

VIM FOR PHP DEVELOPERS

Why on earth would any sane person use Vim for PHP development ? Is it still relevant in the days of full blown IDE's ?



IKE DEVOLDER

@BlackIkeEagle

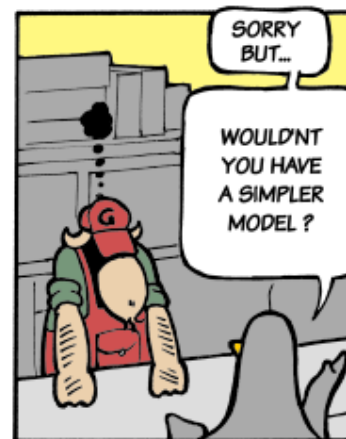
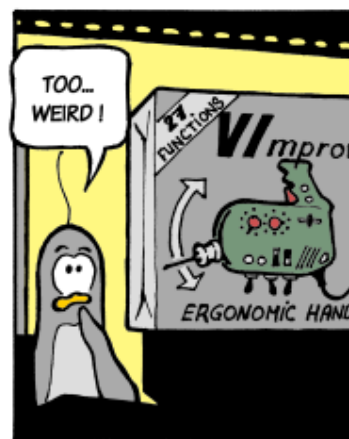
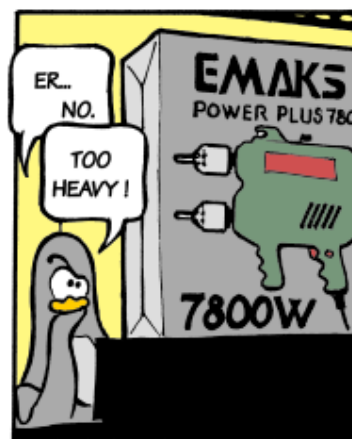
Senior Webdeveloper - Studio Emma

Archlinux Trusted User

Vim user :-)

THE GOAL OF THIS PRESENTATION

- Touch some basics
- Show some PHP specific things
- Inspire you to research more about Vim



WHAT IS VIM ?

“Vim is a highly configurable text editor built to enable efficient text editing. It is an improved version of the vi editor distributed with most UNIX systems.”

“Vim is often called a "programmer's editor," and so useful for programming that many consider it an entire IDE. It's not just for programmers, though. Vim is perfect for all kinds of text editing, from composing email to editing configuration files.”

WHAT IS VIM NOT ?

“Vim isn't an editor designed to hold its users' hands. It is a tool, the use of which must be learned.”

“Vim isn't a word processor. Although it can display text with various forms of highlighting and formatting, it isn't there to provide WYSIWYG editing of typeset documents. (It is great for editing TeX, though.)”

“Vim isn't a desktop environment. If you want integrated twitter, email, webbrowser, ircclient, ... use emacs.”

WHY VIM FOR PHP DEVELOPMENT

Vim has everything (or almost) a PHP developer needs.

We just need to unlock or find the “right” feature.

Once you know it it's FAST!

MODAL EDITOR

Wtf, modal editor?

A modal editor has different modes of operation.

- Normal mode
- Insert mode
- Visual mode

NORMAL MODE

In this mode everything you “type” is interpreted as a command you give.

INSERT MODE

This is what you are used to, start typing and you are adding text

- “i” => insert mode at current position
- “I” => insert mode at beginning of line
- “a” => insert mode at next position (appending)
- “A” => insert mode at end of line (appending)
- “o” => insert mode open new line after current line
- “O” => insert mode open new line before current line
- “r” => insert mode replace current character
- “R” => insert mode start replace from current character

VISUAL MODE

This mode could be compared to selecting some text

- “v” => highlight continuous text from current position
- “V” => highlight complete lines
- “ctrl + v” => highlight block mode (columns)

To define a visual highlight you can use all movement keys

MODES DEMO

MOVING AROUND WITHOUT MOUSE AND WITHOUT ARROW KEYS

If you really want you can move around with your mouse and
with your arrow keys

BASIC MOVEMENT



FASTER MOVEMENT

- by word
 - “w,e” => forward beginning or end of word
 - “W,E” => forward beginning or end of word (context)
 - “b,ge” => backward beginning or end of word
- line
 - “O” => beginning of line
 - “^” => beginning of line (non whitespace char)
 - “\$” => end of line

FASTER MOVEMENT

- “gg” => beginning of document
- “G” => end of document
- “{N}G” => goto linenumber {N}
- “ctrl + f” => 'scroll' page forward
- “ctrl + b” => 'scroll' page backward

SEARCH MOVEMENT

- “/{search}” => search forward for {search}
- “?{search}” => search backward for {search}
- “f{char}” => goto first occurrence of {char} on this line
- “F{char}” => backward of “f”
- “t{char}” => goto char before first occurrence of {char} on line
- “T{char}” => reverse of “t”

MOVEMENT DEMO



<https://bitbucket.org/tednaleid/vim-shortcut-wallpaper>

COMMANDS

Wtf ? Commands, this is just a text editor, why don't you just let me type in some text

COMMON COMMANDS

- Quit: “:q”
- Quit and I don't care about the changes: “:q!”
- Quit everything: “:qa”
- Write: “:w”
- Write and quit: “:wq”
“:help editing.txt”

COPY / PASTE

- Copy: “y” (called yank)
- Cut: “d” (delete)
- Cut Characters: “x” (cut after); “X” (cut before)
- Paste: “p”

“:help change.txt”

TEXT MANIPULATION

- Replace current selection: “r”
- Replace from current position: “R”
- Substitute (search and replace):
“:s/{pattern}/{substitution}/{flags}”
- Flip the case of character: “~”
- Increment number: “ctrl + a”
- Decrement number: “ctrl + x”
“:help change.txt”

UNDO / REDO

- Undo: “u”
- Undo last change on this line: “U”
- Redo: “ctrl + r”
“:help undo.txt”

QUANTIFIERS

Quantifiers can be used to 'extend' the reach of a command

- Move to the 5th word: “5w”
- Yank the next 3 lines: “3yy”
- Delete up til the end of the 3rd word from here: “d3e”

RANGES

Most commands support ranges, ranges are defined like

“:{start},{end}{command}”

- Visual selection: “:’<,’>”
- All lines in file: “:%”
- From line 4 to 10; “:4,10”

“:help range”

FORMATTING TEXT

- Center align: “:ce {width}”
- Right align: “:ri {width}”
- Left align: “:le {indent}”
- Format paragraph: “gqip”
- Format current selection: “gq”
“:help formatting” “:help text-objects”

COMMANDS DEMO

STILL NO WORD ABOUT PHP ?!?



NO PHP YET !

STUFF FOR CODERS, WHAT DO WE WANT?

- syntax highlighting
- completion
- search in files
- syntax errors
- project based configuration
- file/folder navigation

**MANY OF THE THINGS WE NEED ARE
AVAILABLE OUT OF THE BOX**

SYNTAX HIGHLIGHTING

If you are getting on a ubuntu server, first action:
`apt-get install vim`

In most linux distributions vim comes with syntax highlighting enabled by default.

In many distributions the default vim installed is a vi compatible tiny version, install the full version!

COMPLETION

- “ctrl-x ctrl-f” file names
- “ctrl-x ctrl-l” whole lines
- “ctrl-x ctrl-i” current and included files
- “ctrl-x ctrl-k” words from a dictionary
- “ctrl-x ctrl-t” words from a thesaurus
- “ctrl-x ctrl-]” tags
- “ctrl-x ctrl-v” Vim command line

SEARCH IN FILES

- “:grep {search} {infiles}”
- “:lgrep {search} {infiles}”
- “:vimgrep {search} {infiles}”
- “:lvimgrep {search} {infiles}”
- “:cnext” goto the next result
- “:cprevious” goto the previous result

The default search functions in Vim are fairly hard to use

SYNTAX ERRORS

You could manually abuse the makeprg setting in vim and set it to “php -l %” but when using PHP, html, javascript, ... you might want syntax errors for all of those languages.

PROJECT BASED CONFIGURATION

We must explicitly add a configuration option to allow vim to load a .vimrc file from the directory you are starting vim from.

`“:help exrc”`

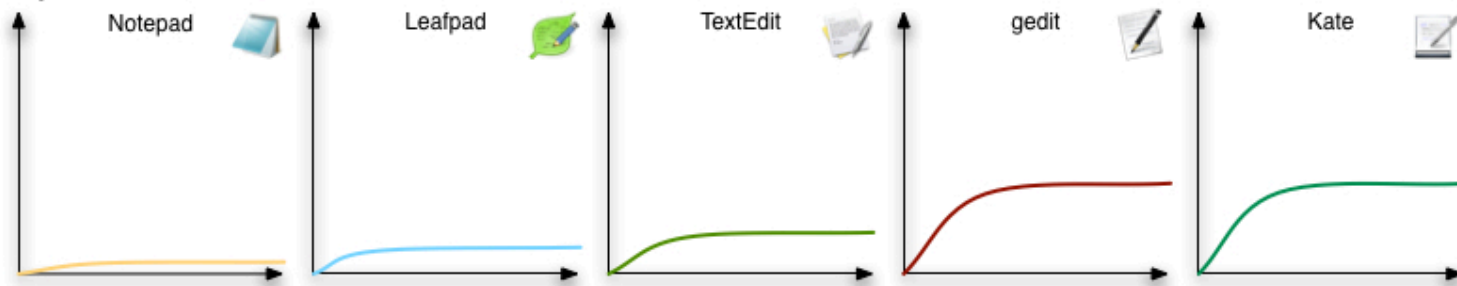
FILE/FOLDER NAVIGATION

There is by default a file navigator built in in vim called netrw, if you have to use it, it does the job fine, but it is not really user friendly.

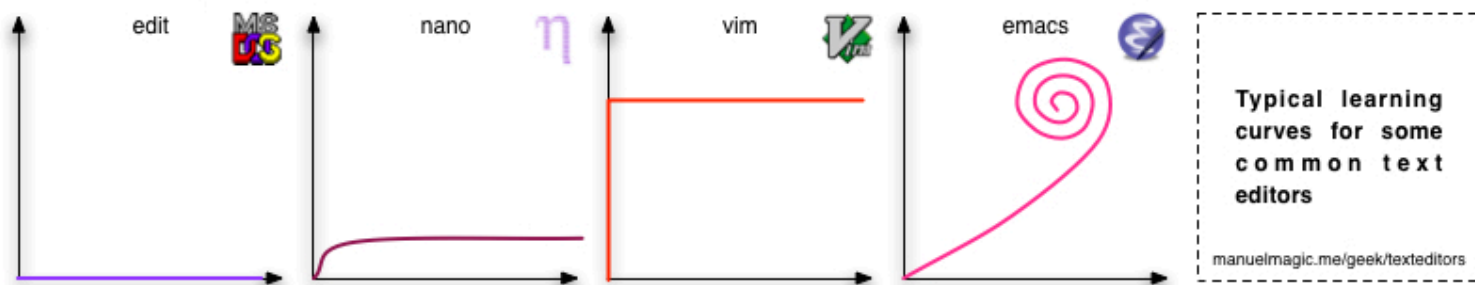
We can also easily navigate our files if we use tags.

OMG VIM IS HARD!

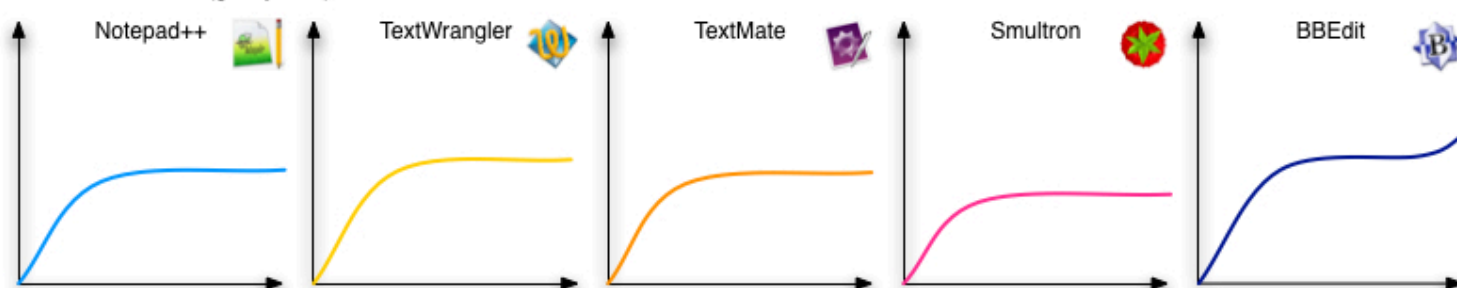
System default editors



Terminal editors (nerdy stuff)



Freeware editors (geeky stuff)



CONFIGURE VIM

The configuration will be done in “~/.vimrc” and additional plugins to make our lives easier go into “~/.vim”

CHOOSE MODE

Vim has a vi compatible mode or you can choose to use the no compatible mode to be able to use all Vim specific features

BACKUP RULES

Vim creates a backup of every file you edit, it also keeps a “swap” file by default. Vim also keeps views around but I personally never saw one saved to disk.

Vim can also persist undo information.

By default all these files are stored next to the file you are editing, this can pollute your source tree.

FILETYPE

We have to enable syntax highlighting and filetype detection, most distributions already enable this by default.

TABSTOP

Tabstop is going to define how the syntax indentation and tab key will behave.

LINENUMBERS

When we are programming knowing what line we are on

COLORSCHEME

And ofcourse we 'never' use the default colorscheme since we all think thatone is not good enough for us

MANY MORE CONFIGURATION POSSIBILITIES

The ones mentioned before are “most” important to do our job

DEMO CONFIGURATION

vim configuration in steps

PLUGINS WILL HELP US

What follows are a list of plugins I use, that does in no way mean those are the best. When you use a plugin make sure you feel an improvement in your workflow.

(C)TAGS

(C)tags is not really a plugin, it is a built in feature of Vim. This built in feature can already start making our lives much easier. There are plugins to generate your tagfiles on save, or on quit. I personally generate them whenever I see fit.

- Jump to tag under cursor: “ctrl+”
 - Go back to originating file: “ctrl+t”
- “:help tagsrch.txt”

PLUGIN MANAGER

In the old days you usually installed plugins by extracting a zipfile in your “~/vim” folder. Luckily, these days are over, now we have several plugins ;) that manage our plugins.

Some “plugin managers”:

- [pathogen](#) is very simple by concept and use
- [Vundle](#) is using a config file and installs plugins automatically
- [NeoBundle](#) is based on Vundle, but adds more features

THE NERD TREE

The NERD Tree is a tree like file explorer for vim

```
.. (up a dir)
<devel/ZendSkeletonApplication/
├─ config/
├─ data/
├─ module/
│   └─ Application/
│       ├── config/
│       ├── language/
│       └─ src/
│           └─ Application/
│               └─ Controller/
│                   └─ IndexController.php
├─ view/
│   └─ Module.php
├─ public/
│   ├── css/
│   ├── fonts/
│   ├── img/
│   ├── js/
│   └─ index.php
├─ vendor/
│   ├── composer.json
│   ├── composer.phar*
│   ├── init_autoloader.php
│   ├── LICENSE.txt
│   └─ README.md
```

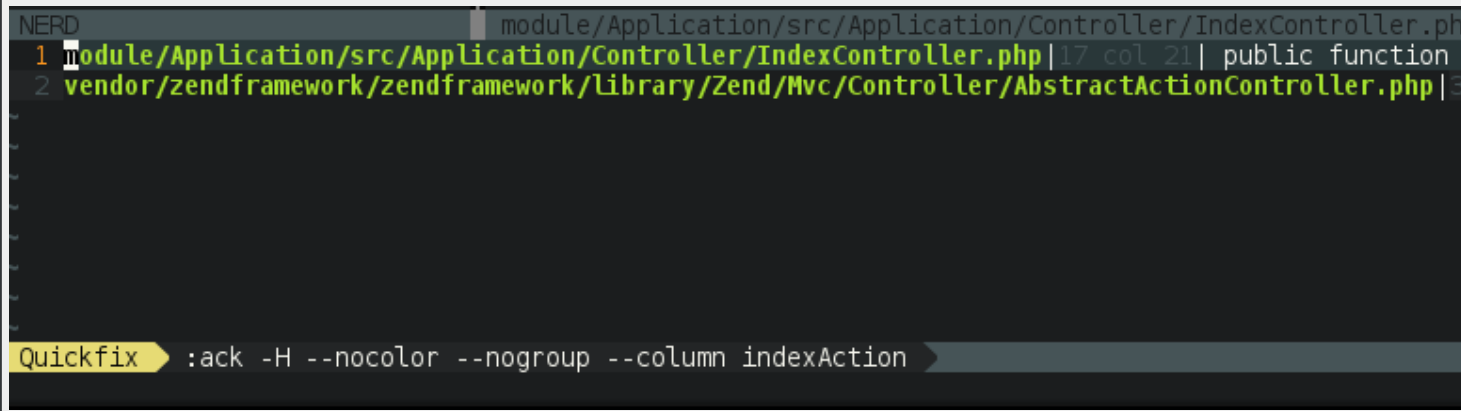

CTRLP.VIM

In short this is a fuzzy typed fast filefinder

```
NERD [Geen naam]
vendor/zendframework/zendframework/library/Zend/Barcode/Barcode.php
vendor/zendframework/zendframework/library/Zend/Cache/composer.json
vendor/zendframework/zendframework/library/Zend/Config/Factory.php
vendor/zendframework/zendframework/library/Zend/Captcha/Figlet.php
vendor/zendframework/zendframework/library/Zend/Server/Cache.php
vendor/zendframework/zendframework/library/Zend/Captcha/README.md
vendor/zendframework/zendframework/library/Zend/Captcha/Image.php
vendor/zendframework/zendframework/library/Zend/Cache/README.md
vendor/zendframework/zendframework/library/Zend/Crypt/Hmac.php
> vendor/zendframework/zendframework/library/Zend/Captcha/Dumb.php
mru files buf <->
>>> zendcac_
```

ACK.VIM

As you could guess the plugin uses the ack command to search in code files.



The screenshot shows a Vim editor window with a dark background. At the top, a tab labeled 'NERD' is visible. Below it, the file path 'module/Application/src/Application/Controller/IndexController.php' is shown. The editor displays two search results:

```
1 module/Application/src/Application/Controller/IndexController.php|17 col 21| public function  
2 vendor/zendframework/zendframework/library/Zend/Mvc/Controller/AbstractActionController.php|3
```

At the bottom, a 'Quickfix' window is open, showing the command used for the search:

```
:ack -H --nocolor --nogroup --column indexAction
```

NERDTREE + ACK.VIM

NERDTree Ack is an extension for nerdtree which adds a few menu items to search with ack

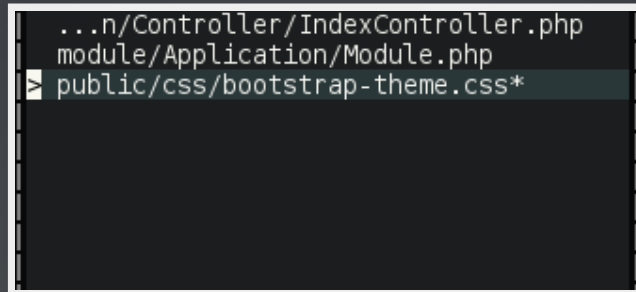
```
NERDTree Menu. Use j/k/enter and the shortcuts indicated
=====
> (a)dd a childnode
  (m)ove the current node
  (d)elele the current node
  (c)opy the current node
  search files, case s(e)nsitive
  (s)earch files, case insensitive
```

SAUCE FOR VIM

Sauce is a “project” like plugin, it creates an extra project specific vimrc file where you can keep the location of the project, maybe indenting options, ...

VIM BUFFERLIST

VIM bufferlist gives you a list of open buffers (files).

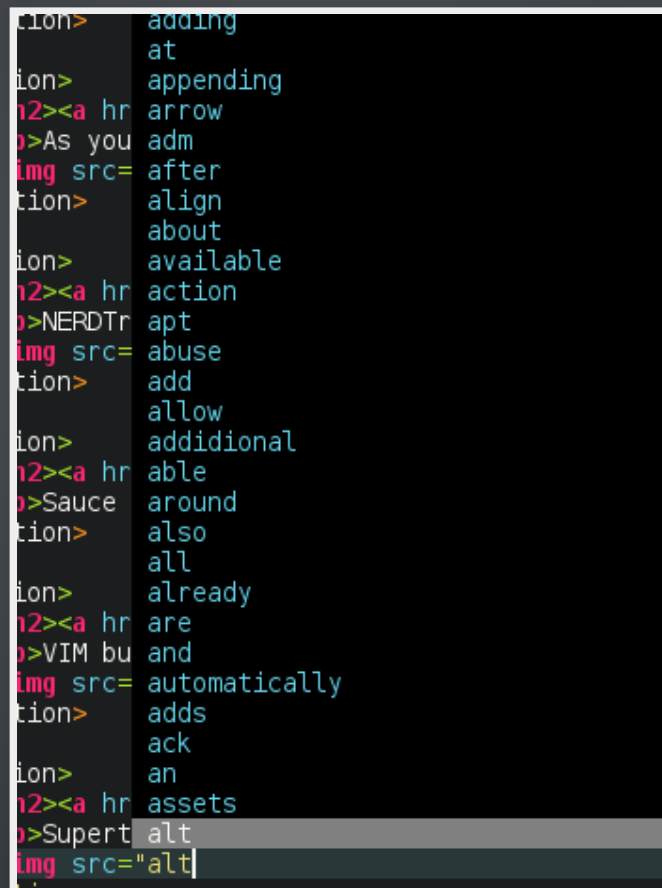


```
...n/Controller/IndexController.php  
module/Application/Module.php  
▸ public/css/bootstrap-theme.css*
```

The image shows a screenshot of the VIM bufferlist window. It displays a list of open buffers. The first two lines are truncated paths: "...n/Controller/IndexController.php" and "module/Application/Module.php". The third line is "public/css/bootstrap-theme.css*", which is highlighted with a dark background and a white cursor icon (a small square with a right-pointing arrow) to its left. The window has a thin white border and a dark background.

SUPERTAB

Supertab is a plugin that helps you to use completion in insert mode, when you press you get a list of possible completions.

A screenshot of a terminal window with a dark background and light-colored text. The terminal shows a list of completion suggestions for the word 'adding'. The suggestions are: adding, at, appending, arrow, adm, after, align, about, available, action, apt, abuse, add, allow, additional, able, around, also, all, already, are, and, automatically, adds, ack, an, assets, alt. The suggestions are listed in a column on the right side of the terminal. The left side of the terminal shows the text being typed: 'ion>', 'ion>', 'n2><a hr', 'o>As you', 'img src=', 'tion>', 'ion>', 'n2><a hr', 'o>NERDTr', 'img src=', 'tion>', 'ion>', 'n2><a hr', 'o>Sauce', 'tion>', 'ion>', 'n2><a hr', 'o>VIM bu', 'img src=', 'tion>', 'ion>', 'n2><a hr', 'o>Supert', 'img src="alt|'. The suggestions are listed in a column on the right side of the terminal. The left side of the terminal shows the text being typed: 'ion>', 'ion>', 'n2><a hr', 'o>As you', 'img src=', 'tion>', 'ion>', 'n2><a hr', 'o>NERDTr', 'img src=', 'tion>', 'ion>', 'n2><a hr', 'o>Sauce', 'tion>', 'ion>', 'n2><a hr', 'o>VIM bu', 'img src=', 'tion>', 'ion>', 'n2><a hr', 'o>Supert', 'img src="alt|'.

```
ion> adding
ion> at
ion> appending
n2><a hr arrow
o>As you adm
img src= after
tion> align
ion> about
ion> available
n2><a hr action
o>NERDTr apt
img src= abuse
tion> add
ion> allow
ion> additional
n2><a hr able
o>Sauce around
tion> also
ion> all
ion> already
n2><a hr are
o>VIM bu and
img src= automatically
tion> adds
ion> ack
ion> an
n2><a hr assets
o>Supert alt
img src="alt|
```

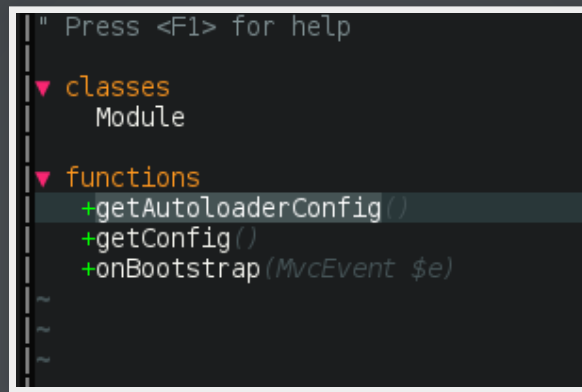
SYNTASTIC

Syntastic is a syntax checker plugin, it will run a syntax check on the open file.

```
module/Application/Module.php
1 module/Application/Module.php | 40 error | unexpected end of file, expecting function (T_FUNCTION)
~
~
~
~
~
~
~
~
~
Location : SyntasticCheck php (php)
```

TAGBAR

Tagbar displays a summary of your class or file. It lists classes, parameters and methods.



```
" Press <F1> for help
▼ classes
  Module
▼ functions
+getAutoloaderConfig()
+getConfig()
+onBootstrap(MvcEvent $e)
```


NERDCOMMENTER

NERDcommenter helps you to quickly add comments to a line or multiple lines.

PHP DOCUMENTOR FOR VIM

PHP Documentor for Vim is a plugin to easily create docblocks on your classes, parameters and methods.

PHP_GETSET

php_getset creates getter and setter methods from the parameters selected in your class.

```
1 <?php
2
3 class Foo {
4     protected $bar;
5 }
```

VIM-FUGITIVE

Fugitive is a git wrapper for vim, you can run practically every git command from fugitive.

```
| ac69ac08 (Maks3w      2012-07-13 00:47:10 +0200) | 3  * Zend Framework (http://framework.zend.com/)
| ac69ac08 (Maks3w      2012-07-13 00:47:10 +0200) | 4  *
| ac69ac08 (Maks3w      2012-07-13 00:47:10 +0200) | 5  * @link      http://github.com/zendframework/ZendSkeletonApplicati
| 66baf0d4 (Bryan Folliot 2013-12-29 15:47:53 +0100) | 6  * @copyright Copyright (c) 2005-2014 Zend Technologies USA Inc. (h
| ac69ac08 (Maks3w      2012-07-13 00:47:10 +0200) | 7  * @license   http://framework.zend.com/license/new-bsd New BSD Lic
| ac69ac08 (Maks3w      2012-07-13 00:47:10 +0200) | 8  */
| f8a26075 (Evan Coury    2011-09-29 20:18:37 -0700) | 9
| f8a26075 (Evan Coury    2011-09-29 20:18:37 -0700) | 10 namespace Application;
| f8a26075 (Evan Coury    2011-09-29 20:18:37 -0700) | 11
| 425c15d4 (Rob Allen     2012-07-02 13:27:39 +0100) | 12 use Zend\Mvc\ModuleRouteListener;
| 9616c475 (Pavel Galaton 2012-09-13 17:00:20 +0300) | 13 use Zend\Mvc\MvcEvent;
| 425c15d4 (Rob Allen     2012-07-02 13:27:39 +0100) | 14
| 19173779 (Matthew Weier O'Phinney 2012-05-16 11:45:29 -0500) | 15 class Module
| f8a26075 (Evan Coury    2011-09-29 20:18:37 -0700) | 16 {
| 9616c475 (Pavel Galaton 2012-09-13 17:00:20 +0300) | 17     public function onBootstrap(MvcEvent $e)
| 425c15d4 (Rob Allen     2012-07-02 13:27:39 +0100) | 18     {
| 425c15d4 (Rob Allen     2012-07-02 13:27:39 +0100) | 19         $eventManager      = $e->getApplication()->getEventManager();
| 425c15d4 (Rob Allen     2012-07-02 13:27:39 +0100) | 20         $moduleRouteListener = new ModuleRouteListener();
| 425c15d4 (Rob Allen     2012-07-02 13:27:39 +0100) | 21         $moduleRouteListener->attach($eventManager);
| 425c15d4 (Rob Allen     2012-07-02 13:27:39 +0100) | 22     }
| 425c15d4 (Rob Allen     2012-07-02 13:27:39 +0100) | 23
| a4621753 (Evan Coury    2011-12-07 14:22:38 -0700) | 24     public function getConfig()
| f8a26075 (Evan Coury    2011-09-29 20:18:37 -0700) | 25     {
| fb614285 (Evan Coury    2011-11-21 18:23:47 -0700) | 26         return include __DIR__ . '/config/module.config.php';
| f8a26075 (Evan Coury    2011-09-29 20:18:37 -0700) | 27     }
| 7ac195f6 (Evan Coury    2012-06-13 15:42:12 -0700) | 28
| 7ac195f6 (Evan Coury    2012-06-13 15:42:12 -0700) | 29     public function getAutoloaderConfig()
| 7ac195f6 (Evan Coury    2012-06-13 15:42:12 -0700) | 30     {
| 7ac195f6 (Evan Coury    2012-06-13 15:42:12 -0700) | 31         return array(
| 7ac195f6 (Evan Coury    2012-06-13 15:42:12 -0700) | 32             'Zend\Loader\StandardAutoloader' => array(
| 7ac195f6 (Evan Coury    2012-06-13 15:42:12 -0700) | 33                 'namespaces' => array(
```

VDEBUG

Vdebug is a debugger for Vim. Every debugger speaking DBGP protocol can be used. Xdebug speaks DBGP ;)

QUESTIONS ?

USEFULL RESOURCES

- <http://www.vim.org/>
- http://vim.wikia.com/wiki/Vim_Tips_Wiki
- <http://vimcasts.org/>
- my vimfiles: <https://github.com/BlackIkeEagle/vimfiles>

THANKS.

<https://joind.in/10940>

