

Re: Ideal Web Development Environment?

Source: <http://coding.derkeiler.com/Archive/PHP/comp.lang.php/2006-05/msg02141.html>

- *From:* "Richard Levasseur" <richardlev@xxxxxxxxxx>
 - *Date:* 31 May 2006 01:01:36 -0700
-

Colin McKinnon wrote:

Richard Levasseur wrote:

(Forewarning, most of these problems and solutions come from being the only developer in a 1 server department with no budget, few resources, unresponsive IT, and non-technical managers, so thats where I'm coming from.)

I've never had to cope with **all** those at the same time – at least not doing real development work. Good luck.

lol, need more than luck ;). Government job, though, go figure.

- Fast speed of development and deployment
- Shared codebase

Don't go there. By all means re-use code, but if it's seperate projects then use a seperate codebase.

Perhaps, but if i did that, i would have, easily, 10x the code base to maintain (every project would have different versions of the same files every project uses). Trade off I guess.

My latest approach is to make every function or group of functions its own file, so if i need to just keep one file, i can drop it in the projects directory and it'll use that one instead of the one from the standard include directory. (Not to mention this makes it easier to maintain commonly used functions)

I'd say the basic tools for real PHP development are:
code editor
documentation generator

Re: Ideal Web Development Environment?

unit test kit
bug tracker
continuous integration kit
style rules
version control (be it policy or software)

For that I favour:

Vim
PHP Documentor
PHPUnit
RT
Reflux
A homegrown document
(different things in different contexts including a homegrown solution and CVS).

RT – Interesting, this would have been useful last year when they hired an intern to make something like this in pagemaker pro (or whatever that mac thing is called, can't recall exactly)

PHPUnit and Reflux looks interesting, too, I'll give those a try when i get the chance.

When I originally started, this elaborate process was the really simple way we all started with: rename index.php to index1.php, modify index1.php, make sure it works, overwrite index.php with the new one.

You really want to keep the old ones too.

The old ones are kept by the version control.

– Setup a Subversion server with a post-commit hook that updates the development server with the latest version (as a working copy, preferably), so that everyone can see what the 'final' version looks like.

Always pushing out the bleeding edge version to a shared development box sounds a bit dangerous to me.

I think you misread – the latest is only put on the shared development box after your changes have been verified by you (pseudo client side, in this case) and then committed. You don't edit-save-commit-refresh-repeat, you just save-refresh-test. You commit your code to the dev box after you're done testing in your

Re: Ideal Web Development Environment?

sandboxed area of the dev box.

Pseudo client side because your 'client side' is really just a separate location on the dev server, so as to avoid having to install apache, php, database, third party tools, onto your own computer (which could be impossible or a PITA in some cases).

– Create a production server in Apache and write a shell script to export the specified tagged version, as well as a shell script to apply patches. Ideally, no person should have to touch the production by hand, except to alter config files (which don't change much, once setup)

I prefer to do this by replicating from the tested environment.

Can you clarify?

- * A few notes about this setup:
- * Config Files: There have to be many types of config files. One for the user (this isn't versioned) so he can override settings as necessary without changing the versioned config file. One for the development server (this isn't versioned as well) so it can override necessary settings. One for the production server. The production server config file conditionally includes the user and dev config files if they exist and the globals are appropriately define()'d. A problem I haven't quite solved is that production configs will be overwritten, so the shell script (that copies from subversion to the server) needs to detect when it has changed to save the old one and notify the person (ala Gentoo's emerge)

hmmm. How much config is there? I prefer to maintain multiple include directories becoming increasingly general as you go along the include path – that way you can override, say an application include file by specifying a file of the same name in the host-specific directory.

Its the config for the application, not 'system wide' settings (though i suppose the same concept would apply to them).

Maybe an example would clarify:

```
/www/myapp/core.php: include('config.php');  
/www/myapp/config.php
```

Then in config.php you have all the constants, servers, aliases, permissions, etc etc.

Thats fine and dandy, but what if you need to change some of those

Re: Ideal Web Development Environment?

values to do some debugging, or include alternate files? In my case, `define('DEBUG', 1)`, `define('DEVELOPMENT', 1)` will modify some standard functions and methods so that they behave differently: queries give full error messages, email won't be sent out, certain conditions are treated as fatal instead of just warnings, etc.

If you change `config.php`, then you'll have to revert before you commit to avoid any changes to the file. But then what if you've added something to `config.php`? Now you have to go through and remove your debug stuff. Tedious to say the least. That's what `user.dev.php` is for – applying user changes to the config without having to fiddle with the actual config file.

The other situation is to handle differences between the dev server and the staging/live server. Stupid little things, you know? Like someone forgot to set the default `include_path` correctly, or something wacky is going on with the server that is easily fixed with a config file change while someone else fixes the root of the problem. That's what `server.dev.php` is for.

This way, you're able to: not change the version'd `config.php`, avoiding any possible confusion why `config.php` was updated without meaningful changes, and not having to deal with tedious back tracking when you do need to update `config.php`, and not having to worry about an update overwriting any custom values you had in the config for your own development.

Though, now that I think about it, it might be easier to have `./dev` on the include path, however, the dev configs are meant to simply override the existing include. If `./dev` was added to the include path the dev file would have to be a full copy of the original config file. It also wouldn't allow per-user and per-server config overrides. Also, how could you conditionally include the configs? If you need to turn `DEBUG` or `DEVELOPMENT` off for some reason, you'd have to change the include path, which would come with, imo, more complications than simply manually conditionally including a couple files in the main config file.

Neither seems ideal, imo. Mine is pretty complicated, though, it may be easier to simply have `./dev` on the include path.

* In cases where you don't have a dedicated SQL server for each of development and production, I've found that creating a development account with access on `dev_%`

I was with you at the first sentence in that paragraph, but lost you on the second. Although I understood the first paragraph, I have trouble believing it. You build your application by structuring data. Where that is not enough you write more complex queries. As a last resort you write code in a

Re: Ideal Web Development Environment?

procedural language. The first two are critically dependant on having a test database. There's lots of ways of managing different versions of code. There's only one way to manage different database schemas.

Let me clarify:

Ideally, you'd have a database server for development and a database server for production – two completely seperate entities so you don't hose the real data. Not everyone is so lucky, though.

In my situation, I only have one physical server and there is no chance of getting a second. In MySQL, you can create masks for permissions to apply to. % is synonomous with *. So if I give you permission on databases dev_%, you can modify dev_foo, dev_bar, but not foo and bar. Your table names stay the same, as you are operating on tables in a database, you don't care what database, just so long as the table names are the same. In Postgres you could simply connect to seperate postgres databases and use the schemas as databases (since you can do schema.table in postgres).

It isn't the most elegant hack and has a few caveats: It makes cross database queries a little tricky. On the dev server you want dev_accounts.users.last_name., but in the code you'll want to write accounts.users.last_name. So unless you have some globals or equiv defining the database names, then you have to manage this manually or come up with a way to obtain the database name you would *want* connect to. Could be tedious. Its not my favorite solution, either – it would probably be better to start another process of the database on a different port using a seperate config file.

An alternate method is to abstract the database structure into code so that you just do \$ACCOUNTS_DB->getDatabaseName();, and override \$ACCOUNTS_DB if necessary. Funnily enough, I have set of classes that completely abstracts a database and its structure into objects like this. But sadly, its new and no projects have been able to fully use it, yet.

* This doesn't quite work for handling SQL changes. If a table was added, modified, etc, then releases must be manually compared and notes checked for changes to the .sql defintion files, then the live database updated accordingly

There's code on PHPClassess for replicating a MySQL database structure. Having said that, it's kinda broken. (Diogo, if you're listening – you going to put in my patches?).

Even when it does work, I often find its better to think about updates as part of the deployment process – and script the changes.

Re: Ideal Web Development Environment?

I remember seeing something like that on phpclasses, but never got around to trying it out.

It would be nice to script the changes.

Identifying dependencies between projects:

No software I know of does this, really, except grep. The goal is to identify other places that might be affected by changes in common code (utilities.php, user authentication code, etc). Grep isn't so bad at it, really, but it would be nice to have a tool that did this, ya know?

Been there. When I used to write MS–Access applications I had a database which managed that (when I remembered to keep it up to date). Where I work now, there's a tool they use from Quest which is OK. Good idea for a software project though. You can build CASE tools ion PHP you know ;).

What would be nice is if PHPDoc had an option to implicitly add a @uses tag for every function call. I manually document important ones, but sometimes...i just want to know how many things use something to know if i should or shouldn't fiddle with something ;)

Bug Reports:

<snip>

– I also would rather say 'Fixed Bug #14729' than 'Fixed issue with auto select not submitting when user entered malformed data into the wrong field' in change logs. Makes me sound cool and I feel like i'm getting somethign done. Plus, management likes seeing a slew of 'Fixed Bug #whatever' instead of technical jargon they don't understand.

Sometimes simple can be good. Like I said I like RT, but there's LOTS of tools written in PHP for this kind of thing. I don't get why you don't just bolt your front–end on to bugzilla.

Thats not such a bad idea – sounds like some good work for the new intern.

Re: Ideal Web Development Environment?

API Documentation:

This is really a matter of discipline while coding: remembering to write the javadoc syntax. The API can be autodocumented with the post-commit hook on commits. Ideally, it would be nice to integrate this into a wiki to preserve code changes and api documentation changes, as well as provide a means of easily editing and commenting the docs.

Sounds a bit impractical having updates to the content driven from different sources (source code and wiki). Go back and read the manual for your documentation tool of choice. PHPDocumentor and doxygen are capable of a lot more than just minimal class and method labelling.

I was a little vague, let me clarify:

The wiki is for the user manual, PHPDoc is for the API documentation.

I chose to use a wiki for the user manual instead of the PHPDoc tutorial/example tags for a few – reasons:

- The user manual is for my users, who, if there is any sort of function signature on the page, I fear they will become confused and will stop reading immediately. A wiki is much friendlier looking than any of the PHPDoc templates i've seen. While I have little faith they'll figure out what `[[Foo|bar]]` does, I think they are much less intimidated by `[[Foo|bar]]` than public static mixed function `foo(string $bar, [mixed $foo=NULL]) ;)`
- The users can edit the manual themselves and add notes in the future. My users have dozens of stickies detailing the most simplistic tasks. How to search for records, print them as PDFs, where to print them, options to set before printing, etc etc etc. Total amount of clicks is about 2 or 3, and there isn't much I can do about that in the PHP code, and I don't feel it belongs in the PHPDocs (which are more intended for the developer than user).
- Its faster than regenerating the PHPDocs for minor revisions. It takes at least 30 minutes for PHPDoc to go through all my code. Partial due to the hardware, partially because of the codebase, and i'm sure it isn't configured optimally, i've only recently installed it.

Tool Support (Editors):

I've yet to find an editor that satisfies me fully. Everything has something nice, but they are all lacking something I really want.

Ideally, it would:

- Support Subversion (or CVS, as the case may be)

Why? Personally I've never found a sensible reason for using an IDE (unless you count Unix as an IDE). Sure there's lots of silly reasons – like its inconvenient not to.

Re: Ideal Web Development Environment?

I should have clarified: by IDE i just mean a set of tools you use. I don't use an IDE per se, but I have a set of tools and programs that help me. It'd be great if we could mix and match all our favorite apps into a single program, but thats quite a pipe dream!

- Save a backup/second copy of the file to another location every save (1: network drives over VPN can be slow, causing the editor to lag,

This ain't going to happen. However if I were faced with the problem and had control over my end, it'd be quite simple to set up some scripting funky distributed filesystems to work around this.

Even on windows? My work does not provide any linux client to VPN in with. The alternative is to SSH in and work directly on the server, but sometimes I like to have an offline copy in case the internet goes down.

- Autocomplete/Intellisense of native PHP functions as well as user defined functions.

I would have thought the latter to be highly improbable if not impossible with a dynamically typed language.

I thought that, too, but the VS.php plugin does a remarkable job at guessing correctly. It is probably my favorite plugin for php development ever. I only wish i could use it with any editor.

- Regular expression searching of files *that display in a seperate area* (ala Dreamweaver, NOT like visual studio, where it flags the line)
- Jump to functionality. Visual studio has *alphabetized* dropdowns that help you jump between classes, functions, and methods within a file, and it is incredibly helpful.

ctags

Clarify? The wikipedia article on it doesn't tell me much, and its quite late to go googling.

Re: Ideal Web Development Environment?

Layer Separation:

I know there are template languages and such out there, but I've never gotten a chance to use them much. How well they work I don't really know. But with things like AJAX autocomplete, it would be nice to separate config, core functionality, and presentation so a simple autocomplete box doesn't need to have the server load the entire class and subsequent includes.

I was going to say that in the absence of graphic designers there's little reason to use a template system but then again my pet project PFP Studio is all about pushing so much more into the presentation layer!

This also ties back to project dependencies. It'd be nice to just have it server parse the config and core methods and not have to parse/load all the presentational code which won't be run. This is sorta related to design methodology, I guess, too. Core functionality and functionality should be separate anyways. And personally, I despise templates. I secretly would rather write some java gridbag than invoke Smarty.

– How does everyone deal with the problems of web development? My solution works, but it is somewhat ragtag and hacked together. Insight is welcome.

Format your harddisk and install Linux. You will hate it for 3 months, after 6 months you'll be at least as productive as you are now and soon after you'll hope you never have to go back to MS. Its just so much easier to automate stuff, to link events to processes, to move files and data around.

I wish I could, but I think it would be more trouble than it is worth. Not on my end, but with everyone else having to deal with whatever I gave them. I've almost got everyone using firefox, though ;)

C.

Thank you Colin!

.