**Policies** are classes that organize authorization logic around a particular model or resource. For example, if your application is a blog, you may have a App\Models\Post model and a corresponding App\Policies\PostPolicy to authorize user actions such as creating or updating posts.

Laravel provides two primary ways of authorizing actions: gates and policies.

**Gates** provide a simple, closure-based approach to authorization while policies, like controllers, group logic around a particular model or resource. In this documentation, we'll explore gates first and then examine policies.

**Gates** are a great way to learn the basics of Laravel's authorization features; however, when building robust Laravel applications you should consider using policies to organize your authorization rules.

**Guards** define how users are authenticated for each request. For example, Laravel ships with a session guard which maintains state using session storage and cookies. Providers define how users are retrieved from your persistent storage.

### Use guard as middleware:

While working with guards we can also **use a guard as a middleware**. To use a guard as a middleware we need to add middleware on the group of routes or any specific route.

```
Route::group(['middleware' => ['auth:admin']], function() {


  Route::get('/users', [UserController::class, 'users']);


});
```

Whenever users URL hits in the browser, the system will first check that the logged-in user is an admin user or not. If the logged user is not admin then it returns an error message and if the user is admin then the request proceeds further.

**JSON web token (JWT)** authentication is used to verify ownership of JSON data. JWT is not encryption, rather it determines if the data can be trusted because its ownership is verified. JWT is an open standard (RFC 7519) that enables information to be securely transmitted between two parties as a JSON object.

JWTs are a good way of securely transmitting information between parties because they can be signed, which means you can be sure that the senders are who they say they are. Additionally, the structure of a JWT allows you to verify that the content hasn't been tampered with.

**Laravel Passport** is an Open Authorization 2.0 server implementation used for authenticating APIs using Laravel. As the tokens are only used in API authentication, Laravel Passport offers an easy way to implement the token authentication on the OAuth server.

## Available Router Methods

The router allows you to register routes that respond to any HTTP verb:

```
Route::get($uri, $callback);
Route::post($uri, $callback);
Route::put($uri, $callback);
Route::patch($uri, $callback);
Route::delete($uri, $callback);
Route::options($uri, $callback);
```

(CSRF) - cross-site request forgery

## Middleware

Laravel includes a middleware that verifies the user of your application is authenticated. If the user is not authenticated, the middleware will redirect the user to your application's login screen. However, if the user is authenticated, the middleware will allow the request to proceed further into the application.

## Controller

Instead of defining all of your request handling logic as closures in your route files, you may wish to organize this behavior using "controller" classes. Controllers can group related request handling logic into a single class. For example, a `UserController` class might handle all incoming requests related to users, including showing, creating, updating, and deleting users. By default, controllers are stored in the `app/Http/Controllers` directory.

## Dependency injection

In Laravel, dependency injection is **the process of injecting class dependencies into a class through a constructor or setter method**. This allows your code to look clean and run faster. Dependency injection involves the use of a Laravel service container, a container that is used to manage class dependencies.

## views

You may create a view by placing a file with the `.blade.php` extension in your application's `resources/views` directory. The `.blade.php` extension informs the framework that the file contains a **Blade template**. Blade templates contain HTML as well as Blade directives that allow you to easily echo values, create "if" statements, iterate over data, and more.

## Session in laravel

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.

## Error Handling

When you start a new Laravel project, error and exception handling is already configured for you. The `App\Exceptions\Handler` class is where all exceptions thrown by your application are logged and then rendered to the user.

### Cache f

Your application's cache configuration file is located at **`config/cache.php`**. In this file, you may specify which cache driver you would like to be used by default throughout your application. Laravel supports popular caching backends like**Memcached**,**Redis**,**DynamoDB**, and relational databases out of the box. In addition, a file based cache driver is available, while **`array`** and "null" cache drivers provide convenient cache backends for your automated tests.

### Contracts

Laravel's "contracts" are a set of interfaces that define the core services provided by the framework. For example, an **`Illuminate\Contracts\Queue\Queue`** contract defines the methods needed for queueing jobs, while the **`Illuminate\Contracts\Mail\Mailer`** contract defines the methods needed for sending e-mail. The decision to use contracts or facades will come down to personal taste and the tastes of your development team. Both contracts and facades can be used to create robust, well-tested Laravel applications. Contracts and facades are not mutually exclusive. Some parts of your applications may use facades while others depend on contracts. As long as you are keeping your class' responsibilities focused, you will notice very few practical differences between using contracts and facades.

### **Benchmarking** in helpers

Sometimes you may wish to quickly test the performance of certain parts of your application. On those occasions, you may utilize the `Benchmark` support class to measure the number of milliseconds it takes for the given callbacks to complete.

### Laravel Pipeline

Laravel's Pipeline facade provides a convenient way to "pipe" a given input through a series of invokable classes, closures, or callables, giving each class the opportunity to inspect or modify the input and invoke the next callable in the pipeline. This Facade has been in the core for years but never documented.

### Laravel **Lottery**

Laravel's lottery class may be used to execute callbacks based on a set of given odds. This can be particularly useful when you only want to execute code for a percentage of your incoming requests. You may combine Laravel's lottery class with other Laravel features. For example, you may wish to only report a small percentage of slow

queries to your exception handler. And, since the lottery class is callable, we may pass an instance of the class into any method that accepts callables.

### Packages in laravel

Packages in PHP are a collection of routes, controllers, and views that are configured to add or extend the functionality of a Laravel application.

### Process in laravel

To invoke a process, you may use the`run`and`start`methods offered by the`Process`facade. The`run`method will invoke a process and wait for the process to finish executing, while the`start`method is used for asynchronous process execution

### Queues in laravel

Laravel queues provide a unified API across a variety of different queue backends, such as Beanstalk, Amazon SQS, Redis, or even a relational database. Queues allow you to defer the processing of a time consuming task, such as sending an email, until a later time.

### Rate limiting

Laravel includes a simple to use rate limiting abstraction which, in conjunction with your application's cache,provides an easy way to limit any action during a specified window of time. If you are interested in rate limiting incoming HTTP requests, please consult the rate limiter middleware documentation.

### Task scheduling

Laravel's command scheduler offers a fresh approach to managing scheduked task on your server. The scheduler allows you to fluently and expressively define your command schedule within your Laravel application itself. When using the scheduler, only a single cron entry is needed on your server.

### Authentication

Authentication is the process of identifying the user credentials. In web applications, authentication is managed by sessions which take the input parameters such as email or username and password, for user identification. If these parameters match, the user is said to be authenticated. Laravel's authentication facilities are made up of "guards" and "providers". Guards define how users are authenticated for each request. For

example, Laravel ships with a `session` guard which maintains state using session storage and cookies.

Your application's authentication configuration file is located at `config/auth.php`. This file contains several well-documented options for tweaking the behavior of Laravel's authentication services.

### Authorization

Laravel provides two primary ways of authorizing actions: gates and policies. Think of gates and policies like routes and controllers. Gates provide a simple, closure-based approach to authorization while policies, like controllers, group logic around a particular model or resource.

### Gates

Gates are simply closures that determine if a user is authorized to perform a given action. Typically, gates are defined within the **`boot`** method of the **`App\Providers\AuthServiceProvider`** class using the **`Gate`** facade. Gates always receive a user instance as their first argument and may optionally receive additional arguments such as a relevant Eloquent model.

### Policies

Policies are classes that organize authorization logic around a particular model or resource. For example, if your application is a blog, you may have `App\Models\Post` model and a corresponding `App\Policies\PostPolicy` to authorize user actions such as creating or updating posts.
You may generate a policy using the `make:policy` Artisan command. The generated policy will be placed in the `app/Policies` directory.

### Verification

Laravel includes the Auth\VerificationController class that contains the necessary logic to send verification links and verify emails. To register the necessary routes for this controller, pass the verify option to the Auth::routes method: Auth::routes(['verify' => true]);

### Encryption

Laravel uses AES-256 and AES-128 encrypter, which uses Open SSL for encryption. All the values included in Laravel are signed using the protocol message authentication code so that the underlying value cannot be tampered with once it is encrypted.

## Hashing

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. The Laravel `Hash` facade provides secure Bcrypt and Argon2 hashing for storing user passwords.

Bcrypt is a great choice for hashing passwords because its "work factor" is adjustable, which means that the time it takes to generate a hash can be increased as hardware power increases. When hashing passwords, slow is good. The longer an algorithm takes to hash a password, the longer it takes malicious users to generate "rainbow tables" of all possible string hash values that may be used in brute force attacks against applications.The default hashing driver for your application is configured in your application's `config/hashing.php` configuration file.

### Resseting password

Most web applications provide a way for users to reset their forgotten passwords. Rather than forcing you to re-implement this by hand for every application you create,Laravel provides convenient services for sending password reset links and secure resetting passwords.

## Database

Currently, Laravel provides first-party support for five databases:

- MariaDB 10.3+ (Version Policy)
- MySQL 5.7+ (Version Policy)
- PostgreSQL 10.0+ (Version Policy)
- SQLite 3.8.8+
- SQL Server 2017+ (Version Policy)

## Query builder in database

Laravel's database query builder provides a convenient, fluent interface to creating and running database queries. It can be used to perform most database operations in your application and works perfectly with all of Laravel's supported database systems.

## Pagintion

The pagination optionspecifies which view should be used to create pagination links. By default, Laravel includes two views. The pagination::slider view will show an intelligent "range" of links based on the current page, while the pagination::simple view will simply show "previous" and "next" buttons.

## Migration

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.

## Database seeders

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.

## Redis in laravel

Redis is an open source, advanced key-value store. It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets, and sorted sets. Before using Redis with Laravel, we encourage you to install and use the phpredis PHP extension via PECL.

## Eloquent

Eloquent is an object relational mapper (ORM) that is included by default within the Laravel framework. An ORM is software that facilitates handling database records by representing data as objects, working as a layer of abstraction on top of the database engine used to store an application's data.

## Laravel chunk

The chunk() method can be used on the DB facade and also on Eloquent models. The chunk()takes care of fetching a small amount of data at a time and the result is present inside the closure for processing.

## Relationship in database in laravel

Database tables are often related to one another. For example, a blog post may have many comments or an order could be related to the user who placed it. Eloquent makes managing and working with these relationships easy, and supports a variety of common relationships:

- One To One
- One To Many
- Many To Many
- Has One Through
- Has Many Through
- One To One (Polymorphic)

- One To Many (Polymorphic)
- Many To Many (Polymorphic)

### Laravel collection

Laravel collections can be regarded as modified versions of PHP arrays. They are located in the Illuminate\Support\Collection directory and provide a wrapper to work with data arrays.

### Mutators/Casts

In Laravel, mutators and accessors allow you to alter data before it's saved to and fetched from a database. To be specific, the mutator allows you to alter data before it's saved to a database. On the other hand, the accessor allows you to alter data after it's fetched from a database.

### API Resources

API Resources acts as a transformation layer that sits between our Eloquent models and the JSON responses that are actually returned by our API. API resources present a way to easily transform our models into JSON responses . This is the representation of the database api we are gonna create.

### Serialization in laravel

It's a way to store the values in an object into a text string format.

### Factories in laravel

When testing your application or seeding your database, you may need to insert a few records into your database. Instead of manually specifying the value of each column, Laravel allows you to define a set of default attributes for each of youreloquent models using model factories.

To see an example of how to write a factory, take a look at the `database/factories/UserFactory.php` file in your application. This factory is included with all new Laravel applications and contains the following factory definition.

### Testing 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 `tests` directory contains two directories: `Feature & 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. Tests within your "Unit" test directory do not boot your Laravel application and therefore are unable to access your application's database or other framework services.

### HTTP testing

Laravel provides a very fluent API for making HTTP requests to your application and examining the responses.To make a request to your application, you may invoke the `get`, `post`, `put`, `patch`, or `delete` methods within your test. These methods do not actually issue a "real" HTTP request to your application. Instead, the entire network request is simulated internally.

### Database Testing

Laravel provides a variety of helpful tools and assertions to make it easier to test your database driven applications. In addition, Laravel model factories and seeders make it painless to create test database records using your application's Eloquent models and relationships.

### Mocking in testing

When testing Laravel applications, you may wish to "mock" certain aspects of your application so they are not actually executed during a given test. For example, when testing a controller that dispatches an event, you may wish to mock the event listeners so they are not actually executed during the test.

## Libraries and Modular

Laravel is very popular as some Object-oriented libraries, and pre-installed libraries are added in this framework, these pre-installed libraries are not added in other php frameworks. One of the most popular libraries is an authentication library that contains some useful features such as password reset, monitoring active users, Bcrypt hashing, and CSRF protection. This framework is divided into several modules that follow the php principles allowing the developers to build responsive and modular apps.

### Laravel Dusk

Laravel Dusk provides an expressive, easy-to-use browser automation and testing API. By default, Dusk does not require you to install JDK or Selenium on your local

computer. Instead, Dusk uses a standalone ChromeDriver installation. However, you are free to utilize any other Selenium compatible driver you wish.

### Laravel Envoy

Laravel Envoy is a tool for executing common tasks you run on your remote servers. Using Blade style syntax, you can easily setup tasks for deployment, Artisan commands, and more. Currently, Envoy only supports the Mac and Linux operating systems. However, Windows support is achievable using WSL2.

### Laravel Fortify

Laravel Fortify is a frontend agnostic authentication backend implementation for Laravel. Fortify registers the routes and controllers needed to implement all of Laravel's authentication features, including login, registration, password reset, email verification, and more.

### Laravel Homestead

Laravel Homestead is an official, pre-packaged Vagrant box that provides you a wonderful development environment without requiring you to install PHP, a web server, and any other server software on your local machine. Vagrant provides a simple, elegant way to manage and provision Virtual Machines.

### Laravel Horizon

Laravel Horizon provides a beautiful dashboard and code-driven configuration for your Laravel powered Redis queues. Horizon allows you to easily monitor key metrics of your queue system such as job throughput, runtime, and job failures.

### Laravel Jetstream

Jetstream provides the implementation for your application's login, registration, email verification, two-factor authentication, session management, API via Laravel Sanctum , and optional team management features. Jetstream is designed using Tailwind CSS and offers your choice of Livewire or Inertia scaffolding.

### Laravel Mix

Laravel Mix, a package developed by Laracasts creator Jeffrey Way, provides a fluent API for defining webpack build steps for your Laravel application using several common CSS and JavaScript pre-processors. In other words, Mix makes it a cinch to compile and minify your application's CSS and JavaScript files.

### Laravel Passport

Laravel Passport **provides a full OAuth2 server implementation for your Laravel application in a matter of minutes**. Passport is built on top of the League OAuth2 server that is maintained by Andy Millington and Simon Hamp. This documentation assumes you are already familiar with OAuth2.

### Laravel Sanctum

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.

### Laravel Telescope

Laravel Telescope **makes a wonderful companion to your local Laravel development environment**. Telescope provides insight into the requests coming into your application, exceptions, log entries, database queries, queued jobs, mail, notifications, cache operations, scheduled tasks, variable dumps, and more.

### Laravel Valet

Laravel Valet is a development environment for macOS minimalists. Laravel Valet configures your Mac to always run Nginx in the background when your machine starts. Then, using DnsMasq, Valet proxies all requests on the *. test domain to point to sites installed on your local machine.

**Chmod -R a+rwx public/assets/image**

**-> gives access to the folder to read, write the data**