

AN OPENSOURCE EBOOK

Laravel



Tips and Tricks

Bobby Iliev

Table of Contents

About the book	14
About the author	15
Sponsors	16
Ebook PDF Generation Tool	18
Book Cover	19
License	20
How to Install Laravel on DigitalOcean with 1-Click?	21
Installing Laravel on DigitalOcean	22
Using the Laravel DigitalOcean 1-Click	23
Completing the Laravel configuration	25
Video Demo	27
Conclusion	28
How to get a free domain name for your Laravel project	29
Choosing a free domain name	30
Adding your Domain to DigitalOcean	33
Updating your Nameservers	34
Conclusion	36
8 Awesome VS Code Extensions for Laravel Developers	37
1. Laravel Blade Snippets	38
2. Laravel Snippets	40
3. Laravel Blade Spacer	42
4. Laravel Artisan	43
5. Laravel Extra Intellisense	45

6. Laravel goto Controller	47
7. Laravel goto View	48
8. DotENV syntax highlighting	49
Book recommendation	52
Conclusion	53
 What is Laravel Jetstream and how to get started	54
What is Laravel Jetstream	55
Installing Laravel Jetstream	56
Authentication	58
Profile management	60
Jetstream Security	62
Jetstream API	64
Jetstream Teams	65
Conclusion	66
 How to check Laravel Blade View Syntax using artisan	67
Installation	68
Testing	69
Conclusion	70
 How to speed up your Laravel application with PHP OPcache ..	71
Enable PHP OPcache	72
Configure PHP OPcache	74
Configure Laravel OPCache	76
Conclusion	78
 What is Laravel Sail and how to get started	79
Installing And Setting Up Laravel Sail	81

Laravel Sail Commands	83
Video Introduction	86
Conclusion	87
How to add simple search to your Laravel blog/website	88
Controller changes	89
Route changes	90
Blade view changes	91
Conclusion	92
How to Create Custom Laravel Maintenance Page	93
Enable Default Maintenance Mode	94
Create Custom Maintenance Page	96
Disable Maintenance Mode	98
Conclusion	99
What is Laravel Blade UI Kit and how to get started	100
Installation	101
Configuration	102
Example	105
Conclusion	107
How to Add a Simple Infinite Scroll Pagination to Laravel	108
Configuring your Controller	110
Configuring your Blade View	111
Adding and configuring jScroll	114
Conclusion	116
How to add a simple RSS feed to Laravel without using a package	117

Configuring RSS Controller	119
Configuring your Blade view	121
Configuring your Route	123
Conclusion	124
What is Laravel Zero and how to get started	125
Installation	126
Commands	129
Configuration	131
Addons	133
Building the application	136
Conclusion	137
How to build a blog with Laravel and Wink	138
Installation	139
Creating a Controller	142
Creating Views	144
Adding routes	148
Conclusion	149
How to copy or move records from one table to another in Laravel	150
Copy all records from one table to another	151
Move all records from one table to another	153
Conclusion	154
How to generate title slugs in Laravel	155
Creating a slug in Laravel	156
Conclusion	157

What is Laravel Enlightn and how to use it	158
Installation	160
Configuration	161
Usage	162
Analyzing the output	164
Conclusion	166
How to consume an external API with Laravel and Guzzle ...	167
Creating a new table	168
Create the Model	170
Create the Controller	171
QuizAPI overview	172
Building the method	174
Add the route	177
Conclusion	178
How to send Discord notifications with Laravel	179
Creating a Discord Channel and Webhook	181
Installing the http-client	184
Adding the Controller	185
Adding the Routes	188
Conclusion	189
How to encrypt and decrypt a string in Laravel	190
App Key Configuration	192
Creating a Route	193
Creating a Controller	194
Conclusion	197

How to remove a package from Laravel using composer ..	198
Adding a package to Laravel	199
Removing a package from Laravel using composer	200
Conclusion	201
What is Laravel Breeze and how to get started ..	202
Overview	204
Installation	206
File structure	208
Video Overview	210
Conclusion	211
What are signed routes in Laravel and how to use them ..	212
Defining a signed route	214
Generating the signature	215
Using the signed route	216
Conclusion	218
How to Quickly Change the Password for a User in Laravel ..	219
Reset password with tinker	220
Reset password via a route	223
Conclusion	224
How to convert markdown to HTML in Laravel and Voyager ..	225
Install the Laravel Markdown Package	227
Configure your Controller	228
Change the Input type in Voyager	230

Conclusion	231
How to Create Response Macros in Laravel	232
Creating a service provider	233
Creating the Response Macro	234
Using the Response Macro	235
Conclusion	236
How to Get the Base URL in Laravel	237
Access the Laravel Base URL	239
Access the current URL in Laravel	240
Named routes	241
Access Assets URLs	242
Conclusion	243
How to limit the length of a string in Laravel	244
Limit string length in Blade	245
Limit string length in Model	246
Limit string length in Controller	247
Conclusion	248
How to check 'if not null' with Laravel Eloquent	249
Check if null: whereNull	250
Check if not null: whereNotNull	251
Conclusion	252
How to get the current date and time in Laravel	253
Using Carbon to get the current date and time	254
Using the now() helper function	257
Conclusion	258

How to Get Current Route Name in Laravel ..	259
Get route name in Controller or Middleware	260
Get route name in Blade View	262
Getting additional information about your route	263
Conclusion	265
How to Count and Detect Empty Laravel Eloquent Collections ..	
266	
Check if a collection is empty	268
Check if a collection is not empty	269
Counting the records of a collection	270
Conclusion	271
How to use Forelse loop in Laravel Blade ..	272
Check if not empty then	273
Forelse example	274
Conclusion	275
How to Delete All Entries in a Table Using Laravel Eloquent ..	
276	
Using truncate to delete all entries from a table	278
Delete all entries using the delete() method	279
Conclusion	280
How to check if a record exists with Laravel Eloquent ..	281
Check if a record exists	282
Create record if it does not exist already	283
Conclusion	285

How to Add Multiple Where Clauses Using Laravel Eloquent ..	
286	
Chaining multiple where clauses together	287
Passning an array to the where method	288
Using orWhere Clauses	289
Conclusion	290
How to Add a New Column to an Existing Table in a Laravel Migration ..	291
Creating a new table with a migration	292
Add a New Column to an Existing Table	295
Conclusion	298
How to Rollback Database Migrations in Laravel ..	299
Rollback the Last Database Migration	300
Rollback Specific Migration	301
Rollback All Migrations	302
Conclusion	303
How to Remove a Migration in Laravel ..	304
Creating a Laravel migration	305
Remove a Migration in Laravel	306
Conclusion	308
How to create a contact form with Laravel Livewire ..	309
Installing Livewire	310

Adding new Livewire component	311
Creating your Blade view	312
Prepare your Livewire view	314
Prepare your Livewire Logic	317
Add routes	320
Adding SMTP details	321
Conclusion	322
 How To Display HTML Tags In Blade With Laravel	323
Display HTML In Laravel Blade Views	324
Conclusion	326
 How to Set a Variable in Laravel Blade Template	327
Returning a variable from a controller	328
Setting variables in Laravel Blade Template	329
Conclusion	330
 How to limit the result with Laravel Eloquent	331
Limiting the result with pure SQL	333
The limit() method	334
The take() method	335
The paginate() method	336
Conclusion	337
 How to Select Specific Columns in Laravel Eloquent	338
Select specific columns with SQL only	339
Select specific columns with Laravel Eloquent	340
Conclusion	341
 How to get the Laravel Query Builder to Output the Raw SQL	

Query	342
The toSql() method	343
Laravel Debugbar	345
Conclusion	346
How to Get the Last Inserted Id Using Laravel Eloquent	347
Last Inserted Id Using Laravel Eloquent	348
Conclusion	350
How to Order the Results of all() in Laravel Eloquent	351
Ordering the results on a query level	353
Ordering the results on a collection level with all()	355
Conclusion	357
How to fix Laravel Unknown Column 'updated_at'	358
Disable the timestamp columns in your Model	360
Change the name of the timestamp tables	361
Conclusion	362
How to Define Custom ENV Variables in Laravel	363
Defining a variable in .env	364
Accessing the env variable with env()	365
Accessing the env variable with config()	366
Conclusion	368
How to fix 'Please provide a valid cache path' error in Laravel	369
Creating the cache folders	370
Clearing the cache	371
Conclusion	372

How to check your exact Laravel version	373
Check your Laravel version with artisan	374
Check your Laravel version via your text editor	375
Conclusion	376
Contact Form with Voyager and Laravel	377
Steps	378
Configuration	379
Conclusion	386
Scaling Laravel App with Multiple Databases	387
Prerequisites	388
Steps	389
1. Deploy 3 servers on Digital Ocean	390
2. Install and Configure our Web server	391
Install and configure our database servers along with our MySQL replication	396
Configure Laravel to work with the Master-Slave MySQL setup	399
Install the Voyager admin panel on the new setup	401
Conclusion	403
Conclusion	405
Other eBooks	406

About the book

- **This version was published on July 25 2021**

This is an open-source [Laravel Tips and Tricks eBook](#) that is a collection of my own notes that I've put together for myself throughout the years. You would likely than not need many of those tips at some point in your career as a Laravel Developer.

The guide is suitable for anyone working as a Laravel developer and would love to learn some random Laravel tips and tricks. You can read the chapters in this book in a random order as they are completely separate tips or tricks.

About the author

My name is Bobby Iliev, and I have been working as a Linux DevOps Engineer since 2014. I am an avid Linux lover and supporter of the open-source movement philosophy. I am always doing that which I cannot do in order that I may learn how to do it, and I believe in sharing knowledge.

I think it's essential always to keep professional and surround yourself with good people, work hard, and be nice to everyone. You have to perform at a consistently higher level than others. That's the mark of a true professional.

For more information, please visit my blog at <https://bobbyiliev.com>, follow me on Twitter [@bobbyiliev_](#) and [YouTube](#).

Sponsors

This book is made possible thanks to these fantastic companies!

DigitalOcean

DigitalOcean is a cloud services platform delivering the simplicity developers love and businesses trust to run production applications at scale.

It provides highly available, secure, and scalable compute, storage, and networking solutions that help developers build great software faster.

Founded in 2012 with offices in New York and Cambridge, MA, DigitalOcean offers transparent and affordable pricing, an elegant user interface, and one of the largest libraries of open source resources available.

For more information, please visit <https://www.digitalocean.com> or follow [@digitalocean](#) on Twitter.

If you are new to DigitalOcean, you can get a free \$100 credit and spin up your own servers via this referral link here:

[Free \\$100 Credit For DigitalOcean](#)

DevDojo

The DevDojo is a resource to learn all things web development and web design. Learn on your lunch break or wake up and enjoy a cup of coffee with us to learn something new.

Join this developer community, and we can all learn together, build together, and grow together.

[Join DevDojo](#)

For more information, please visit <https://www.devdojo.com> or follow [@thedevdojo](#) on Twitter.

Ebook PDF Generation Tool

This ebook was generated by [Ibis](#) developed by [Mohamed Said](#).

Ibis is a PHP tool that helps you write eBooks in markdown.

Book Cover

The cover for this ebook was created with [Canva.com](https://www.canva.com).

If you ever need to create a graphic, poster, invitation, logo, presentation – or anything that looks good — give Canva a go.

License

MIT License

Copyright (c) 2020 Bobby Iliev

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

How to Install Laravel on DigitalOcean with 1-Click?

Laravel is an amazing PHP framework that makes working with PHP great.

As of the time of writing this tutorial, Laravel has more than 105 million installs according to Packagis, so as you can imagine there are multiple ways of **installing Laravel on DigitalOcean** or any other cloud provider.

In this tutorial, we will go through a few ways of installing Laravel and also show you how to do this with just 1-Click on DigitalOcean!

All you need in order to follow along is a DigitalOcean account. To make things even better you can use the following referral link to get **free \$100 credit** that you could use to deploy your servers and test the guide yourself:

[DigitalOcean \\$100 Free Credit](#)

Installing Laravel on DigitalOcean

If you wanted to configure your LEMP server from scratch you could, for example, do it manually by following this amazing step by step guide provided by DigitalOcean:

- [How to Install and Configure Laravel with LEMP on Ubuntu 18.04](#)

Another option of automating the above steps is to use the [LaraSail](#) open-source script to set up your server and install Laravel:

- <https://github.com/thedevdojo/larasail>

Using the Laravel DigitalOcean 1-Click

DigitalOcean has a Marketplace, where you could get a lot of various 1-Click Applications which you can deploy on your servers and Kubernetes clusters.

Recently we built a Laravel image and submitted a listing to DigitalOcean. The image is now available on the DigitalOcean Marketplace:

<https://marketplace.digitalocean.com/apps/laravel>

The image comes with a preconfigured LEMP stack and certbot installed. The software stack that comes with the image is:

- Laravel 7.20.0
- Nginx 1.18.0
- MySQL server 8.0.20
- PHP 7.4.3
- Certbot 0.40.0
- Composer 1.10.1

In order to use the image just go to the 1-Click Laravel application page and hit the **Create Laravel Droplet** button:

The screenshot shows the DigitalOcean Marketplace interface. At the top, there's a navigation bar with 'Marketplace' (highlighted with a blue outline), 'Solutions ▾', 'For Vendors', 'Sign in', 'Sign up', and a three-dot menu icon. Below the navigation is a search bar with the placeholder 'Search for other solutions'. The main content area displays a listing for 'Laravel' by DevDojo. The listing includes a thumbnail icon, the name 'Laravel', the developer 'by DevDojo', the version 'Version 7.20.0', the OS 'Ubuntu 20.04', and two buttons: 'FRAMEWORKS' and 'NEW'. To the right of the listing is a blue button labeled 'Create Laravel Droplet'. Below the listing, there's a section titled 'Description' containing a brief text about Laravel and a 'Support Details' section with a support URL link.

All Solutions > Frameworks

Laravel
by DevDojo

Version 7.20.0 OS Ubuntu 20.04 FRAMEWORKS NEW

Create Laravel Droplet

Description

Laravel is an open-source PHP framework that provides a set of tools and resources to build modern PHP applications.

Support Details

Support URL: <https://devdojo.com>

After that just follow the standard flow and choose the details that match your needs like:

- The size of the Droplet
- Choose a datacenter region
- Choose your SSH keys
- Choose hostname
- I would recommend enabling backups as well

Finally hit the **Create Droplet** button. Then in around 30 seconds or so your Laravel server will be up and running!

Completing the Laravel configuration

To complete the installation, copy your IP address and [SSH to the Droplet](#).

You would get to an interactive menu that would ask you for the following details:

```
| Enter the domain name for your new Laravel site.  
(ex. example.org or test.example.org) do not include www or  
http/s  
-----  
Domain/Subdomain name:
```

There just type your domain name or subdomain name, I would go for `laravel.bobbyiliev.com` for my example.

After that you would see the following output:

```
| Configuring Laravel database details  
Generating new Laravel App Key  
Application key set successfully.
```

Then you will be asked if you want to secure your Laravel installation with an SSL certificate:

Next, you have the option of configuring LetsEncrypt to secure your **new** site.

Before doing this, be sure that you have pointed your domain or subdomain to this server's IP address.

You can also run LetsEncrypt certbot later with the command
`'certbot --nginx'`

Would you like to use LetsEncrypt (certbot) to configure SSL(https) for your new site? (y/n) :

Note that before hitting **y** make sure to have your domain name or subdomain DNS configured so that your A record points to the Droplet's IP address, otherwise Let's Encrypt will not be able to validate your domain and would not issue a certificate for you.

If you don't want a certificate, just type **n** and hit enter.

That is pretty much it! Now if you visit **yourdomain.com** in your browser you would see a fresh new installation!

Video Demo

Here's also a quick video demo on how to do the above:

[How to Install Laravel on DigitalOcean with 1-Click](#)

Conclusion

Thanks to the Laravel 1-Click installation from the DigitalOcean's Marketplace, we can have a fully running LEMP server with Laravel installed in less than 30 seconds!

[Originally posted here.](#)

How to get a free domain name for your Laravel project

There could be various reasons why you would need a free domain for your project.

For example, having multiple side projects could be quite costly in case that your projects are not generating any income. So saving costs could be crucial.

Another reason why you might need a free domain name is that you might want to do some just testing and not really need an actual domain which could cost you anywhere from \$5 to +\$50 dollars depending on the provider and the domain extension.

In this tutorial, I will show you **how to get a free domain name** from [Freenom](#) and use it with your Laravel Project.

Choosing a free domain name

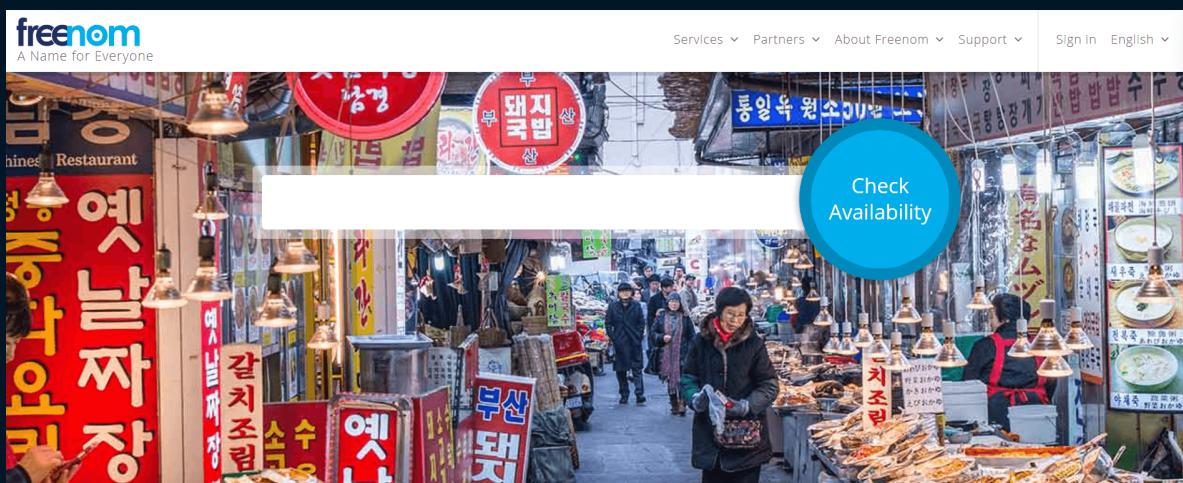
In this tutorial, we will use [Freenom](#) to register a free domain name! As far as I am aware, Freenom is the world's only free domain name provider.

They provide free domain names with the following domain extensions:

- .TK
- .ML
- .GA
- .CF
- .GQ

Unfortunately, Freenom does not offer free [.com](#) domains but for testing purposes, only the above domain extensions are a great alternative!

To check if a specific domain is available, just visit the [Freenom](#) website, you would get to a page where you can search and check if a specific domain is available:



In my case I will look for [bobbyiliev](#) and then choose the available extension:

The screenshot shows the freenom website's domain search interface. A search bar at the top contains the text "bobbyiliev". To its right is a blue circular button labeled "Check Availability". Below the search bar, three domain options are listed in a table:

Domain	Status	Price	Action
bobbyiliev.tk	• FREE	EUR 0.00	Get it now!
bobbyiliev.ml	• FREE	EUR 0.00	Get it now!
bobbyiliev.ga	• FREE	EUR 0.00	Get it now!

In my case, **bobbyiliev.ml** is available so I will go for that one by clicking on the **Get it now!** button, and then click on **Checkout**.

You would get to a page where you could choose for how long would you like to keep the domain name for:

The screenshot shows the freenom website's domain registration configuration page. At the top, there is a search bar with "Find a new FREE domain" and a "Check Availability" button. Below the search bar, the domain "bobbyiliev.ml" is selected. There are two buttons: "Forward this domain" and "Use DNS". To the right of the domain, a dropdown menu titled "Period" is open, showing various options for the duration of the domain registration:

- 1 Month @ FREE
- 2 Months @ FREE
- 3 Months @ FREE** (selected)
- 4 Months @ FREE
- 5 Months @ FREE
- 6 Months @ FREE
- 7 Months @ FREE
- 8 Months @ FREE
- 9 Months @ FREE
- 10 Months @ FREE
- 11 Month @ FREE
- 12 Months @ FREE
- 1 Year @ EUR 8.22
- 2 Years @ EUR 16.44
- 3 Years @ EUR 24.66
- 4 Years @ EUR 32.88
- 5 Years @ EUR 41.10
- 6 Years @ EUR 49.32
- 7 Years @ EUR 57.54
- 8 Years @ EUR 65.76

Choose the period from the dropdown menu and then hit next. After that follow the steps and sign up for a free account.

Once you are ready with the registration process, then go ahead and

proceed to the next step where you will need to add your new free domain name to your DigitalOcean account.

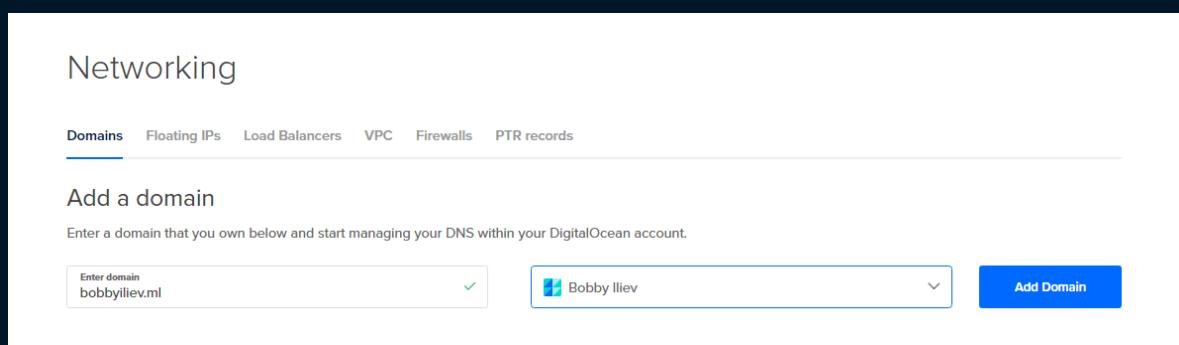
Adding your Domain to DigitalOcean

If you don't have a DigitalOcean account already make sure to create one via this link here:

[Sign up for DigitalOcean](#)

Once you've created a DigitalOcean account follow these steps here:

- First go to your [DigitalOcean Control Panel](#)
- Then click on Networking on the left
- After that click on Domains
- And there add your new domain name button and then add the domain name that you just registered:



Then you would see your new DNS zone where you would need to add an A record for your domain name and point it to your Laravel server! For more information on [how to manage your DNS](#) make sure to read this guide!

Updating your Nameservers

Once you have your Domain name all configured and ready to go in DigitalOcean, then you need to update your Nameservers, so that your domain would start using your new DigitalOcean DNS zone.

To do so just copy the following 3 nameservers:

- ns1.digitalocean.com
- ns2.digitalocean.com
- ns3.digitalocean.com

And then go back to your Freenom account, there follow these steps:

- From the menu click on Services
- Then click on My Domains
- After that click on Manage Domain for your new domain
- Then from the "Management Tools" dropdown click on "Nameservers"
- There choose "Use custom nameservers (enter below)" and enter the 3 DigitalOcean nameservers from above and click "Save"



Finally, it could take up to 24 hours for the DNS to propagate over the

Globe and after that, you will be able to see your Laravel application when visiting your free domain name in your browser!

Conclusion

Now you know how to get a free domain name for your Laravel Project and add it to DigitalOcean where you could manage your DNS zone and point it to your Laravel server.

[Originally posted here](#)

8 Awesome VS Code Extensions for Laravel Developers

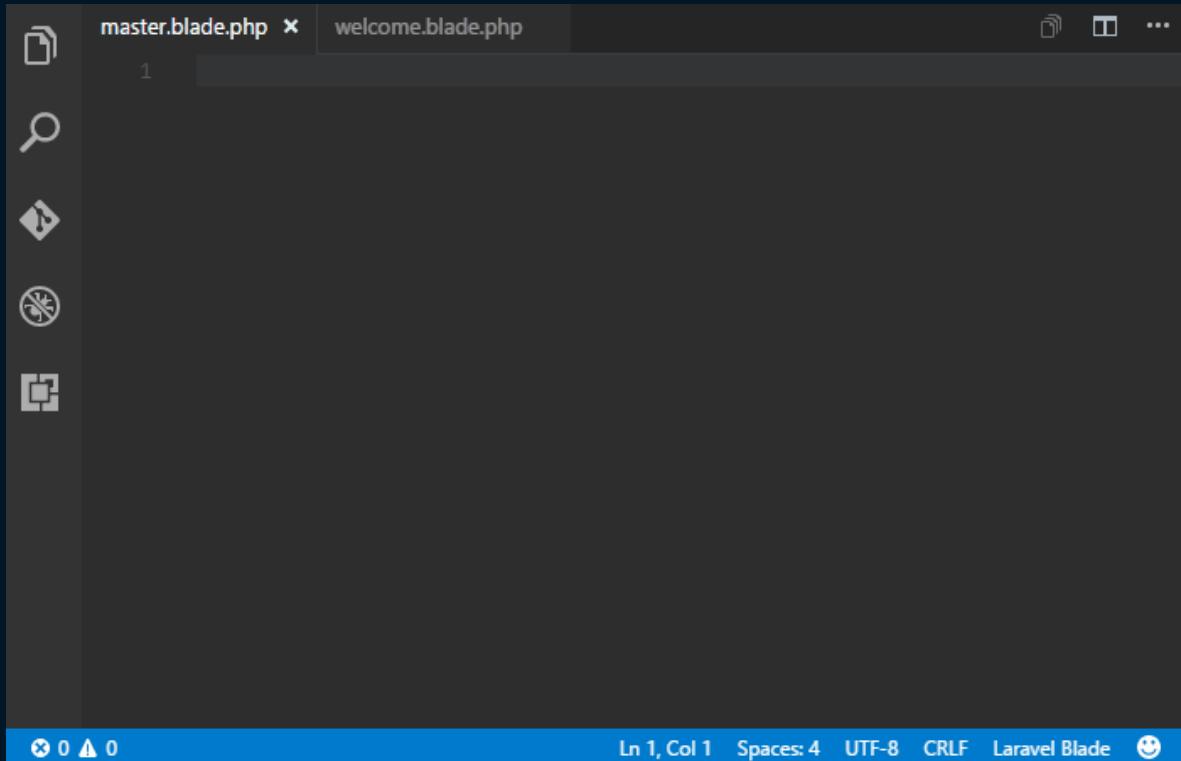
While I'm still a sublime fan for quite some time, I've been mainly using VS Code.

For anyone who is just getting started with Laravel, I would recommend going through this [Laravel basics course here!](#)

Here is a list of my top 8 VS Code extensions for Laravel developers, which would help you be more productive!

1. Laravel Blade Snippets

The [Laravel blade snippets](#) extension adds syntax highlight support for Laravel Blade to your VS Code editor.



Some of the main features of this extension are:

- Blade syntax highlight
- Blade snippets
- Emmet works in blade template
- Blade formatting

In order to make sure that the extension works as expected, there is some additional configuration that needs to be done. Go to **File -> Preferences -> Settings** and add the following to your `settings.json`:

```
"emmet.triggerExpansionOnTab": true,  
"blade.format.enable": true,  
"[blade)": {  
    "editor.autoClosingBrackets": "always"  
},
```

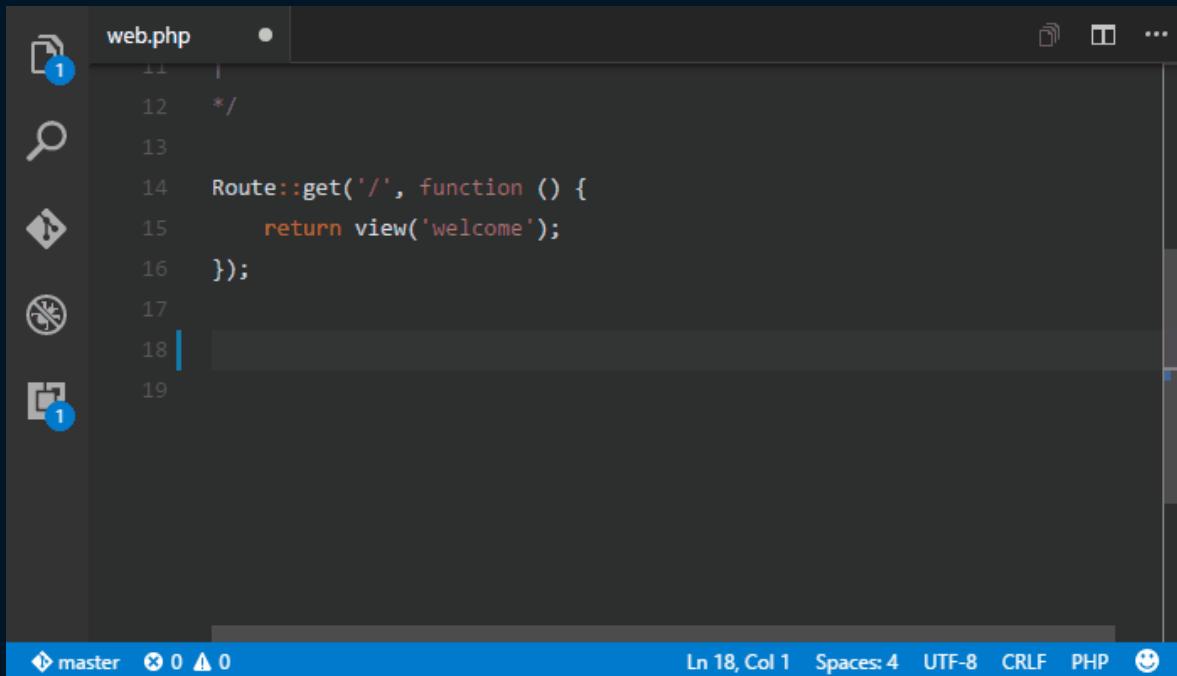
This will enable tab completion for emmet tags and if enable blade formatting.

For more information on the available snippets, make sure to check the documentation here:

[VSCode extensions for laravel](#)

2. Laravel Snippets

This one is probably my personal favorite! The [Laravel Snippets extension](#) adds snippets for the Facades like `Request::`, `Route::` etc.



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with icons for file operations. The main area displays a PHP file named "web.php". The code contains the following snippet:

```
11
12     */
13
14 Route::get('/', function () {
15     return view('welcome');
16 });
17
18
19
```

The cursor is positioned at line 18, where the closing brace of the Route::get() call is located. The status bar at the bottom indicates "Ln 18, Col 1".

Some of the supported snippet prefixes include:

- Auth
- Broadcast
- Cache
- Config
- Console
- Cookie
- Crypt
- DB
- Event
- View

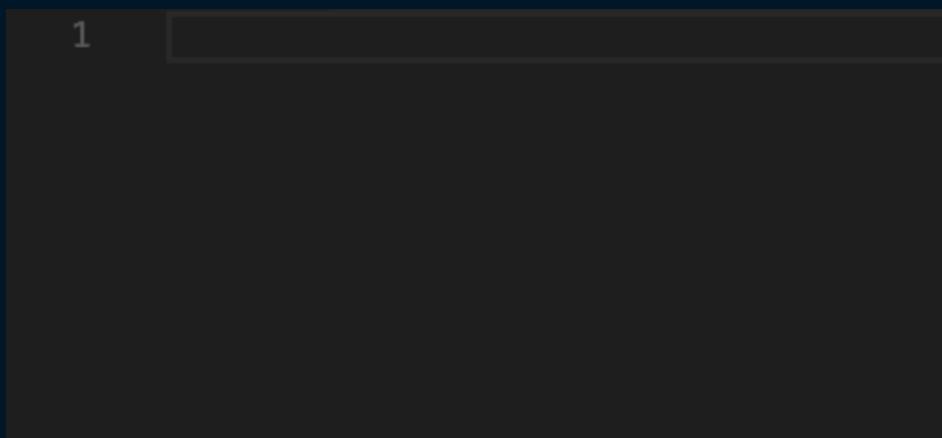
For more information on the available snippets, make sure to check the documentation here:

VSCode extensions for laravel

3. Laravel Blade Spacer

Isn't it annoying when you try to echo out something in your Blade views with `{{ }}` and your whole line going back 4 spaces? Well, luckily, the [Laravel Blade Spacer](#) fixes that!

The Laravel blade spacer extension automatically adds spacing to your blade templating markers:

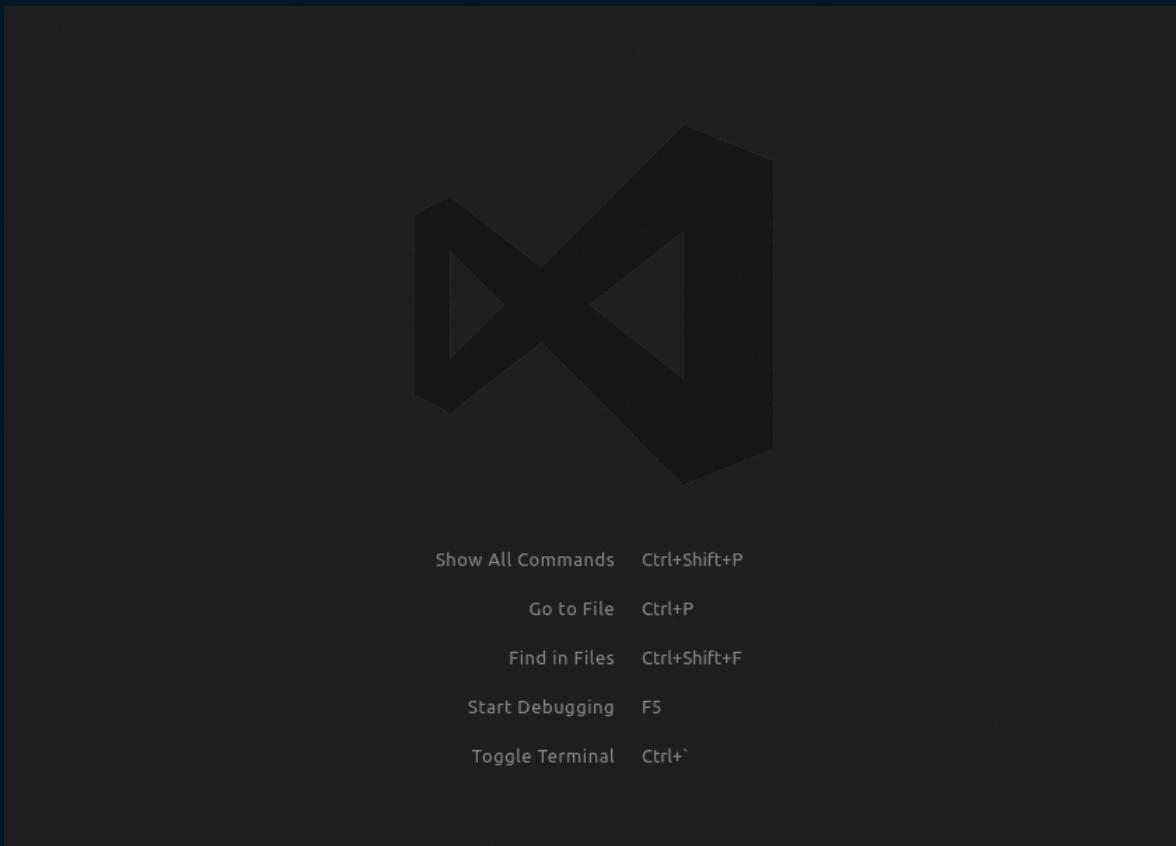


For more information, make sure to check the documentation here:

[VSCode extensions for laravel](#)

4. Laravel Artisan

I personally like to use the command line all the time, but I have to admit that the [Laravel Artisan](#) extension is awesome! It lets you run Laravel Artisan commands from within Visual Studio Code directly!



Some of the main features are:

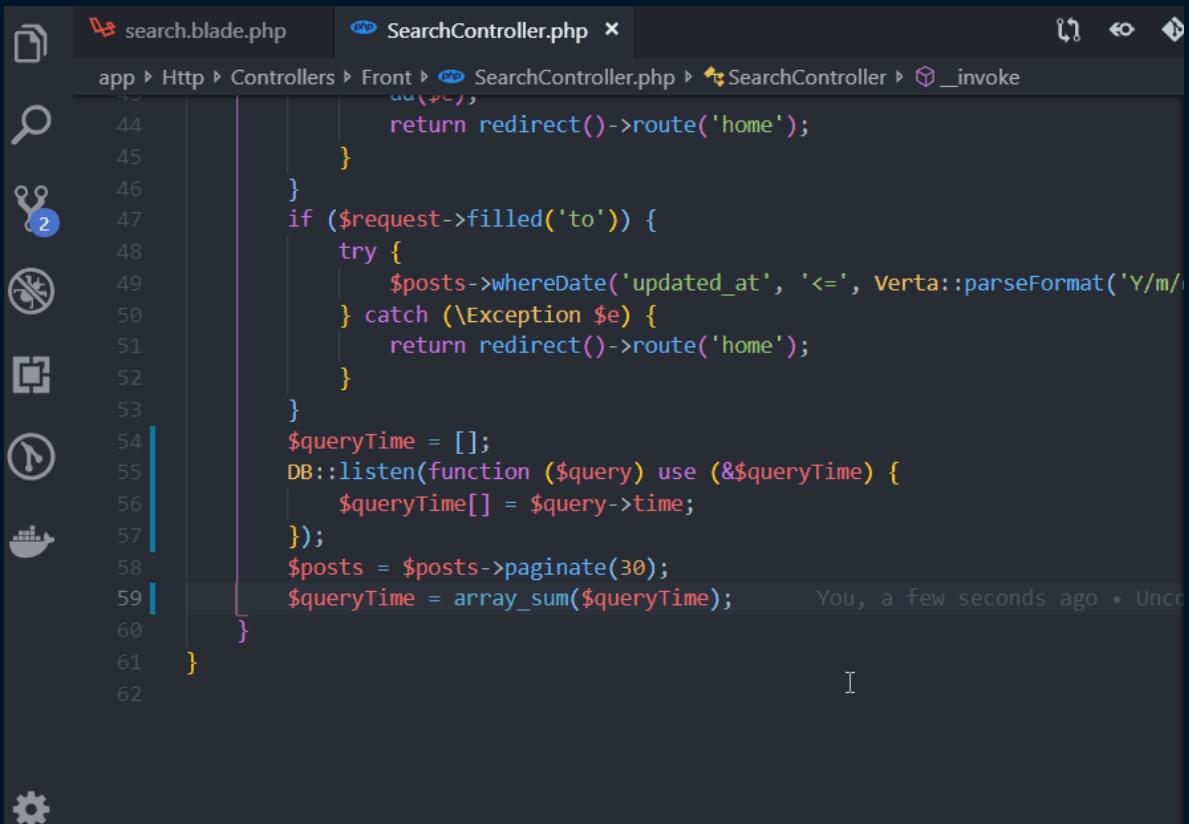
- Make files like Controllers, Migrations, etc.
- Run Your own Custom Commands
- Manage your database
- Clear the Caches
- Generate Keys
- View all application routes
- Manage your local php server for test purposes

For more information, make sure to check the documentation here:

VSCode extensions for laravel

5. Laravel Extra Intellisense

The [Laravel Extra Intellisense](#) extension provides autocompletion for Laravel in VSCode.



The extension comes with auto-completion for:

- Route names and route parameters
- Views and variables
- Configs
- Translations and translation parameters
- Laravel mix function
- Validation rules
- View sections and stacks
- Env
- Route Middlewares

For more information, make sure to check the documentation here:

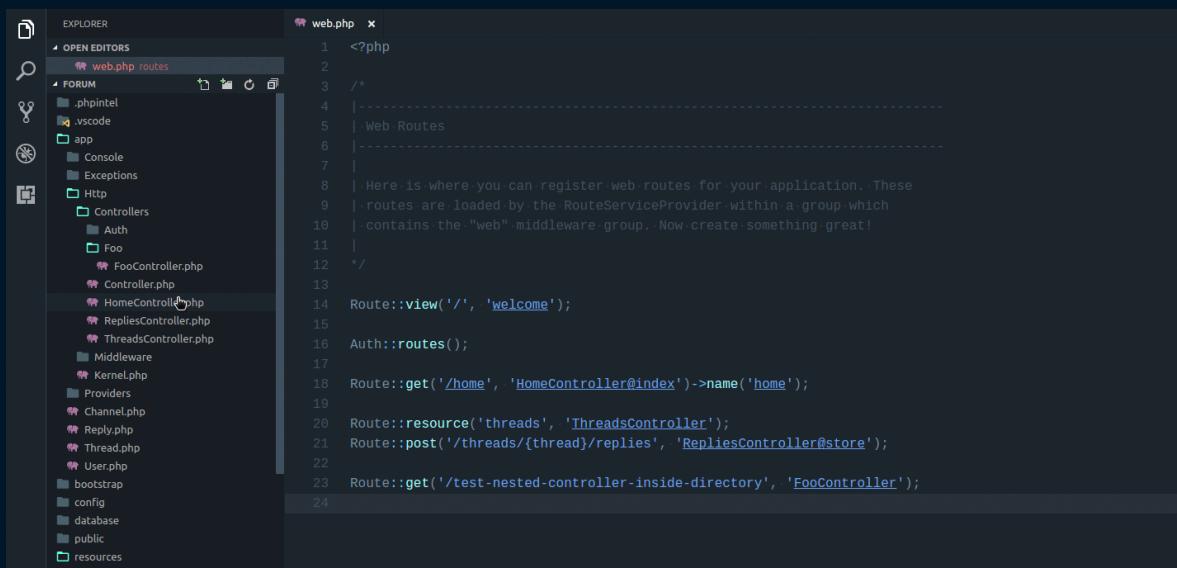
VSCode extensions for laravel

6. Laravel goto Controller

As your application grows, the number of your Controllers grows as well, so at some point, you might end up with hundreds of controllers. Hence finding your way around might get tedious.

This is the exact problem that the [Laravel-goto-controller](#) VScode extension solves.

The extension allows you to press **Alt** + click on the name of the controller in your routes file, and it will navigate you from the route to the respective controller file:



```
<?php
/*
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
*/
Route::view('/', 'welcome');

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');

Route::resource('threads', 'ThreadsController');
Route::post('/threads/{thread}/replies', 'RepliesController@store');

Route::get('/test-nested-controller-inside-directory', 'FooController');
```

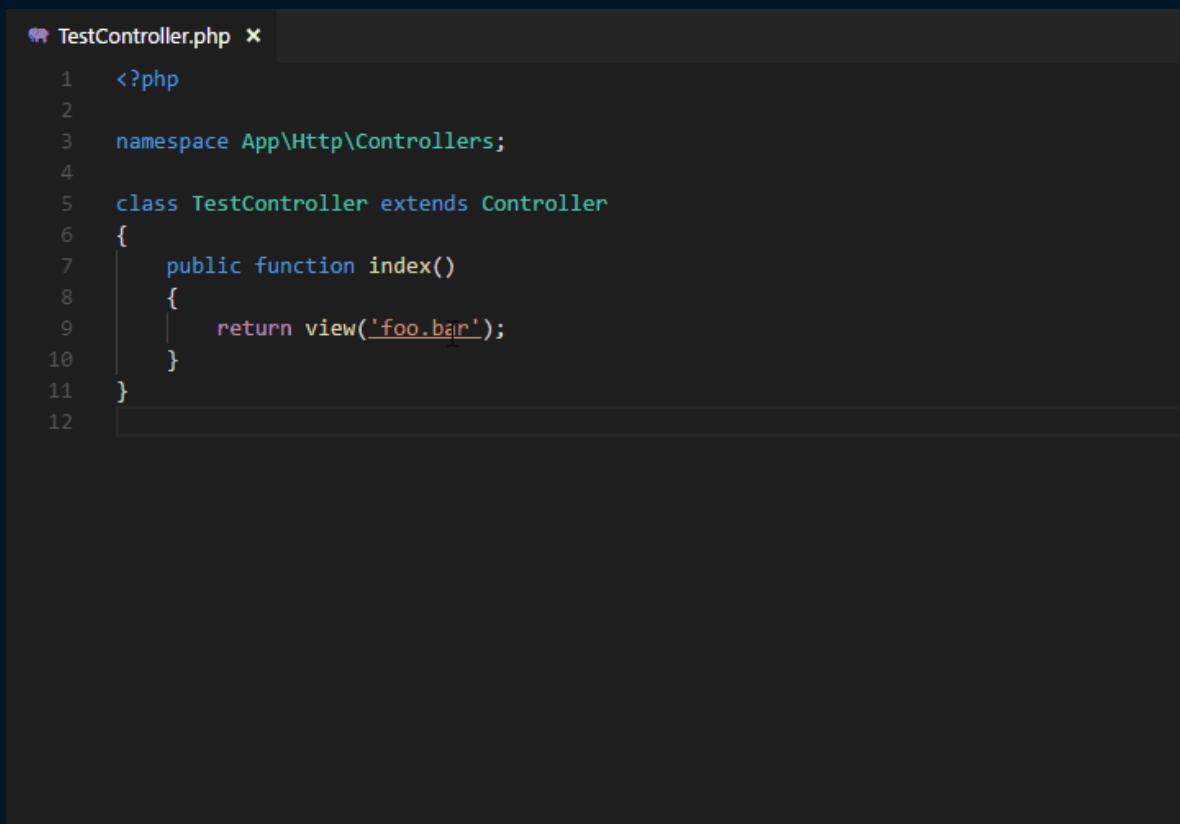
For more information, make sure to check the documentation here:

[VSCode extensions for laravel](#)

7. Laravel goto View

Similar to the Laravel goto Controller extension, the [Laravel goto View VScode extension](#) allows you to go from your Controller or Route to your view. This can save you quite a bit of time!

You can use **Ctrl** or **Alt** + click to jump to the first matched Blade View file:



```
TestController.php x
1 <?php
2
3 namespace App\Http\Controllers;
4
5 class TestController extends Controller
6 {
7     public function index()
8     {
9         return view('foo.bar');
10    }
11 }
12
```

For more information, make sure to check the documentation here:

[VSCode extensions for laravel](#)

8. DotENV syntax highlighting

This one is pretty simple but handy. The [DotENV](#) VS Code extension is used to highlight the syntax of your `.env` file, which could be quite handy for spotting some problems:

```

1 # Keyword
2 export TEST_KEYWORD="bar"
3 export TEST_KEYWORD=12345
4 export TEST_KEYWORD=TRUE
5
6 # Variable
7 TEST_VARIABLE="Hello"
8
9 # String interpolation
10 TEST_INTERPOLATION_VARIABLE="$VAR1 test test$VAR2test test"
11 TEST_INTERPOLATION_SYNTAX_ONE="test test${NVAR1}test${NVAR2}test test"
12 TEST_INTERPOLATION_SYNTAX_TWO="test test${NVAR1}test${NVAR2}test test"
13 TEST_INTERPOLATION_SYNTAX_ALL="test$VAR1test test ${VAR2}test test${VAR3}test"
14
15 # Unquoted
16 TEST_UNQUOTED=bar
17 TEST_UNQUOTED_NO_VALUE=
18
19 # White spaced
20 TEST_WHITE_SPACE =
21 TEST_WHITE_SPACE_STRING = "Hello"
22 TEST_WHITE_SPACE_UNQUOTED = bar
23 TEST_WHITE_SPACE_UNQUOTED_BOOL = false
24 TEST_WHITE_SPACE_UNQUOTED_NUM = 20
25
26 # language constants
27 TEST_TRUE=true
28 TEST_FALSE=false
29 TEST_NULL=null
30 TEST_TRUE_CAPITAL=TRUE
31 TEST_FALSE_CAPITAL=FALSE
32 TEST_NULL_CAPITAL=NULL
33
34 # Numerical values
35 TEST_NUM_DECIMAL=54
36 TEST_NUM_FLOAT=5.3
37 TEST_NUM=1e10
38 TEST_NUM_NEGATIVE=-42
39 TEST_NUM_OCTAL=057
40 TEST_NUM_HEX=0x1A
41
42 # Comments
43 #TEST_ONE=foobar
44 # TEST_TWO='foobar'
45 # TEST_THREE="foobar" # a comment on a commented row
46 TEST_FOUR="test test test" # this is a comment
47 TEST_FIVE="comment symbol # inside string" # this is a comment
48 TEST_SIX="comment symbol # and quotes \" \' inside quotes" # this is a comment
49
50 # Escape sequences
51 TEST_ESCAPE="escaped characters \n \t \r \" \' \$ or maybe a backslash \\..."
52
53 # Double Quotes
54 TEST_DOUBLE="Lorem ${VAR1} ${VAR2} $VAR3 ipsum dolor sit amet\n\r\t\\"
55
56 # Single Quotes
57 TEST_SINGLE='Lorem ${VAR1} ${VAR2} $VAR3 ipsum dolor sit amet\n\r\t\\'

```

For more information, make sure to check the documentation here:

[VSCode extensions for laravel](#)

Book recommendation

If you are a Laravel fan, make sure to check out the [The Laravel Survival Guide](#) ebook!

Conclusion

If you like all those extensions, you can take a look at the [Laravel Extension Pack for Visual Studio Code](#), where you could get all of the mentioned extensions as 1 bundle!

The only extension not included in the pack is the Laravel Blade Spacer, so make sure to install it separately!

[Originally posted here.](#)

What is Laravel Jetstream and how to get started

[Laravel 8](#) was released on September 8th 2020 along with Laravel Jetstream.

[Laravel Jetstream](#) is a new application scaffolding for Laravel. Laravel Jetstream replaces the legacy Laravel authentication UI available for previous Laravel versions.

In this tutorial, I will give you a quick introduction to what exactly Laravel Jetstream is and how to get started with it.

If you want to follow along, you would need a LEMP server together with [composer](#) or the latest Laravel installer.

I will use DigitalOcean for the demo. If you do not have a DigitalOcean account yet, you can use the following referral link to get free \$100 credit that you could use to deploy your servers and test the guide yourself:

[**DigitalOcean \\$100 Free Credit**](#)

What is Laravel Jetstream

Jetstream gives you a better starting point for your new projects. It includes the following components:

- Login and registration functionality
- Email verification
- Two-factor authentication
- Session management
- API support via Laravel Sanctum

Laravel Jetstream replaces the legacy Laravel authentication UI available for previous Laravel versions.

Jetstream uses Tailwind CSS, and you can choose between Livewire or Inertia.

Laravel Jetstream is free and opensource.

Installing Laravel Jetstream

You can choose between a couple of ways of installing Laravel Jetstream. You could either use [composer](#) or the Laravel installer.

Installing Jetstream with Laravel installer

If you already have the latest version of the [Laravel installer](#), you just need to use the `--jet` flag in order to install a new Laravel Jetstream project:

```
| laravel new project-name --jet
```

After that, as usual, make sure to run your migrations:

```
| php artisan migrate
```

Installing Jetstream with Composer

If you prefer using composer, you need to run the following command inside your Laravel directory just like you would with any other package:

```
| composer require laravel/jetstream
```

Note: you need to have Laravel 8 installed. Otherwise, the above command will fail.

After that, you would need to run `artisan jetstream:install` and specify the stack that you want to use:

- If want to use Livewire with Blade run:

```
| php artisan jetstream:install livewire
```

- And if you want to use Inertia with Vue run:

```
| php artisan jetstream:install inertia
```

You may also add the `--teams` flag to enable [Laravel Jetstream team support](#).

After that, execute:

```
| npm install && npm run dev
```

The command above will build your assets.

Finally, make sure to run your migrations:

```
| php artisan migrate
```

Authentication

Your new Jetstream application comes out of the box with:

- Login form
- Two-factor authentication
- Registration form
- Password reset
- Email verification

You can find those views at:

```
| resources/views/auth
```

The backend logic is powered by [Laravel Fortify](#).

You can find the Fortify actions at the following directory:

```
| app/Actions/Fortify/
```

And you can find the Fortify configuration at:

```
| config/fortify.php
```

In the `fortify.php` config file, you can make some changes like enable and disable different features like:

```
'features' => [
    Features::registration(),
    Features::resetPasswords(),
    // Features::emailVerification(),
    Features::updateProfileInformation(),
    Features::updatePasswords(),
    Features::twoFactorAuthentication(),
],
```

Profile management

Right out of the box, Jetstream provides you and your users with user profile management functionality which allows users to update their name, email address, and their profile photo.

The user profile view is stored at:

```
| resources/views/profile/update-profile-information-  
| form.blade.php
```

And in case you are using Inertia, the view can be found at:

```
| resources/js/Pages/Profile/UpdateProfileInformationForm.vue
```

The following file handles the user update logic:

```
| app/Actions/Fortify/UpdateUserProfileInformation.php
```

In case that you want to, you could also disable the user profile picture via your Jetstream config file at:

```
| config/jetstream.php
```

Just comment out the `Features::profilePhotos()` line:

```
'features' => [
    // Features::profilePhotos(),
    Features::api(),
    // Features::teams(),
],
```

Jetstream Security

Laravel Jetstream comes with the standard functionality that allows users to update their password and log out:

The screenshot shows a 'Update Password' form. It includes fields for 'Current Password', 'New Password', and 'Confirm Password'. A 'SAVE' button is located at the bottom right. Below the form, a note says: 'Ensure your account is using a long, random password to stay secure.'

However, what's more, impressive is that Jetstream also offers two-factor authentication with QR code, which the users could enable and disable directly:

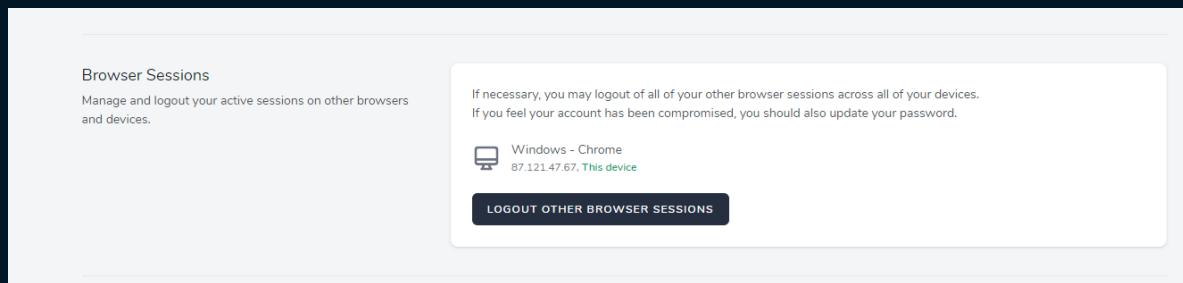
The screenshot shows a 'Two Factor Authentication' setup page. It displays a QR code for enabling two-factor authentication. A note below the QR code says: 'Store these recovery codes in a secure password manager. They can be used to recover access to your account if your two factor authentication device is lost.' A block of recovery codes is shown:

```
sFlk8RnmpY-iYmA35yYko  
U0k6NZ0zYb-fNHENK3EgR  
dczs0VvYfj-xR3nnQH3b  
HDFgvcIwlq-WhYRyVgTh0  
2N27Vcw3K8-1NyJkluwNI  
fuqXe7yQ-7BWFZtH1D  
AGAwIhn513-cIIGJW5Kxk  
wElwTRqb4s-geEI2wJhIK
```

 At the bottom, there are 'REGENERATE RECOVERY CODES' and 'DISABLE' buttons.

Another brilliant security feature is that users can logout other browser

sessions as well.



The profile Blade views can be found at:

| resources/views/profile/

And the if you are using Inertia, you can find them at:

| resources/js/Pages/Profile/

Jetstream API

Laravel Jetstream uses [Laravel Sanctum](#) to provide simple token-based API.

With Sanctum, each user can generate API tokens with specific permissions like Create, Read, Update, and Delete.

Then to check the incoming requests, you can use the `tokenCan` method like this:

```
| $request->user()->tokenCan('read');
```

Again you can disable API support in your `config/jetstream.php` config file.

Jetstream Teams

If you used the `--team` flag during your Jetstream installation, your website would support team creation and management.

With the Jetstream teams feature, each user can create and belong to multiple different teams.

For more information about Jetstream teams, you can take a look at the official documentation [here](#).

Conclusion

Laravel Jetstream gives you a great head start when starting a new project!

I also suggest going through this post here on [what's new in Laravel 8!](#)

[Originally posted here](#)

How to check Laravel Blade View Syntax using artisan

The Laravel community has been growing exponentially, and there are a lot of fantastic packages out there. I recently came across a Laravel package that provides an artisan command to check the syntax of your blade templates.

In this tutorial, I will show you how to use the [Laravel Blade Linter package](#) to check your blade views syntax!

Before getting started, you need to have a Laravel project up and running.

You can use the following referral link to get free \$100 credit that you could use to deploy your servers and test the guide yourself:

[DigitalOcean \\$100 Free Credit](#)

After that, you can follow the steps on [How to Install Laravel on DigitalOcean with 1-Click here!](#)

Installation

```
| composer require magentron/laravel-blade-lint
```

Testing

Once you have the package installed, in order to test your Blade syntax, you need to run the following command:

```
php artisan blade:lint
```

If you don't have any errors, you would see the following output:

```
All Blade templates OK!
```

On another note, if there are any issues anywhere in your blade files, you would get an error like this one:

```
PHP Parse error:  syntax error, unexpected ':', expecting '('  
in /var/www/html/resources/views/welcome.blade.php on line 103  
Found 1 errors in Blade templates!
```

In my case, it tells me that I have a syntax error on line 103 in my `welcome.blade.php` file.

If you don't want to check all of your views, you can specify the path to a specific directory:

```
php artisan blade:lint resources/views/blog
```

Conclusion

The Laravel Blade Linter is a great package that can help you avoid errors before pushing your code to production!

If you like the package, make sure to contribute at on [Github](#)!

[Originally posted here](#)

This is a sample from "Laravel Tips and Tricks" by Bobby Iliev.

For more information, [Click here](#).