### The App Directory

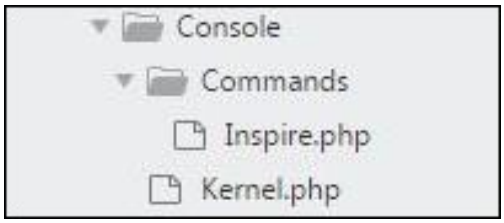
The `app` directory contains the core code of your application.

### The Bootstrap Directory

The `bootstrap` directory contains the `app.php` file which bootstraps the framework. This directory also houses a `cache` directory which contains framework generated files for performance optimization such as the route and services cache files. You should not typically need to modify any files within this directory.

## Console

Console includes the artisan commands necessary for Laravel. It includes a directory named **Commands**, where all the commands are declared with the appropriate signature. The file **Kernal.php** calls the commands declared in **Inspire.php**. If we need to call a specific command in Laravel, then we should make appropriate changes in this directory.




### The Events Directory

This folder includes all the events for the project.

This directory does not exist by default, but will be created for you by the `event:generate` and `make:event` Artisan commands. The `Events` directory houses event classes. Events may be used to alert other parts of your application that a given action has occurred, providing a great deal of flexibility and decoupling.

## Exceptions

This folder contains all the methods needed to handle exceptions. It also contains the file **handle.php** that handles all the exceptions.

The `Exceptions` directory contains your application's exception handler and is also a good place to place any exceptions thrown by your application. If you would like to customize how your exceptions are logged or rendered, you should modify the `Handler` class in this directory.

### The Http Directory

The `Http` directory contains your controllers, middleware, and form requests. Almost all of the logic to handle requests entering your application will be placed in this directory.

The **Http** folder has sub-folders for controllers, middleware and application requests. As Laravel follows the MVC design pattern, this folder includes model, controllers and views defined for the specific directories.

The **Middleware** sub-folder includes middleware mechanism, comprising the filter mechanism and communication between response and request.

The **Requests** sub-folder includes all the requests of the application.

## Models

The Models directory contains all of your Eloquent model classes. The Eloquent ORM included with Laravel provides a beautiful, simple ActiveRecord implementation for working with your database. Each database table has a corresponding "Model" which is used to interact with that table.

## Service Priovisers

Service providers are the central place to configure your application. If you open the config/app.php file included with Laravel, you will see a providers array. These are all of the service provider classes that will be loaded for your application.

**Default Laravel Service Providers**

- AppServiceProvider.
  Service providers are the central place to configure your application.

- AuthServiceProvider.
  AuthServiceProvider is the default guard that Laravel uses to give the service authentication in the system. But if you need you can make your own guards for specific situations, in that case you will have you own AuthServiceProvider.

- BroadcastServiceProvider.

Laravel's event broadcasting allows you to broadcast your server-side Laravel events to your client-side JavaScript application using a driver-based approach to WebSockets. Currently, Laravel ships with Pusher Channels and Ably drivers.

- EventServiceProvider.

  The EventServiceProvider included with your Laravel application provides a convenient place to register all event listeners. The listen property contains an array of all events (keys) and their listeners (values). Of course, you may add as many events to this array as your application requires.

- RouteServiceProvider.

  All Laravel routes are defined in your route files, which are located in the `routes` directory. These files are automatically loaded by your application's `App\Providers\RouteServiceProvider`. The `routes/web.php` file defines routes that are for your web interface. These routes are assigned the `web` middleware group, which provides features like session state and CSRF protection. The routes in `routes/api.php` are stateless and are assigned the `api`middleware group.

### View/Components

The Laravel components are a small entity with an interface that is well defined. These serve as the building block for a large software system. All the related data is encapsulated in the reusable unit.

### App Layout

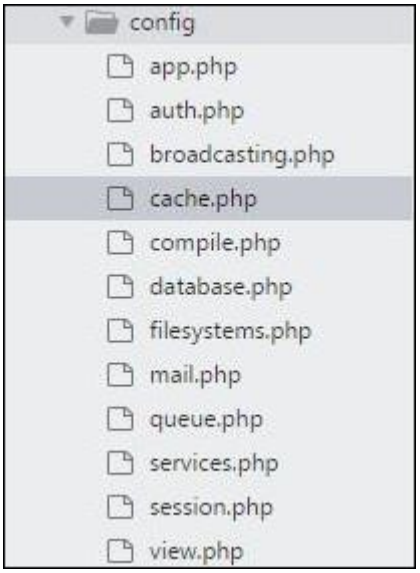
Layouts are often the starting point of many web development projects.Laravel Blade template engine enables the developer to produce HTML based sleek designs and themes. All views in Laravel are usually built in the blade template.

### The Bootstrap Directory

The `bootstrap` directory contains the `app.php` file which bootstraps the framework. This directory also houses a `cache` directory which contains framework generated files for performance optimization such as the route and services cache files. You should not typically need to modify any files within this directory.

### The Config Directory

The `config` directory, as the name implies, contains all of your application's configuration files. It's a great idea to read through all of these files and familiarize yourself with all of the options available to you.




**cache.php**

The cache configuration is located at config/cache. php . In this file you may specify which cache driver you would like used by default throughout your application. Laravel supports popular caching backends like Memcached and Redis out of the box.

**database.php**

Laravel makes connecting with databases and running queries extremely simple. The database configuration file is app/config/database. php . In this file you may define all of your database connections, as well as specify which connection should be used by default.

**Filesystem.php**

Laravel's filesystem configuration file is located at config/filesystems.php . Within this file, you may configure all of your filesystem "disks". Each disk represents a particular storage driver and storage location.

**Hashing.php**

Hashing is the process of transforming a string of characters into a shorter fixed value or a key that represents the original string. Laravel uses the Hash facade which provides a secure way for storing passwords in a hashed manner.

**Logging.php**

Laravel logging is based on "channels". Each channel represents a specific way of writing log information. For example, the single channel writes log files to a single log file, while the slack channel sends log messages to Slack. Log messages may be written to multiple channels based on their severity.

**Mail.php**

Laravel provides a clean, simple API over the popular SwiftMailer library. The mail configuration file is app/config/mail. php , and contains options allowing you to change your SMTP host, port, and credentials, as well as set a global from address for all messages delivered by the library.

**Queue.php**

The Laravel queue service provides a unified API across a variety of different queue back-ends. Queues allow you to defer the processing of a time consuming task, such as sending an e-mail, until a later time which drastically speeds up web requests to your application.

**sanctum.php**

Laravel Sanctum provides a featherweight authentication system for SPAs (single page applications), mobile applications, and simple, token based APIs. Sanctum allows each user of your application to generate multiple API tokens for their account.

**Services.php**

The Laravel service container is a powerful tool for managing class dependencies and performing dependency injection. Dependency injection is a fancy phrase that essentially means this: class dependencies are "injected" into the class via the constructor or, in some cases, "setter" methods.

**Session.php**

Laravel session is a way of storing the user information across the multiple user requests. It keeps track of all the users that visit the application. Let's understand the session through an example. First, we create a form on which we apply the properties of the session.

The Database Directory

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.

**Cors.php**

Cross-Origin Resource Sharing (CORS) is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources. CORS also relies on a mechanism by which browsers make a "preflight" request to the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In that preflight, the browser

sends headers that indicate the HTTP method and headers that will be used in the actual request.

### View.php

Views separate your controller / application logic from your presentation logic and are stored in the resources/views directory. When using Laravel, view templates are usually written using the Blade templating language.

### Factory in laravel

As you can see, in their most basic form, factories are classes that extend Laravel's base factory class and define a definition method. The definition method returns the default set of attribute values that should be applied when creating a model using the factory.

### Migration in laravel

Migrations are like version control for your database, allowing your team to modify and share the application's database schema. Migrations are typically paired with Laravel's schema builder to build your application's database schema.

### Seeder in laravel

Laravel includes the ability to seed your database with data using seed classes. All seed classes are stored in the database/seeders directory. By default, a DatabaseSeeder class is defined for you. From this class, you may use the call method to run other seed classes, allowing you to control the seeding order.

### Node modules

In this folder you have got all the pacakge dependencies of the laravel. And All those packages and their dependencies are listed in composer. json. Same as node_modules folder also contains the package dependencies related to your javascript projects. So, that you can require it in your projects.

### Assets

The asset class is **a set of methods to help with the collection, grouping and displaying of assets** (js, css, img).

### htaccess file

The . htaccess file is **a special Apache file that you can use to manipulate the behavior of your site**. These manipulations include things such as redirects that force all of your domain's pages to https or www. You can even redirect all users to one page, while your IP loads another page.

### Auth in laravel

Laravel includes built-in authentication and session services which are typically accessed via the Auth and Session facades. These features provide cookie-based authentication for requests that are initiated from web browsers. They provide methods that allow you to verify a user's credentials and authenticate the user.

### Components in laravel

The Laravel components are a small entity with an interface that is well defined. These serve as the building block for a large software system. All the related data is encapsulated in the reusable unit. Web development, programming languages, Software testing & others.

### App.blade.php

Blade is the simple, yet powerful templating engine that is included with Laravel. Unlike some PHP templating engines, Blade does not restrict you from using plain PHP code in your templates.

### Layout

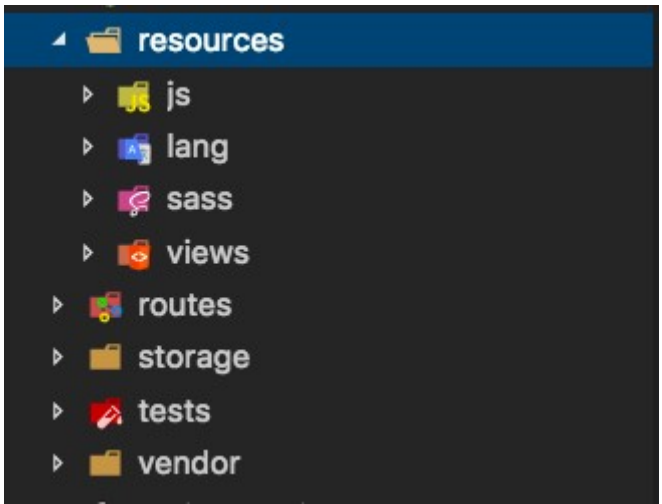
Layouts are often the starting point of many web development projects. Laravel Blade template engine enables the developer to produce HTML based sleek designs and themes. All views in Laravel are usually built in the blade template.

### Laravel blade partials

A blade partial is similar to an include or require in PHP. It's an easy way to include contents of another file inside of a template. A PHP include would look something like 'include file. php' whereas a blade @include looks like @include('partials. file.

### Route

What Is a Route? The route is a way of creating a request URL for your application. These URLs do not have to map to specific files on a website, and are both human readable and SEO friendly. In Laravel, routes are created inside the routes folder. They are are created in the web.



The `routes` directory contains all of the route definitions for your application. By default, several route files are included with Laravel:`web.php`,`api.php`,`console.php`, and `channels.php`.

The `web.php` file contains routes that the `RouteServiceProvider` places in the `web` middleware group, which provides session state, CSRF protection, and cookie encryption. If your application does not offer a stateless, RESTful API then all your routes will most likely be defined in the `web.php` file.

The `api.php` file contains routes that the `RouteServiceProvider` places in the `api` middleware group. These routes are intended to be stateless, so requests entering the application through these routes are intended to be authenticated via tokens and will not have access to session state.

The `console.php` file is where you may define all of your closure based console commands. Each closure is bound to a command instance allowing a simple approach to interacting with each command's IO methods. Even though this file does not define HTTP routes, it defines console based entry points (routes) into your application.

The `channels.php` file is where you may register all of the event broadcasting channels that your application supports.

### The Storage Directory

The `storage` directory contains your logs, compiled Blade templates, file based sessions, file caches, and other files generated by the framework. This directory is segregated into `app`, `framework`, and `logs` directories. The `app` directory may be used to store any files generated by your application. The `framework` directory is used to store framework generated files and caches. Finally, the `logs` directory contains your application's log files.

The `storage/app/public` directory may be used to store user-generated files, such as profile avatars, that should be publicly accessible. You should create a symbolic link at `public/storage` which points to this directory. You may create the link using the `php artisan storage:link` Artisan command.

### Log in laravel

By default, Laravel is configured to create a single log file for your application, and this file is stored in app/storage/logs/laravel.Log.

In Laravel, you can configure logging from a single configuration file located in config/logging.php. The configuration file comes with predefined log drivers to choose from, and the default driver is a stack that uses the single channel to log to a laravel. log file found in the storage/logs folder.

## Test in laravel

Laravel is built with testing in mind. In fact, support for testing with PHPUnit is included out of the box and a `phpunit.xml` file is already set up for your application. The framework also ships with convenient helper methods that allow you to expressively test your applications.

By default, your application's `ests` directory contains two directories:`Feature` and `Unit`. Unit tests are tests that focus on a very small, isolated portion of your code. In fact, most unit tests probably focus on a single method. Feature tests may test a larger portion of your code, including how several objects interact with each other or even a full HTTP request to a JSON endpoint.

An `ExampleTest.php` file is provided in both the `Feature` and `Unit` test directories. After installing a new Laravel application, run `phpunit` on the command line to run your tests.

## Larael vendor directory

The vendor directory contains the composer dependencies, for example, to install Laravel setup, the composer is required. The vendor folder contains all the composer dependencies.

## Laravel env file

Laravel's default .env file contains some common configuration values that may differ based on whether your application is running locally or on a production web server. These values are then retrieved from various Laravel configuration files within the config directory using Laravel's env function.