Introduction to Laravel

The PHP Framework for Web Artisans

Laravel is a web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you to create without sweating the small things.

Laravel is a powerful and elegant PHP web application framework that streamlines web application development through an **MVC (Model-View-Controller) architecture**.

Laravel has gained immense popularity due to its simplicity and ease of use, making it a top choice for web developers worldwide.

Is Laravel Frontend or Backend?
Laravel has both backend development framework tools and also provides frontend functionality.

Is Laravel a Programming Language?
Laravel **is a PHP framework** and uses a scripting language rather than being a strict PHP programming language. While scripting languages and programming languages are related, they have several noticeable differences, primarily in ease of use and speed of execution.

What Is a PHP Framework?
PHP (a recursive acronym for PHP Hypertext Preprocessor) is an open source, server-side scripting language widely used for web development. As of early 2021, nearly 80% of all websites are using PHP.

A PHP framework provides a set of code libraries containing pre-programmed modules that allow a user to build applications faster.
They offer web developers a number of benefits including

- more rapid development,
- a reduced need to write code, and
- enhanced security.
- They also help novice developers build up good coding practices, since they require specific organization of code.
- PHP frameworks typically require less maintenance than applications built from the ground up.

Many modern PHP frameworks are object-oriented. Because of this, it’s beneficial to have a basic understanding of concepts like classes, objects, and inheritance before diving into a framework.

Model-View-Controller (MVC)

The Model-View-Controller (MVC) architecture is a widely adopted design pattern in web development, and Laravel, a popular PHP framework, follows this pattern closely. Understanding MVC is essential for building scalable and maintainable web applications. In this article, we will explore the MVC architecture in Laravel and delve into the roles and interactions of its three core components: the Model, View, and Controller.

1. Model: Managing Data and Business Logic
   The Model represents the data and business logic of an application. In Laravel, the Model is responsible for interacting with the database and encapsulating data-related operations. It serves as an intermediary between the application and the database, providing an object-oriented interface to work with data.
   Laravel’s Eloquent ORM (Object-Relational Mapping) simplifies database interactions within the Model. By defining Eloquent models, you can easily perform CRUD (Create, Read, Update, Delete) operations on database records. Eloquent models also enable you to define relationships between different models, making it easier to handle complex data associations.

2. View: Rendering User Interfaces

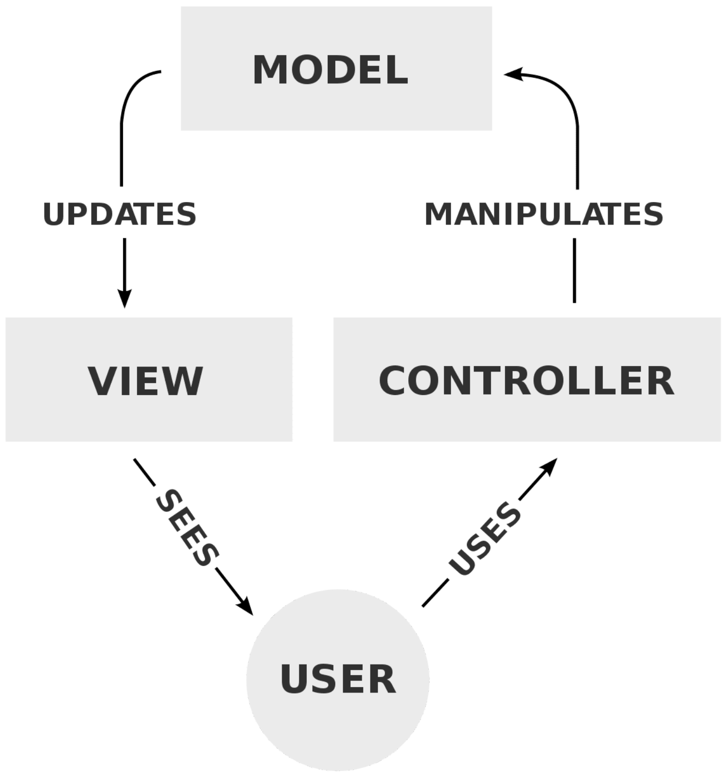
   The View represents the user interface (UI) of the application. It is responsible for presenting data to the user and receiving user input. In Laravel, Views are typically written using the Blade templating engine, which provides a clean and concise syntax for creating dynamic and reusable UI components.

   Views in Laravel are separate files that contain HTML markup along with embedded PHP code. They can receive data from the Controller and display it using the Blade syntax. Views also support conditionals, loops, and partials, making it easy to build modular and interactive UIs.

3. Controller: Handling User Requests

   The Controller acts as an intermediary between the Model and the View. It receives and processes user requests, interacts with the Model to retrieve or modify data, and passes the processed data to the appropriate View for rendering.

   In Laravel, Controllers are responsible for defining methods, or actions, that correspond to different user requests. These actions encapsulate the logic necessary to handle specific requests and manage the flow of data. Controllers also facilitate the validation of user input and perform any required authorization checks before interacting with the Model.



The MVC Workflow in Laravel

1. User request: When a user interacts with your Laravel application, a request is sent to the server.
2. Routing: Laravel's routing system directs the incoming request to the appropriate Controller method based on the defined routes.
3. Controller: The Controller method receives the request and performs any necessary data retrieval, validation, and manipulation. It interacts with the Model to fetch or update data.
4. Model: The Model handles database interactions, such as querying or updating records, based on the instructions from the Controller.
5. Data preparation: The processed data is then passed from the Controller to the appropriate View.
6. View: The View receives the data and uses the Blade templating engine to render the final HTML output, which is sent back to the user's browser.
7. User response: The rendered HTML, along with any associated styles and scripts, is displayed to the user, completing the request-response cycle.

Benefits of MVC in Laravel

The MVC architecture in Laravel provides several benefits for web application development:

1. Separation of concerns: MVC promotes the separation of application logic, presentation, and data manipulation. This separation makes the codebase more organized, maintainable, and reusable.
2. Code reusability: Views and Controllers can be reused across different parts of the application, reducing code duplication and improving development efficiency.
3. Collaboration and scalability: The clear separation of responsibilities in MVC allows multiple developers to work on different components simultaneously, making it easier to scale and maintain the application.

PHP applications, such as WordPress, have some known vulnerabilities. The most notable examples are code injection and SQL injection. Laravel includes features that help prevent SQL injection and other attacks.

However, developers should undertake additional efforts, such as **penetration testing**, to ensure the security of their applications.

Penetration tests are intended to be more complex tests performed by qualified cybersecurity professionals, who will attempt to break into your system without any prior knowledge of its development in order to identify unpatched security vulnerabilities.

Key Features and Functionality

Some of the key features and functionality offered by Laravel include:

Eloquent ORM For Database Interactions
Laravel's **Object-Relational Mapping (ORM)** allows for smooth and intuitive interactions with your database, making it easy to perform CRUD (Create, Read, Update, Delete) operations and manage relationships between tables.

Blade Templating Engine
Blade is Laravel's simple yet powerful templating engine, allowing you to create **dynamic and reusable HTML templates** with ease. With its concise syntax, Blade makes it easy to separate your application's logic from its presentation.

Artisan Command-Line Tool
Laravel's Artisan command-line tool offers a range of helpful commands for common tasks, such as generating controllers, migrations, and more. This speeds up the development process and helps you maintain a clean, organized codebase.

Built-in Support For Tasks Scheduling and Authentication
Laravel offers built-in support for task scheduling, making it easy to automate tasks like sending emails or cleaning up old data. Additionally, Laravel's authentication system simplifies the process of adding secure user authentication to your web applications.

Who Uses Laravel?
Laravel is a popular choice among various types of users, each of whom finds different benefits in using this powerful PHP framework. Some of the key user groups include:

- **Startups**: For startups, Laravel offers a **rapid development process**, which helps bring their ideas to life quickly and efficiently. The framework's built-in tools and libraries save precious time and resources during the initial stages of a project.
- **Established companies**: Laravel's scalability and maintainability make it an attractive choice for established companies looking to build or upgrade their web applications. The framework's **strong community support** ensures businesses can rely on Laravel for long-term projects.
- **PHP web developers:** Developers appreciate Laravel's elegant syntax and ease of use, which streamline the web application development process. Laravel's features allow for efficient and effective coding practices.
- **Backend engineers:** For backend engineers, Laravel's powerful features, including its support for t**ask scheduling, authentication, and the Artisan command-line tool**, make it an essential tool in their toolkit. Laravel enables backend engineers to create **robust and secure web applications** with ease.

Now, here are some examples of companies that use Laravel in their projects:

Bankrate

Bankrate is a leading financial services company that uses Laravel to build and maintain its online applications. The framework helps Bankrate manage complex data structures and speed up development cycles.

The New York Times

The **New York Times** uses Laravel for its internal web applications, helping the company to build and maintain complex web applications quickly and effectively.

St Jude Children's Research Hospital

St Jude Children's Research Hospital uses Laravel to power its online donation platform and other web applications. The framework helps the hospital efficiently manage large amounts of data and ensure secure user authentication.

Geocodio

Geocodio is a geocoding service that converts addresses to latitude and longitude coordinates and vice versa. Laravel powers Geocodio’s web application, allowing it to provide its customers a reliable and efficient geocoding service.

October CMS

October CMS is a free, open-source content management system (CMS) built on top of Laravel. By leveraging the power and flexibility of the Laravel framework, October CMS provides a user-friendly and customizable solution for managing website content.

As you can see, Laravel is a powerful framework used by companies of all sizes, **from startups to large enterprises.** But you still might not be sure about the benefits this framework offers to developers. Let’s explore that now.

Advantages of Using Laravel

Laravel offers a number of advantages that make it a popular choice among web developers:

Rapid Development Process

Laravel provides intuitive syntax, built-in tools, and extensive documentation, enabling developers to create web applications quickly and efficiently, significantly reducing development time.

Readable and Maintainable Code

Laravel promotes clean and well-structured code, making it easier to read and maintain. This is particularly beneficial when working in a team or on large projects, as it helps ensure consistency and code quality.

Strong Community and Support

Laravel has a large and active community of developers who contribute to its growth and provide support through forums, blogs, and social media. This makes it easy to find help and resources when needed.

Built-In Tools and Libraries

Laravel comes with many built-in tools and libraries, such as **Eloquent ORM**, **Blade templating engine**, and **Artisan command-line tool**, which simplify the development process and reduce the need for external dependencies.

Laravel Prerequisites

Before diving into Laravel, there are a few things you should be familiar with:

- Basic knowledge of PHP programming
- Understanding of the MVC (Model-View-Controller) architectural pattern
- Familiarity with command line/terminal
- Experience with HTML, CSS, and JavaScript

System Requirements

Ensure your system meets the following requirements:

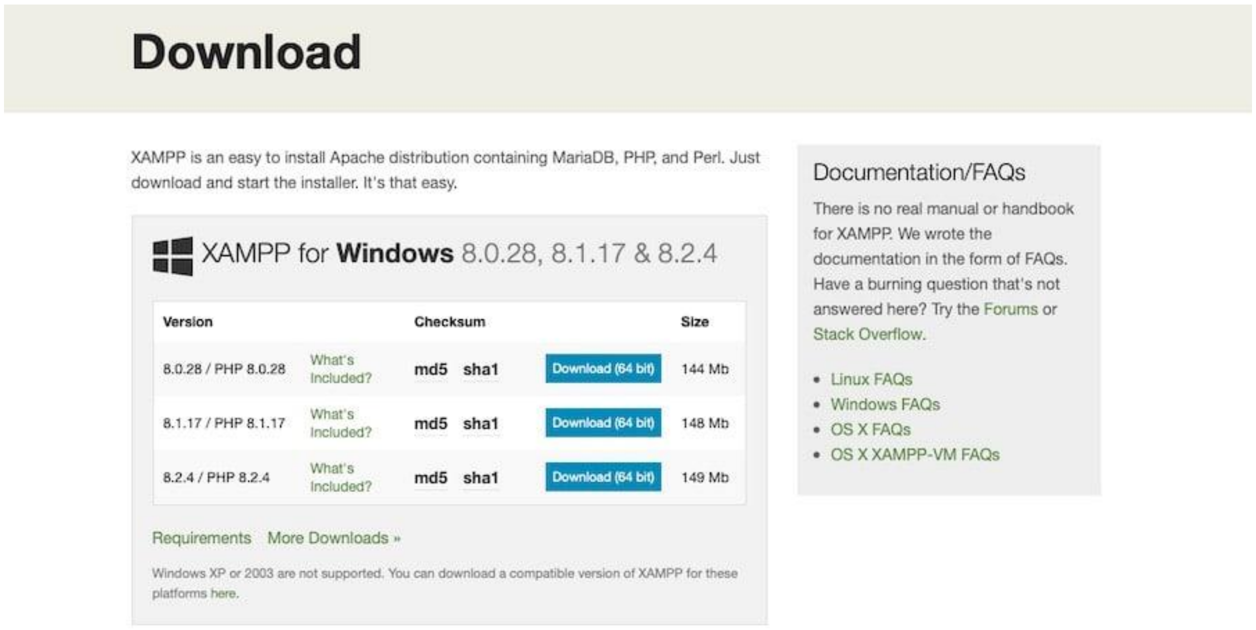
- Supported operating systems: Windows, macOS, Linux
- Hardware requirements: Minimal (dependent on PHP and Composer)
- PHP installation: Required before installing Laravel
- Composer installation: Required for Laravel installation

How To Install Laravel

Once you’ve familiarized yourself with Laravel and the system requirements, you can begin installing it. The installation process is straightforward and consists of straightforward steps but they do vary from operating system to operating system.

1. Install XAMPP

XAMPP is a free and open-source web server solution stack that includes PHP, MySQL, and Apache. To install XAMPP on Windows, follow these steps:



Download

XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

XAMPP for **Windows** 8.0.28, 8.1.17 & 8.2.4

Version		Checksum			Size
8.0.28 / PHP 8.0.28	What's Included?	md5	sha1	Download (64 bit)	144 Mb
8.1.17 / PHP 8.1.17	What's Included?	md5	sha1	Download (64 bit)	148 Mb
8.2.4 / PHP 8.2.4	What's Included?	md5	sha1	Download (64 bit)	149 Mb

Requirements   More Downloads »

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms here.

Documentation/FAQs

There is no real manual or handbook for XAMPP. We wrote the documentation in the form of FAQs. Have a burning question that's not answered here? Try the Forums or Stack Overflow.

- Linux FAQs
- Windows FAQs
- OS X FAQs
- OS X XAMPP-VM FAQs

The XAMPP download page for Windows.

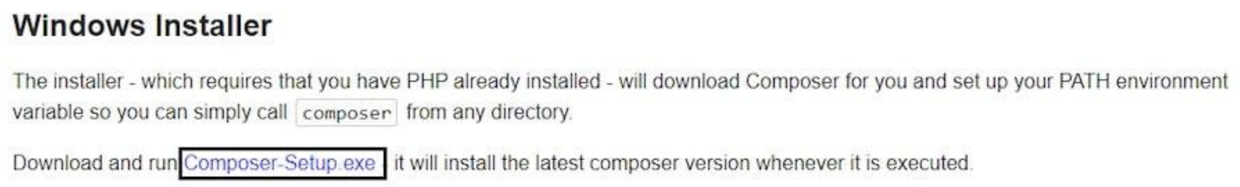
Visit the XAMPP download page and download the appropriate installer for your Windows version.

Run the installer and follow the on-screen instructions. During the installation process, you can choose which components to install. Make sure to select PHP and MySQL.

Once installed, launch the XAMPP Control Panel and start the Apache and MySQL services.

2. Install Composer

Composer is a dependency management tool for PHP that is required to install Laravel. To install Composer on Windows, follow these steps:
Visit the Composer download page and download the Composer-Setup.exe file.


Windows Installer

The installer - which requires that you have PHP already installed - will download Composer for you and set up your PATH environment variable so you can simply call `composer` from any directory.

Download and run Composer-Setup.exe it will install the latest composer version whenever it is executed.

Composer for Windows download page.

Run the Composer-Setup.exe file and follow the on-screen instructions.

You’ll also be prompted to select the install mode. Be sure to pick Install for all users (recommended).

Make sure to select the correct PHP executable during the installation process (usually located in the XAMPP installation folder under **xampp/php/php.exe**).

Click Next to move through the on-screen instructions then click Install.

Once installation is completed, click Finish.

3. Verify Composer Installation

To verify that Composer was installed correctly, open the Command Prompt and run the following command:

```
composer --version
```

If the installation was successful, you should see the Composer version displayed.

4. Install Laravel Using Composer

You can use Composer, which is now installed, to install Laravel globally on your system. To do so, open the Command Prompt and run the given command:

```
composer create-project laravel/laravel app-name
```

This will automatically download all the relevant Laravel files to create a new project.

5. Verify Laravel Installation

To verify that Laravel was installed correctly, open the Command Prompt and run the following command:

```
laravel --version
```

After a successful installation, you will be able to see the Laravel version.

6. Start The Server

With your new app project created, you will then need to start a server. To do this, type in the following:

```
cd app-name
php artisan serve
OR
php artisan serve --host=0.0.0.0 --port=8888
```

7. Run The Project In Your Browser

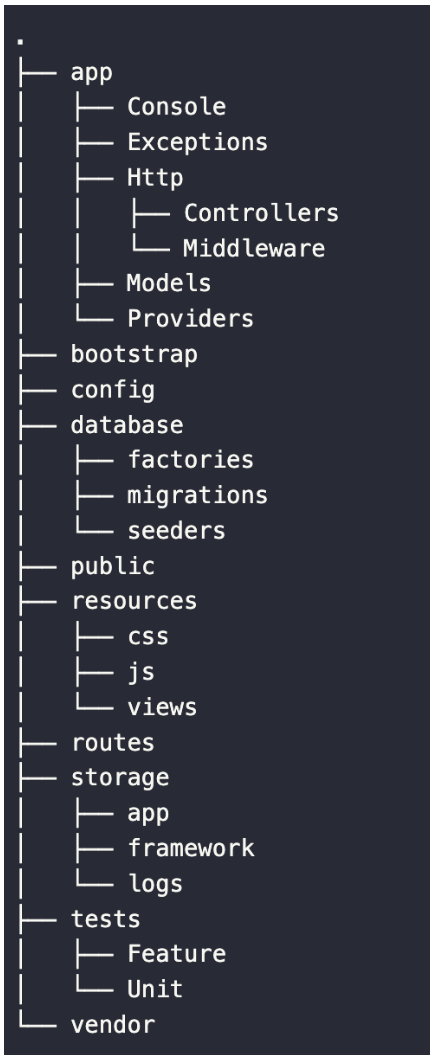
With the server started you should then be able to access your app project via your web browser. To do this, open your browser and go to the following:

```
https://localhost:8000
ORR
http://0.0.0.0:8888
```

With this, you can start developing web applications using Laravel on your Windows machine.

Exploring Laravel application structure

Before we start coding, let’s look at what has been created inside our project. Here is an overview of the project root directory.The app directory: This directory is the core component of our project, most importantly the controllers, middleware, and models. The controller defines the core logic of the app, the middleware defines what actions should be taken before the controller is called, and the model provides an interface allowing us to deal with databases. We’ll discuss each of them in detail later.


```
.
├── app
│   ├── Console
│   ├── Exceptions
│   ├── Http
│   │   ├── Controllers
│   │   └── Middleware
│   ├── Models
│   └── Providers
├── bootstrap
├── config
├── database
│   ├── factories
│   ├── migrations
│   └── seeders
├── public
├── resources
│   ├── css
│   ├── js
│   └── views
├── routes
├── storage
│   ├── app
│   ├── framework
│   └── logs
├── tests
│   ├── Feature
│   └── Unit
└── vendor
```

- **The bootstrap directory**: This directory contains the app.php file which bootstraps the entire project. You don’t need to modify anything in this directory.
- **The config directory**: As the name suggests, it contains the configuration files. We don’t need to care about these configurations in this tutorial.
- **The database directory:** Contains migration files, factories and seeds. The migration files describes the structure of the database. The factories and seeds are two different ways we can fill the database with dummy data in Laravel.
- **The public directory:** Contains index.php, which is the entry point of our app.
- **The resources directory**: Contains view files, which is the frontend part of a Laravel application.
- **The routes directory:** Contains all URL routers for our project.

- **The storage directory:** A storage space for the entire project. Contains logs, compiled views, as well as user uploaded files.
- **The tests directory:** Contains test files. Testing is a relatively more advanced subject in Laravel, so we are not going to cover it in this tutorial, but you may check the linked article for more details on testing in Laravel .
- **The vendor directory:** Includes all dependencies.

Environmental variables

We also need to take a closer look at the environmental variables. All the environmental variables are stored in the `.env` file. Alot of configurations are set by default, but we still need to talk about what they do.

App URL

The APP_URL variable defines the URL of the application. By default, it is `http://localhost`, but you might need to change it to `http://127.0.0.1` if `http://localhost` is giving you the apache2 default page.

```
APP_URL=http://127.0.0.1

APP_URL=http://localhost
```

Database

By default, Laravel Sail will installed MySQL as the database application, and the database connections are defined as follows:

```
DB_CONNECTION=mysql
DB_HOST=mysql
DB_PORT=3306
DB_DATABASE=curl_demo
DB_USERNAME=sail
DB_PASSWORD=password
```

If you need to, you may also define your own custom environmental variables, and then you can access them anywhere in this project.

```
CUSTOM_VARIABLE=true
```

This variable can be accessed like this:

```
    env('CUSTOM_VARIABLE', true)
```

The second parameter is the default value, if CUSTOM_VARIABLE does not exist, then the second parameter will be returned instead.