications nailed down. Easy as that. Once you have a blue print, the code flows like butta. And, as anyone else would probably tell you, flip a coin and choose a framework (CI is awesome when used properly). Then learn that framework inside out. This way, you'll learn to build modular code that can be \*properly\* re-used on other projects, thus saving precious time and almost never re-inventing the wheel.

first, a lot of being a php programmer is creating "my own little library". 2nd, if this is the route you want to take you might as well build your own framework. If you don't want to make your own then learn one that has already been built.

Ha. ColdFusion has better docs, clearer organization and better error reporting than any language I've seen. That said, I still use PHP for almost everything I do. I just do it wishing I had CF's tools available to me.

web applications aren't so fundamentally different from desktop applications. How do you deal with all the typical functionality one expects from an any ui-based application?

IMO, one of the best things about PHP is that a quick google turns up just about any functionality you could ever need. CodeIgniter makes that even better. If you're already using CI, then you've already cut a huge slow headache out of most development. try leaning on other's work for some functionality. Something like TankAuth would take care of 4 out of 5 on your list. Other than that, you just need experience to know what to do when and then to remember that all good things take time. If this were easy and fast, I wouldn't have a job. At least not a well-paying one. EDIT: After reading some of OP's other replies, I see your skill level in programing, so I'm not telling you anything you don't know in that last paragraph.

I'm a web developer, and I look at it this way. The more development becomes attaching pre-made components, the less customizable it becomes. Like legos: you're limited to the parts on hand. Modern programming languages are as abstract as they can be without losing expression. Any more abstract, programs and websites start to look and behave the same. It takes the skill of real programmers and a lot of time to get exactly what you want.

WinSCP allows you to edit a file with Notepad++ and when it is saved will automatically upload it. This cuts out the having to confirm the auto upload. Notepad++ also has a plugin called NppFTP I believe which you can browse and open files for editing over FTP, SFTP. They are automatically uploaded upon saving locally.

I can definitely respect that. That's part of the reason I opted not to use a CMS. If every website used the same messaging plugin from drupal, well, the web would cease to be an interesting place, partly because \*never\* reinventing the wheel can stagnate innovation and also because nearly every feature that a website uses, no matter how generic, can usually be tailored to fit better into the overall experience than a plugin ever could.

As far as I know most Windows IDEs allow you to edit files right where they are, through SFTP. Even Edit+ did that back when I was still on Windows.

Wouldn't that mean you're working on the live site? I imagine Gargan\_roo was \*probably\* talking about deployment strategy.

Has everyone skimmed over that I've been using CI?

My point is that adding extra layers; WebDAV in your framework seems like a recipe to add both security holes, size and use up dev resources all at the same time. Seeing as you have to SFTP (which is what I meant instead of SCP) it up anyway at least once, it seems like a cool but bad idea. Again, building a custom SSH server in PHP sounds like a horrible idea for the same reasons but with everything turned up a little. Still, having said that - building these things in PHP is awesome, if they were standalone tested libraries I'm sure I'd find a use for them.

Glad to help - if you wind up having some KO questions either ask me or head to #kohana on freenode.org (IRC) or to the forums! :-)

I usually develop on a Linux VM. Setup a Apache Virtual Host that points to the local Git repository on the VM and develop directly there. It means you've got to work with Git on the command line but you could always X forward a Git GUI if you prefer. Once you've done your feature/bug fix or whatever, merge it back into master and push it to a central repository, then on your live site you can git pull and get the latest version. I was only really answering ruko's Q re: a better approach to editing files over FTP. Deployment strategies hadn't specifically been brought up.

That's the thing! We use a stand-alone tested library called SabreDav. http://code.google.com/p/sabredav/ I would love to switch to to an SSH server instead! Multiple file uploads right now are just like FTP. One call per file. Rather then one call per upload session like SSH.

Any of the big frameworks will \*do a lot of shit for you\* (it's why they're big...). The problem is knowing \*how to let them\*. If you haven't done OOP PHP, you'll want to learn that first. If you can't explain what [MVC](http://en.wikipedia.org/wiki/Model-view-controller) ([link](http://book.cakephp.org/view/890/Understanding-Model-View-Controller),[link](http://net.tutsplus.com/tutorials/other/mvc-for-noobs/)) generally means to someone else, you'll want to be able to do so. Part of the problem is that the developers have been working at such a high level and deep in the framework, that they have trouble finding the right level to communicate to new users. For example, Symfony, in its getting started page, seems to assume you already know exactly how MVC works, and just wants to show you how it's implemented in Symfony 2. Lastly, when they do offer you a lot of functionality, in the end there's no other way to finding out what a framework can do for you than reading it's documentation. Because that's... the thing that tells you what it can do you for you. [Start here](http://symfony.com/doc/current/book/http\_fundamentals.html).

1. Try to group your functions into objects. 2. There isn't really much to be told - the 3 files you gave so far are extremely simple and can actually be part of any type of application - not necessarily an invoice system. I suggest coming back with more code when you have it because as of now, there really is nothing much to be commented on.

\* Efficiency: Make sure username is indexed in the database or else it will cause huge delays in login times when you have a large userbase. \* Security: Once the client has logged in they stay on the same page and you are not persisting the login information. At some point you are going to need to keep track of who is logged in and who isn't. Do this with sessions, and not cookies. If you do it with cookies you have to re-invent the wheel and your system may have security vulnerabilities. \* You should be covered on basic SQL injections with that prepare statement, which is good.

Yea i am defiantly not jacking with cookies. Sessions are simple, so right now im just getting the grit of the code down. Optimization will be a primary goal, so i will look into indexing usernames. as far as the same page thing goes, if i decide to use an MVC structure, will this help avoid such things like presisting the login info? as far as the prepare statement goes, i knew starting this project i did not want to use depreciated mysql functions, and realized mysqli and prepared statements (after a bit of research) would probally be the best way to go as opposed to old techniques. This is actually my first time using prepared statements, and i have to say it aint so bad.

thanks for the info. I assume what you mean by grouping functions is to put similar functions into a class which i could then call as an object, right?

\* don't htmlentities there. It's not needed. Use htmlentities when you're just about to output data. \* Parameter binding protects against SQL injection; if that's what you're using htmlentities for, it's redundant. \* You're checking that usernames and passwords match twice. Once in the database, once in the code. Only do it in the database. \* You're not doing salted hashing. Salted hashing protects your users against your database being compromised, and protects their passwords from being found out and used on other sites. To hash &amp; salt a password: $salt = 'ewjfoia342874023jof3jnoifv32f0j23fj0'; $result = $salt . $password; for ($i = 0; $i &lt; 1000; $i++) { $result = hash('sha256', $result . $salt); // sha2 hash; don't use md5. } `$salt` is a random, long, unique value. The hash is performed iteratively, multiple times to protect against brute force attacks. You store `$result` (the hash) in the database. To compare, generate `$result` again and compare the two hashes. \* I would advise `invoice\_login()` return either [ a username string or a user array of data] or false. You then check that `($user = invoice\_login($user,$pass) !== false)` to check that the user logged in correctly. \* `$login\_stmt` would be a better variable name, imho; just nitpicking. \* House your config variables in a `$config` array; it achieves better compartmentalisation. You want as few variable in the global context as possible. There are thousands of restructurings avialable, but for a start I would create a function which given a `$config` array returns a mysqli connection; you then pass that mysqli connection along to invoice\_login as an extra parameter.

MVC as a coding pattern doesn't have much to do with how an app persists its state within the application / per user.

i was using html entities to sanatize input to curb xss attacks, i know its not needed for sql injections because im already sanatizing sql. i was not too sure about my login design on the frontend of things, but your right. i need to restructure my login code just a wee bit to reduce redundancy. i know im not doing salted hashing, it'll be added soon enough (thanks for the heads up though) good idea on invoice\_login returning true/false instead of what its doing, actually. i completely forgot my config.php file! and the $config array thing, im afraid im not sure i know what you mean, but i really could not find much about custom functions and global variables, do you think you could show me an example?

As I said, I think it's best to sanitize output when you output it - not when it's input. Completely untested: http://pastie.org/2583993

This from a skateboarding teenager who can't even get fizzbuzz right. http://www.reddit.com/r/PHP/comments/k86yp/wondering\_about\_the\_fizzbuzz\_test\_and\_looking\_for/c2ia571 &gt; It is accepted as good practice by 99.999% of coders that separating logic/presentation is a good idea. Yours is all mixed up. Gauranteed 99% of the people in /r/PHP would agree with me. Mmm. Helpfully it seems you follow your own advice, le cough. Here's who's advocating readability: http://www.reddit.com/r/PHP/comments/k7baf/help\_me\_improve\_this\_code/c2i59kg?context=3 Hush, now. People who've been doing this longer than you've been alive are talking; you shouldn't be using the word rookie at your age.

Well, they're easy to exploit, at least. There are amazingly ugly injections I've seen which are not, in fact, easy to prevent. For example, one interesting and frankly horrifying injection I've seen is bit leakage. Consider the case of where you're trying to get information about a table you have no access to, but you do have access to any query which has botchable math. For this case, I'll assume there's an exposed divisor, but there's no shortage of paths. The attack exploits the order in which things happen inside systems. If the developer is relying solely on SQL to arbit what is and what is not accessable, \*and\* if they aren't careful, you can get SQL to look up the information before it tells you you can't have it. Why would that be useful? Well, here's the schtick. If you can \*also\* force SQL to do bad math, but after it looked things up, \*and\* if you can force that bad bath to be contingent on the information you want that you aren't allowed to have, then you've got an in. Check it out. Consider an example case where you have a system which, I don't know, maybe it's a tool for your salespeople. It produces the salesperson's last name, their total sales, and it lets you pick a list of customers, and will there return the average profit per customer. Salesforce does this; it's a pretty common report type for CRM. Let's also assume there's a generic filter field in reporting, which is pretty much universal. And let's suppose that you, the outside attacker, want the names of their customers, so you can go poach said customers. However, you are not permitted to see the customer list. So. Request the average. Do \*not\* name any customers. That means that the system will divide by zero to produce the average. Now set the filter to the first letter of the verboten column in the verboten table, and list things starting with A. If you get a math error, there was something starting with A. Set the filter to AA and repeat. If there was no math error, but instead you were told your search was forbidden, then there was no A, and move on to AB. You can, one piece at a time, index information out of the system by reading which error message you get. Injection isn't always as easy to prevent as people imagine; only the basics are. Upboated for interesting conversation.

Why would you do a hash 1000 times? That's just chewing up server resources. If they log in incorrectly more than 5 times then lock their IP out for 15mins and autoredirect any attempts to the login page to the lockout page.

This is TrollPointerOuter, aka KravenC, who has already had several sockpuppet accounts shut off, and who makes new sockpuppets every week to follow his grudge through. It's quite entertaining to watch just how much effort he puts into this - it's been years, and he hasn't learned that it doesn't work yet - and it's also sort of interesting to see him call other people trolls. Curious how his mind works, inasmuch as it does. Instructive, too: if you get into calling someone else troll, making sock puppets and vote manipulating to feel powerful, you never actually get out, \*\*\*\*\*because the only way you can feel powerful is to tell yourself you're achieving something, so you can never move on.\*\*\*\*\* I quite enjoy knowing how much of his life I've wasted by simply ignoring him. :) Once every week or two he leaves a comment on my blog trying to brag about how much comment karma he's cost me, as if I care. I'm not sure why; the comments are never posted, and always get deleted. Maybe he thinks I'm frustrated by them. Not really certain. It's kind of hard to tell what a failed bully thinks they're achieving.

the mantra i learned a long time ago: filter input, escape output. xss falls in the output category.

For fast development, you have 2 choices: You can use code that is not yours, from a library, or a full script and use it in your own project. You can also use a cms or a framework that already has all the modules you'll need. This is fast but it might not work exactly like you would like and can be a pain if you need to start making modifications. Or you can decide to code everything yourself, but you need to make sure it's reusable code. (OO preferably). It takes longer to develop your own code but once you start having a good amount development is real fast. If you're doing this, might as well make your own framework, and every new functionality becomes a module. That's what I did.

&gt; You have an identical control structure and calls to the exact same accessors, but you've also created an additional control structure, a function, make use of an additional data structure, and you use concatenation as well. And you've done all of this to accomplish the task of rendering a single table row. I love how you continue to act as if these are criticisms. Gasp! You have a function! WITCH! WITCH! I'm going to say it again: so what? This is like those people whose idea of code criticism is to point out how many letters got typed. Yes, I get that I didn't follow your preferences. This is not a criticism, no matter how badly you want for it to be. I think my favorite part is that this all comes from a guy who wrapped a PHP comment in a PHP block, which has an actual cost, instead of just using an HTML comment. Clearly, wrapping a single behavior in a function to get it out of the markup and to make it more maintainable is bad, but throwing away cycles for literally no benefit ... is good, and defense-and-argument worthy. C'mon. &gt; You just added more logic to an otherwise straightforward and concise task. Let me know if you ever actually read what I said, and realize that there's a "why" there, instead of just continuing to insist that if you don't know why something was done then it must be wrong. Also continue to downvote for disagreement, because that's the new Redditor spirit.

&gt; Why would you do a hash 1000 times? That's just chewing up server resources. Because it chews up the attacker's resources too. We're paying it once at login. They're paying it billions of times during brute force attack. That isn't a downside; \*\*\*\*\*that's the whole point\*\*\*\*\*. This is the basis of bcrypt, scrypt and pcrypt. There are also people who advocate a hashing process that intentionally takes large amounts of memory, to make the cost of attack hardware stratospherically more expensive. &gt; If they log in incorrectly more than 5 times then lock their IP out for 15mins It's for "we have your data on our servers" post-compromise protection, not on-the-wire-brute protection. If you think that isn't important, just repeat the mantra "Sony Sony Sony Enron Minnesota State; Kaiser Permanente SUNY State Farm AIG." I wish someone hadn't downvoted you for disagreeing with you, as is becoming so common in `/r/php`. I upboated you back to 1, because it's a legitimate question that people need to know the answer to.

thats not my fizzbuzz, i found it online, and it is right. you don't really make much sense, and youve done your research on my age all wrong =)

Except ColdFusion is expensive to license, and practically dead.

&gt; thats not my fizzbuzz, i found it online, You had to find fizzbuzz online? &gt; and it is right. You can't see any of the problems? I think I've made my point.

[Key stretching](http://en.wikipedia.org/wiki/Key\_stretching). It's to protect the hashed password itself in case the database is stolen by severely decreasing the speed of brute forcing the hashes—locking out incorrect login attempts after a certain number of failures is a separate but also important step… which won't help once your database has been breached.

To start with, don't send passwords over the wire. If there's anything we should have learned from the recent crack of TLS, it's that it doesn't matter if you're using HTTPS;, you still shouldn't send credentials over the wire. Hash it against a fixed salt, an account salt and the HTTPS nonce. For two, start requiring at the beginning of your PHP scripts that a define has been set, and set it only in index, to prevent direct loading of your scripts. For three, perform the comparison inside the DB, not outside. It's frequently easy to trick a given PHP script into revealing the contents of some variable, which in your system currently contains an account name and password. The same commentary applies regarding your database authentication. Given that you're talking about moving to prepared statements, why not just go the extra mile and work in stored procedures? That is quite a bit more secure - the PHP just pushes data to a black box. The echoing server self as the action is a security problem waiting to happen; it's also completely unnecessary, as "this server" is implicit if you don't give anything more than a URN. You \*really\* \*can't\* keep passwords in your database plaintext. Hash them, and salt them against both a system-wide salt and an account-specific salt. (The system-wide salt is so that if you want to kill every password at once, you can by just changing one value; the account-specific salt is to fight broad-target attacks like rainbow tables.) It's good that you're encoding to prevent XSS, but you should do it at output, not at input; doing it at input makes it harder for you to deal with the data interoperably, and makes it very easy to accidentally create repeat encoding problems like WordPress has. It'll be easier to make more substantive criticisms when there's more code to look at.

if you wouldve taken 2 secodns to run it youd see that its a valid fizzbuzz. im thoroughly convinced youre not a real person.

I sent a recommendation in private mail.

If it's not your price, it's someone else's. There are "contractors" out there, in America, who charge $12/hr. Let the buyer set their price; they'll get what they pay for.

&gt; if you wouldve taken 2 secodns to run it youd see that its a valid fizzbuzz. (facepalm) And yet you call other people novices. &gt; im thoroughly convinced youre not a real person. That's nice. Are we done here?

Please stop using annotations until PHP actually supports them, if ever. Just my opinion.

&gt; My question is, what book or books can anyone recommend for me? http://www.amazon.com/PSP-Self-Improvement-Process-Software-Engineers/dp/0321305493/ref=sr\_1\_2?s=books&amp;ie=UTF8&amp;qid=1316875968&amp;sr=1-2 http://www.amazon.com/TSP-Leading-Development-Software-Engineering/dp/0321349628/ref=pd\_sim\_b1 http://sunset.usc.edu/csse/research/COCOMOII/cocomo\_main.html http://www.amazon.com/Software-Estimation-Demystifying-Practices-Microsoft/dp/0735605351 Tell me when you're done; I'll recommend more.

Did you not read? As I said, to protect against brute force attacks of various nature. Iterative hashing is \*not\* designed to protect against scenarios where you can limit password attempts. It protects against scenarios when they've already got your database of passwords. See the sibling comments to this for more info.

It's a low level abstraction. People will often want a higher one, and so wrap it. It's low level precisely so people have that choice.

Kinda what I figured, thanks for the advice! I guess I'm just getting bogged down since I have a lot of basic ground to cover just starting out.

I have a pretty good understanding of the language, and I am familiar with frameworks. Maybe I should have just said "nothing that's too dry".

Nice, thank you. Is that the order I should read them?

if you want stuff like a prebuilt authentication system, use Drupal. Frameworks like CodeIgniter or Zend can't offer that kind of functionality because its just too specific to one type of application, it wouldn't fit. ( although ruby on rails does have a gem, devise, which gives you an authentication system with password resets etc. So maybe it could be done in php too... ) For the newsletter system, you can use an external API like MailChimp.

It's a leaky abstraction -- the way it has statements and cursors are mixed together is reminiscent of how MySQL API works but, on it's own, isn't very logical.

Nifty! I've been something similar in a framework of mine for some time https://github.com/doomspork/sleepy

It's certainly not perfect but there's no reason not to experiment with possibilities.

That's a judgement call. PSP/TSP are basically accounting systems. They teach you how to keep statistics on how long it takes you to do X, how many defects to expect, how many missed items to expect to fill in, etc. PSP is for you; TSP is for your small team. (Later you can get up to CMMI too.) Software estimation is more of a "generally this is how you go about guessing." It'll be more useful early on, when you don't have stats, and less later on, when you do. The systems are very different, and both are good; sometimes having a different way to look at it is useful, for the same reason that learning a radically different programming language can make you better at the original one too. COCOMO II is actually 1970s old-sauce. However, it's big, it's industrial, and it works. It shows you how to do this stuff at scale, and it shows you how this works when you're dealing with gigantic - and therefore by definition very average - talent pools. It's also a lot more analytical than the other two; it's something you can run kind of like a spreadsheet. COCOMO is what the Ohloh cost model is built on, for example. COCOMO can tell you what it believes it should cost, money-wise, to replace a module based on a given salary. There's lots of other stuff. PSP is, in my opinion, the best place to start if this is about personal discipline and a learning experience, but it takes months to start paying off, because you have to build up a body of measurement. The Microsoft book is, in my opinion, the best place to start if you're trying to do honest contracting or work on the job immediately. You can read it and start immediately, and it's still quite a good method, though I have a preference for PSP and a belief that it develops better predictions. Read TSP immediately after PSP. Read Cocomo last.

I don't know if this is still the case, but some op code cachers like eAccelerator don't cache comments, unless they're specifically compiled with certain switches. Apps that reply on information found in comments work on the first request, and fail on all subsequent requests. The trick I've seen to get around that "bug", is to touch the file before including it. That invalidates the op code cache. Of course now you're getting into a situation of adding a lot of additional over head, and constantly invalidating your op code caches, for very little return. Just a thought.

are you done? You cant look at the code and see its valid, cant run it and see its valid, and you still echo html like i did in my first week of coding. So, yeah, i guess we are done.

I feel like nobody really answered your question. There's a library of reusable code [here](http://codeigniter.com/wiki/Contributions) as well as a bunch of stuff [here](http://www.rlmseo.com/blog/codeigniter-library-80-free-addons-tutorials-and-videos/) and [this guy](http://www.haughin.com/code/) wrote a few API tie-ins that I use a lot. Whatever you're looking for, google it + codeigniter and there's usually at least one implementation out there (usually short and needing customization, but there). You can also take a look at some open source projects that have implementations of what you're trying to do and pull and customize code from them (like [bamboo invoice](http://bambooinvoice.org/)).

It should be reminiscent of all SQL-based RDBMSes with transactions and bindings... as that's generally what it's seeking to abstract.

In PDO (and the MySQLi API), the statement object is also the resultset/cursor object. Half the methods aren't useable until you execute the statement at which point the other methods stop being usable. It's a rather poor design. The small abstraction I use splits those two concepts into two separate objects.

ahhh its ok if its untested, i can go off of that, i see what you mean now, thanks!

Hahaha alright alright, i'll start doing this, ya fuckers :) Its all good, im open to ideas - im trying to move away from how i have done php for many many years...lets just say sometimes it would have made the worst perl code seem like a good idea. old habits die hard.

Actually I am not TrollPointerOuter. I don't see the point in making such an account name. Continue with your persistent beliefs in untruth when it suits you.

What would I do from here? Would the feedback appear on a separate page?

Here's what I do when I don't want to code something: 1. Go to Google. 2. Query "[language] [feature]". 3. ??? 4. Profit!!!

Apologies for being a bit late to the party, but I feel as if you're advocating a non-optimal way of approaching this issue and wanted to throw my 2 cents in there. Mostly I think you fail to appreciate that these parts of an application are often worked on by designers not coders. Also, the design isn't necessarily decided at the time you make these templates, so you shouldn't be making any assumptions. Here you've tied the markup to a specific implementation and made the whole thing overly complicated for the web designers who are there to edit the HTML markup and doesn't (and shouldn't need to) know more than absolute basic PHP. In light of this, I don't see how this as a template is an improvement over the original. I know for sure the designers I work with would be scratching their heads over this, even though they know all there is to know about HTML and CSS. This isn't particulary easy to change. What if, say, the designer wants the description &lt;td&gt; to have its own CSS class? This would require refactoring (if not a full rewrite) of your version, ie. more work for you. And you would most likely wind up with a version equally vulnerable to futher design changes. In the original version this change would be trivial and the designer wouldn't have to bug you about it. In the original it is abundantly clear that we're dealing with a set of data that should be iterated over, and what that data is. It is dead simple for someone with only HTML knowledge to totally change the markup, and if needed move around or remove parts of it with little risk of breaking anything. Heck, they could even use Dreamweaver to drag and drop things around. PHP has constructs that make it very usable as a templating language. When you make HTML templates. you want the markup to be prominent, not the code. You want to focus on just making the data available for the designers to do what they want with, and not lock yourself to a specific markup through rigid and potentially extraneous logic. Ironically, your way is more reminiscent of how things were done 15 years ago than the original. I'm sorry, but I almost feel like you already know this and is just jerking everyones chain ... :)

http://improvise.it/core/content/output.php?id=0&amp;oid=267&amp;always=false http://improvise.it/core/content/output.php?id=0&amp;oid=267&amp;always=true

Why? The routes are parsed once and then cached. I probably should've included it, but [here's the default dispatch](https://gist.github.com/1239909).

[Here's](https://gist.github.com/1239909) the default dispatch. The comments only need to be parsed once, subsequent requests just pull the routes array and dispatch

&gt; Mostly I think you fail to appreciate that these parts of an application are often worked on by designers not coders. This person here is not such a person, and `/r/php` is not for such people. Programming strategies should not be chosen on the grounds that someone might be bad at programming. &gt; I don't see how this as a template is an improvement over the original I'm sure. &gt; This isn't particulary easy to change. Okay. :) &gt; Ironically, your way is more reminiscent of how things were done 15 years ago than the original. This just isn't true, no matter how often you lot say this. I've given several examples of major projects which started with code woven throughout markup - WordPress, MediaWiki, SupportSuite, Django, et cetera - and they've \*all\* moved away from it. This is the only way you \*could\* do PHP at the start. You're just blindly repeating things you heard. You weren't writing PHP 15 years ago. Stop pretending. Give \*one\* example of a major project which started as code separated from content and moved towards code scattered throughout markup. Just one.

Thanks, just backed up my site :)

Symfony2 supports this kind of approach pretty extensively. So does Doctrine, their ORM of choice.

Have a table of companies and a company\_id row in the employees table.

I know this /r/php but that register page is goddam unusable so I couldn't be bothered to register. Could you please remove the CSS?

Can you elaborate on 'unusable' ? It works for me. edit: So I guess "don't use javascript when you don't have to" is what you're saying? I removed it.

That's some pretty ugly code as it stands right now. Get rid of those global variables for a start. Never, ever use global variables. They are horrible.

Also - you shouldn't be echoing directly from within a function. Functions should return true or false (booleans), or values or objects. There's no reason to echo something out in the middle of a function unless you're debugging the function itself. Echoing from within a function will make it much more difficult to change your design at a later point in time.

~~Well for a start, that cross-hair is ridiculous and eliminates my natural instinct to click when my cursor turns to a pointer, and then the link text looks the same as the text around it anyway so I'm only guessing that clicking actually does something. Then when I do finally figure out that I need to click it, some more low-contrast dark grey quadrilaterals appear on my screen, but at least these are labeled so I click on them to type and nothing happens. Turns out you have to click on the other arbitrarily coloured quadrilaterals to actually input your data! And then find another randomly coloured and positioned quadrilateral somewhere off the the bottom right to actually submit my input.~~ EDIT: Thanks for fixing :).

Thats what i hear, but im not sure how to pass through variables that contain data from outside a custom function to in it. i actually can't find much info on global variables with custom functions in php, but another guy has posted a solution i think i will follow.

Don't use global variables at all. Have a config.php file that contains all your config settings. Have a database array in there, and use that to set up connections to your database. You could initialise the db object in the same file and make that object available to every script. I'm not used to writing without using objects, but I'd imagine that'd be a good way of doing it. Then any function that needs to use the database would accept the db object as one of it's parameters. You'd be much better off using objects anyway. This code is going to get messy pretty quickly.

If I do this for each user: $salt = hash('sha512', generateLongRandomNum()); $passwordHash = hash('sha512', $salt . $password); I end up with a 512bit salt and a 512bit hashed password. Then I store those two values in a table which are unique for each user. If they somehow managed to steal the entire user table they'd have the username, a 512bit hashed password and a 512bit unique salt for each user. Now to calculate the original password from only having the hashed password and the salt is going to take \*\*many\*\* years to brute force even with a supercomputer. Even then they're only going to have cracked \*\*one\*\* password from the entire list of users. Even if you find out your user table has been compromised you can just tell your users to generate a new password over the course of the month. The attackers will still be many years out from cracking even one password. As for running the hashing 1000 times each login and registration that is a crazy waste of server CPU resources especially for a high traffic site. Even if you're running the most basic VPS with dedicated CPU of 1.2Ghz that is $38 USD a month. I calculated it takes 0.05secs to do 1000 hashes on a 1.6Ghz machine. Now if I'm locking the single CPU for 0.05secs each time someone logs in or registers an account because I'm hashing something 1000 times this slows down the server as it has to wait for the task to complete. Add a hundred concurrent users logging in that slows your site for 5 seconds. A thousand concurrent users, 50 seconds. Pretty soon you'll be adding more CPU or getting additional servers just to pay for your hashing algorithm. You mention password fiascos like Sony. There is a massive difference there, they were storing their passwords in \*\*plain text\*\*. For my sites I'm storing unique salts and hashed passwords for \*each\* user. Huge difference. Sorry pal but I'll never be hashing my passwords 1000 times on any of my sites - it's a complete waste of time, money and resources.

In addition to what XyploatKyrt said (a lot of which you've now fixed—much better), the contrast of the labels' background and text color is far too low, the horizontal dark gray bars of the labels is distracting and unnecessary, and the submit buttons' being the same height and color as the input fields is confusing. That's still nothing to do with PHP, but I figured I'd mention it since this discussion has already begun.

Look for an ACL component in any modern framework.

I replied [here](http://www.reddit.com/r/PHP/comments/kpwyn/need\_some\_advice\_on\_my\_php\_invoicing\_system/c2mc8tj).

I replied [here](http://www.reddit.com/r/PHP/comments/kpwyn/need\_some\_advice\_on\_my\_php\_invoicing\_system/c2mc8tj).

&gt; This person here is not such a person, and /r/php is not for such people. Programming strategies should not be chosen on the grounds that someone might be bad at programming. But we're talking about PHP in the context of HTML templates, and in the real world designers deal with templates, and the programmers job is to make the data to be presented available to them in a way that makes sense. WHy shouldn't this be the base on which we form our strategy? &gt; This just isn't true, no matter how often you lot say this. I've given several examples of major projects which started with code woven throughout markup - WordPress, MediaWiki, SupportSuite, Django, et cetera - and they've all moved away from it. [You mean like this?](http://core.svn.wordpress.org/trunk/wp-content/themes/twentyeleven/content-gallery.php) How is the way you're gluing strings of markup together and echoing them out seperating code from content? &gt; Give one example of a major project which started as code separated from content and moved towards code scattered throughout markup. Just one. Just about every PHP-based framework or CMS under the sun, for instance [Drupal](http://drupalcode.org/project/drupal.git/blob/HEAD:/themes/bartik/templates/page.tpl.php), utilizes the fact that PHP can be used as a templating language and implements views in a way similar to the original post. Smarty et. al. works the same way; it merely adds (a rather pointless) layer of abstraction where you basically can do the same things as in pure PHP with some syntactic sugar. Code within templates isn't inherently bad. You just have to restrict it to presentation logic, and keep the business logic somewhere else. &gt; &gt; I don't see how this as a template is an improvement over the original &gt;I'm sure. &gt;&gt;This isn't particulary easy to change. &gt; Okay. :) &gt;You're just blindly repeating things you heard. You weren't writing PHP 15 years ago. Stop pretending. I'm going to assume you missed the parts where I tried to explain why this is and therefore felt a couple of snide remarks and ad hominems were appropriate instead of actually addressing what I wrote.

Duly noted, and changed. Thanks.

You mean after you log in, right? That is because the chat is fixed to the bottom of the screen and is above (z-index) all the other aspects. So I added a margin to the bottom so the bottom of the lowest aspect can scroll above the chat.

&gt; Now to calculate the original password from only having the hashed password and the salt is going to take many years to brute force even with a supercomputer. That's not the point. Number one, that's well within the reach of botnets that are currently being used for this purpose by blackhats. Number two, that's well within the reach of $50,000 GPU clusters on the order of weeks. Number three, computing hardware gets cheaper year by year. Again, this is standard practice in the industry, whether or not you agree with it. &gt; As for running the hashing 1000 times each login and registration that is a crazy waste of server CPU resources This from a guy using PHP. &gt; Even if you're running the most basic VPS with dedicated CPU of 1.2Ghz that is $38 USD a month. I ... question this judgement openly. These numbers appear out of thin air, and VPS certainly isn't fourty dollars a month. This kind of stretching is, \*again\*, common practice, and will work just fine on $3/mo shared hosting. &gt; I calculated it takes 0.05secs to do 1000 hashes on a 1.6Ghz machine Gasp! A twentieth of a second! So if you have one point seven million logins a day, a single core of a cheap server from five years ago is saturated! &gt; Add a hundred concurrent users logging in that slows your site for 5 seconds A hundred people logging in every second on an ancient garbage server, yes. &gt; A thousand concurrent users, 50 seconds. A thousand people logging in every second on an ancient garbage server, yes. You do realize that these are very high levels of traffic, right? &gt; Pretty soon you'll be adding more CPU or getting additional servers just to pay for your hashing algorithm. If you honestly believe that one login models any legitimate amount of the cost of running someone through a day's course of a dynamic site, I would recommend to you that you begin to profile. &gt; You mention password fiascos like Sony. There is a massive difference there, they were storing their passwords in plain text. Mmm hmm. AIG wasn't. Kaiser Permanente wasn't. &gt; Sorry pal but I'll never be hashing my passwords 1000 times on any of my sites Feel free to ignore basic common practice as practiced by security professionals for the last 20 years if you want to. &gt; it's a complete waste of time, money and resources. You'll pardon me if I believe Schneier before I believe you, I hope.

&gt; But we're talking about PHP in the context of HTML templates You are. I'm not. When you feel the need to insist that something someone has repeatedly explicitly said they aren't talking about is what "we" are talking about, you make it clear that you're not actually reading what's been said, and speaking largely for the goal of feeling correct. &gt; and in the real world designers deal with templates, Again, I'm not talking to a designer, and I'm not in a design place. It doesn't matter what designers need here. Are you a designer? Is that why you keep bringing this up inappropriately? &gt; How is the way you're gluing strings of markup together and echoing them out seperating code from content? Perhaps you missed the part where the person you're trying to back up was criticizing the use of functions. Perhaps you'll look at that code again. &gt; Just about every PHP-based framework or CMS under the sun, for instance Drupal, utilizes the fact that PHP can be used as a templating language and implements views in a way similar to the original post. Oh. You're an MVC believer, aren't you? That ... pretty much cuts my interest right there. &gt; Smarty et. al. works the same way; it merely adds (a rather pointless) layer of abstraction where you basically can do the same things as in pure PHP with some syntactic sugar. Which is exactly what the person you're trying to argue for was arguing against. &gt; Code within templates isn't inherently bad. And yet the things you're pointing to aren't that, by grandparent's argument; they're function calls. &gt; I'm going to assume you missed the parts I'm sure you will. &gt; felt a couple of snide remarks and ad hominems And now we get to pretending that insults are ad hominem in order to sound Reddit smart. Ho hum. Not really interested. Feel free to tell me how I'm a condescending jerk because you told me a bunch of things which are opinions are somehow actually facts, tried to reinforce someone else's argument but ended up actually arguing against it, and generally trying to take a technical argument into a pissing match because your technical side wasn't up to snuff. Not interested. Have a nice night; this conversation was not, in my opinion, improved by your presence.

I think you may be overly optimistic about the amount of time it will take to crack a single password. Consider the Bitcoin network, which earlier this year was generating over 3.5 trillion SHA-256 hashes per second. The problem with using hashing algorithms like MD5 and SHA is that they're very fast. That's good for their primary intended purposes, but not good for securing passwords, because a single modern computer can generate millions of hashes per second with them. See: \* [How to safely store a password](http://codahale.com/how-to-safely-store-a-password/) \* [How to manage a PHP application's users and passwords](http://www.openwall.com/articles/PHP-Users-Passwords) by [Alexander Peslyak](http://en.wikipedia.org/wiki/Alexander\_Peslyak)

That's very clever. I like it.

Alright. Yea too many people have said objects, objects, objects already. I will work on it for sure, thanks for the info

When you first start working with objects, you'll just see them as a slightly different way of doing the same stuff you were doing before. Keep reading and practising and improving, and looking at other peoples code, and in time you begin to look at things in a fundamentally different way - and you'll end up actually writing really neat code that you (in some cases) can re-use in other projects with no fuss at all, and that is easy to come back to a long time later and adjust/extend, without breaking stuff that already worked. Objects are the way forward, take a leap of faith for now and you'll realise over time just what I mean. Linkage: http://net.tutsplus.com/tutorials/php/object-oriented-php-for-beginners/

Try Zend ACL. You don't even need to use the entire Zend Framework, it can be used as a standalone. http://framework.zend.com/manual/en/zend.acl.html

&gt; Feel free to tell me how I'm a condescending jerk I won't. I was actually looking for a constructive discussion, but you seem to be more interested in insulting me for whatever reason and I'm not playing that game. You have yourself a nice night as well.

&gt; I was actually looking for a constructive discussion Then maybe you should have tried not making the other person so annoyed at you. &gt; but you seem to be more interested in insulting me You keep pretending this, as if it's not what's keeping you from getting what you say you're looking for. You have not been insulted. Please stop pretending that you have been.

Set up the development stack on your local machine, work there, commit your files to the version control of your choice. When done, deploy with a script that looks up what files have changed (in version control), and uploads it to the remote site.

I had a suggestion for how you could improve it but it looks like you beat me to it :) http://improvise.kingston-fear.com/

I would use [HTML Purifier](http://htmlpurifier.org/) for the input filtering. It works really well.

Hmm yes that does look nice!

Actually I think you are underestimating how many possible combinations of 512bit there are: 2^512 = 1.34078079 × 10^154 possible combinations. That's more than a googol. (3,500,000,000,000 \* 60secs \* 60mins \* 24hours \* 365days) = 1.10376 × 10^20 hashes per year ( 2^512) / (1.10376 × (10^20 ) ) = \*\*1.21473943 × 10^134 years\*\* to crack just one password Ok so maybe before the end of time you might have cracked one password.

I'm no designer, that's just an improvement I came up with in a few minutes as an alternative to figuring out how to use the original site design that put me off before, though it could be used as a basis for something nicer still I guess.

yea, right now my code is half object, half proceedueral (spelling?). I actually already have that tutorial bookmarked from googling the other day. I love the tutsplus network as a whole, they really do have some top notch stuff. When i learned php years ago, oop wasn't in my vocabulary at all. Part of the reason im doing this project is to throw out the old and bring in the new. I have always wanted to do oop but i just never buckled down and did it, thats what one of my goals is with this project. I know a little on oop because of the code igniter framework, so its not a complete strange concept. from what i have seen, oop is a much cleaner well structured way of doing things, not to mention extremely flexible in being modular. To get a general idea of my end goal, look at bamboo invoice. I am basically doing that. I am building a multi user invoice/point of sale system being cross device agnostic. I want it to function and work on iphones, tablets, android devices, with touch screen capabilities all from the browser. Quite a task i have ahead of me, but im sick of all these invoicing "apps" that just plain suck. On top of sucking, they are restricted to one device. since mine is web based, that is not a restriction. Since it is open source, people can download and use it on their server privately if they wish, rebrand it, expand contribute code/etc. I am basically looking to be the wordpress of invocing software. Thats why im asking reddit every chance i get. Even though its nothing more than a simple login system with little security, i feel getting input in every step of the process will keep me in line to make sure its done right. anyways, thanks for the info :)

thought about it, but its a bit too bulky for my taste. i have used it in the past though and it does work wonders

&gt; That's not the point. Number one, that's well within the reach of botnets that are currently being used for this purpose by blackhats. Number two, that's well within the reach of $50,000 GPU clusters on the order of weeks. Number three, computing hardware gets cheaper year by year. I did some calculations [here](http://www.reddit.com/r/PHP/comments/kpwyn/need\_some\_advice\_on\_my\_php\_invoicing\_system/c2mcltw). You can run them again using google if you like. Even 3.5 trillion hashes a sec is still going to take them 1.21473943 × 10^134 years to crack just one password. I'd like to see your proof that it will take a $50k GPU cluster to find the original password from a salted + hashed sha512 string in just two weeks. Besides two weeks would be enough time to get your users to change their passwords if your database was compromised. &gt; I ... question this judgement openly. These numbers appear out of thin air, and VPS certainly isn't fourty dollars a month. You can get a bog standard VPS with \*\*no\*\* dedicated CPU for $15-30 USD a month. The CPU will be shared across the whole server I imagine and how much you get depends on the utilisation and number of VPSs running on it. If you want dedicated CPU then you go to a provider that gives you that. I just pulled some numbers from [here](http://vps.net/product/cloud-servers/). So drag the slider along and 1.2Ghz is $38/month. Admittedly I now see that's actually for 2 nodes. &gt; This from a guy using PHP. What web scripting language is faster than PHP? It suits me just fine. If I really need to micro-optimise it like Facebook then maybe I could something like HipHop. &gt; Mmm hmm. AIG wasn't. Kaiser Permanente wasn't. I can't find the articles about them. But probably they weren't salting the passwords or they were using something like md5 hashes which can be trivially cracked or rainbow tables used. sha512 is a whole other story.

Cool. I've had a look at a drupal addon called OG which might work. I've used a Cake ACL component before, half-heartedly. Will check this out. Thanks.

Ok, will take a look, thanks. I've used the Zend Lucene Search port before.

I appreciate the links! Apparently you're right about "Part of the problem is that the developers have been working at such a high level...". Only one comment. lol

Your maths are completely unrelated to the question at hand. If I'm brute-forcing, I'm not searching for a 512bit key (I already know it, because I owned your box, remember?). I'm searching for a password, which is probably 8-20 characters long and probably comprised of [a-zA-Z0-9] (plus maybe some punctuation). Hashing algorithms are \*designed to be fast\* -- fast hashing is exactly the opposite of what you want in this situation. You want to make computing a hash hard, to slow down brute-forcers.

Oops, try now. I left out a bracket after adding proper username validation.

A hash is a one-way function - given \*y\* and \*z\* such that \*z\* = hash(\*y\* + \*x\*), it must not be feasible to compute x. You can't 'decrypt' a hash unless there's some flaw in the hashing algorithm. The only way to figure out what x (the password) is you have to brute force calculate all the possible hash outputs. So you're sitting there with your computer trying: if (hash('sha512', ('aaaaaa' + $salt)) == $hashedPassword) { echo 'found the password!'; } Continue on through a massive loop trying passwords 'aaaaab' and 'aaaaac' etc all the way up to 20 characters. Maybe you've got a dictionary of common words to try first and see if the hash matches so you run those through first. Still going to take you a truckload of time and resources just to get even one password. Remember there's no rainbow tables available for sha512 like there are for MD5 so it would just take too long to generate all the possible combinations. Assuming no-one uses a password longer than 20 chars for a website you have to calculate all the possible hash calculations with all the salt combinations (random 20 chars + random 512bits) to build your rainbow table. So to calculate all the possible combinations will likely take until the end of the universe. Even storing all the possible hash calculations could be 7.80437138 × 10^143 TiB. I can't even be bothered working that out in yobibytes. Even after some years you've found your first password it's no longer useful as the website has forced all the users to change their password after a few weeks when they discovered the data theft. Also you're assuming you even know the hash function used. Maybe you've only hacked the database server and dumped that out. Maybe you don't have access to the PHP code so what if they're using whirlpool or some other hashing function.

Do you mean identity, authentication, or authorisation?

I've been using Codeigniter a lot recently. It uses the traditional Model, view, controller which typically: - Controllers map to URLs, and use models and load views. - Models provide access to a database or files on disk. - Views output HTML using data given from the controller. CodeIgniter also has Libraries, which are classes, and helpers which are basically a file of functions. Codeigniter provides it's own auto loading system. I suspect the 'objects' you're talking about would map to libraries quite well? You can organise your library objects and create what ever patterns you want eg abstract classes. The controllers would create use your library objects. I'd still keep models as models, then your library objects and controllers can use models as needed.

To do even a simple prepared statement query in PDO and do the connecting and proper exception handling it is like 50+ lines of code. Do you really want to copy paste that around your app 100 times everytime you want to run a query? It's way cleaner to just go: $query = 'SELECT \* FROM users WHERE user\_id = :user\_id' $params = array('user\_id' =&gt; 123); $result = $db-&gt;select($query, $params); $result can return false on error or the result set. You can handle that in your app however you choose. So yeah definitely wrap it up in a database class. Or use an ORM.

&gt; and practically dead CF is alive, growing and being actively developed. It's no PHP, but its still got plenty of life. Also, alternative engines such as Railio and BlueDragon are free and open source. Again, not that I'm trying to claim CF is better than \*, but its a great system in the right situations.

Yeah, I know what a hash is, thanks. Just to be clear here, here's the scenario: 1. I'm an evil blackhat. 2. I own your box. I have your database here locally now. 3. In your database, I have rows that have user's email address, salt, and single-iteration sha512 hash of password+salt. 4. My goal is to get as many plaintext passwords as I can. I don't need them all, but the more the merrier. So what do I do? I get a big dictionary, and a nice algo that makes variations on dictionary words (1337-alpha/digit-substitutions, etc). So for each word in my dictionary, I hash it with the users salt and see if it's a hit (and then do all the variations from my algo, maybe a few thousand of them). Whenever I get a hit, I take that user's record off my list, because now I know their password. Then I repeat for the next word in the dictionary. The size of my search domain, relative to an exhaustive search of all 20-char passwords, is incredibly tiny. Now assume 50% of your users have passwords based on dictionary words. Your math answers "If i have a good entropy-full password, how long will it take a cracker to find it?". \*That's the wrong question\* The right question is: "How many of my users' shitty, obvious, passwords can this guy guess in 24 hours?" \*\*EDIT\*\*: &gt; Also you're assuming you even know the hash function used. Maybe you've only hacked the database server and dumped that out. Maybe you don't have access to the PHP code so what if they're using whirlpool or some other hashing function. Hey, sure. Even better, maybe I didn't even get access to the database! What are we even talking about, just store those puppies in plaintext!

From what I've seen with Codeigniter, it's autoloading is not real autoloading. In Codeigniter you just make a list of classes to be included in every page load. With the SPL autoloading you do not need to include the file. The first time you use a class does not exist, your autoloader class will take over and include the missing class. You set what class does this with spl\_autoload\_register(); I know its not the end of the world but when starting a new project it would be nice to have. Also, it shows the framework is up to date with the newest stuff in PHP. From what I've seen with forum posts of Codeigniter abstract classes are either a hack or a plugin. Which I guess isn't the end of the world but it shows no consideration was given for this. It seems you need to do a hard include or put two classes in a file.

How are you calling test()?

Well isn't that why you have password strength enforcing techniques that have a minimum length, some random characters etc? What kind of system are we trying to protect here? What's so interesting about my site that they want to spend time cracking the passwords? If I'm Joe Bloggs and you've found my Facebook password. Is it really that big of a deal? You gonna post some bad stuff about me on my wall? :O Even if Amazon lost their usernames and password hashes. Is there a big deal? Unless Amazon have saved the credit card details in the system as well then perhaps some people could log into the accounts, change the addresses and buy stuff with it. That's a whole other ball game if you're storing credit card details as well though. What if you're designing for something serious like the US defense network or a bank? If you're a bank, then yeah maybe some money is at stake if they can access the user's account and transfer the money out. Then maybe you force everyone to have a 20char strong password and 2-factor authentication. That way if they get the password then it's useless without the random generating number token device which the user is carrying around as well. Irregardless it's a lot more difficult to hack a bank's internal systems, penetrate their multiple zones of firewalls and somehow breach their mainframe as well. US DoD network, forget it. Their interesting stuff probably isn't even on the internet.

I think Zend is about as OO as you can get in PHP. You might want to check that out. CodeIgniter will let you do OOP as well. Yes there is a loader but you don't have to use it you can use standard OOP practices in most places if you want.

You're right about Zend. One of the first topics it covers in the documentation is SPL autoloading. It may be large and complicated but that is not a bad thing.

You spelled \_\_construct wrong in the Test class. Because of that, the test() function actually takes over as the constructor (PHP 4 backwards compatibility). So func\_get\_args is returning the Load and Config objects you meant to pass to the constructor. To resolve it, you should fix the spelling mistake in \_\_construct. You should probably also consider not naming methods the same as the class name to avoid similar issues.

I've been using Zend Framework for a couple of years now and it has increased my knowledge of OOP greatly. The documentation isn't the best but it is a powerful framework and provides a lot of functionality to create great applications.

Now you're grasping at straws. Seriously, stop and think. This stuff is important. This is not about protecting your system -- it's already been owned. This is about protecting your users. As a programmer, you should at all times consider your users to drooling idiots of the highest order. Of course all of them won't be, but many, many, many of them will, and those are the ones you need to consider when you're thinking about security (and data integrity, and a host of other concerns) Your users will use the simplest passwords possible. No amount of fancy requirements on the password will change that. "hunter" becomes "Hunter123". And they will use the same password on your site as they use on the computers at their job, on Amazon, and on their bank's web site, with only minor variations which will be easy for a computer to guess. Your job is to protect these idiots. Your reason for giving them less protection is that you're concerned about thousands of them logging into your $20/month VPS in the course of a few seconds. You back it up with math that makes assumptions about password selection that are completely divorced from reality. Do you have no sense of how absurd that sounds? Do you really think all these people using 1000 iterations of SHA512 or using bcrypt are crazy and you're the only sane one?

When you talk about spl autoloader, are you thinking that it uses that for controllers too, or just that you want a framework that does use it and is built around it? Can you give an example about how you are thinking abstract classes would be used?

Oh wow, that was dumb haha. Thanks!

I really like the symfony framework

[Symfony 2](http://symfony.com/) is nice.

Absolutely. ZF is my go-to framework for a lot of stuff. I'm trying to find time to look at symfony2, now that it's done -- I did some stuff in 1.4 and it went ... okay. I like ZF because it's not so tightly coupled. When I did stuff in Cake years and years ago, and symfony just a few years ago, everything was awesome until I started doing something the framework didn't anticipate. Then I had to resort to reading framework code, since all the internal bits weren't very well documented. Lots of times, I found out the component I wanted to bend to my will was just too rigid. ZF makes fewer assumptions about what you want to do. This means there's a lot of configuration/dependency-injection necessary to do things that would be like zero-to-two lines of code in other frameworks. But the learning curve is worth it because you inevitably learn the framework inside-and-out. The fact that the "model" bit of the mvc is so anemic (boring table-gateway pattern) encourages you to actually do your job, which is to figure that shit out for your own application. I've been using the Doctrine2 ORM alongside ZF with great success for the past year or so. The default controller/view stuff is good enough to make those boring bits fairly painless. And then you've got a huge library of things that are useful in a vacuum, but also work together. Check out Zend\_Log, Zend\_Mail, and Zend\_Log\_Writer\_Mail, for instance. It's easy to set things up so your logger will email you any log message more severe than NOTICE, or whatever level you want. Useful components on their own, with nice synergies when you combine them.

It's going to depend on complexity and your style. There are really 2 types of frameworks that both have their strengths and weaknesses. There is "Convention over Configuration" which typically dictates that there is a set convention that you have to follow with little opportunity for deviation. Frameworks that follow this paradigm are typically quicker to develop with because there's not as much open for interpretation by the programmer. CakePHP and CodeIgniter follow this paradigm and as such, you get the "bake" console in CakePHP that lets you generate controllers, models and views. The other paradigm is the opposite "Configuration over Convention". This paradigm is slower to develop because you get maximum ability to tell the system how it should go together and gives you the ability to customize what components go where and how they should be used, there are also less assumptions made about how things work. Zend follows this paradigm and as you would expect, to develop a simple website, it's slower then CakePHP, CodeIgniter, etc but if your doing something that is outside of the "comfort zone" of those frameworks you are much better off. I prefer Configuration over Convention because the projects I work on are rarely a simple CMS or website and involve background processing with job queuing systems, data analysis or other tasks that don't fall inside of your usual web based controller/view request. Also, be aware that alot frameworks talk about the Model being just your access to the database. This is wrong. Your model is your data source and your business logic - Lookup something like "fat model skinny controller" and you will find info about how the controllers should be very minimal with the majority of the business logic is in the model. This is important because when I first used CakePHP they drive into you that models are your database access and not much else. Thats my rant...

You're the one grasping at straws. Your 1000 extra iterations of hashing is a mild bandaid on the actual problem. The main problem is you lost all your data in the first place. That's the real security problem. You need to take major preventions against that first. &gt; Your users will use the simplest passwords possible. No amount of fancy requirements on the password will change that. "hunter" becomes "Hunter123". And they will use the same password on your site as they use on the computers at their job, on Amazon, and on their bank's web site, with only minor variations which will be easy for a computer to guess. If they're stupid enough to repeat the passwords on every site they use even their banking sites then that's their problem! Also just because you have the username + password they use on one site, doesn't mean it's going to work on their banking site. A bank usually have their own online user ID/customer number you have to use. That's the second half of the puzzle you don't have. &gt; Your job is to protect these idiots. Actually no it's not. If they want to use insecure passwords then there's no liability on my site/company if their passwords or account is compromised because they used a poor password. It's in the terms and conditions. &gt; Do you really think all these people using 1000 iterations of SHA512 are crazy and you're the only sane one? Yes. Tell you what. Open challenge to the internet. Here's my online banking login page: https://secure.nbnz.co.nz/IBCS/pgLogin I hashed my customer number and password with the method I was talking about earlier. Here's the random sha512 salt: ca6ff1df5e2f75080c103aee6bca791b687334651457c13f876eeae9aee01186beb937bc21c69bec26b487f7eec89d75d431efbcf8f5a141ad4bef1ac26243dc Here's the sha512 hashed username and password: 4c77e09d28f24d427698c343e64f94655b59c3ce54ab70426ba126a6454e59a37dc960d5d1ae76a50e7cbc0e81d2a2beedc8548db2d837e6060d7f49d4f479b8 You should be able to reverse engineer that and get my login credentials in about 2 weeks according to your own logic. When you're done you're free to transfer a million dollars to yourself.

Well done, sir.

No. The fact that your database is compromised is \*\*assumed\*\*. Otherwise, why not just store passwords in plaintext? The fact that your users are dumb might be "their problem", but protecting them from their stupidity is \*\*your responsibility as a professional\*\*. Regarding your "challenge", I don't have the tools the real bad guys do, and your password is probably better than 99% of your stupid users. But it's not about your single, lonesome, password. It's about "how many passwords can I guess out of these N passwords chosen by dumb users". Get that through your head. The fact remains that if you have thousands of users, a dictionary-based attack might well reveal a substantial portion of their passwords in "days" if you hash with only one iteration of SHA512. You can trade a few scant milliseconds computation-time and turn days into \*\*thousands of days\*\*. And yet, you think it's a bad idea because you think "gee, maybe all that extra hashing for my non-existent millions of users will chew up resources on my shitty $20/month vps". I can nearly guarantee you that the big boys like Google, Facebook and Reddit are using some very expensive hashing, and they could save BIG MONEY if they did it your way. They don't. You know why? Because they have a bunch of engineers working for them who are smarter than both you and me. But they're all crazy and throwing away buckets of money, because zoszsoz is god's gift to software development. Seriously. This is an issue of \*\*professional responsibility\*\*. Lose the ego, and find your brain. It's in there somewhere.

And just to be clear: &gt; You should be able to reverse engineer that and get my login credentials in about 2 weeks according to your own logic. I never said any such thing. The fact that you'd say so illustrates how poor your comprehension is.

Thanks for pointing out the proper use of models. CakePHP was also my first MVC framework, and I also was led to believe models are nothing more than front ends to the database. This is so terribly wrong. The controller is meant to be a thin-client sitting between the view and the model. It's sole purpose is formatting information going to and from the view and model. I think people learn this early on when they find themselves duplicating logic in a lot of different places, and finally realize that logic should have been inside the models the whole time.

&gt; Regarding your "challenge", I don't have the tools the real bad guys do, and your password is probably better than 99% of your stupid users. But it's not about your single, lonesome, password. It's about "how many passwords can I guess out of these N passwords chosen by dumb users". Get that through your head. Yeah I already understood that genius. The argument was already anulled as there are already minimum strength password policies in place that make this magical algorithm of yours that guesses all the low hanging fruit passwords a lot less useful than you're making it out to be. Hell if I wanted to be really clever at registration time I could run the password against a dictionary and if the password contained a string in that then they wouldn't be allowed to use it. That would foil your dictionary attack and you'd be left to brute force combinations of all the passwords instead. That'll take you a while. &gt; I can nearly guarantee you that the big boys like Google, Facebook and Reddit are using some very expensive hashing, and they could save BIG MONEY if they did it your way. They don't. You know why? Because they have a bunch of engineers working for them who are smarter than both you and me. Well Facebook apparently processes 175 million logins a day. Assuming your 1000 hashes that's a phenomenal waste of CPU hours each day for a theoretical attack that may or may not ever happen. Reddit sure as hell doesn't have the servers or money to waste doing that much hashing for its millions of users. &gt; Seriously. This is an issue of professional responsibility. Lose the ego, and find your brain. It's in there somewhere. Better yet rather than wasting time and money hashing, how about protecting your database server better in the first place. Ie. not exposing the database server directly to the internet, properly locking down the OS, protecting against SQL injection, putting it behind some firewalls and IDS systems, encrypt all the confidential data. These few options are a much better idea than implementing some magical 1000x hashing algorithm that may one day save your data but cost you 10s of thousands doing it. Because at the end of the day the username and password isn't that useful if you've got the entire database anyway. You've already compromised the privacy of all your users. But hey the magical 1000x hashing will save you.

&gt; This is important because when I first used CakePHP they drive into you that models are your database access and not much else. Who's \*they\*? Where did they drive that into you? Practically, yes that is the most obvious thing about models, is that they act as the abstraction layer to the database, but I've never got the idea that that would be all they did. Behaviours for them illustrate this; some of the most powerful functionality you get is from modifying models. I think convention over configuration is about having sensible \*\*defaults\*\* that \*work\*. But there is \*nothing\* to prevent you from overriding those defaults.

Any framework using php 5.3 and namespaces will in all likelyhood be using the autoloader. Yii is the most object-oriented framework I've seen, although I'm not sure I've come across any class that doesn't use classes.

Assuming "good OOP design" is an euphemism for excessive design patterism and not reasoned usage, then look into Lithium, Symfony, FLOW3. On the more practicable end I'd consider CI, Lion, ZooP.

Not a PHP programmer for about six years now - just a security person. Turn off directory listings. ([example](http://improvise.it/core/database/) ) Use a better solution for escaping. I should be able to discuss the '&lt;' character or the &amp;amp;amp; character sequence character. I haven't properly audited most of those php scripts, but the ones in [/core/content/](http://improvise.it/core/content/) look unauthenticated in parts? [logout CSRF](http://improvise.it/core/logout.php) Please use SSL or HTTPS, or atleast make it an option. Also, what's this function doing? function updateArb(xprompt, xdefault, table, column, row\_id){ var xvalue = window.prompt(xprompt,xdefault); if (xvalue != '' &amp;&amp; xvalue != null) { $.post("database/update.php", { table: table, column: column, row\_id: row\_id, value: xvalue }, function() { fresh(); } ); } }

Doesn't have to be git, but it would be a good choice. Try looking at mercurial, it is ~90% the same as git, but more user friendly. Also, read through [hginit](http://hginit.com). After you grasp mercurial (the concepts behind distributed version control) it will be easy to switch between any distributed version control system.

&gt; A super easy example of this is the mysql\_query function, it's easier to pass everything straight in as a string rather than escape everything yourself Two things wrong with this. 1. Nobody should be using this anymore in PHP. For a long time now, PDO has been the recommended way of accessing MySQL or pretty much any database. mysql\_query is deprecated and positively ancient in PHP history - even mysqli, which replaced it and which does have parameterised queries, has also largely fallen out of favour since PDO. 2. The function mysql\_query comes from MySQL's C API. Any other language, not just PHP, which uses this API will have this function. Thankfully with PHP, there is a better and more secure way (PDO).

Kohana. I think it is very well laid out. Additionally, I use the mustache module for it and think that that helps make it even better It autoloads based on name. For instance, Model\_Part will search application/classes/model/part.php then all the modules for $modules/class/model/part.php and then the system for $system/classes/model/part.php. This happens when you load the class, you don't have to give it more information.

&gt; Oh wow, that was dumb haha. You should crosspost this to `/r/