THE LARAVEL SURVIVAL GUIDE

WRITTEN AND UPDATED FOR LARVEL 6

LEARN LARAVEL AND SAVE YOURSELF FROM BECOMING A ZOMBIE DEVELOPER!

EDITION 3 BY TONY LEA & THE DEVDOJO

The Laravel Survival Guide

Tony Lea

This book is for sale at http://leanpub.com/laravelsurvivalguide

This version was published on 2019-10-19

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Introduction

Why The Book? Well, it's not really a book… It's more of a guide (hence the title). A guide to save yourself and others from becoming a zombie developer!

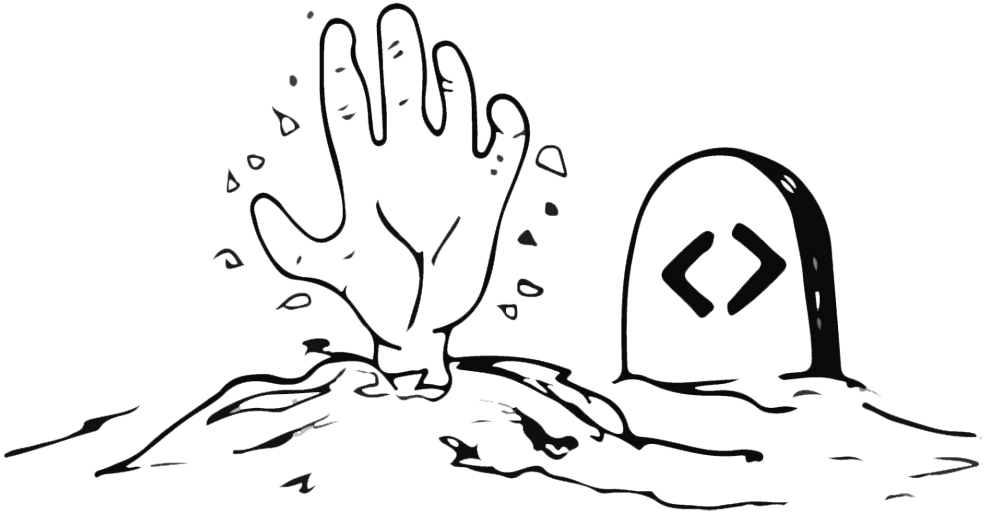
What exactly is a zombie developer? Well, a zombie developer is a developer like you or I, yet they mindlessly hack on PHP apps and do the same thing over and over. These repetitive tasks are incredibly time consuming, and make the developer brain dead. When this happens it gives them a thirst for blood and an urge to kill.

So, instead of letting this happen the developer could have used the amazing Laravel framework for rapid application development. This will help them keep their sanity and it will make coding enjoyable again. Oh, yeah… And it'll save lives.

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

Don't let that inner zombie revive, be sure to keep in hand this Laravel survival guide.

Chapter 1 - Getting Started



A zombie developer is very slow at setting up a new project, whereas a Laravel developer can run a few commands to set up a new project in a matter of seconds!

Being aware of one's environment is crucial to surviving the zombie developer apocalypse. In this chapter, we will briefly go over how to setup your local environment.

> Quick note: We will not go into full detail of installing system requirements such as PHP, MySQL, and Apache. Instead, we are just going to give a quick overview on setting up your local development environment so that way we can start getting into the code as soon as possible.

Your Local Development Environment

A local environment, also referred to as the "development environment", is when you work on your web app from your computer. When you are ready to make your application available for the world to see you will move your code to another server referred to as the "production environment".

So, when we set up your local development environment we are referring to setting up the required applications to run a Laravel app on your computer.

A basic local development environment for Laravel typically requires 3 applications, which are:

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)

There are many programs for each operating system that will install all these applications for you. Just to name a few there is MAMP, WampServer, and XAMPP. You can feel free to check out the links below for these applications:

- https://www.mamp.info/en/ (Mac and Windows)
- http://www.wampserver.com/en/ (Windows)
- https://www.apachefriends.org/ (Mac, Windows, and Linux)

Be sure to read the docs for all the system requirements at http://laravel.com/docs.

There are also 2 other ways to get your local development environment setup. One is referred to as installing a virtual machine which already has all the system requirements installed. This is called Laravel Homestead and you can learn more about installing this at http://laravel.com/docs/homestead.

The other method is called Laravel Valet which is a Mac only development environment. Laravel Valet makes getting started with a new laravel app faster than ever before. You can learn more about setting up the Laravel Valet development environment by visiting https://laravel.com/docs/valet.

There is no right or wrong way to setup your local development environment as long as you meet the minimum system requirements for Laravel. Find a way that works for you, so you can start building the next greatest web app!

Okay, Let's move on to Composer and the Laravel Installer.

Chapter 2 - Composer & The Laravel Installer

A zombie developer manually moves files into their project, whereas a Laravel developer leverages composer to install tools and libraries.

Taking on the zombie developer apocalypse on your own would be almost impossible. Getting help from others is essential, and that's exactly what composer allows us to do. Composer is used to include libraries from other developers into our application.

Let's continue.

Composer & The Laravel Installer

How cool would it be if you could open up a command prompt and type in:

```
$ laravel new blog
```

And a Laravel app gets created inside a folder named “blog”. Well, that’s what the laravel installer does. So, let’s learn how we can use this on our computer.

The Laravel installer, as well as many PHP packages, make use of a dependency manager called Composer (http://getcomposer.org) to add this functionality.

So, What exactly is a dependency manager? A dependency manager is a tool to manage your dependencies.

“WHAT!!!?”, the definition sounds pretty weird, right? Maybe it would help if I explain how composer works with a “Pizza Analogy”.

Let’s pretend we could use a command to make us a pizza:

```
$ composer make pizza
```

By default, we are given a pepperoni pizza. But let's say we wanted this command to make us a pizza with different toppings. Perhaps we wanted a meat-lovers pizza. We would probably need the following toppings:

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

Now, if we save this file in our current directory and name it 'composer.json' and run the command again:

```
$ composer make pizza
```

DING! We now get our meat-lovers pizza instead of our pepperoni pizza.

Hazzzaa!

As you can see Composer is a way of managing the things we need to build our app (or pizza).

Composer is also a command line tool which can be used to install other tools, such as the laravel installer. So, before adding the laravel installer we need to install composer.

Installing Composer

Visit https://getcomposer.org/, click on the 'Getting Started' button. If you are on Windows you can click to download the installer and run through the .exe setup.

If you are on Mac you will need to click on 'Downloading the Composer Executable' and download the file. Then navigate to where you downloaded the installer file and run the following command:

```
$ php installer
```

Then to install composer globally you will need to run:

```
$ mv composer.phar /usr/local/bin/composer
```

And now you will be able to run the `composer` command anywhere on your machine.

Adding the Laravel Installer

After installing Composer we are now ready to add the Laravel Installer. To do this we can simply run the following command:

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

And now we can easily create a new laravel app by typing:

```
$ laravel new app_name
```

Then navigate to your new `app_name` folder in a command prompt. And run:

```
$ php artisan serve
```

Finally, navigate to http://localhost:8000/ in your browser and you will see a welcome 'Laravel' screen.

Now you're ready to start building your amazing app!

> **Warning**: If the laravel installer does not globally work you may need to specify where the composer bin directory is located on your machine. Visit https://devdojo.com/articles/comp bin-directory-path to learn how to do this.

How awesome is this?

With a single command, you can be up and running with a new Laravel app in a few seconds!

You can learn more about the Laravel installer at https://laravel.com/docs/installation#installing-laravel.