AN OPENSOURCE EBOOK

Laravel

Tips and Tricks

Bobby Iliev

Table of Contents

About the book **15**

- About the author 16
- Sponsors 17
- Ebook PDF Generation Tool 19
- Book Cover 20
- License 21

How to Install Laravel on DigitalOcean with 1-Click? **22**

- Installing Laravel on DigitalOcean 23
- Using the Laravel DigitalOcean 1-Click 24
- Completing the Laravel configuration 26
- Video Demo 28
- Conclusion 29

How to get a free domain name for your Laravel project **30**

- Choosing a free domain name 31
- Adding your Domain to DigitalOcean 34
- Updating your Nameservers 35
- Conclusion 37

8 Awesome VS Code Extensions for Laravel Developers **38**

- 1. Laravel Blade Snippets 39
- 2. Laravel Snippets 41
- 3. Laravel Blade Spacer 43
- 4. Laravel Artisan 44
- 5. Laravel Extra Intellisense 46

6. Laravel goto Controller 48
7. Laravel goto View 49
8. DotENV syntax highlighting 50
Book recommendation 51
Conclusion 52

What is Laravel Jetstream and how to get started 53
What is Laravel Jetstream 54
Installing Laravel Jetstream 55
Authentication 57
Profile management 59
Jetstream Security 61
Jetstream API 62
Jetstream Teams 63
Conclusion 64

How to check Laravel Blade View Syntax using artisan 65
Installation 66
Testing 67
Conclusion 68

How to speed up your Laravel application with PHP OPcache .. 69
Enable PHP OPcache 70
Configure PHP OPcache 72
Configure Laravel OPCache 74
Conclusion 76

What is Laravel Sail and how to get started 77
Installing And Setting Up Laravel Sail 79

Laravel Sail Commands 80
Video Introduction 82
Conclusion 83

How to add simple search to your Laravel blog/website 84
Controller changes 85
Route changes 86
Blade view changes 87
Conclusion 88

How to Create Custom Laravel Maintenance Page 89
Enable Default Maintenance Mode 90
Create Custom Maintenance Page 92
Disable Maintenance Mode 94
Conclusion 95

What is Laravel Blade UI Kit and how to get started 96
Installation 97
Configuration 98
Example 101
Conclusion 103

How to Add a Simple Infinite Scroll Pagination to Laravel 104
Configuring your Controller 106
Configuring your Blade View 107
Adding and configuring jScroll 110
Conclusion 112

How to add a simple RSS feed to Laravel without using a package 113

Configuring RSS Controller 115
Configuring your Blade view 117
Configuring your Route 119
Conclusion 120

What is Laravel Zero and how to get started 121
Installation 122
Commands 125
Configuration 127
Addons 129
Building the application 132
Conclusion 133

How to build a blog with Laravel and Wink 134
Installation 135
Creating a Controller 137
Creating Views 139
Adding routes 143
Conclusion 144

How to copy or move records from one table to another in Laravel 145
Copy all records from one table to another 146
Move all records from one table to another 148
Conclusion 149

How to generate title slugs in Laravel 150
Creating a slug in Laravel 151
Conclusion 152

What is Laravel Enlightn and how to use it **153**
Installation 155
Configuration 156
Usage 157
Analyzing the output 159
Conclusion 161

How to consume an external API with Laravel and Guzzle ... **162**
Creating a new table 163
Create the Model 165
Create the Controller 166
QuizAPI overview 167
Building the method 169
Add the route 172
Conclusion 173

How to send Discord notifications with Laravel **174**
Creating a Discord Channel and Webhook 176
Installing the http-client 177
Adding the Controller 178
Adding the Routes 181
Conclusion 182

How to encrypt and decrypt a string in Laravel **183**
App Key Configuration 185
Creating a Route 186
Creating a Controller 187
Conclusion 190

How to remove a package from Laravel using composer **191**
Adding a package to Laravel 192
Removing a package from Laravel using composer 193
Conclusion 194

What is Laravel Breeze and how to get started **195**
Overview 197
Installation 198
File structure 200
Video Overview 202
Conclusion 203

What are signed routes in Laravel and how to use them **204**
Defining a signed route 206
Generating the signature 207
Using the signed route 208
Conclusion 210

How to Quickly Change the Password for a User in Laravel .. **211**
Reset password with tinker 212
Reset password via a route 215
Conclusion 216

How to convert markdown to HTML in Laravel and Voyager .. **217**
Install the Laravel Markdown Package 219
Configure your Controller 220
Change the Input type in Voyager 222

Conclusion 223

How to Create Response Macros in Laravel 224
Creating a service provider 225
Creating the Response Macro 226
Using the Response Macro 227
Conclusion 228

How to Get the Base URL in Laravel 229
Access the Laravel Base URL 231
Access the current URL in Laravel 232
Named routes 233
Access Assets URLs 234
Conclusion 235

How to limit the length of a string in Laravel 236
Limit string length in Blade 237
Limit string length in Model 238
Limit string length in Controller 239
Conclusion 240

How to check 'if not null' with Laravel Eloquent 241
Check if null: whereNull 242
Check if not null: whereNotNull 243
Conclusion 244

How to get the current date and time in Laravel 245
Using Carbon to get the current date and time 246
Using the now() helper function 249
Conclusion 250

How to Get Current Route Name in Laravel **251**
Get route name in Controller or Middleware 252
Get route name in Blade View 254
Getting additional information about your route 255
Conclusion 256

How to Count and Detect Empty Laravel Eloquent Collections .. **257**
Check if a collection is empty 259
Check if a collection is not empty 260
Counting the records of a collection 261
Conclusion 262

How to use Forelse loop in Laravel Blade **263**
Check if not empty then 264
Forelse example 265
Conclusion 266

How to Delete All Entries in a Table Using Laravel Eloquent .. **267**
Using truncate to delete all entries from a table 269
Delete all entries using the delete() method 270
Conclusion 271

How to check if a record exists with Laravel Eloquent **272**
Check if a record exists 273
Create record if it does not exist already 274
Conclusion 276

How to Add Multiple Where Clauses Using Laravel Eloquent .. **277**

Chaining multiple where clauses together 278
Passning an array to the where method 279
Using orWhere Clauses 280
Conclusion 281

How to Add a New Column to an Existing Table in a Laravel Migration **282**

Creating a new table with a migration **283**

Add a New Column to an Existing Table **286**

Conclusion **289**

How to Rollback Database Migrations in Laravel **290**

Rollback the Last Database Migration 291
Rollback Specific Migration 292
Rollback All Migrations 293
Conclusion 294

How to Remove a Migration in Laravel **295**

Creating a Laravel migration 296
Remove a Migration in Laravel 297
Conclusion 299

How to create a contact form with Laravel Livewire **300**

Installing Livewire 301

Adding new Livewire component 302
Creating your Blade view 303
Prepare your Livewire view 305
Prepare your Livewire Logic 308
Add routes 311
Adding SMTP details 312
Conclusion 313

How To Display HTML Tags In Blade With Laravel 314
Display HTML In Laravel Blade Views 315
Conclusion 317

How to Set a Variable in Laravel Blade Template 318
Returning a variable from a controller 319
Setting variables in Laravel Blade Template 320
Conclusion 321

How to limit the result with Laravel Eloquent 322
Limiting the result with pure SQL 324
The limit() method 325
The take() method 326
The paginate() method 327
Conclusion 328

How to Select Specific Columns in Laravel Eloquent 329
Select specific columns with SQL only 330
Select specific columns with Laravel Eloquent 331
Conclusion 332

How to get the Laravel Query Builder to Output the Raw SQL

Query **333**
The toSql() method 334
Laravel Debugbar 336
Conclusion 337

How to Get the Last Inserted Id Using Laravel Eloquent **338**
Last Inserted Id Using Laravel Eloquent 339
Conclusion 341

How to Order the Results of all() in Laravel Eloquent **342**
Ordering the results on a query level 344
Ordering the results on a collection level with all() 346
Conclusion 348

How to fix Laravel Unknown Column 'updated_at' **349**
Disable the timestamp columns in your Model 351
Change the name of the timestamp tables 352
Conclusion 353

How to Define Custom ENV Variables in Laravel **354**
Defining a variable in .env 355
Accessing the env variable with env() 356
Accessing the env variable with config() 357
Conclusion 359

How to fix 'Please provide a valid cache path' error in Laravel **360**
Creating the cache folders 361
Clearing the cache 362
Conclusion 363

How to check your exact Laravel version **364**
Check your Laravel version with artisan 365
Check your Laravel version via your text editor 366
Conclusion 367

Contact Form with Voyager and Laravel **368**
Steps 369
Configuration 370
Conclusion 377

How to filter routes in Laravel **378**
Steps 379
Filter routes by term 380
Conclusion 382

How to add a gravatar profile picture in Laravel **383**
Steps 384
Create a trait to handle the gravatar 385
Render an image in a blade view: 387
Use gravatars with Jetstream 388
Conclusion 389

How to append a mutator to a collection **390**
Append a mutator on a single model 391
Usage 392
Append a mutator to a collection 393
Conclusion 394

How to use env variables in javascript **395**

An example 396
Conclusion 397

How to customize forgot password notification 398
Create a custom notification 399
Override the notification 400
Conclusion 401

How to fire an event when user is logged in 402
Create a custom service provider 403
Conclusion 404

Scaling Laravel App with Multiple Databases 405
Prerequisites 406
Steps 407
1. Deploy 3 servers on Digital Ocean 408
2. Install and Configure our Web server 409
Install and configure our database servers along with our MySQL replication 413
Configure Laravel to work with the Master-Slave MySQL setup 416
Install the Voyager admin panel on the new setup 418
Conclusion 420

Conclusion 422
Other eBooks 423

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 more 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.

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 Packagist, 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:

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 domina 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

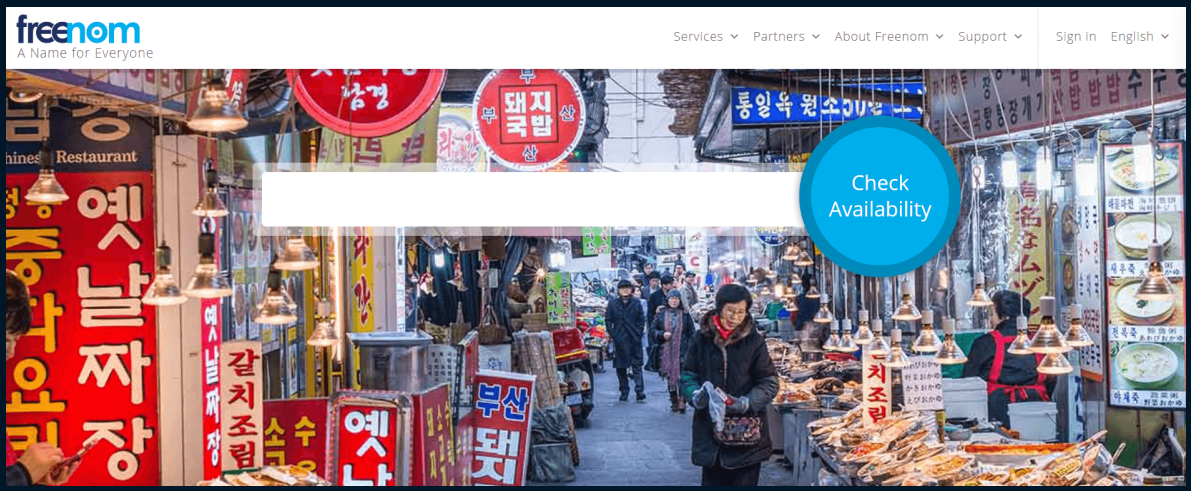
In this tutorial, we will use Freenom to register a free domain name! As far as I am aware, Freenom is the world's only free domain name provider.

They provide free domain names with the following domain extensions:

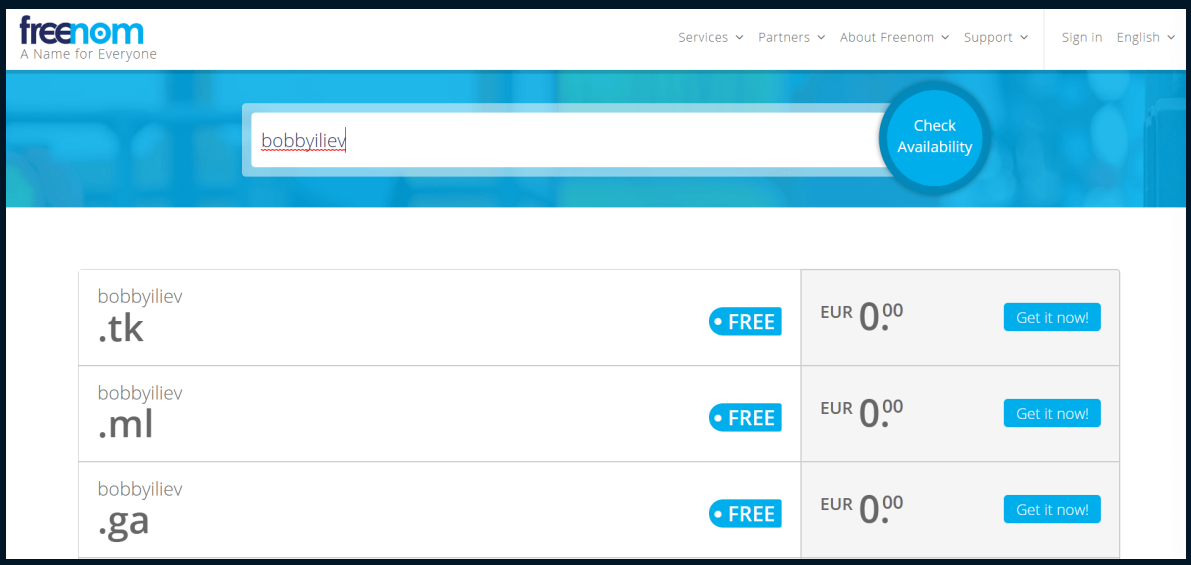
- .TK
- .ML
- .GA
- .CF
- .GQ

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:

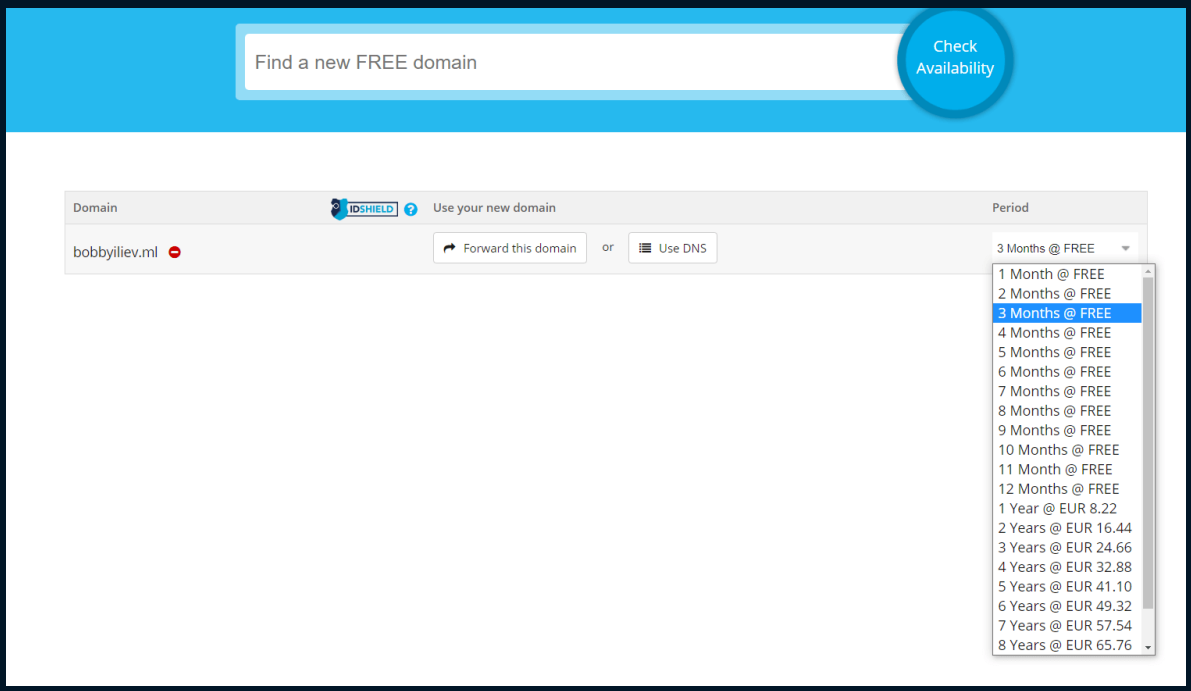


In my case I will look for `bobbyiliev` and then choose the available extension:



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:



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.

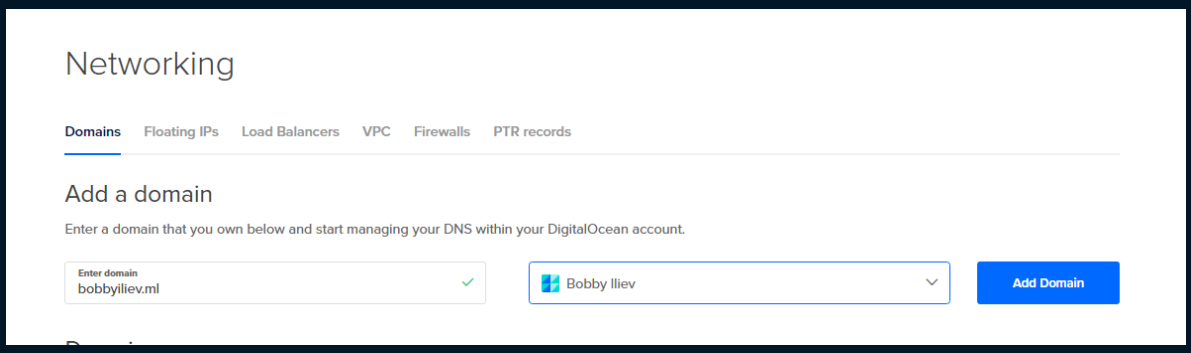
Adding your Domain to DigitalOcean

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 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

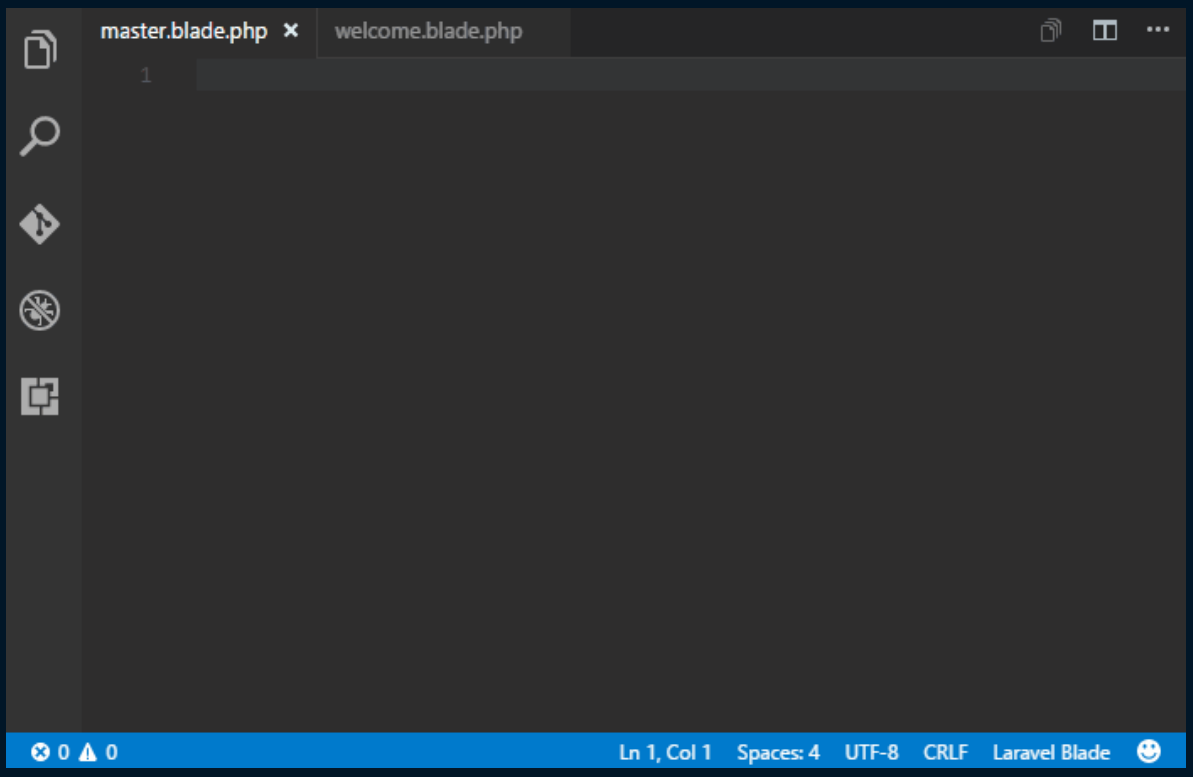
While I'm still a sublime fan for quite some time, I've been mainly using VS Code.

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"
},
```

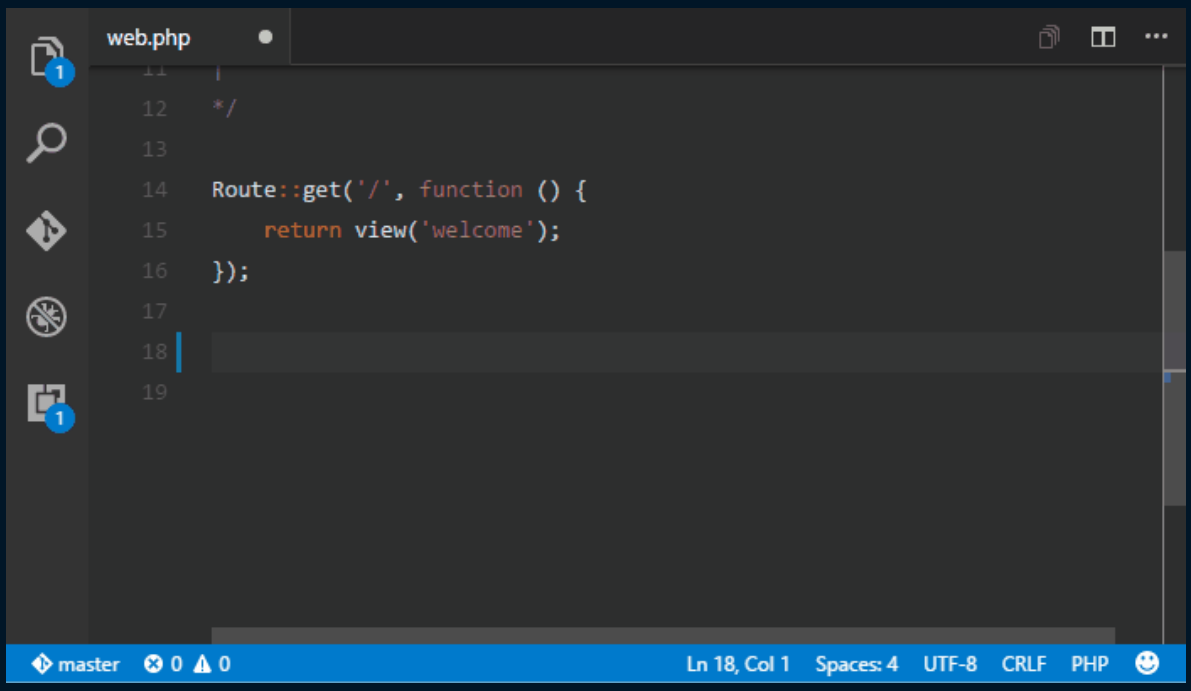
This will enable tab completion for emmet tags and if enable blade formatting.

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.



Some of the supported snippet prefixes include:

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

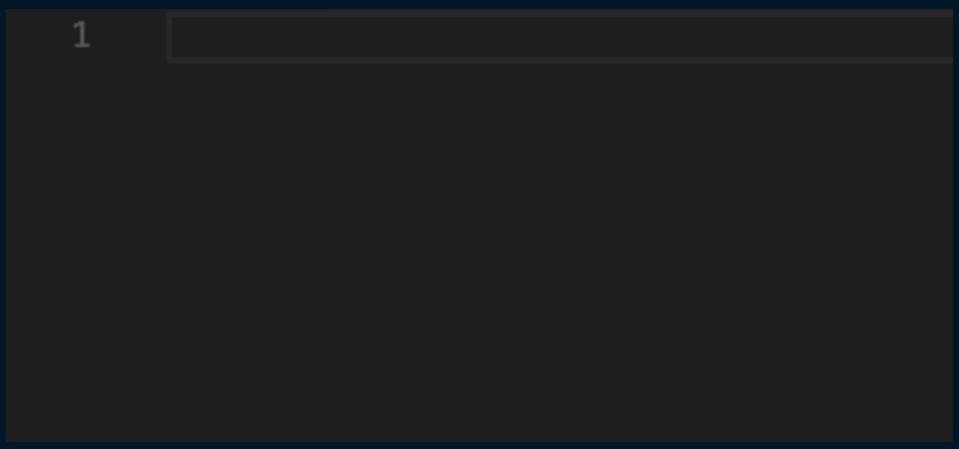
For more information on the available snippets, make sure to check the documentation here:

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:

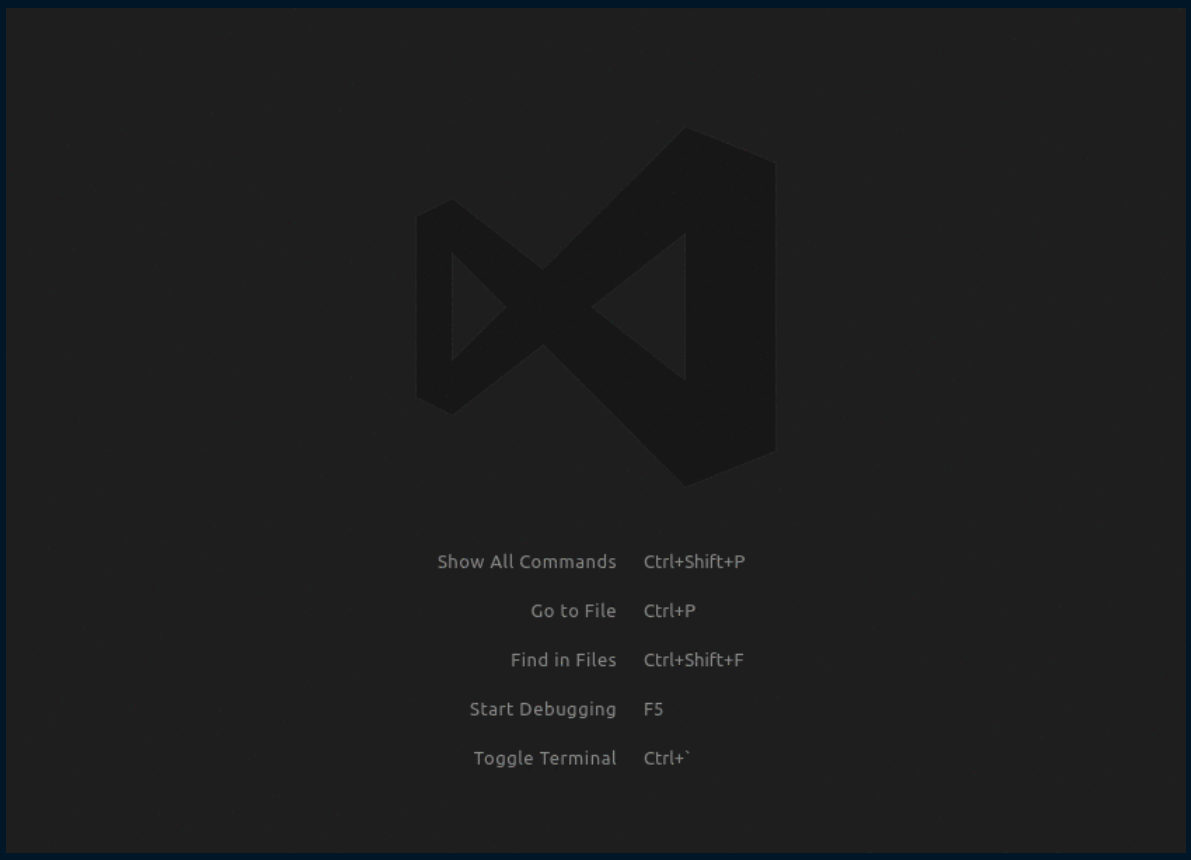


For more information, make sure to check the documentation here:

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:

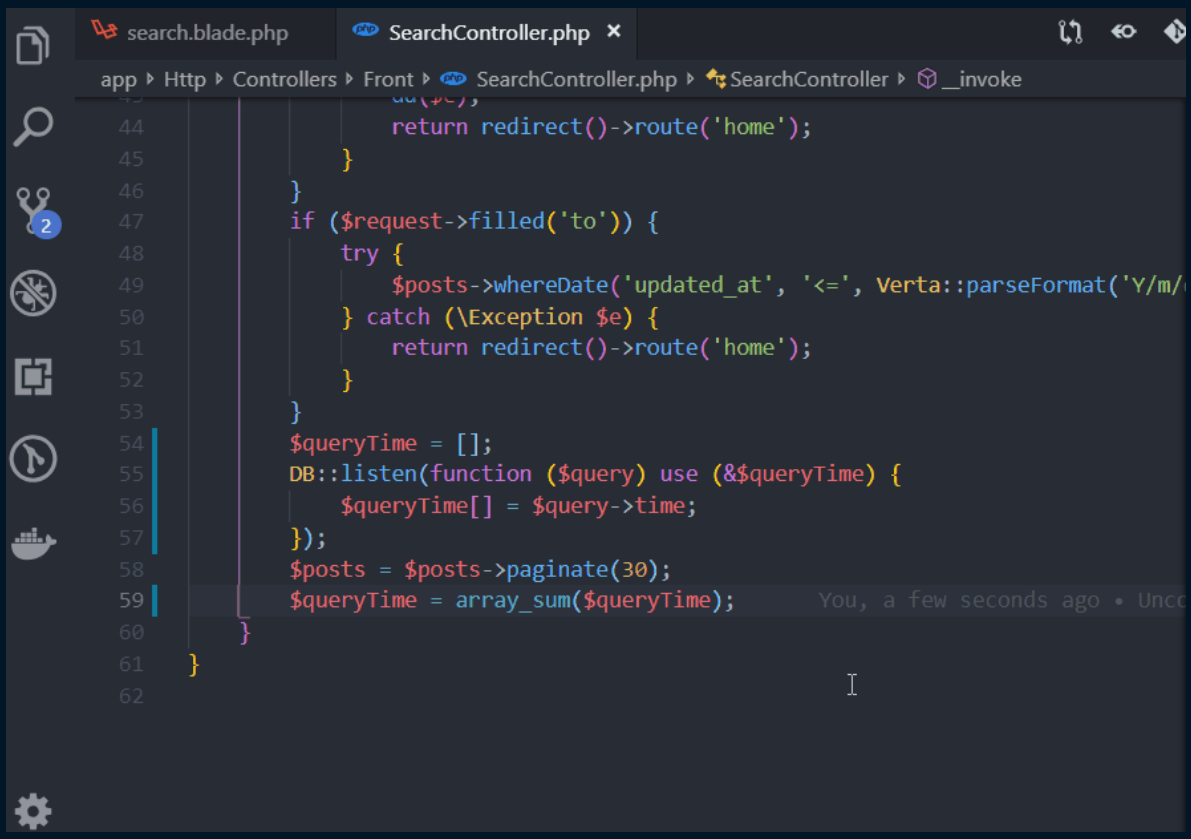
- Make files like Controllers, Migrations, etc.
- Run Your own Custom Commands
- Manage your database
- Clear the Caches
- Generate Keys
- View all application routes
- Manage your local php server for test purposes

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:

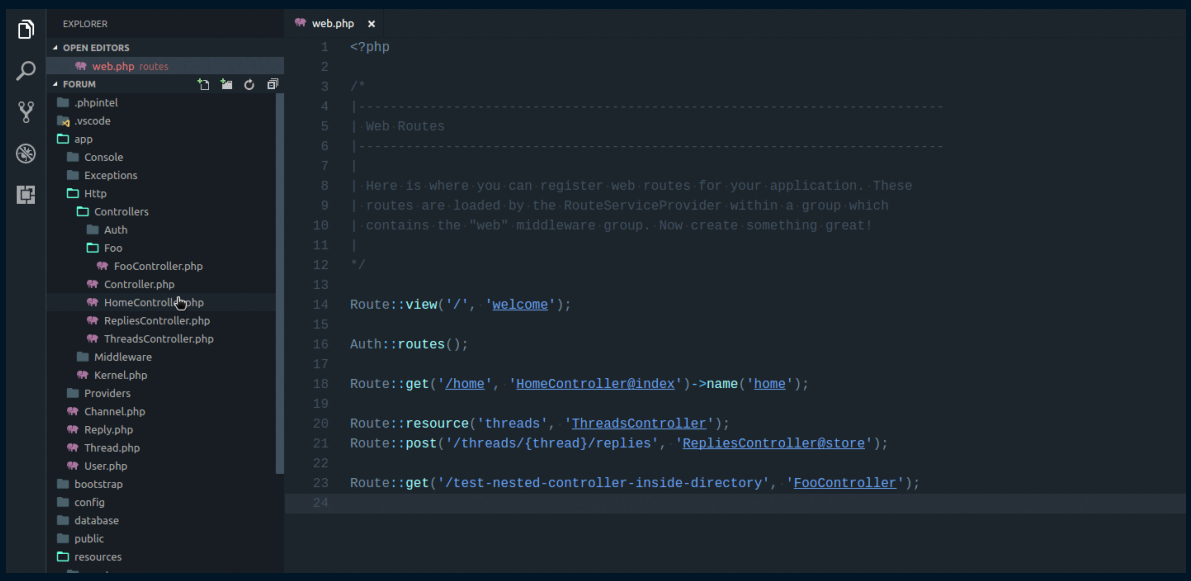
VSCode extensions for laravel

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. Hance 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:



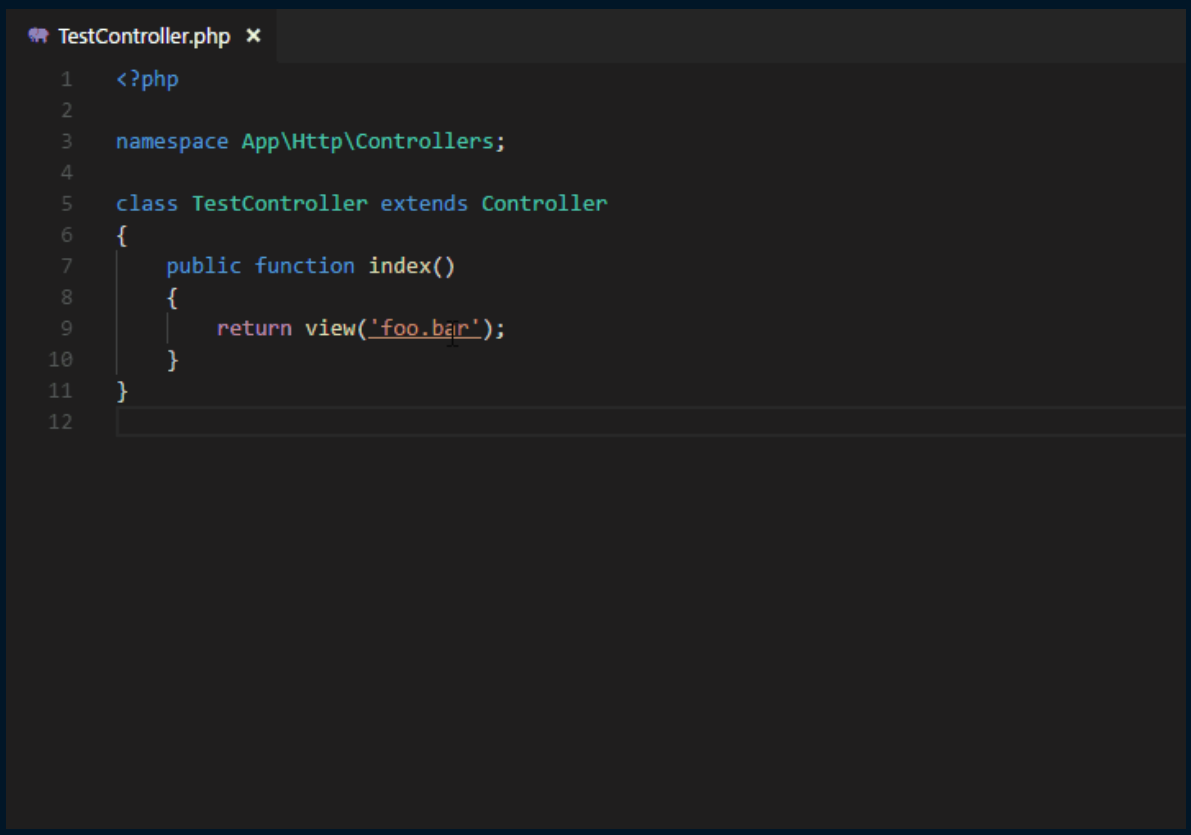
For more information, make sure to check the documentation here:

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:



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:

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:

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

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 a 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

How to speed up your Laravel application with PHP OPcache

Using PHP OPcache is a great way to improve your overall performance. OPcache stores pre-compiled script bytecode in memory, which eliminates the need for PHP to load and parse scripts on every request.

In this tutorial, you will learn how to use the Laravel along with OPcache to speed up your website!

In order for you to be able to follow along, 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**!

Enable PHP OPcache

There are a couple of ways of checking if you already have PHP OPcache enabled.

You could either add a PHP info file in your Laravel `public` folder with the following content:

```
<?php

phpinfo();
```

And then visit the file via your browser. After that, use `CTRL+F` and search for OPcache. You will see the following information:

Another way of checking if OPcache is enabled is to run the following command via your terminal:

```
php -m | grep -i opcache
```

The output that you would see the following result:

```
Zend OPcache
```

If you don't have OPcache enabled, you can install it with the following command on Ubuntu:

```
sudo apt install php-opcache
```

If you are not using Ubuntu, you can install PHP OPcache using `pecl`:

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

For more information, Click here.