
sublime-phpcs

This is a plugin for Sublime Text which provides checkstyle reports using the following tools (all optional):

- PHP_CodeSniffer ([phpcs](#))
- Linter ([php -l](#))
- PHP Mess Detector ([phpmd](#))

You can also configure the plugin to fix the issues using either

- PHP Coding Standards Fixer ([php-cs-fixer](#))
- PHP Code Beautifier ([phpcbf](#)) application

Requirements

Requirements for this plugin, should you want all the options to work:

- PHP_CodeSniffer 3.5+
- PHPMD 2.8+
- PHP CS Fixer 2.6+

This plugin has been tested on:

- Mac OS X 10.8.2
- Ubuntu 11.10
- Windows 7
- Sublime Text 2
- Sublime Text 3
- Sublime Text 4

It may work with other versions, but we cannot confirm that.

Installation

Use Sublime Text's Package Control (Tools -> Command Palette -> Package Control: Install Package -> Phpcs) to install this plugin. This is the recommended installation path.

Or

Simply checkout the git repo into “~/Library/Application Support/Sublime Text/Packages/” or the equivalent folder on Windows or Linux.

```
1 cd ~/Library/Application\ Support/Sublime\ Text/Packages/  
2 git clone git://github.com/benmatseby/sublime-phpcs.git Phpcs
```

In both cases, you may need to then configure the following with the actual path to the application:

- “phpcs_php_path”
- “phpcs_executable_path”
- “phpmd_executable_path”
- “php_cs_fixer_executable_path”

They are optional for the plugin. The path needs to include the application such as `/usr/local/bin/phpcs`.

In order to get the path of the application (On a Mac/Linux based environment), you can use:

```
1 which phpcs  
2 which phpmd  
3 which php-cs-fixer  
4 which phpcbf
```

Features

- Ability to run PHP_CodeSniffer
- Ability to run `php -l` on the open file
- Ability to run PHP Mess Detector on the open file
- Show cached results from PHP_CodeSniffer in open file
- Show errors in the Quick Panel
- Show errors in the Gutter
- Highlight the errors in the editor
- Show the error for a given line in the status bar
- Ability to specify the regular expression of the linter errors
- Ability to specify the location of the PHP_CodeSniffer application
- Ability to specify the location of the PHP Mess Detector application
- Ability to run the PHP Coding Standards Fixer tool which fixes most issues in your code when you want to follow the PHP coding standards as defined in the PSR-1 and PSR-2 documents
- Ability to run the PHP Code Beautifier tool which fixes most issues in your code when you want to follow PHP coding standards

Once you have right clicked on a file and selected “PHP CodeSniffer” > “Sniff this file...” you will get the output as shown below (depending on the settings you have defined):

Configuration

You can also define the configuration for the following settings, be it for a project, user settings or the default settings:

Plugin

- `show_debug` – Do you want the debug information to be sent to the console?
- `extensions_to_execute` – Which filetypes do you want the plugin to execute for?
- `extensions_to_blacklist` – Override the `extensions_to_execute` in case you have a sub extension such as `twig.php` etc.
- `phpcs_execute_on_save` – Do you want the code sniffer plugin to run on file save for php files?
- `phpcs_show_errors_on_save` – Do you want the errors to be displayed in `quick_panel` on save?
- `phpcs_show_gutter_marks` – Do you want the errors to be displayed in the gutter?
- `phpcs_outline_for_errors` – Do you want the errors to be highlighted in the editor?
- `phpcs_show_errors_in_status` – Do you want the errors to be displayed in status bar when clicking on the line with error?
- `phpcs_show_quick_panel` – Do you want the errors to be displayed in the quick panel?
- `phpcs_php_prefix_path` – Needed on windows for phar based applications. Also if you cannot make phar executable. Avoid if possible
- `phpcs_commands_to_php_prefix` – List of commands you want the php path to prefix. This would be useful, if you have some commands as a phar that cannot be run without the php prefix, and others using native command.
- `phpcs_icon_scope_color` – What colour to stylise the icon. This needs knowledge of theming of Sublime Text, as it uses scope colours from the theme to “tint” the dot icon. See [here](#)

PHP_CodeSniffer

- `phpcs_sniffer_run` – Do you want the PHPCS checker to run?
- `phpcs_command_on_save` – Do you want the command to execute on save?
- `phpcs_executable_path` – The path to the phpcs executable. If empty string, use `PATH` to find it
- `phpcs_additional_args` – This is the extra information you want to pass to the phpcs command. For example which “standard” you want to run, and if you want to show warnings or not

PHP CodeSniffer Fixer

- `php_cs_fixer_on_save` – Do you want to run the fixer on file save?
- `php_cs_fixer_show_quick_panel` – Do you want the quick panel to display on execution?
- `php_cs_fixer_executable_path` – The path to the php-cs-fixer application.
- `php_cs_fixer_additional_args` – This is the extra information you want to pass to the php-cs-fixer command. For example which “fixers” you want to run

PHP Code Beautifier

- `phpcbf_on_save` – Do you want to run the fixer on file save?
- `phpcbf_show_quick_panel` – Do you want the quick panel to display on execution?
- `phpcbf_executable_path` – The path to the phpcbf application.
- `phpcbf_additional_args` – This is the extra information you want to pass to the phpcbf command. For example which “standard” to use in order to fix the issues

PHP Linter

- `phpcs_linter_run` – Do you want the PHP linter to run?
- `phpcs_linter_command_on_save` – Do you want the command to execute on save?
- `phpcs_php_path` – The path to the PHP executable. If empty string, use PATH to find it
- `phpcs_linter_regex` – The regex for the PHP linter output

PHP Mess Detector

- `phpmd_run` – Do you want the PHPMD to run? Off by default
- `phpmd_command_on_save` – Do you want the command to execute on save?
- `phpmd_executable_path` – The path to the phpmd executable. If empty string, use PATH to find it
- `phpmd_additional_args` – This is the extra information you want to pass to the phpmd command. For example which “rulesets” you want to run

Project Based Settings

Your .project file should look something like this:

```
1 {
2   "folders": [{}],
3   "settings": {
4     "phpcs": {
5       "phpcs_additional_args": {
6         "--standard": "/path/to/.composer/vendor/drupal/coder/
          coder_sniffer/Drupal"
7       }
8     }
9   }
10 }
```

Of course this is a example to apply Drupal code sniffer. This could be anything. Whatever you can have on this package settings it can be overwritten under the settings -> phpcs

FAQ

What do I do when I get “OSError: [Errno 8] Exec format error”?

- This seems to be an issue you may get with regards to wrapper scripts.
- Please make sure that the application/script you are referencing has the correct shebang line, as per GH-79

What do I do when I get “OSError: [Error 2] No such file or directory”?

- Well, first of all you need to check that you have PHP_CodeSniffer, and if being used, the phpmmd application.
- If you have these applications installed, then it sounds like those applications are not in your PATH, or cannot be found in your PATH by the Python runtime, so configure “phpcs_php_path”, “phpcs_executable_path”, “phpmd_executable_path” and “php_cs_fixer_executable_path” with the actual paths to those applications

What do I do when I get “OSError: [Errno 13] Permission denied”?

- It sounds like your path settings are incorrect.
- You need to make sure that when you specify the path you include the entire path including the application

```
1 $ which phpcs
2 /usr/local/bin/phpcs
```

-
- That entire output is the path you need in your configs.

What if I've installed the applications using Homebrew?

If you have installed php-cs-fixer, phpmid or phpcs via homebrew then please make sure that you define the “`_executable_path`” option to the .phar application and not the wrapper script that is placed in your bin folder, as this will cause odd behaviour.

What other Key Bindings can I setup?

The following is a list of commands that you can bind to a keyboard shortcut:

- `phpcs_fix_this_file`
- `phpcs_clear_sniffer_marks`
- `phpcs_goto_next_error`
- `phpcs_show_previous_errors`
- `phpcs_sniff_this_file`

In order to achieve this you need to add the following to one of your key bindings settings file:

```
1 { "keys": ["ctrl+super+t"], "command": "phpcs_clear_sniffer_marks" }
```

To decide which “Fixer” to use, you can do:

```
1 { "keys": ["super+k", "super+f"], "command": "phpcs_fix_this_file", "
  args": {"tool": "CodeBeautifier"}},
```

or

```
1 { "keys": ["super+k", "super+f"], "command": "phpcs_fix_this_file", "
  args": {"tool": "Fixer"}},
```

You can then change the ctrl+super+t combination to something of your choosing.