Updated to the latest version of **Laravel**. Written by the **DevDojo**

Table of Contents

1. Getting Started **8**
- Setting up a Local Dev Environment 8
- Composer & The Laravel Installer 10
- The Laravel Folder Structure 14

2. Routing **18**
- Routing overview 18
- Routing in Laravel 19
- Quick Routing example 21
- Route Closures vs Route Controller Actions 22
- Route Parameters 23

3. Models **25**
- What Are Models? 25
- Eloquent: The Laravel ORM 26

4. Model Relationships **33**
- Model Relationships 33

5. Mutators and Accessors **40**
- Understanding Mutators and Accessors 40

6. Views **44**
- Understanding Views 44

7. Blade **48**

Blade Templating 48

8. Controllers **52**
Understanding Controllers 52

9. Piecing It Together **56**
Piecing It Together 56

10. Artisan **61**
Introducing Artisan 61

11. Middleware **65**
Understanding Middleware 65
Web Middleware 68
Nesting and Using Multiple Middleware 69
Route-Specific Middleware 69
Controller Middleware 69

12. Authentication **72**
Authentication with Laravel Breeze 72
1. Setting Up a New Laravel Application: 72
2. Integrating Laravel Breeze: 72
3. Database Configuration: 73
4. Exploring Authentication: 73
5. Crafting a User Profile: 74
6. Wrapping Up: 75

13. Requests **77**
Requests 77

14. Responses **82**
Responses 82

15. Migrations **87**
Migrations 87

16. Seeds **94**
Seeds & Seeders 94

17. Security **99**
Security 99

18. Testing **104**
Testing 104

19. Wrapping Up **111**
Wrapping Up 111
Resources 111
Words of Encouragement 112
A New Reality 112


WELCOME TO THE
LARAVEL
SURVIVAL GUIDE
WELCOME

Why This Book? Actually, it's not quite a book; it's more of a guide—a guide to prevent you and others from becoming "zombie developers".

What exactly is a 'zombie developer'? It's a developer, much like us, who finds themselves mindlessly hacking away on PHP apps, repeating the same tasks over and over. These repetitive tasks can be incredibly draining and eventually turn one's brain into mushy goo. When this happens, developers everywhere transform into mindless zombies with a thirst for blood and an urge to kill.

However, there's a remedy: the Laravel framework, designed for rapid application development. By embracing Laravel, you can rediscover your passion for coding and fend off the "zombie" within. This guide aims to preserve your sanity, making coding enjoyable once more. And yes, it might just save lives!

By mastering the basics of Laravel, you can save yourself, and possibly others, from becoming a mindless zombie developer.

Don't let the inner zombie thrive; keep close the Laravel survival guide.


GETTING STARTED

1. Getting Started

In this first chapter, we'll explore:

- Setting up a Local Dev Environment
- Composer & The Laravel Installer
- The Laravel Folder Structure

Let's do this!

Setting up a Local Dev Environment

Being aware of one's environment is the key to surviving the zombie developer apocalypse. To craft exceptional web applications, we first need to master setting up a local developer environment.

A local environment, often dubbed the "development environment," is when you develop your web app on your personal computer. Once you're ready to unveil your creation to the world, you will move your code to another server, commonly known as the "production environment".

Below are the instructions on how to add a local environment on your machine.

Local Development on a Mac

If you are a Mac user it will be scary easy to setup a local development environment on your machine. Laravel now offers a native application called Herd. Simply download the application here, install it, and you're ready to start building.

For Mac users, setting up a local development environment is a breeze. Laravel introduces a native application called Herd. Simply download

the application at https://herd.laravel.com, install, and you're ready to start building.

Local Development on Windows

The easiest solution on a Windows machine is to use Laragon, a longtime community favorite. However, there are several other alternatives worth considering:

- https://www.mamp.info/en/
- http://www.wampserver.com/en/
- https://www.apachefriends.org/

Local Development on Ubuntu

If you are on an Ubuntu machine you can use Xampp, or you may choose to install all the applications individually. You can learn more about how to do that from this article here.

To explore other methods of setting up a local environment, refer to the Laravel installation documentation, there you will find many other options that may suit your use-case.

It's important to understand the three essential services that are needed to run a typical local environment:

1. Apache or Nginx (the **web server** for your application)
2. MySQL (the **database** for your application)
3. PHP (the **server-side scripting** language for your application)

Once these services are in place on your machine, you're all set! At the end of the day there is no right or wrong way to setup a local dev environment. Find a way that works best for you and you'll be on the road to creating some killer web applications in no time!

Composer & The Laravel Installer

Laravel leverages Composer to manage its external libraries or packages. Your application's dependencies are defined inside of a file called `composer.json`.

Composer

If you're new to the concept of Composer and its functionalities, don't worry. Let's simplify it with a fun analogy.

Understanding Composer with a Pizza Analogy

Think of Composer as a pizza-making command. If you were to order a pizza using a command, it might look something like this:

```
$ composer make pizza
```

By default, this command gives us a pepperoni pizza. But what if we wanted a different type of pizza, say, a meat-lovers pizza? We'd specify our desired toppings like this:

```
{
    "toppings" : [
        "pepperoni", "ham", "bacon", "beef", "sausage"
    ]
}
```

To customize our pizza order, we save this list in a file named 'composer.json' in our current directory. Running the command again:

```
$ composer make pizza
```

Voilà! Instead of the default pepperoni pizza, we now have our meat-lovers pizza!

Composer, in essence, helps manage the components (or toppings) required to build our applications.

Composer will already be installed if you used Herd or Laragon; however, if you need to install it manually you can do so by visiting https://getcomposer.org/download/

The Laravel Installer

The Laravel installer is a tool that allows developers to quickly scaffold a new Laravel project from the command line. To create a new Laravel project using the Laravel installer you can run:

```
laravel new project-name
```

Replace project-name with the desired name for your new project. This command will create a directory with the given name and install a fresh Laravel application inside it.

Installing the Laravel Installer

If you've already installed Herd or Laragon, skip this step.

After setting up Composer, it's time to integrate the Laravel Installer. Execute the following command:

```
$ composer global require "laravel/installer"
```

Using the Laravel Installer

To use the Laravel installer, open up a command prompt and type the following command:

```
$ laravel new folder_name
```

When you run this command you'll encounter several prompts; choose `No starter kit`, `PHPUnit`, and `No` respectively. Also, when asked about the database, opt for `MySQL`.

```
 _                              _
| |                            | |
| |     __ _ _ __ __ ___   _____| |
| |    / _` | '__/ _` \ \ / / _ \ |
| |___| (_| | | | (_| |\ V /  __/ |
|______\__,_|_|  \__,_| \_/ \___|_|


┌ Would you like to install a starter kit? ──────────────┐
│ No starter kit                                         │
└────────────────────────────────────────────────────────┘

┌ Which testing framework do you prefer? ────────────────┐
│ PHPUnit                                                │
└────────────────────────────────────────────────────────┘

┌ Would you like to initialize a Git repository? ────────┐
│ ○ Yes / ● No                                           │
└────────────────────────────────────────────────────────┘

┌ Which database will your application use? ─────────────┐
│ › ● MySQL                                              │
│   ○ PostgreSQL                                         │
│   ○ SQLite                                             │
│   ○ SQL Server                                         │
└────────────────────────────────────────────────────────┘
```

You'll now have a new application in the `folder_name` you specified. Navigate to this folder using `cd folder_name`, then launch:

```
$ php artisan serve
```

This fires up a local server at http://localhost:8000/. Accessing this URL displays Laravel's welcome screen.

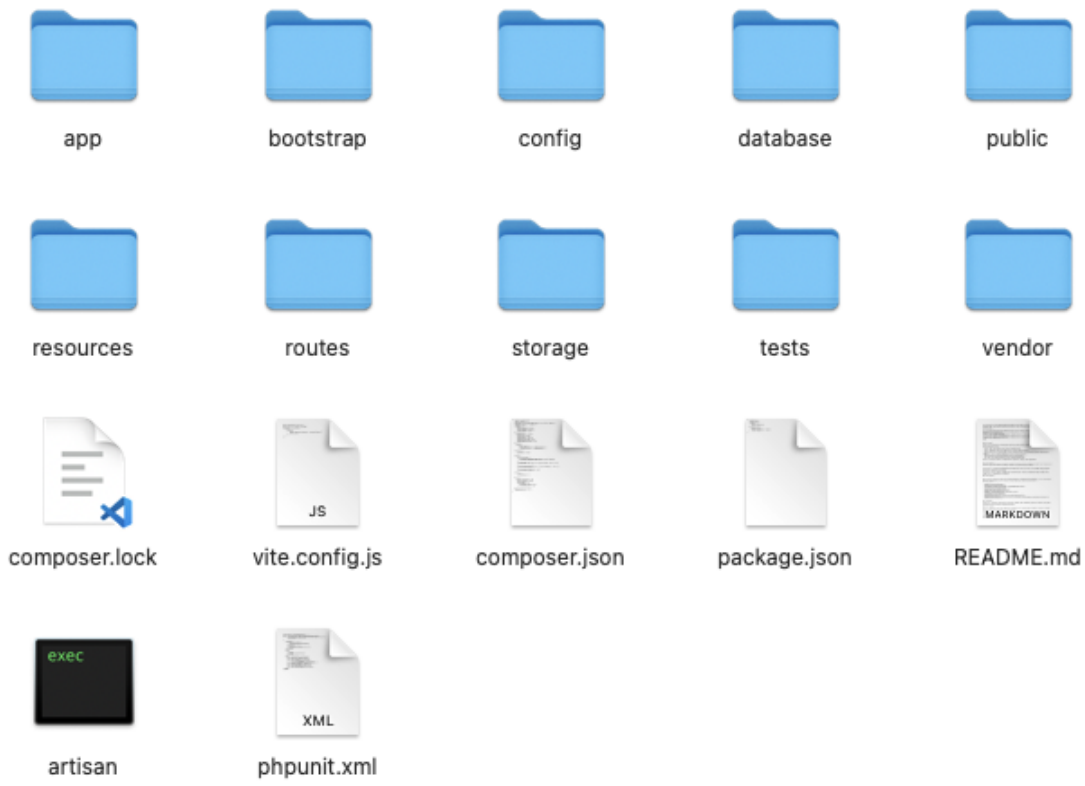
Note: If you're using Laravel Herd, your new Laravel apps will use the `.test` domain. For instance, https://folder_name.test will present the welcome page.

Congrats! You're now ready to start building amazing applications. The ease of Laravel ensures you can resurrect a new project in mere moments.

Before we dive into the code let's briefly get acquainted with Laravel's folder structure.

The Laravel Folder Structure

In a new laravel project you will have the following code structure:



You'll encounter 10 directories:

1. app
2. bootstrap
3. config
4. database
5. public

6. resources
7. routes
8. storage
9. tests
10. vendor

We will not go into full detail with all the files; however, it's important to have a brief understanding of each folder.

App

This is the directory that has all our application logic. In this folder, we will put all our models, controllers, services, and many other classes.

Bootstrap

This folder is used to bootstrap laravel (startup laravel).

Config

This file will contain many of our global configurations for our application.

Database

This folder contains our database files such as migrations and seeds.

Public

This public folder contains many of the applications assets such as images, stylesheets, and scripts.

Resources

We will put our view files in this folder. The views are the pages that a user sees.

Routes

This folder contains all the routes for our application.

Storage

Laravel uses this folder to store sessions, caches, and logs in this folder.

Test

This folder contains files that we use to test the logic of our application.

Vendor

This is the folder that contains our dependencies. When you add new libraries (toppings) to your app, this is the folder that will contain those libraries.

Do you recognize the composer.json file from the image above? Remember this is where we define our dependencies (pizza toppings) for our app. Another important file is `.env`, which contains all our environment variables like debug mode and database credentials.

That's the basic structure of a Laravel application. It will all start becoming more familiar to you as you continue to work with Laravel.

Great job! You're going to love working with Laravel. Next, let's get our hands dirty and dig into some code.