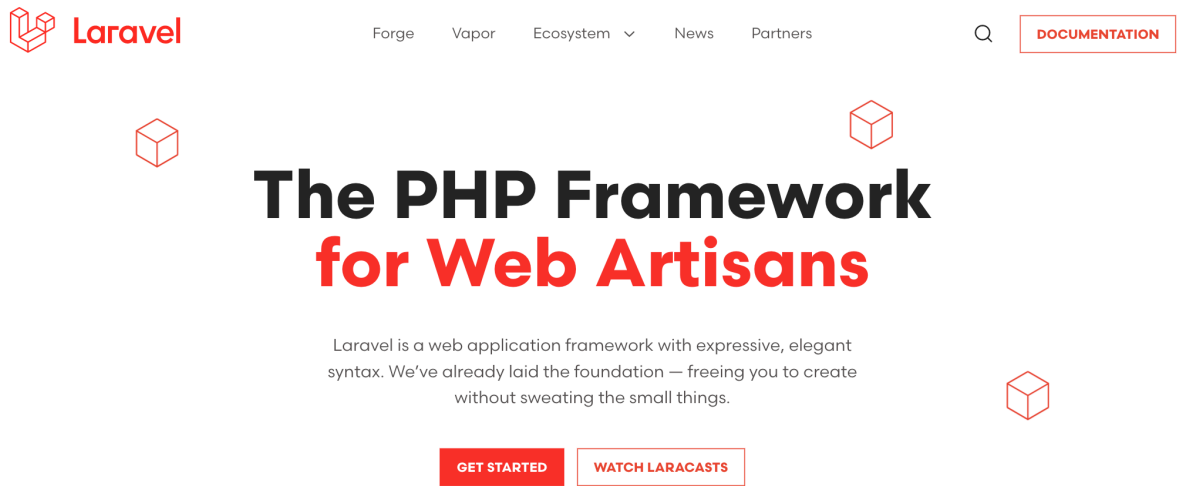
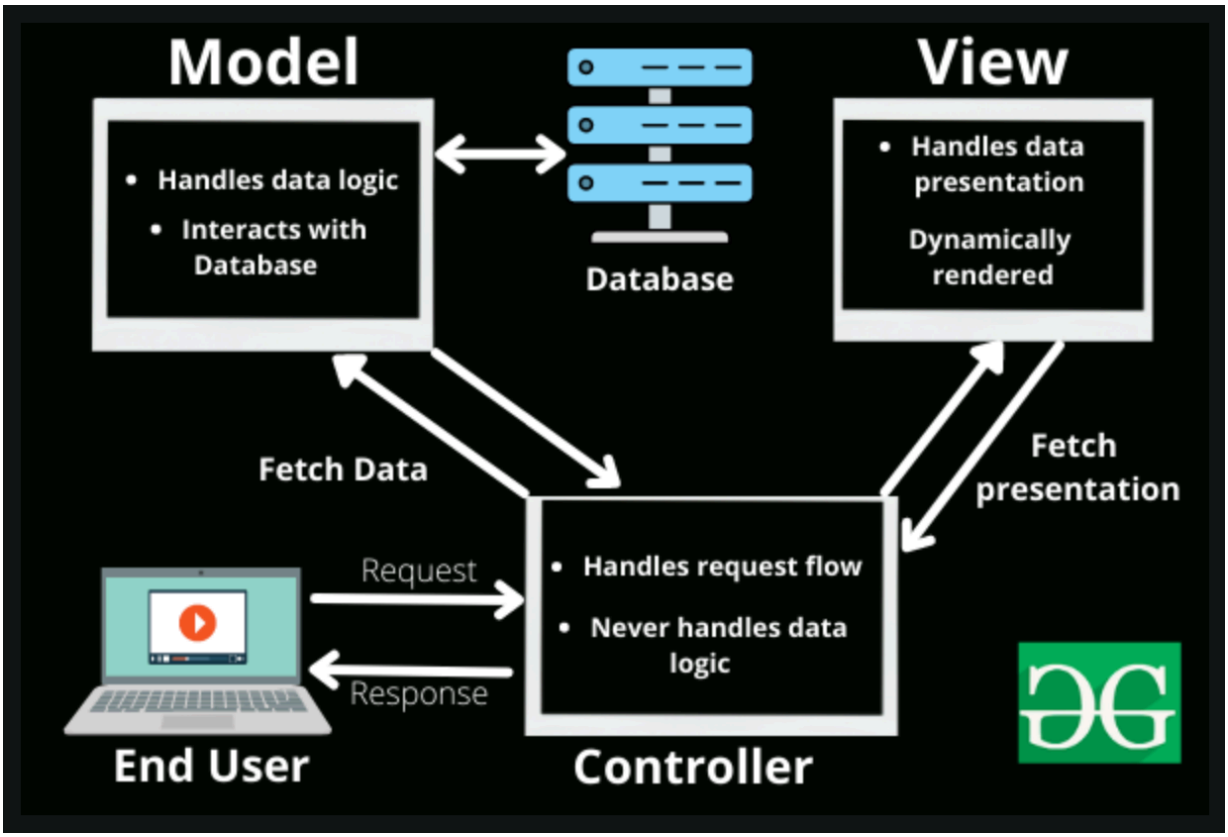
Laravel - Written by Anthony Adams

What is Laravel?



Laravel is a PHP framework. This framework is open source so it is free to use and build on top of. Laravel is not a CMS system. Rather, it is an MVC (model view controller) architectural pattern, which separates the application logic into three interconnected components to improve code organization and maintainability. Laravel provides a clean and elegant syntax, along with a rich set of tools and libraries, making it a favourite among developers.



More on MVC architecture here: https://www.youtube.com/watch?v=DUg2SWWK18I

How does Laravel work?

Laravel follows the convention-over-configuration principle, which means it provides sensible defaults and conventions, allowing developers to focus on the unique aspects of their application. It comes with a wide range of features and functionalities, including routing, database ORM (Object-Relational Mapping),

caching, session management, authentication, and more. Laravel also incorporates modern PHP practices and utilizes Composer, the PHP package manager, to manage dependencies.

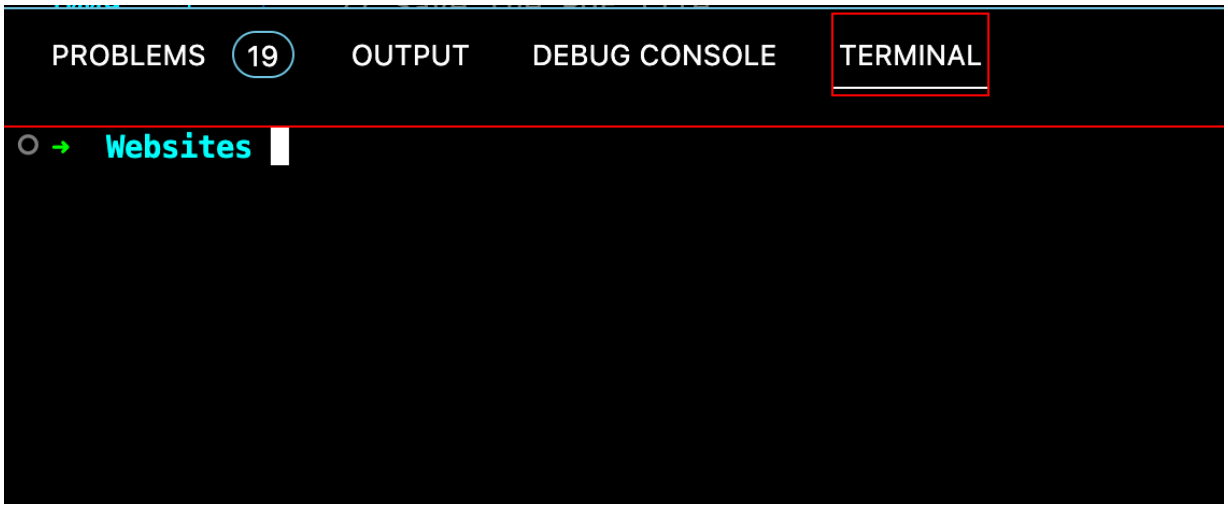
Running Laravel locally

Dependancies

We can run laravel locally on your machine and build something in a virtual server environment (like on XAMPP). To do so we must 1st install some other things first using command line.

Laravel requires PHP version 7.4 or higher, along with extensions like OpenSSL, PDO, Mbstring, Tokenizer, and XML. Additionally, it requires a database, such as MySQL or PostgreSQL, for storing application data.

Again we will be doing some things in command line so if you are on a Mac, open your terminal. If you are on a PC, open CMD:



First lets check what version of PHP we are running:

Type: php -v

➜ php -v
PHP 7.3.29 (cli) (built: Aug 15 2021 23:10:16) (NTS)
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.29, Copyright (c) 1998-2018 Zend Technologies

If you don't have PHP installed do the following:

Mac:

Install brew by running the following command:
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

See more on brew here: https://brew.sh/

PC:

Check to see if your computer is X64 or X86 in your:

Settings>Choose System>About

Once you know what your system is head over here:

https://windows.php.net/download

And download the zip package, extract it and rename the folder to “PHP”, then place the

folder in your computers root folder (most likely c:\)

C:\PHP

XAMPP

We already have XAMPP installed from our 1st class but just in case anyone reading this does not have it installed, here are the instructions!

Head over to:

https://www.apachefriends.org/

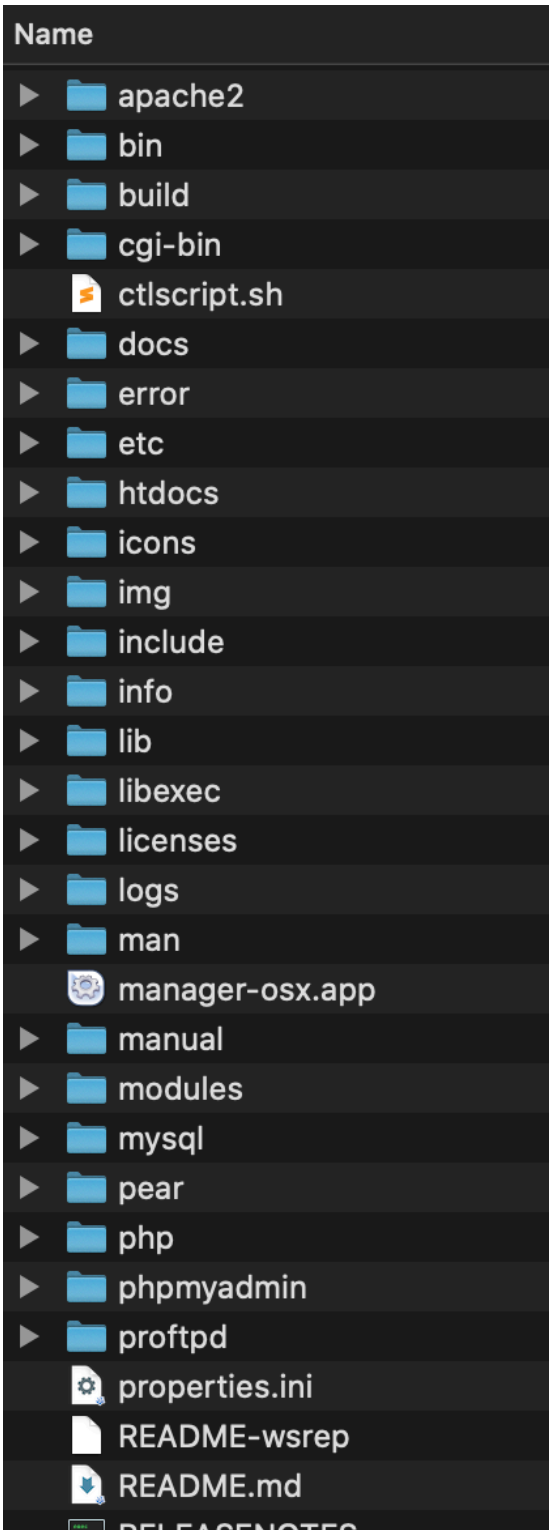
Download XAMPP for your system and install it.

When you launch the application you will see this:

From this screen we can do a few things. First off, if you click “Go to Application” it will open a web browser and load the index.html in the HTDOCS folder within your server environment.

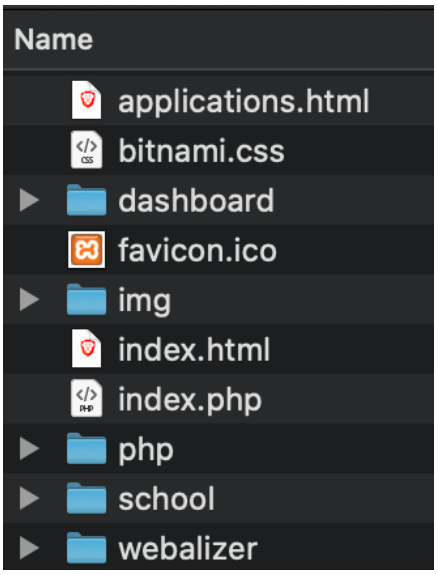
Which brings us to the next part we need to understand. Like stated earlier XAMPP is like running your own apache server on your computer. As such there is a file system within it. You should be saving your files within it. Click the “Open Application Folder” button.

It will open this:



Within this file system, you want to be working in the “htdocs” folder for the most part. This is no different than when working on an actual hosting server. The folder to publish web pages will either be named public_html or htdocs.

In the htdocs folder you will see this (minus the “school” folder):



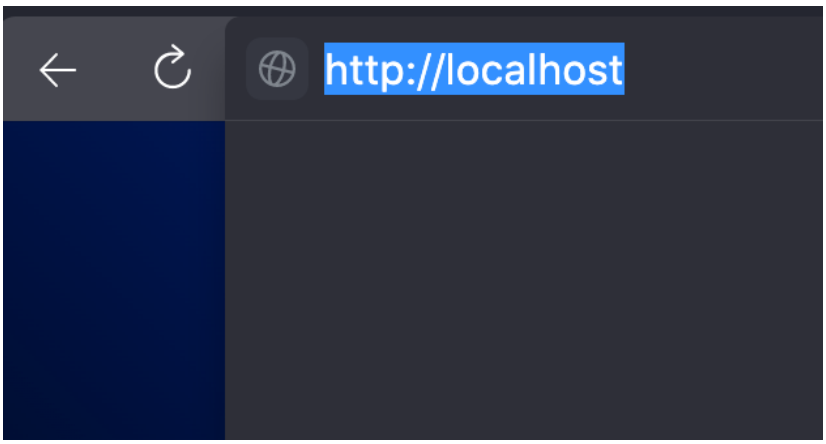
If you were to change the content of the

“index.html” folder it will change what you see when you click the “Go to Application” button.

In here we can add more folders and files, the same as we would when working on an actual apache server.

Im my case I have added a “school” folder and inside that folder it has all the different courses. Remember to stay organized!

When navigating to pages within the server, you have to go to “localhost” to do so.

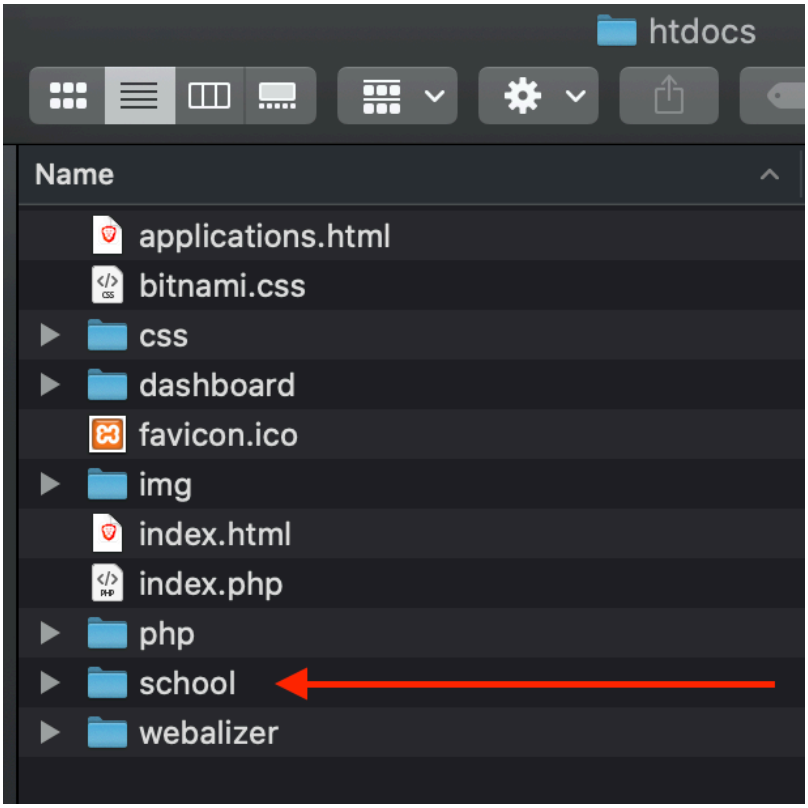


This is your home page. Which resolves to: http://localhost/index.html

If you want to make a child folder, you can do

so inside the “htdocs” folder. For example:

Make a folder like mine, names “school”.

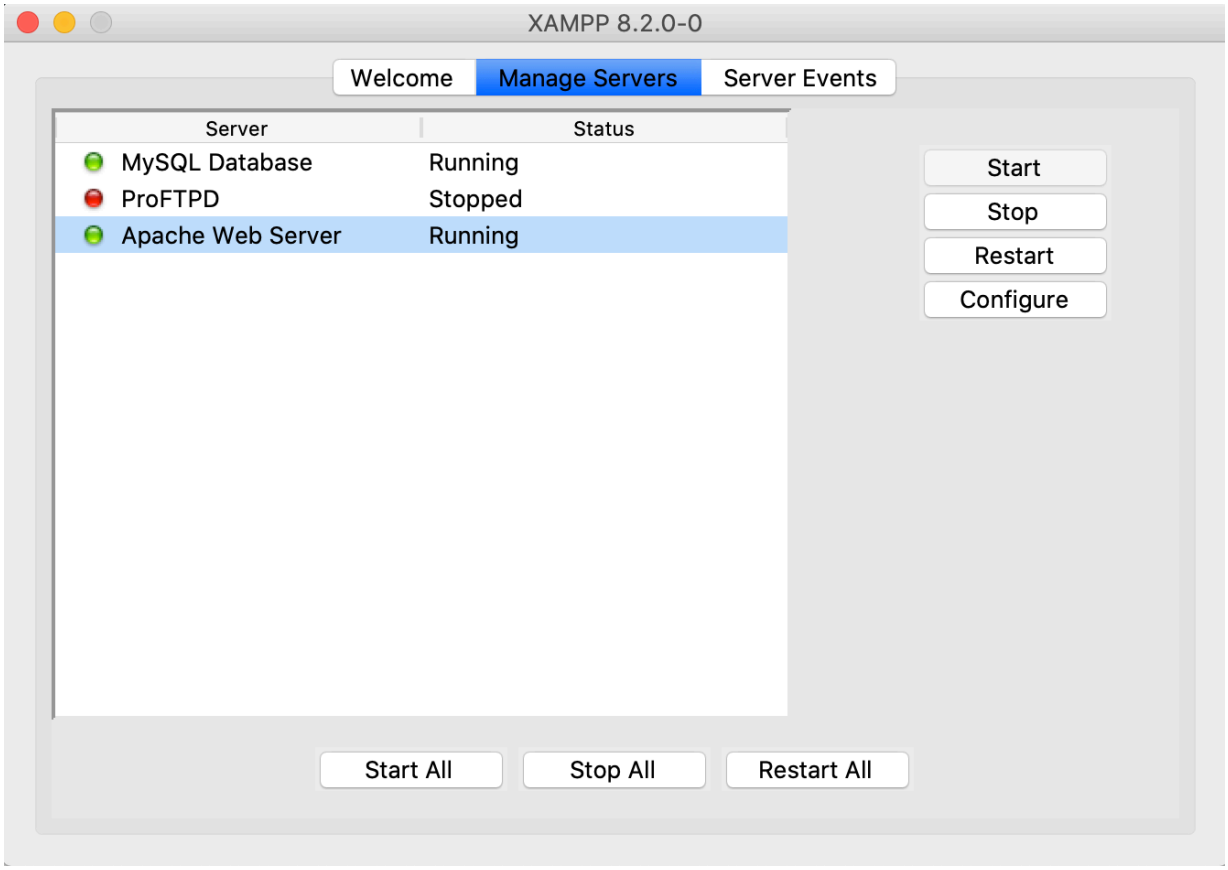


If you were to make a file named index.php in the “school” folder, then the URL would be:

http://localhost/school/

Which would resolve to:

http://localhost/school/index.php



If you click “Manage Servers” you will see this screen. Here you can, as the name states manage servers. The Apache Web Server must be running for you to be able to run the files saved in it.

The MySQL Database must be running to access the DB’s.

You can connect to the Apache server via FTP,

if you wish to you must enable it. Be advised this can allow incoming connections.

Composer

Composer is a package manager. It provides a standard format for managing dependencies for PHP software and required libraries. We need to install that on our machines.

Note* You may need to install curl for this part to work properly. Do so with this command:

```
brew install curl
```

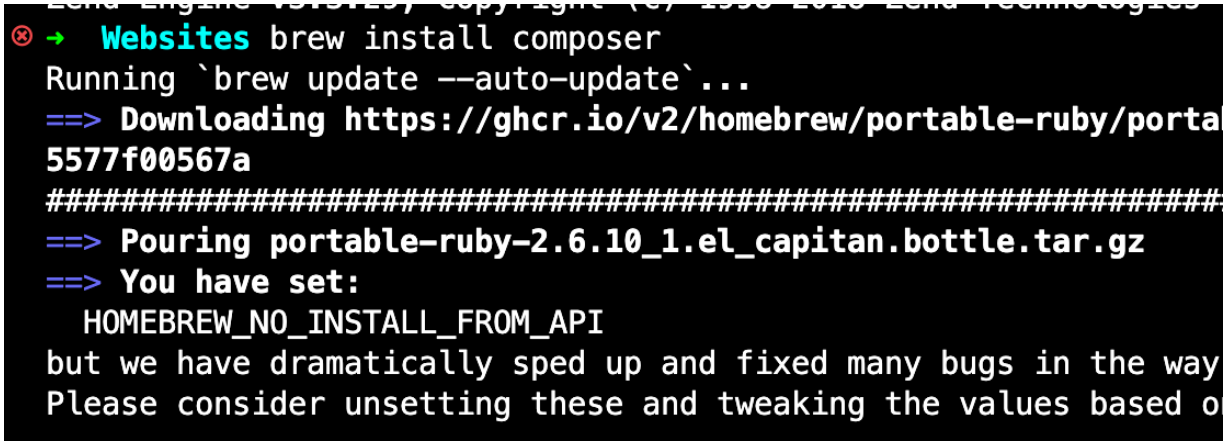
https://getcomposer.org/

Installing composer

Mac:

Now that you have brew installed, you can use brew to install composer real easy by running the following command:

$ brew install composer



PC:

Seeing as you by this point SHOULD have camp installed on your machine, you thus have PHP installed on your machine. So now you can install composer.

Use this link:

https://getcomposer.org/Composer-Setup.exe

Install using the wizard, go through all the steps and once it is installed. Open CMD and run this command:

```
composer
```

Which should return this:

```
   ______
  / ____/___  ____ ___  ____  ____  ________  _____
 / / / __ / __ `__ / __ / __ / ___/ _ / ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__ ) __/ /
____/____/_/ /_/ /_/ .___/____/____/___/_/
                  /_/
Composer version 2.4.2 2022-09-14 16:11:15
```

Linux / Server:

The previous steps won't really apply for installing on a server because the server should already have PHP installed.

However, Laravel won't work on *every* server. The server environment must meet the criteria to run Laravel. You have to be running PHP 7.4 or higher, you also have to have terminal access via SSH as well as some other dependencies stated earlier. If not you will need to find a hosting server that has this access.

Connect to your hosting account via SSH (SSH allows you to connect to your hosting server and run command line remotely).

Run the command to download Composer:

```
php -r "copy('https://getcomposer.org/installer',
'composer-setup.php');"
```

Now install it:

```
php composer-setup.php --install-dir=/usr/local/bin --
filename=composer
```

After installing it terminal should return:

```
All settings correct for using Composer
Downloading…

Composer (version 2.4.2) successfully installed to: /
usr/local/bin/composer
Use it: php /usr/local/bin/composer
```

Now you can remove the installer:

```
php -r "unlink('composer-setup.php');"
```

Test the install:

```
composer
```

You should get this result:

```
   ______
  / ____/___  ____ ___  ____  ____  ________  _____
 / / / __ / __ `__ / __ / __ / ___/ _ / ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__ ) __/ /
____/____/_/ /_/ /_/ .___/____/____/___/_/
                  /_/
Composer version 2.4.2 2022-09-14 16:11:15
```

Laravel

Now that composer is installed we can use Composer to create a laravel project.

See more here:
https://laravel.com/docs/9.x/installation#installation-via-composer

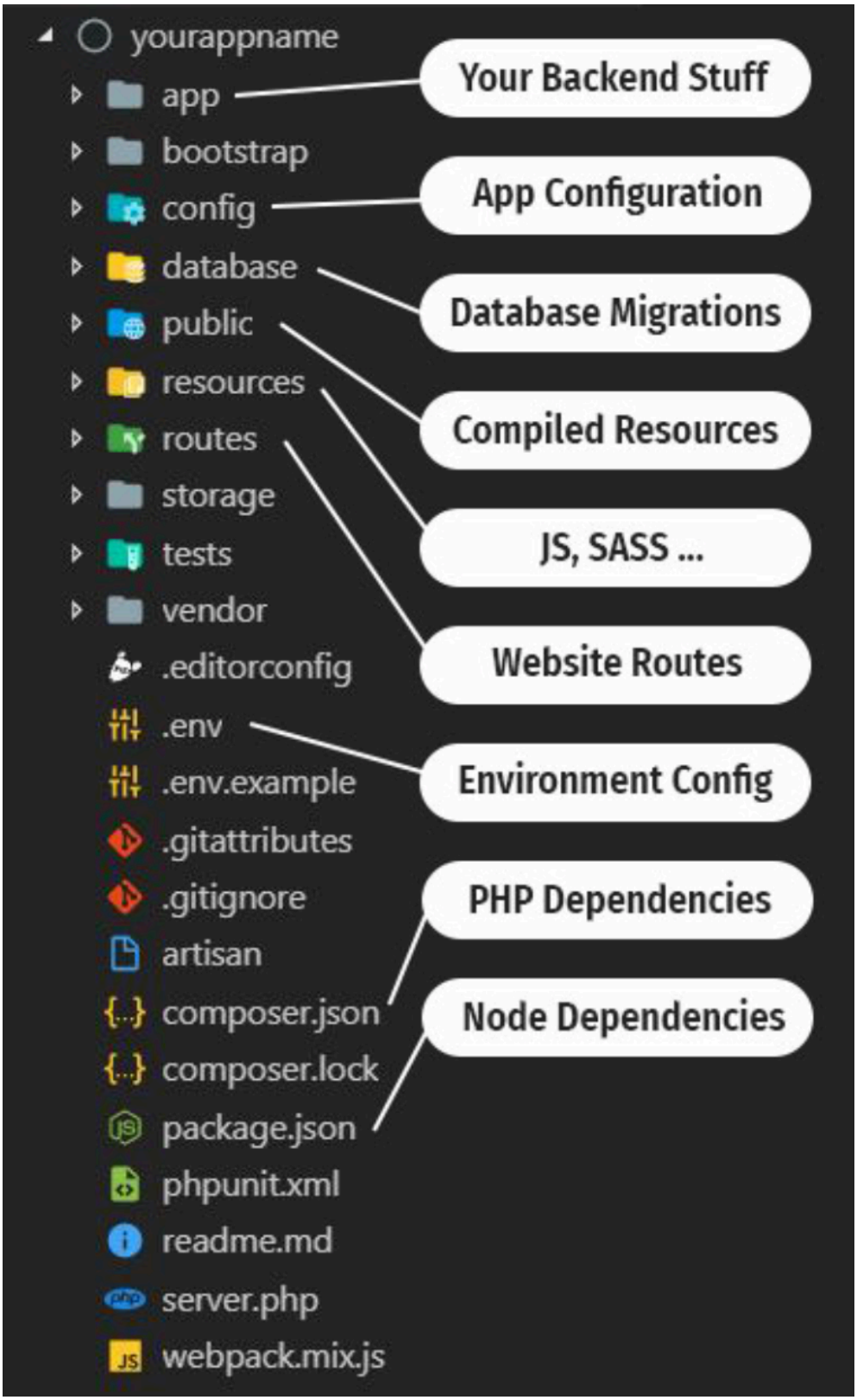
Use Composer to create a Laravel project:

```
composer create-project laravel/laravel:^9.0 example-app
```

Use Composer to install the Laravel Installer:

```
composer global require laravel/installer

laravel new example-app
```


yourappname
app
bootstrap
config
database
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package.json
phpunit.xml
readme.md
server.php
webpack.mix.js
Your Backend Stuff
App Configuration
Database Migrations
Compiled Resources
JS, SASS ...
Website Routes
Environment Config
PHP Dependencies
Node Dependencies

- **app –** The app directory contains the core code of your application.
- **bootstrap –** The bootstrap directory contains the app.php file which bootstraps the framework.
- **config –** The config directory, as the name implies, contains all of your application's configuration files.
- **database –** The database directory contains your database migrations, model factories, and seeds. If you wish, you may also use this directory to hold an SQLite database.
- **public –** The public directory contains the index.php file, which is the entry point for all requests entering your application and configures autoloading. This directory also houses your assets such as images, JavaScript, and CSS.
- **resources –** The resources directory contains your views as well as your raw, un-compiled assets such as CSS or JavaScript. This directory also houses all of your language files.
- **routes –** The routes directory contains all of the route definitions for your application.
- **storage –** The storage directory contains your compiled Blade templates, file based sessions, file caches, and other files generated by the framework.
- **tests –** The tests directory contains your automated tests.
- **vendor –** The vendor directory contains your Composer dependencies.