Laravel Guide

Adegoke Akintoye

Laravel Guide

Adegoke Akintoye

First edition. May 6, 2024.

Juri Books (email: call.juri@outlook.com tel: +2349012885870)

Disclaimer

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Other Books By The Author:

- Mastering JavaScript Array Methods: A Beginner's Guide to Simplifying Array Manipulation
- Mastering JavaScript String Methods: A Beginner's Guide to Simplifying String Manipulation
- Mastering Coding Test: 50 Problems with Solutions
- Mastering Design Patterns in TypeScript: An Approachable Guide
- Object-Oriented Programming In TypeScript: A Beginner's Guide
- Mastering TypeScript: A Beginner’s Guide
- JavaScript: The Ultimate Guide to Interview Questions
- Integrating HTMX with Laravel: An Approachable Guide
- Object-Oriented Programming in PHP:An Approachable Guide
- Functional Programming in TypeScript: An Approachable Guide
- Laravel Guide
- Mastering API Development with Laravel
- HTMX Guide
- Lumen Illuminated
- Easy, Fast, and Practical PWA

Table of Contents

Introduction

Welcome to your journey into the world of Laravel, a journey that will take you from the foundational principles of web development to the advanced techniques used by professionals to build robust, scalable applications. Laravel, known for its elegance, simplicity, and readability, has emerged as one of the most popular PHP frameworks, empowering developers to craft their ideas into reality efficiently.

This book is designed to be your companion as you explore Laravel, whether you're just starting out or looking to enhance your existing skills. We'll start with the basics, setting up your development environment and understanding the core concepts that make Laravel stand out.

Our aim is not just to teach you how to code in Laravel but to help you think like a Laravel developer. By the end of this book, you'll have a solid understanding of the framework, equipped with the skills to tackle real-world challenges and the confidence to explore the vast ecosystem that Laravel offers. Whether you're building a personal blog, a dynamic web application, or contributing to the Laravel community, the knowledge you gain here will be a valuable asset on your development journey.

Let's embark on this exciting adventure together, turning your ideas into reality with Laravel.

Chapter 1: Getting Started with Laravel

This chapter will guide you through installing Laravel, understanding its folder structure, and setting up a basic project. By the end of this chapter, you'll have a solid foundation to start building your web applications with Laravel.

Laravel Installation

Prerequisites:

- PHP (version 7.3 or higher)
- Composer (Dependency Manager for PHP)

Installing Laravel: Laravel utilizes Composer to manage its dependencies. So, before installing Laravel, make sure you have Composer installed on your machine. Once Composer is installed, you can create a new Laravel project using the following command:

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

This command creates a new directory named `myLaravelApp` with a fresh Laravel installation inside it. It might take a few minutes as Composer downloads the necessary files.

Understanding the Laravel Folder Structure

Once Laravel is installed, you'll notice several directories and files within your project. Here's a brief overview of the key directories:

- `app/`: This directory contains the core code of your application. It includes models, controllers, and other PHP classes.
- `bootstrap/`: Contains files that bootstrap the framework and configure autoloading.
- `config/`: Houses all of your application's configuration files.
- `database/`: This directory contains your database migrations and seeds.

- `public/`: The `public/` directory contains the `index.php` file, which is the entry point for all requests entering your application. This directory also houses your assets like images, JavaScript, and CSS.
- `resources/`: Contains your views as well as your raw, uncompiled assets such as LESS, SASS, or JavaScript.
- `routes/`: All of the route definitions for your application are located in this directory.
- `storage/`: Compiled Blade templates, file-based sessions, file caches, and other files generated by the framework are stored here.
- `tests/`: Contains your automated tests. PHPUnit is set up by default.
- `vendor/`: Composer dependencies are installed here.

Setting Up a Basic Project

Let's set up a basic route and view to make sure our Laravel installation is working correctly.

1. **Routes**: Open the `routes/web.php` file. This file contains all the web routes for your application. Add the following code to define a new route:

```
Route::get('/', function () {
    return view('welcome');
});
```

This code defines a route for the application's root URL (`/`) and returns the `welcome` view.

2. **Views**: The `welcome` view referred to in the route is located at `resources/views/welcome.blade.php`. Open this file, and you'll see the default Laravel welcome page HTML. You can modify this file to customize the welcome page.
3. **Running Your Application**: To see your application in action, you need to start the Laravel development server. Run the following command in your terminal:

```
php artisan serve
```

This command starts a development server at `http://localhost:8000`. Open this URL in your web browser, and you should see the Laravel welcome page.

Conclusion

Congratulations! You've successfully installed Laravel, explored its folder structure, and set up a basic project. This chapter has laid the groundwork for you to dive deeper into Laravel's features and start building your web applications. Laravel's elegant syntax and powerful tools make web development an enjoyable experience.

Chapter 1: The Basics of Laravel

In this chapter, we'll dive into the foundational concepts of Laravel, a powerful and elegant PHP framework that simplifies the development of web applications. By the end of this chapter, you'll understand the MVC architecture, routing, and middleware, and you'll have built a simple blog with static pages. Let's break these concepts down into manageable parts, with clear examples and explanations.

MVC Architecture

What is MVC? MVC stands for Model-View-Controller. It's a design pattern that separates the logic of your application into three interconnected parts. This separation helps manage complex applications, because you can focus on one aspect at a time without worrying about the rest.

- **Model**: Represents the data and business logic. It's where your database queries will live.
- **View**: The presentation layer. This is what your users will see—it's the HTML and CSS that make up your webpage.
- **Controller**: Acts as an intermediary between Models and Views. It handles the user's requests, interacts with the Model, and decides which View to render.

Example: Imagine you're building a simple blog. The Model retrieves the blog posts from the database, the View displays them in a list, and the Controller handles the request to "show all blog posts" and uses the Model to retrieve the posts and the View to display them.

Routing

Routing in Laravel is incredibly powerful and flexible. It allows you to map URLs to specific controller actions or closures, making it easy to define how your application responds to user requests.

Basic Route: A route can be defined in the `routes/web.php` file. Here's how you can define a route to return a simple "Hello, World!" message:

```
Route::get('/', function () {
    return 'Hello, World!';
```

```
});
```

Route to Controller: Instead of defining the logic directly in the route closure, you can point a route to a controller action. First, let's create a controller:

```
php artisan make:controller BlogController
```

Then, define a method in `BlogController`:

```
public function index()
{
    return view('welcome');
}
```

And finally, update your route to use this controller action:

```
Route::get('/', 'BlogController@index');
```

Middleware

Middleware provides a convenient mechanism for filtering HTTP requests entering your application. For example, Laravel includes a middleware that verifies the user is authenticated. If not, it redirects the user to the login screen.

Creating Middleware: Let's say you want to create a middleware that checks if the user's name is "Admin".

First, create the middleware:

```
php artisan make:middleware CheckName
```

Then, in your newly created middleware (`app/Http/Middleware/CheckName.php`), add your logic:

```
public function handle($request, Closure $next)
{
    if ($request->name !== 'Admin') {
        return redirect('home');
    }

    return $next($request);
}
```

Registering Middleware: To use this middleware, you must register it in the `app/Http/Kernel.php` file. You can add it to the global middleware array or assign it to a specific route.

Project: Simple Blog with Static Pages

Setting Up: Ensure you have Laravel installed and create a new project:

```
laravel new simple-blog
```

Creating Views: In `resources/views`, create a new file `welcome.blade.php`. This will be our blog's homepage.

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple Blog</title>
</head>
<body>
    <h1>Welcome to My Blog</h1>
</body>
</html>
```

Defining Routes: Use the route to the `BlogController@index` method we defined earlier to display this view.

Conclusion: You've just created a basic structure for a Laravel application, understanding MVC, routing, and middleware along the way. This foundation will serve you well as we continue to build more complex features.

In the next chapter, we'll explore how to work with databases in Laravel, introducing migrations, Eloquent ORM, and more.

Appendix

In this appendix, we'll provide a comprehensive list of tools, resources, and additional information to support your Laravel development journey. Whether you're just starting out or looking to expand your knowledge, these resources will help you work more efficiently and effectively with Laravel.

Development Tools

- **Composer**: The PHP dependency manager, essential for managing Laravel's libraries and packages. Get Composer
- **Laravel Installer**: A CLI tool for creating new Laravel projects. Install it globally via Composer to quickly start new projects. `composer global require laravel/installer`
- **Valet** (Mac): A Laravel development environment for Mac minimalists. No Vagrant, no `/etc/hosts` file. Laravel Valet
- **Homestead**: The official, pre-packaged Vagrant box that provides a wonderful development environment without requiring you to install PHP, a web server, or any other server software on your local machine. Laravel Homestead
- **Envoyer**: A zero-downtime deployment tool for Laravel, ensuring your users experience no service interruption. Laravel Envoyer
- **Forge**: A server management and site deployment tool, making it easy to launch and manage Laravel applications. Laravel Forge
- **Horizon**: A beautiful dashboard and configuration system for Laravel-powered Redis queues. Laravel Horizon
- **Telescope**: An elegant debug assistant for the Laravel framework, providing insight into requests, exceptions, database queries, and more. Laravel Telescope
- **Nova**: A beautifully designed administration panel for Laravel, crafted by the creators of Laravel. Laravel Nova

Learning Resources

- **Laravel Documentation**: The official Laravel documentation is an excellent starting point and reference for development. Laravel Docs

- **Laravel News**: A portal and newsletter offering the latest news, tutorials, package development, and more. Laravel News
- **Laravel Podcast**: Listen to interviews and discussions about Laravel and its ecosystem. Laravel Podcast
- **Packagist**: The main Composer repository. It aggregates all types of PHP packages that can be installed with Composer. Packagist

Community and Support

- **Laravel GitHub Repository**: Contribute to the framework, report issues, or browse the source code. Laravel on GitHub
- **Laravel Forums**: Engage with the Laravel community, ask questions, and share knowledge. Laravel.io Forum
- **Laravel Reddit**: A place to discuss Laravel, share projects, and get help from fellow developers. r/laravel
- **Stack Overflow**: Use the `laravel` tag to find answers or ask questions about Laravel-specific issues. Stack Overflow