1. Getting started

1.1. Meet Laravel

Laravel is a web application framework with expressive, elegant syntax. A web framework provides a structure and starting point for creating your application, allowing you to focus on creating something amazing while we sweat the details.

Laravel strives to provide an amazing developer experience while providing powerful features such as thorough dependency injection, an expressive database abstraction layer, queues and scheduled jobs, unit and integration testing, and more.

Whether you are new to PHP web frameworks or have years of experience, Laravel is a framework that can grow with you. We'll help you take your first steps as a web developer or give you a boost as you take your expertise to the next level. We can't wait to see what you build.

1.2. Why Laravel?

1.2.1. Best choice

There are a variety of tools and frameworks available to you when building a web application. However, we believe Laravel is the best choice for building modern, full-stack web applications.

1.2.2. A Progressive Framework

We like to call Laravel a "progressive" framework. By that, we mean that Laravel grows with you. If you're just taking your first steps into web development, Laravel's vast library of documentation, guides, and video tutorials will help you learn the ropes without becoming overwhelmed.

If you're a senior developer, Laravel gives you robust tools for dependency injection, unit testing, queues, real-time events, and more. Laravel is fine-tuned for building professional web applications and ready to handle enterprise workloads.

1.2.3. A Scalable Framework

Laravel is incredibly scalable. Thanks to the scaling-friendly nature of PHP and Laravel's built-in support for fast, distributed cache systems like Redis, horizontal scaling with Laravel is a breeze. In fact, Laravel applications have been easily scaled to handle hundreds of millions of requests per month.

Need extreme scaling? Platforms like Laravel Vapor allow you to run your Laravel application at nearly limitless scale on AWS's latest serverless technology.

1.2.4. A Community Framework

Laravel combines the best packages in the PHP ecosystem to offer the most robust and developer friendly framework available. In addition, thousands of talented developers from around the world have contributed to the framework. Who knows, maybe you'll even become a Laravel contributor.

1.3. Your First Laravel Project

1.3.1. Axtyng

Before creating your first Laravel project, you should ensure that your local machine has PHP and Composer installed. If you are developing on macOS, PHP and Composer can be installed via Homebrew. In addition, we recommend installing Node and NPM.

1.3.2. New project

After you have installed PHP and Composer, you may create a new Laravel project via the Composer create-project command:

{code-section}

composer create-project Laravel/Laravel example-app

{/code-section}

Or, you may create new Laravel projects by globally installing the Laravel installer via Composer:

{code-section}

composer global require Laravel/installer

Laravel new example-app

{/code-section}

1.3.3. Start local development

After the project has been created, start Laravel's local development server using the Laravel's Artisan CLI serve command:

{code-section}

cd example-app

php artisan serve

{/code-section}

1.3.4. Started development server

Once you have started the Artisan development server, your application will be accessible in your web browser at http://localhost:8000. Next, you're ready to start taking your next steps into the Laravel ecosystem. Of course, you may also want to configure a database.

1.3.5. Started kits

If you would like a head start when developing your Laravel application, consider using one of our starter kits. Laravel's starter kits provide backend and frontend authentication scaffolding for your new Laravel application

2. Laravel & Docker

2.1. Interface and containers

2.1.1. Easy use

We want it to be as easy as possible to get started with Laravel regardless of your preferred operating system. So, there are a variety of options for developing and running a Laravel project on your local machine. While you may wish to explore these options at a later time, Laravel provides Sail, a built-in solution for running your Laravel project using Docker.

2.1.2. Docker

Docker is a tool for running applications and services in small, light-weight "containers" which do not interfere with your local machine's installed software or configuration. This means you don't have to worry about configuring or setting up complicated development tools such as web servers and databases on your local machine. To get started, you only need to install Docker Desktop.

2.1.3. Laravel Sail

Laravel Sail is a light-weight command-line interface for interacting with Laravel's default Docker configuration. Sail provides a great starting point for building a Laravel application using PHP, MySQL, and Redis without requiring prior Docker experience.

2.2. Getting Started On macOS

2.2.1. Fast Mac creates

If you're developing on a Mac and Docker Desktop is already installed, you can use a simple terminal command to create a new Laravel project.

For example, to create a new Laravel application in a directory named "example-app", you may run the following command in your terminal:

{code-section}

curl -s "https://laravel.build/example-app" | bash

{/code-section}

Of course, you can change "example-app" in this URL to anything you like - just make sure the application name only contains alpha-numeric characters, dashes, and underscores. The Laravel application's directory will be created within the directory you execute the command from.

2.2.2. Sail installation Mac

Sail installation may take several minutes while Sail's application containers are built on your local machine.

After the project has been created, you can navigate to the application directory and start Laravel Sail. Laravel Sail provides a simple command-line interface for interacting with Laravel's default Docker configuration:

{code-section}

cd example-app

./vendor/bin/sail up

{/code-section}

Once the application's Docker containers have been started, you can access the application in your web browser at: <http://localhost>.

2.3. Getting Started on Windows

2.3.1. WSL2

Before we create a new Laravel application on your Windows machine, make sure to install Docker Desktop. Next, you should ensure that Windows Subsystem for Linux 2 (WSL2) is installed and enabled. WSL allows you to run Linux binary executables natively on Windows 10. Information on how to install and enable WSL2 can be found within Microsoft's developer environment documentation.

After installing and enabling WSL2, you should ensure that Docker Desktop is configured to use the WSL2 backend.

2.3.2. First Windows project

Next, you are ready to create your first Laravel project. Launch Windows Terminal and begin a new terminal session for your WSL2 Linux operating system. Next, you can use a simple terminal command to create a new Laravel project. For example, to create a new Laravel application in a directory named "example-app", you may run the following command in your terminal:

{code-section}

curl -s https://laravel.build/example-app | bash

{/code-section}

Of course, you can change "example-app" in this URL to anything you like - just make sure the application name only contains alpha-numeric characters, dashes, and underscores. The Laravel application's directory will be created within the directory you execute the command from.

2.3.3. Sail installation Windows

Sail installation may take several minutes while Sail's application containers are built on your local machine.

After the project has been created, you can navigate to the application directory and start Laravel Sail. Laravel Sail provides a simple command-line interface for interacting with Laravel's default Docker configuration:

{code-section}

cd example-app

./vendor/bin/sail up

{/code-section}

Once the application's Docker containers have been started, you can access the application in your web browser at: <http://localhost>.

2.3.4. Developing Within WSL2

Of course, you will need to be able to modify the Laravel application files that were created within your WSL2 installation. To accomplish this, we recommend using Microsoft's Visual Studio Code editor and their first-party extension for Remote Development.

Once these tools are installed, you may open any Laravel project by executing the code. command from your application's root directory using Windows Terminal.

2.4. Getting Started on Linux

2.4.1. First create Linux

If you're developing on Linux and Docker Compose is already installed, you can use a simple terminal command to create a new Laravel project. For example, to create a new Laravel application in a directory named "example-app", you may run the following command in your terminal:

{code-section}

curl -s https://laravel.build/example-app | bash

{/code-section}

Of course, you can change "example-app" in this URL to anything you like - just make sure the application name only contains alpha-numeric characters, dashes, and underscores. The Laravel application's directory will be created within the directory you execute the command from.

2.4.2. Sail installation Linux

Sail installation may take several minutes while Sail's application containers are built on your local machine.

After the project has been created, you can navigate to the application directory and start Laravel Sail. Laravel Sail provides a simple command-line interface for interacting with Laravel's default Docker configuration:

{code-section}

cd example-app

./vendor/bin/sail up

{/code-section}

Once the application's Docker containers have been started, you can access the application in your web browser at: http://localhost.