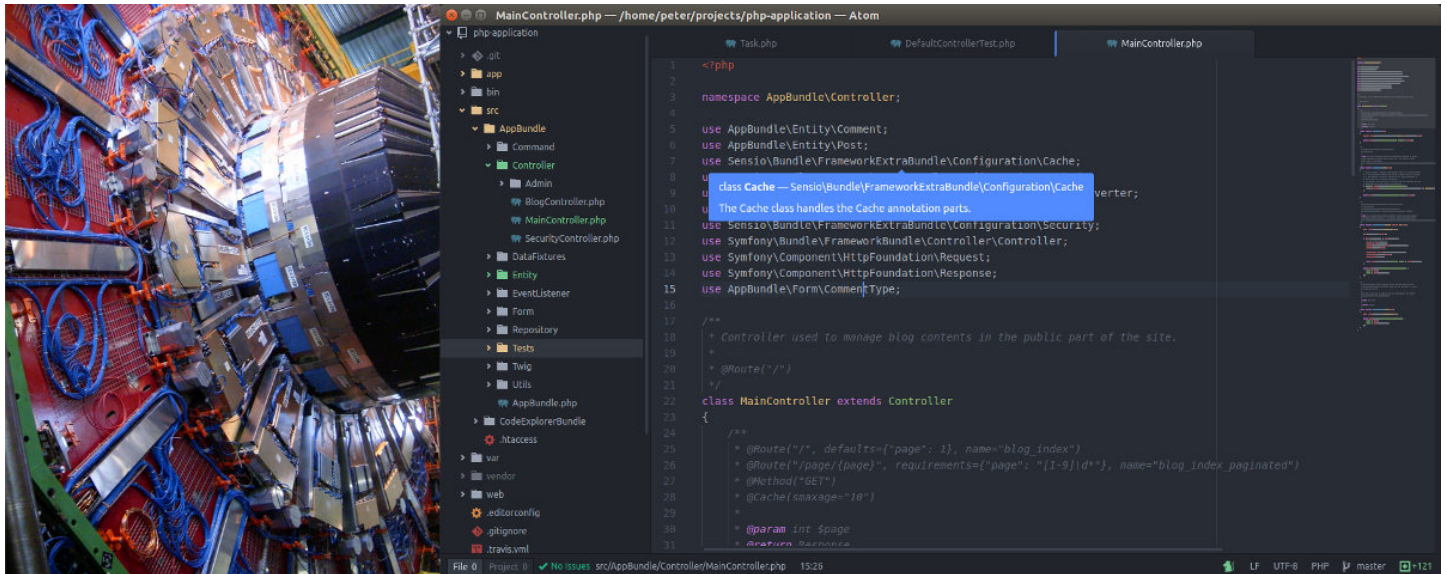




Atom editor for PHP developers



Atom is an advanced open source text editor for developers. This article will explain some useful packages and tricks how to make the most out of it when developing with PHP.

IDE or editor?

First, a bit of an introduction and comparison between IDE (integrated development environment) and editor. IDE provides more functionality out-of-the-box compared to an editor. So why would you want to use an editor in the first place?

Editor usually requires less computer resources and can work faster compared to an IDE. Also the learning curve and time to get used to an editor is less steep. And on top of that editor can still be customized with additional packages to behave a lot like an IDE.

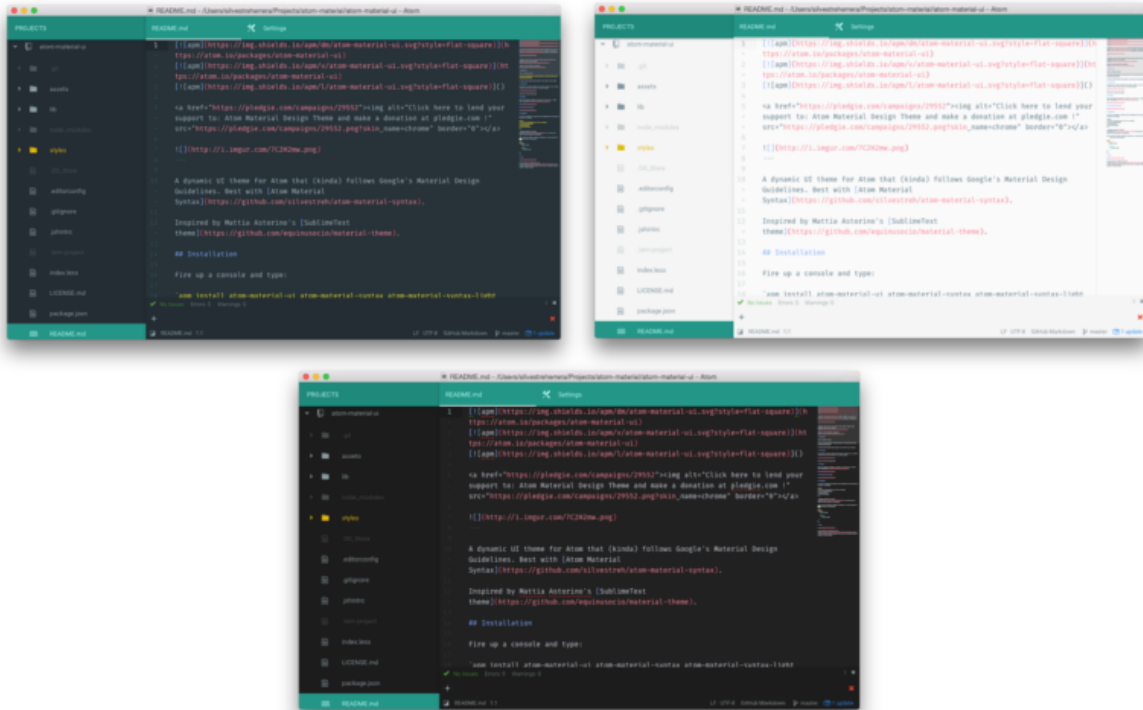
One of the recommended ways to develop applications is for example, to use an IDE for developing the project at large and an editor for quick edits across different projects or folders. However you can successfully develop PHP projects also without an IDE. Explore and try different tools to see what suits your development workflow. Make sure that the tools you use, make you productive.

Atom installation

Installation of Atom is as simple as it gets and works on all widely used operating systems (Linux, Windows and macOS). Visit the [Atom homepage](#) and the releases on [GitHub](#) to download the installation for your system.

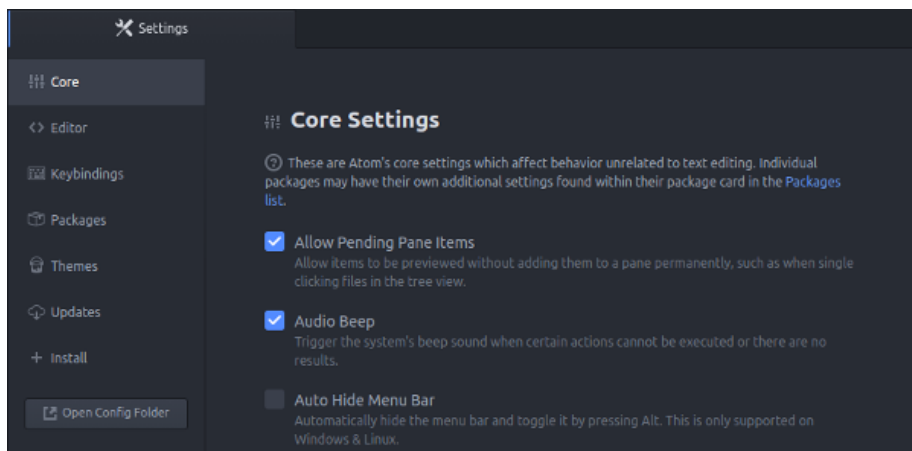
Themes

By default, Atom comes with multiple nice themes you can use out of the box. Additional themes can be obtained from [Atom themes repository](#).



Configuration

Nice thing about Atom is that it can be customized to do almost anything. Configurations can be set via the menu [Edit > Preferences](#).



The configuration files are saved in your home directory `~/ .atom`, so you can backup them to your [dotfiles](#) and have a portable configured development environment for multiple development machines.

Some of the portable files in that folder are `config.cson`, `keymap.cson`, `snippets.cson`, and `styles.less`. The installed packages can also be exported to a custom list file via command line:

```
apm list --installed --bare > packages.list
```

When you will need to reinstall Atom, the saved packages list can be used to quickly install all your favorite packages:

```
apm install --packages-file packages.list
```

Code style

By default, Atom already supports syntax highlighting and snippets for a lot of languages. For PHP it uses the [language-php](#) package. To set your preferred coding style such as spaces/tabs, line ending, and similar go to **Edit > Preferences > Packages** and type **language-php** and set some default behaviors you use. In the following section we'll check some packages that can improve and extend managing code style in your PHP project.

Packages

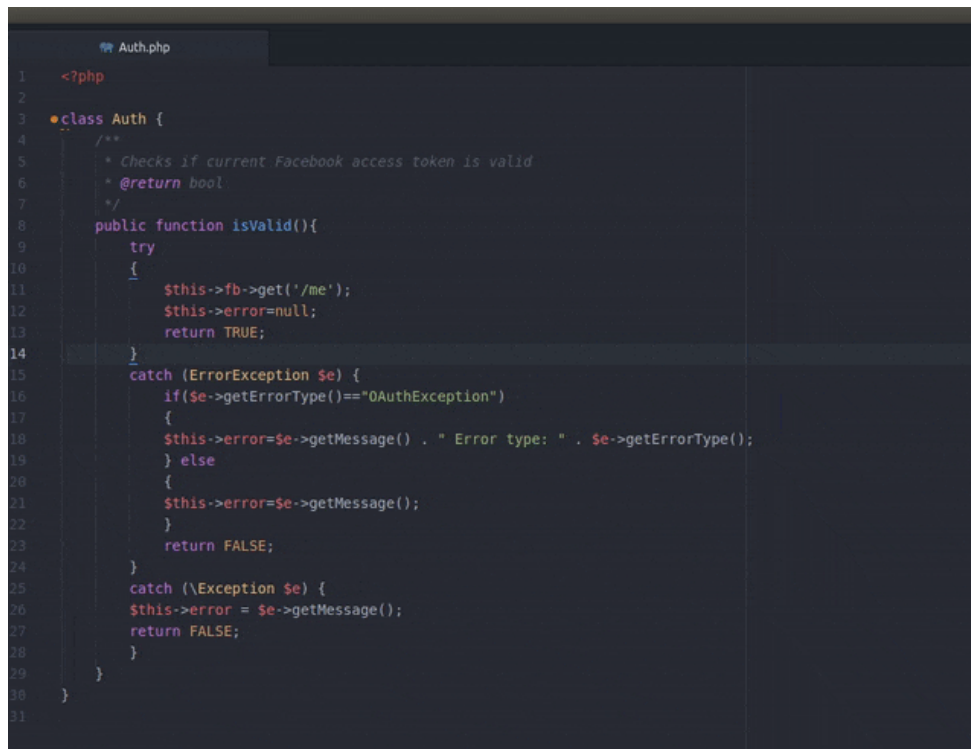
Atom packages extend core editor functionality. They can be obtained from the [Atom packages repository](#).

Packages can be installed via the GUI **Edit > Preferences > Install** or with the command line Atom package manager **apm**:

```
apm install package-name
```

By default, Atom provides basic editing capabilities and everything you need to write PHP code. There are many packages to extend and make writing PHP code more efficient and provide a much better development experience. If you are used to some advanced IDE functionality already, you're familiar with things such as autocompletion, code inspection, generating code snippets and more. There are a lot of Atom packages for PHP development, however you might want to take a look at some of the following highlighted packages for having PHP development more efficient.

Atom beautify



```
1  <?php
2
3  class Auth {
4      /**
5       * Checks if current Facebook access token is valid
6       * @return bool
7       */
8      public function isValid(){
9          try
10         {
11             $this->fb->get('/me');
12             $this->error=null;
13             return TRUE;
14         }
15         catch (ErrorException $e) {
16             if($e->getErrorType()=="OAuthException")
17             {
18                 $this->error=$e->getMessage() . " Error type: " . $e->getErrorType();
19             } else
20             {
21                 $this->error=$e->getMessage();
22             }
23             return FALSE;
24         }
25         catch (\Exception $e) {
26             $this->error = $e->getMessage();
27             return FALSE;
28         }
29     }
30 }
31
```

Did you get a code that is not suited for your coding style? The [atom-beautify](#) package is a must check for having a consistent code style and to beautify the code on the fly. It cleans the code for multiple languages according to the predefined code style settings. To successfully beautify PHP code, you must also separately install either the [php-cs-fixer](#) or the [PHP Code Sniffer](#).

Editorconfig

To have a consistent code style across different editors for multiple people, you might want to take a look at [Editorconfig](#) initiative. The [editorconfig](#) package adds support for **.editorconfig** file to Atom.

PHP Integrator

PHP Integrator is a collection of multiple Atom packages to provide a better PHP development experience. To install and successfully use PHP Integrator packages you will first need to install some prerequisites:

PHP

php-sqlite extension

php-mbstring

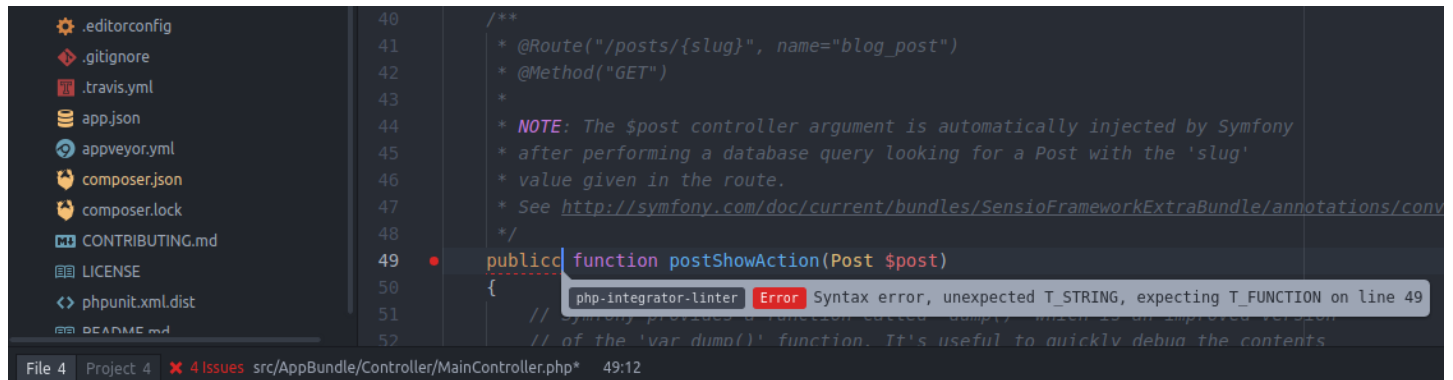
openssl

project-manager package

php-integrator-base package

After base PHP Integrator installation save your PHP project in Project Manager ([Packages > Project Manager > Save Project](#)) and index it with PHP integrator ([Packages > PHP Integrator > Set Up Current Project](#)) which will index project files for further usage.

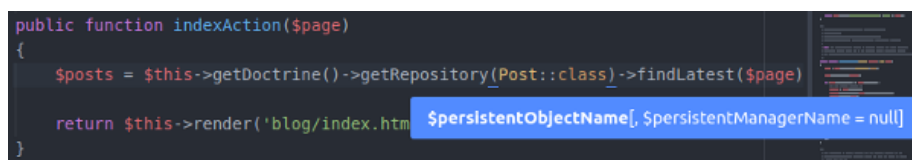
PHP Integrator and Linter inspect and validate your PHP code for errors, check [PHP DocBlocks](#) and more.



PHP Integrator shows tooltips (e.g. for methods and classes) in your PHP source code.



PHP Integrator includes call tips with parameter information in your PHP source code for functions and methods.



With PHP Integrator installed and project prepared, you can check the following additional packages from PHP Integrator collection:

PHP Integrator Autocomplete Plus



```

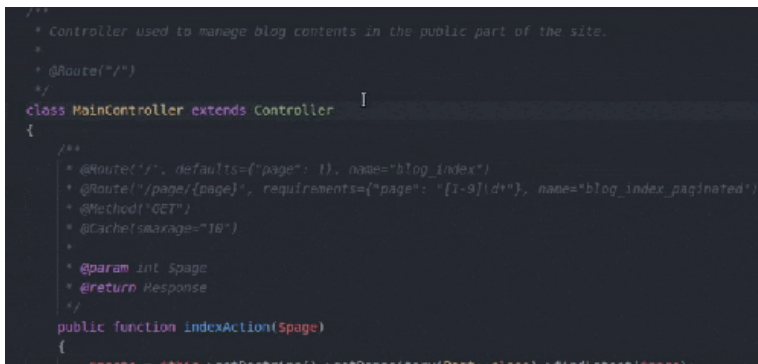
/**
 * Controller used to manage blog contents in the public part of the site.
 *
 * @Route("/")
 */
class MainController extends Controller
{
    /**
     * controller_func()
     * controller_function($foo, $foobar)
     * @Route("/")
     * @Route("/")
     * @Method("GET")
     * @Cache(smaxage="10")
     * @param int $page
     * @return Response
     */
    public function indexAction($page)
    {
        $posts = ...;

        return $this->render('blog/index.html.twig', ['posts' => $posts]);
    }
}

```

The [php-integrator-autocomplete-plus](#) package provides autocompletion for your PHP source code. By indexing project files, autocompletion works also for classes from project and Composer's vendor folder. Class is automatically added to the list of used classes on top of a current PHP file.

PHP Integrator Navigation



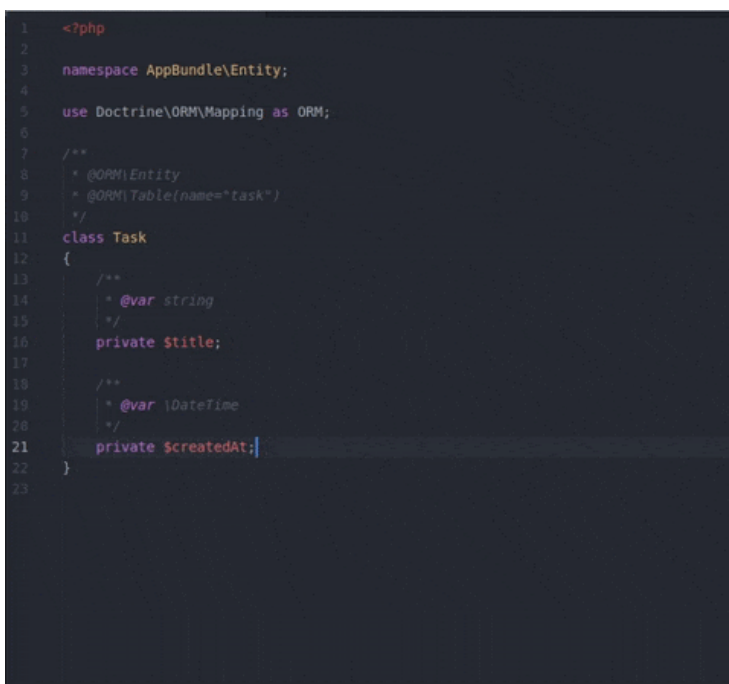
```

/**
 * Controller used to manage blog contents in the public part of the site.
 *
 * @Route("/")
 */
class MainController extends Controller
{
    /**
     * @Route("/", defaults={"page": 1}, name="blog_index")
     * @Route("/page/{page}", requirements={"page": "[1-9]+"}, name="blog_index_paginated")
     * @Method("GET")
     * @Cache(smaxage="10")
     *
     * @param int $page
     * @return Response
     */
    public function indexAction($page)
    {
        ...
    }
}

```

The [php-integrator-navigation](#) package provides code navigation and go to functionality for your PHP source code. For example, clicking a class opens the source code file. Opening can be done by adjusting the settings (**Alt** + left mouse click, **Shift** + left mouse click or similar).

PHP Integrator Refactoring



```

1  <?php
2
3  namespace AppBundle\Entity;
4
5  use Doctrine\ORM\Mapping as ORM;
6
7  /**
8   * @ORM\Entity
9   * @ORM\Table(name="task")
10  */
11  class Task
12  {
13      /**
14       * @var string
15       */
16      private $title;
17
18      /**
19       * @var \DateTime
20       */
21      private $createdAt;
22  }
23

```

The [php-integrator-refactoring](#) package provides refactoring capabilities for your PHP source code. It generates getters and setters for PHP classes and similar. Keybinding: **Alt** + **Enter**.

PHP Integrator Annotations



The [php-integrator-annotations](#) package shows which methods override their interface implementations.

PHP Debug

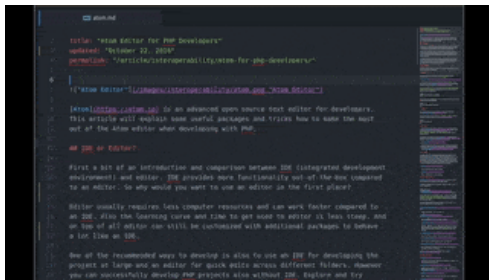
The [php-debug](#) package is PHP code debugging package using the Xdebug extension.

PHP Server

The [php-server](#) package runs PHP development server on your localhost.

Above we've gone through some of the useful packages for PHP. To improve development experience and add additional functionality you might want to check these awesome packages.

Minimap



The [minimap](#) package adds file preview on the sidebar. Once you get used to it, it is a must have package in your collection. It is useful for easier overview of the source code and quicker navigation over the file.

Highlight Selected

The [highlight-selected](#) package highlights the current word selected when double clicking. If you're using the minimap package there is also [minimap-highlight-selected](#).

Atom Minify

The [atom-minify](#) package minifies JavaScript and CSS files. It is useful when you come across a need to manually quickly minify a required file on the fly.

Todo Show

The [todo-show](#) finds all `TODO`, `FIXME`, `CHANGED` and similar comments in your project and shows them in a nice overview list.

Color Picker and Pigments

The [color-picker](#) package opens color picker for color codes such as HEX by using right click or pressing `CMD-SHIFT-C` / `CTRL-ALT-C`. [Pigments](#) displays colors for color codes in the editor itself.

Ctrl-dir-scroll

The [ctrl-dir-scroll](#) package is convenient for users who are used to shortcuts from some other IDEs and editors. By default, Atom uses [Ctrl](#) and up/down keys to move the line up and down. This package adds scrolling up and down with keyboard - [CTRL](#) and up/down keys.

Project Manager

The [project-manager](#) package helps you manage multiple projects.

Emmet

[Emmet](#) adds support for [Emmet](#).

Docblockr

[Docblockr](#) is a helper package for writing documentation.

File Icons

The [file-icons](#) package adds icons in tree view for recognized file types.

Framework specific packages

PHP frameworks are supported by multiple packages which provide additional functionality such as code snippets.

[CodeIgniter](#)

[Drupal](#)

[Laravel](#)

[Symfony](#)

[WordPress](#)

Docker

Developing with different PHP installations, extensions and dependencies soon requires virtualization and/or containerization such as Docker. Docker can be optionally integrated in Atom with additional plugins:

[language-docker](#)

Adds Dockerfile syntax highlighting.

[docker](#)

Integrates Docker with Atom editor.

Shortcuts and useful features

Keyboard shortcuts can make you more productive. These are some of the keyboard shortcuts you might find useful. On Linux and Windows use the [Ctrl](#) key, and on macOS the [Cmd](#).

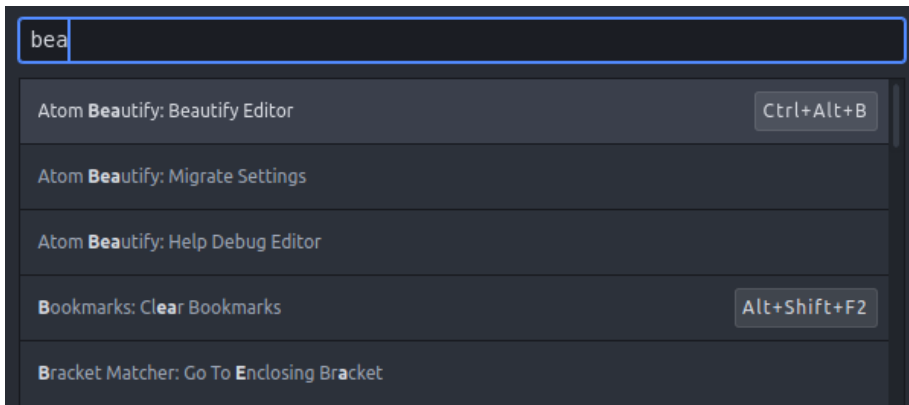
Comments: [Ctrl](#) + [/](#)

Comments/uncomments selected code.

Find and open a file: [Ctrl](#) + [p](#)

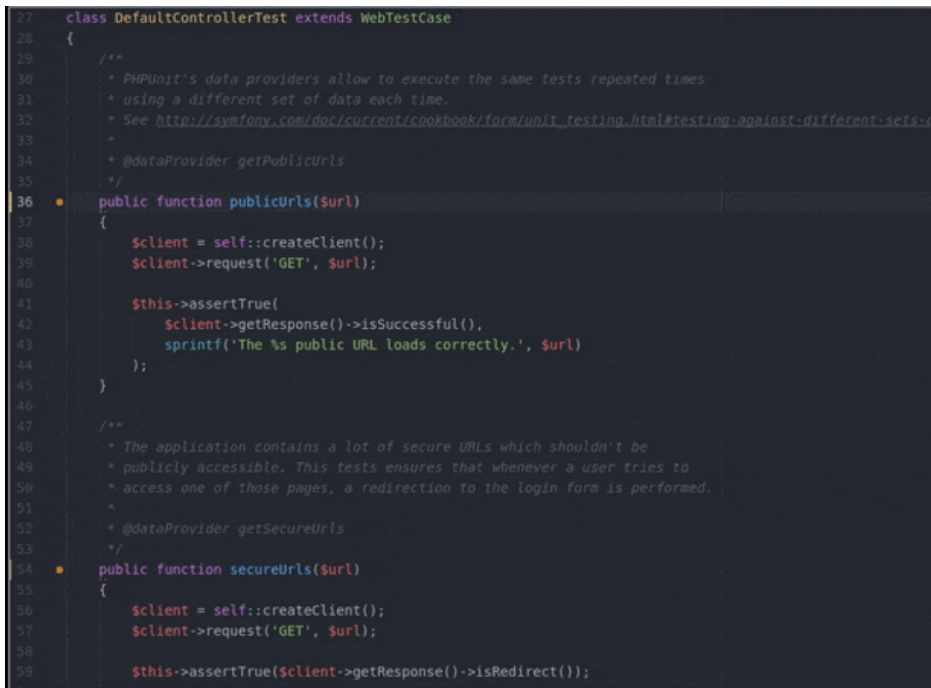
Command Palette: [Shift](#) + [Ctrl](#) + [p](#)

Command palette simplifies access to find and run available commands instead of going through the menu manually.



Multiple cursors:

With `Ctrl` and left mouse click you can add more cursors to the editor which behave the same as a single cursor. This is useful for changing multiple same values on different parts of the file.



Caveats and final thoughts

Performance

Having installed and enabled a lot of packages might also slightly slow down the performance. When opening very large files (few MB), the editor performance can decrease and the editor becomes unresponsive. Currently Atom supports files up to few MB (~10MB), and works a bit better on Linux and macOS systems. This has been already noted on [Atom issue tracker](#), so it might be fixed in the future.

How to deal with the issue of very large files (for example, log files)?

Many GUI editors and IDEs have same issues when opening and working on large files. People usually solve this with the command line tools like `grep`, `cat`, `head`, `tail`, and `sed`. For compressed files there are also `zless`, `zmore`, `zcat`, and `zgrep`.

Learning curve

At first, Atom might seem a bit overwhelming, specially to fine tune it for your needs. So, as with every tool, take some extra time and read the [documentation](#) to get to know it in more details.

Multiple community packages can have the same keybindings and in this case need to be reconfigured manually.

Which packages to install and which not is your choice and preference. For PHP development in particular, many tasks can be done by using native tools in the command line instead of using editor plugins. For example, unit testing, running Docker containers, deployment, running up local development server and similar.

Atom truly is a magnificent editor worth taking a look. It has a huge community behind and it is open sourced.

See also

Some additional useful resources:

[Atom Flight Manual](#) - Official book about Atom.

[Atom in Orbit](#) - Atom in the browser.

[Awesome Atom](#) - A curated list of delightful Atom packages and resources.

[Nuclide](#) - A package built on top of Atom for React Native, Hack and Flow projects.

About PHP.earth

[Get Involved](#)
[Internal FAQ](#)
[Status](#)

Documentation

[Index](#)
[FAQ](#)
[<?php tips](#)
[Linux repositories](#)

Community

[Facebook Group](#)
[GitHub](#)

Powered by

