

Contents

Pre-Requisites:	2
Composer	2
Download Composer:	2
Install the Composer:	2
composer.json.....	3
The <code>require</code> key#.....	3
Laravel Installation:.....	3
Laravel Architecture.....	4
Console	5
Events	5
Exceptions	5
Authentication	6
Create a Model:	7
Single Cmd to Create Migration and Model at a time	7
Creating Controller.....	7
KODIARY Technologies.....	7
Integrating with Views with Laravel	7
Integrating with Admin Panel with Laravel.....	7
A project from Scratch:	8
Create a project:	8
Login / Registration Module	9

Pre-Requisites:

Composer: Composer is a tool for **dependency management** in PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you.

Composer is not a package manager in the same sense as Yum or Apt are. Yes, it deals with "packages" or libraries, but it manages them on a per-project basis, installing them in a directory (e.g. vendor) inside your project. By default, it does not install anything globally. Thus, it is a dependency manager.

The current stable Version is: 2.0.8 2020-12-03.

This idea is not new and Composer is strongly inspired by node's npm and ruby's bundler.

Suppose:

1. You have a project that depends on a number of libraries.
2. Some of those libraries depend on other libraries.

Composer:

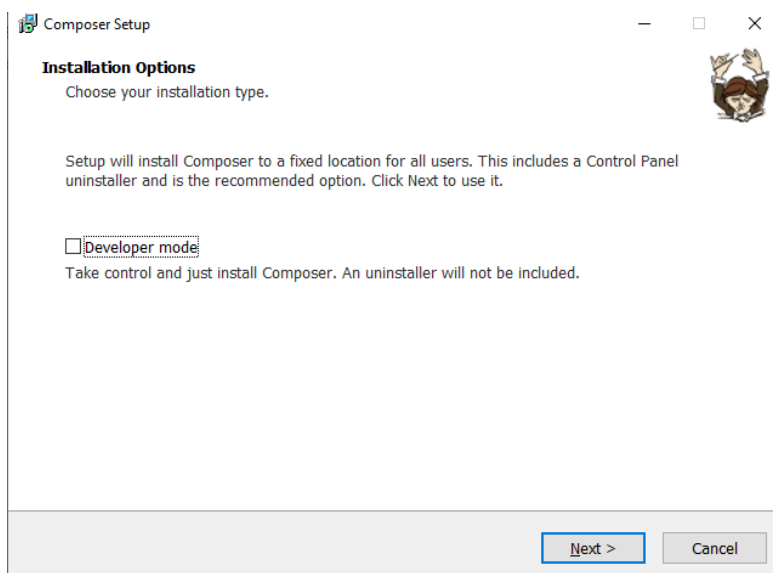
1. Enables you to declare the libraries you depend on.
2. Finds out which versions of which packages can and need to be installed, and installs them (meaning it downloads them into your project).
3. You can update all your dependencies in one command.

Download Composer:

<https://getcomposer.org/download/>

Install the Composer:

Windows:



composer.json

To start using Composer in your project, all you need is a composer.json file. This file describes the dependencies of your project and may contain other metadata as well. It typically should go in the top-most directory of your project/VCS repository. You can technically run Composer anywhere but if you want to publish a package to Packagist.org, it will have to be able to find the file at the top of your VCS repository.

The `require` key#

```
{
  "require":{
    "monolog/monolog":"1.0.*"
  }
}
```

How does Composer download the right files? When you specify a dependency in `composer.json`, Composer first takes the name of the package that you have requested and searches for it in any repositories that you have registered using the `repositories` key. If you have not registered any extra repositories, or it does not find a package with that name in the repositories you have specified, it falls back to Packagist (more [below](#)).

When Composer finds the right package, either in Packagist or in a repo you have specified, it then uses the versioning features of the package's VCS (i.e., branches and tags) to attempt to find the best match for the version constraint you have specified. Be sure to read about versions and package resolution in the [versions article](#).

Note: If you are trying to require a package but Composer throws an error regarding package stability, the version you have specified may not meet your default minimum stability requirements. By default, only stable releases are taken into consideration when searching for valid package versions in your VCS. You might run into this if you are trying to require dev, alpha, beta, or RC versions of a package. Read more about stability flags and the `minimum-stability` key on the [schema page](#).

Laravel Installation:

Step 1 – Visit the following URL and download composer to install it on your system.

<https://getcomposer.org/download/>

Step 2 – After the Composer is installed, check the installation by typing the Composer command in the command prompt as shown in the following screenshot.

Step 3 – Create a new directory anywhere in your system for your new Laravel project. After that, move to path where you have created the new directory and type the following command there to install Laravel.

```
composer create-project laravel/laravel --prefer-dist
```

Step 4 – The above command will install Laravel in the current directory. Start the Laravel service by executing the following command.

```
php artisan serve
```

Step 5 – After executing the above command, you will see a screen as shown below –

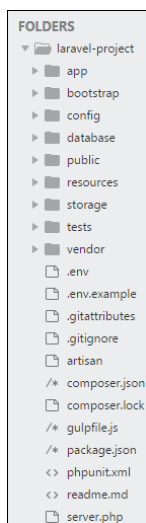
You will get the link with localhost

Step 6 – Copy the URL underlined in gray in the above screenshot and open that URL in the browser. If you see the following screen, it implies Laravel has been installed successfully.

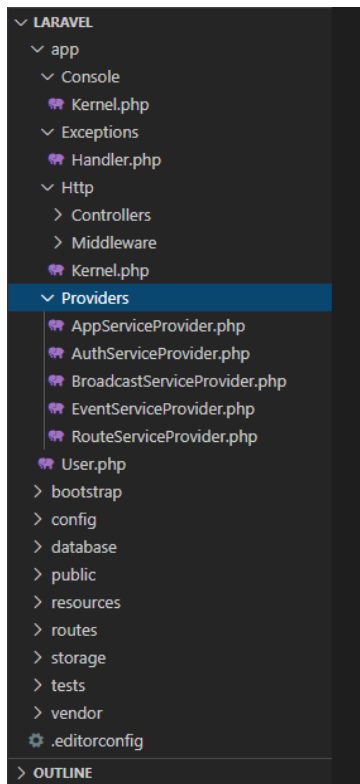
Laravel Architecture

The application structure in Laravel is basically the structure of folders, sub-folders and files included in a project. Once we create a project in Laravel, we get an overview of the application structure as shown in the image here.

The snapshot shown here refers to the root folder of Laravel namely **laravel-project**. It includes various sub-folders and files.



App is application folder and includes entire source code of the project. It contains events, exceptions and middleware declaration. The app folder comprises various sub folders as explained below –



Console

Console includes the artisan commands necessary for Laravel. It includes a directory named Commands, where all the commands are declared with the appropriate signature. The file Kernel.php calls the commands declared in Inspire.php.

If we need to call a specific command in Laravel, then we should make appropriate changes in this directory.

Events

This folder includes all the events for the project.

Events are used to trigger activities, raise errors or necessary validations and provide greater flexibility. Laravel keeps all the events under one directory. The default file included is event.php where all the basic events are declared.

Exceptions

AuthenticationSystem in Laravel 8

<https://laravelarticle.com/laravel-8-authentication-tutorial>

1. Open htdocs→ Type CMD in address bar Enter Laravel new *Project name*
2. Type `cd project namerun php artisan serve`
3. Open the URL <http://127.0.0.1:8080> in browser and check the Larvel 8.0 default page
4. To install Authentication Module:
Install Laravel UI packages:

```
composer require laravel/ui
```

Generate auth scaffolding.

```
php artisan ui:vue --auth
```

Install NPM Dependencies

```
npm install
```

Compile assets

```
npm run dev
```

Test Authentication System

```
example.com/login  
example.com/register
```

If you want you can disable Registration go to web.php route file and change the auth route

```
Auth::routes(['register' => false]);
```

To Create the database table:

```
php artisan make:migrationcreate_news_table
```

Then Open migrate folder and create schema under `public function up()` as below:
You can see id and timestamp already exist can create the below in between those 2

```
$table->string('title');  
    $table->text('description');  
    $table->string('author');  
    $table->date('date');
```

Once the file is updated then run migrate to create table in db

```
php artisan migrate
```

Create a Model:

```
Php artisan make:model News
```

Single Cmd to Create Migration and Model at a time

```
Php artisan make:model News --migration
```

Creating Controller

```
Php artisan make:controller TestController
```

For each and every URL to make it work we have to write it route which is in web.php

This is how you can write it:

```
Route::get('/test', [App\Http\Controllers\TestController::class, 'index'])->  
>name('home');
```

We can have 2 different routes for same function and same controllers

KODIARY Technologies

Integrating with Views with Laravel

Integrating with Admin Panel with Laravel

A project from Scratch:

Prerequisites Installed Composer, Php, MySQL(I prefer Xampp or Amp)

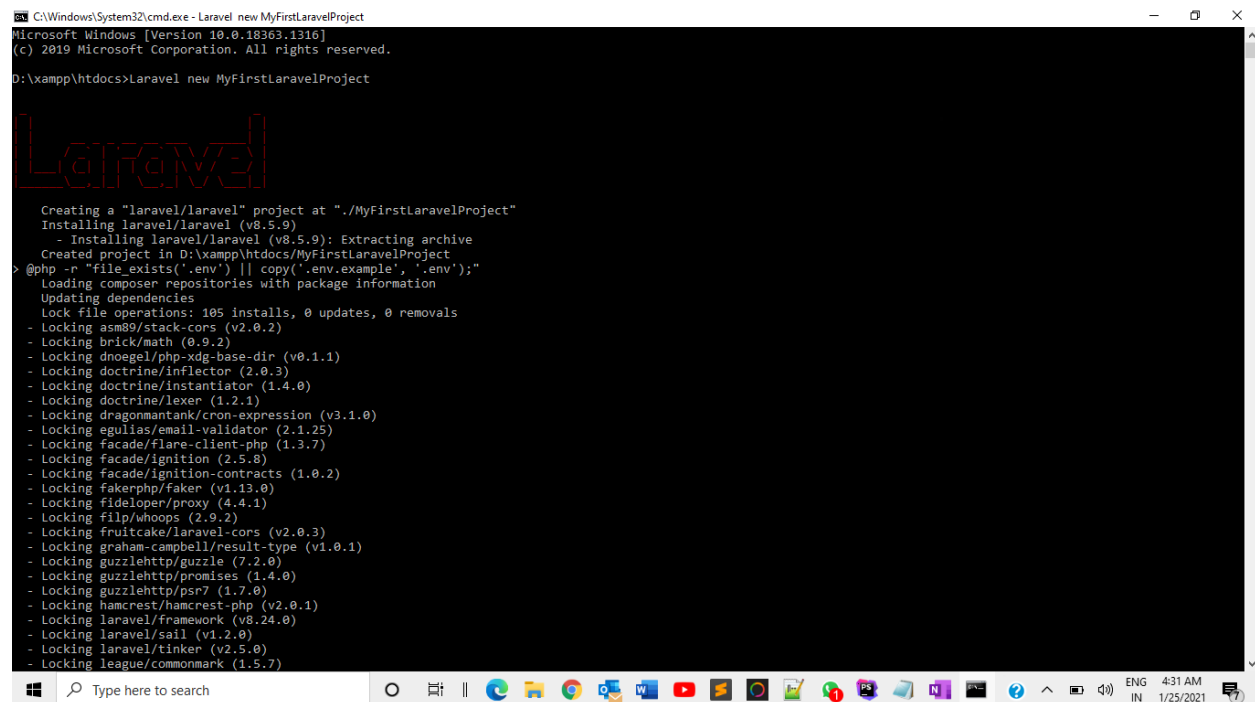
Create a project:

Go to htdocs folder open command prompt by typing cmd in address bar and then type below command

```
Laravel new projectname
```

Example: Laravel new MyFirstLaravelProject

Below is the screen shows after you enter the above command:



```
C:\Windows\System32\cmd.exe - Laravel new MyFirstLaravelProject
Microsoft Windows [Version 10.0.18363.1316]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\xampp\htdocs>Laravel new MyFirstLaravelProject

Laravel

Creating a "laravel/laravel" project at "D:\xampp\htdocs\MyFirstLaravelProject"
Installing laravel/laravel (v8.5.9)
- Installing laravel/laravel (v8.5.9): Extracting archive
Created project in D:\xampp\htdocs\MyFirstLaravelProject
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 105 installs, 0 updates, 0 removals
- Locking asm89/stack-cors (v2.0.2)
- Locking brick/math (0.9.2)
- Locking dnoegel/php-xdg-base-dir (v0.1.1)
- Locking doctrine/infllector (2.0.3)
- Locking doctrine/instantiator (1.4.0)
- Locking doctrine/lexer (1.2.1)
- Locking dragonmantank/cron-expression (v3.1.0)
- Locking egulias/email-validator (2.1.25)
- Locking facade/flare-client-php (1.3.7)
- Locking facade/ignition (2.5.8)
- Locking facade/ignition-contracts (1.0.2)
- Locking fakerphp/faker (v1.13.0)
- Locking fideloper/proxy (4.4.1)
- Locking filp/whoops (2.9.2)
- Locking fruitcake/laravel-cors (v2.0.3)
- Locking graham-campbell/result-type (v1.0.1)
- Locking guzzlehttp/guzzle (7.2.0)
- Locking guzzlehttp/promises (1.4.0)
- Locking guzzlehttp/psr7 (1.7.0)
- Locking hamcrest/hamcrest-php (v2.0.1)
- Locking laravel/framework (v8.24.0)
- Locking laravel/sail (v1.2.0)
- Locking laravel/tinker (v2.5.0)
- Locking league/commonmark (1.5.7)
```

To check the Laravel is installed properly run

First enter the folder by typing:

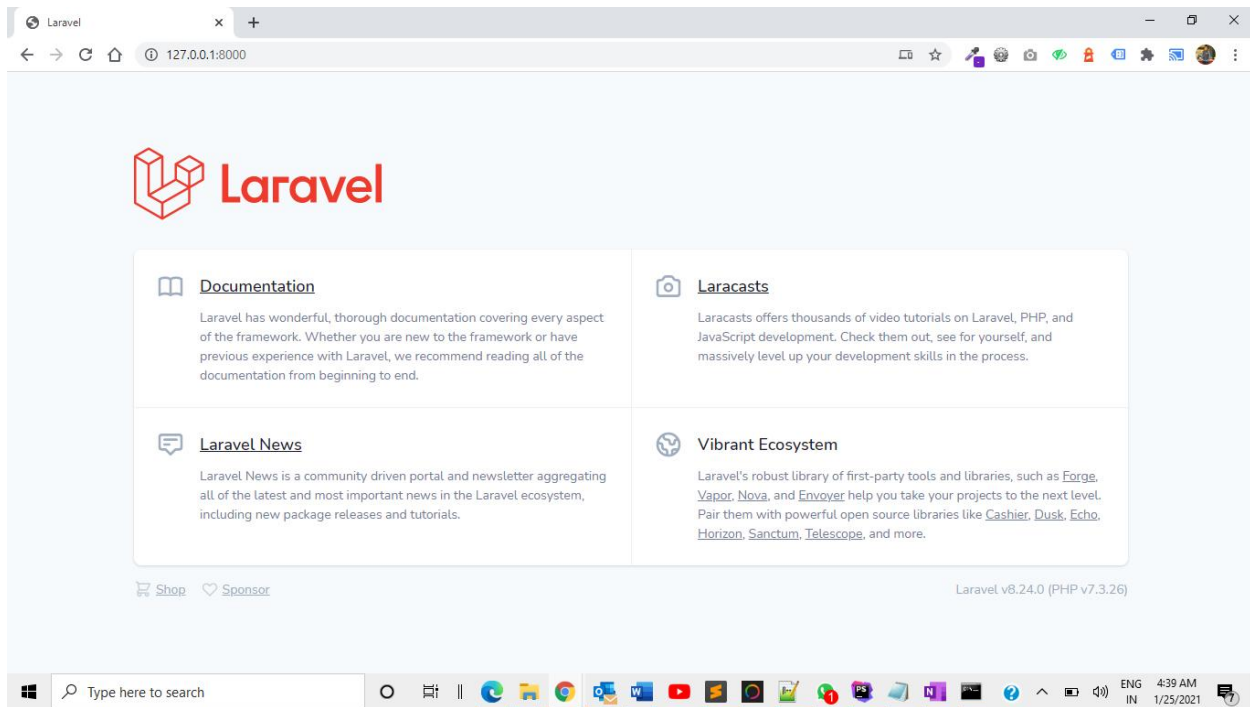
```
cd MyFirstLaravelProject
```

And run the below command.

```
Php artisan serve
```

You can will be asked to open <http://127.0.0.1:8000/> or you can also go to the path <http://localhost/MyFirstLaravelProject/public/>

You can see the below screen:



Login / Registration Module

You can Install Registration / Login module with just 5 commands.

Prerequisites: Create a db with the name **myfirstlaravelproject** in Phpmyadmin

1. Install Laravel UI packages:

```
composer require laravel/ui
```

2. Generate auth scaffolding.

```
php artisan ui:vue --auth
```

3. Install NPM Dependencies.

```
npm install
```

If you get any issue while running the above command run `npm audit fix`

4. Compile assets

```
npm run dev
```

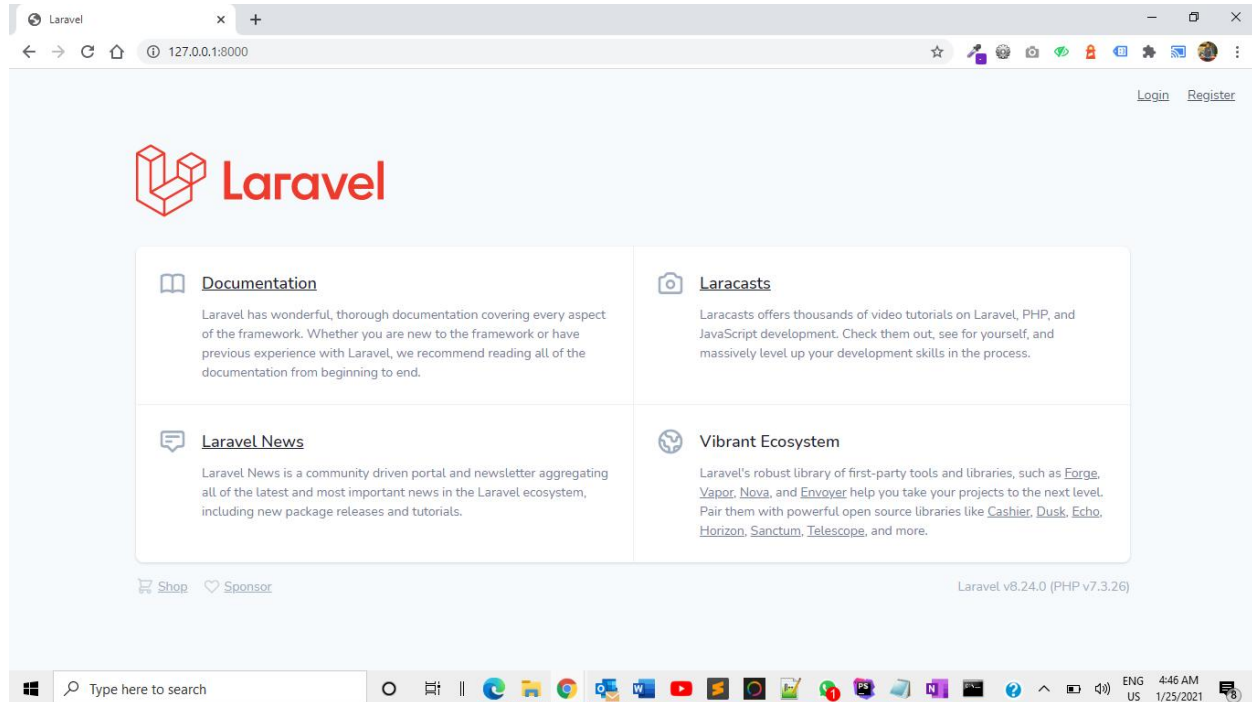
If it prompts "Please run Mix again." Do it with same command `npm run dev`

You will get a notification in windows as Laravel Build Successful.

5. Run migration to create database table for authentication module:

```
Php artisan migrate
```

And you can see below Login Register at top right



Click on Register and Register a user, it will automatically be logged in – Logout and Login with created credentials.

You are done with Authentication Module!

Extras:

If you want you can disable Registration go to web.php route file and change the auth route

```
Auth::routes(['register' => false]);
```

Dashboard

We can download and integrate

Package fzaninotto/faker is abandoned, you should avoid using it. No replacement was suggested.

Package phpunit/php-token-stream is abandoned, you should avoid using it. No replacement was suggested.

