

11. Laravel 8

- 1) What is Framework & Why?
- 2) PHP frameworks
- 3) Laravel Directory Structure
- 4) Namespace
- 5) MVC
- 6) Routing
- 7) View Template Engine (blade)
- 8) Controller
- 9) Request & Response
- 10) Model (Database)
- 11) Eloquent ORM
- 12) Middleware
- 13) CSRF Protection
- 14) Session
- 15) Validation & Error Handling
- 16) Logging
- 17) HTTP Client
- 18) Localization
- 19) File Storage Management
- 20) Sending Mail
- 21) Factory & Seeding

- 22) Pagination
- 23) Eloquent Relationship & Serialization
- 24) Encryption & Hashing
- 25) Laravel Mix
- 26) Authentication & Authorization
- 27) API

1) What is Framework & Why?



Framework ဆုတေယာကတော့ programming Language တခုခုပေါ်မှာ အခြေခံပြီး software development တွေဖန်တီးတဲ့အခါ ပိုမိုလွယ်ကူစေနိုင်ရန်အတွက် ဖြစ်ပါတယ်။ ဥပမာ အိမ်တလုံးစောက်တဲ့အခါကိုယ်တိုင်တည်စောက် နေတာထက်ကျမ်းကျင်တဲ့ လူတွေခေါ်ပြီး တည်စောင်ခိုင်းတာနဲ့ အချင်နဲ့ Qualityဟာ သိသိသာသာ ကွာခြားစေပါတယ်။

Framework သုံးရတဲ့ ရည်ရွယ်ချက်တွေကတော့

1. Security ပိုမိုလုံခြုံစေရန်
 2. Testing နဲ့ Debugging တွေလုပ်နိုင်အောင်
 3. ထပ်နေတဲ့ code တွေကို ရှင်းလင်းနိုင်အောင်
 4. Development Processing ရှင်းလင်းလွယ်ကူနိုင်အောင်
- စတဲ့အချက်တွေ ကြောင့် framework တွေကို သုံးရတာဖြစ်ပါတယ်။

2) PHP Frameworks

Programming language တိုင်းမှာ framework တွေရှိကြပါတယ်။ PHP ဟာ Web Development ပိုင်းမှာ အားကောင်းတဲ့ language တခုဖြစ်တာကြောင့် framework တွေဟာ များပြားပါတယ်။ Laravel,cake php,Yii,Zend Framework,Fuel PHP,Slim .., စတဲ့ framework တွေ များပြားစွာ ရှိပါတယ်။



Extension list for vscode

1. Laravel Artisan
2. Laravel Blade Snippets
3. Laravel Blade Spacer
4. Laravel Easy Blade Snippets
5. Laravel Extra Intellisense
6. Laravel goto view
7. Laravel Snippets
8. Path Intellisense

3) Laravel Directory Structure

■ app

App ထဲမှာတော့ Business Logic တွေ အတွက် အသုံးပြုရမယ့် File တွေ၊ error handling, controller , database modal file တွေ ပါဝင်ပါတယ်။

■ bootstrap

Environment setup အတွက် bootstrap file တွေပါဝင်ပါတယ်။

■ config

Project configuration setting တွေသတ်မှတ်ဖို့အတွက် file တွေပါဝင်ပါတယ်။

■ database

ကိုယ့်ရဲ Database schema တွေ Modify လုပ်တဲ့အခါ အသုံးပြုတဲ့ file တွေ ပါဝင်ပါတယ်။

■ public

Public folder ထဲမှာတော့ application ကို စစချင်းဝင်ရောက်တဲ့ entry file တွေ ပါဝင်ပါတယ်။

Example index.php

■ resources

Resource ထဲမှာတော့ Css, js စတဲ့ file တွေအပြင် Authentication, pagination, validation စတဲ့ file တွေပါဝင်ပါတယ်။

■ routes

Application စစချင်း ဝင်လာတဲ့အခါ သတ်မှတ်ထားတဲ့လမ်းကြောင်းတွေကို လမ်းလွှဲပေးဖို့ route file,api response တွေအတွက် api file စတဲ့ file တွေပါဝင်ပါတယ်။

■ storage

File တွေသိမ်းဆည်းဖို့ အတွက် လမ်းညွှန်ပေးတဲ့ file တွေပါဝင်ပါတယ်။

■ tests

Testing အတွက် testcase file တွေပါဝင်ပါတယ်။

■ vendor

Applicationမှာအသုံးပြုထားတဲ့ third-party file တွေရဲ့ library file တွေပါဝင်ပါတယ်။

4) Namespace

Class ရဲ့ name တွေ သို့မဟုတ် class တွေထဲမှာရှိတဲ့ method name တွေတူညီနေပြီး
တို့၏ php file ကနေ required နဲ့ခေါ်သုံးဖို့လိုတဲ့အခါ name တွေတူညီနေတဲ့အတွက်
Already use ဆိုတဲ့ error message တွေပြပါတယ်။ဒါပြသာကို ဖြေရှင်းဖို့အတွက်
namespace ကို အသုံးပြနိုင်ပါတယ်။ဒါလိုတည်ဆောက်ထားတဲ့ Class တွေအတွက်

Employee.php

```
class payment
{
    public function pay()
    {
        echo "Pay with CB Bank";
    }
}
```

Customer.php

```
class payment
{
    public function pay()
    {
        echo "Pay with Kpay";
    }
}
```

Index.php ကနေ ခေါ်တဲ့အခါ

```
require "Employee.php";
require "Customer.php";

$empPay = new payment();
$empPay->pay();
echo "<br>";
$cusPay = new payment();
$cusPay->pay();
```

Fatal Error တတ်ပါတယ်။

Fatal error: Cannot declare class payment, because the name is already in use in /Applications/XAMPP/xamppfiles/htdocs/Customer.php on line 3

Namespace သုံးပြီးခေါ်တဲ့အခါမှာတော့ Class တွေရဲ့ declaration တွေအပေါ်မှာ

```
namespace empPayment;
class payment
{
    public function pay()
    {
        echo "Pay with CB Bank";
    }
}
namespace cusPayment;
class payment
{
    public function pay()
    {
        echo "Pay with Kpay";
    }
}
```

Index.php မှာ namespaceနဲ့သတ်မှတ်ထားတဲ့ Classကိုခေါ်တဲ့အခါ\ ခံပြီးခေါ်သုံးရပါတယ်။

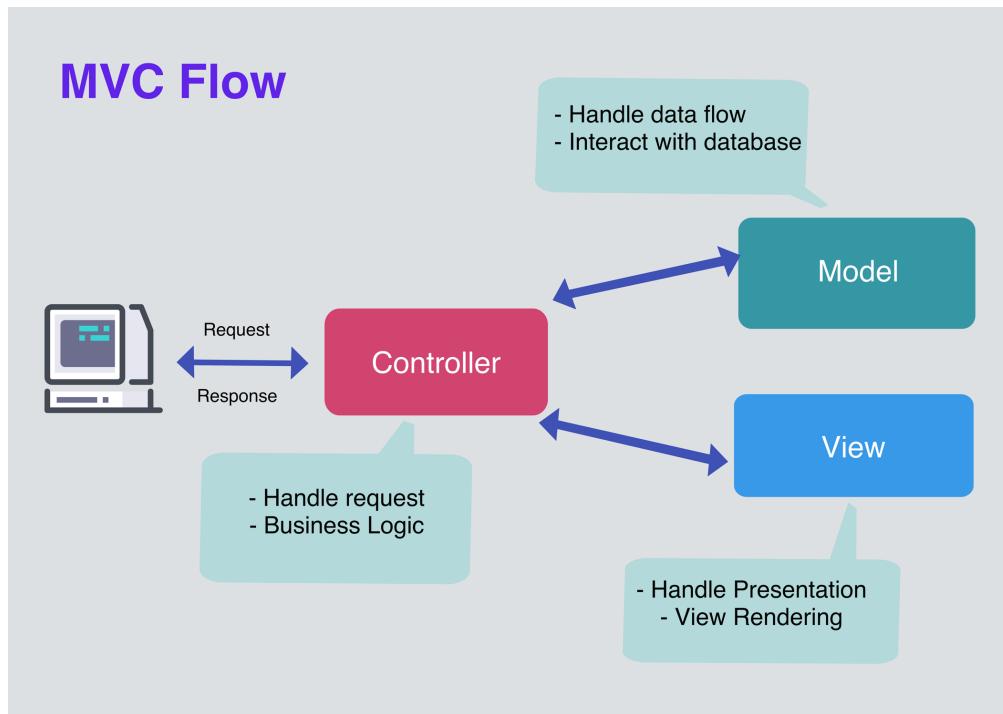
```
require "Employee.php";
require "Customer.php";

$empPay = new empPayment\payment();
$empPay->pay();
echo "<br>";
$cusPay = new cusPayment\payment();
$cusPay->pay();
```

Browser result မှာတော့သတ်မှတ်ထားတဲ့ အတိုင်းပေါ်လာမှာဖြစ်ပါတယ်။

```
//Pay with CB Bank
//Pay with Kpay
```

5) MVC



MVC (Model view Controller) မှာဆိုရင် web app တဲကို route ကနေတစ်ခု ဝင်လာတဲ Request တွေအရ business logic တွေကို controller က စီမံပြုးလိုအပ်တဲ data တွေကို database မှာ CRUD လုပ်ပြုး View မှာ render ပြချပြတဲ flow ကို MVC လိုခေါ်ပါတယ်။

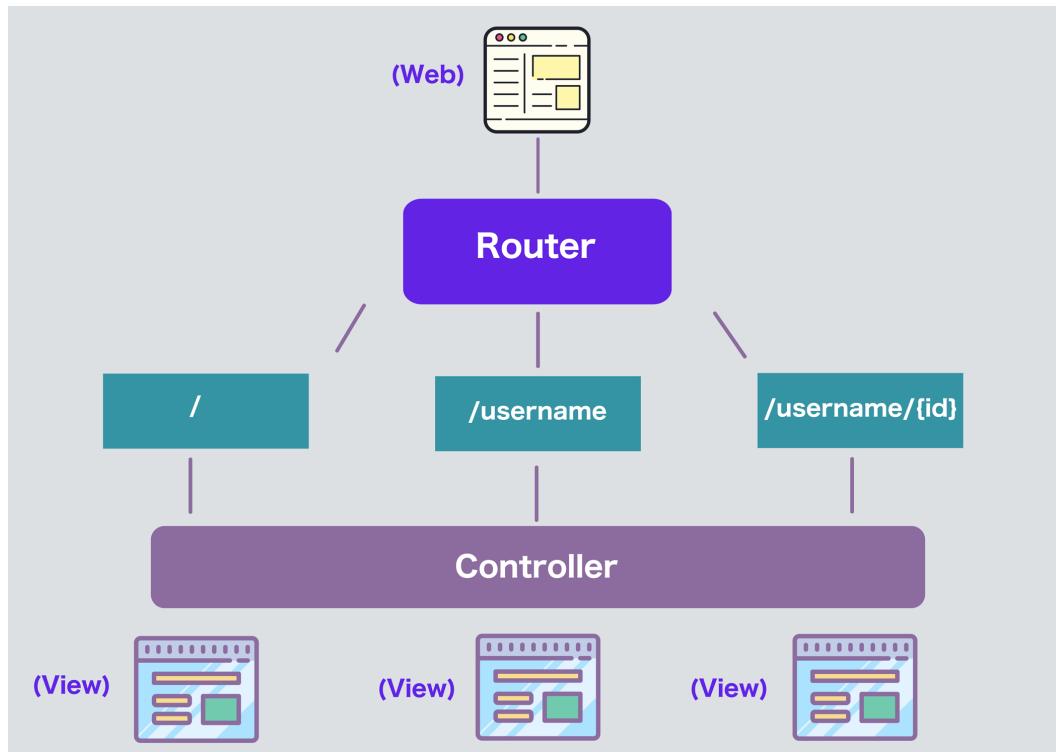
Controller Client ဘက်က request တွေကို ထိန်းချုပ်ပေးပြုး လိုအပ်တဲ logic တွေကို ရေးသားနိုင်တဲ နေရာဖြစ်ပါတယ်။

Model Controller မှာလိုအပ်တဲ data တွေကို ပိုပေးနိုင်ဖို့ database နဲ့ချိတ်ဆတ်ပေးတဲ နေရာ ဖြစ်ပါတယ်။

View Controller ကနေ Data တွေကို user ဆီ presentation လုပ်ပေးတဲ နေရာ ဖြစ်ပါတယ်။

6) Routing

Web Application ရဲ့ entry လို့ပြောလို့ရတဲ့ နေရာ တခုဖြစ်ပါတယ်။ ဝင်လာတဲ့ route တွေပေါ်မှာမူတည်ပြီး လမ်းကြောင်းလွှဲပေးရတဲ့ အလုပ်ကို လုပ်ပါတယ်။



Route တခုစွဲပြီး ဝင်လာတဲ့ အခါ controller ထဲ သွားမလား View ကို
သွားမလားစတဲ့ လမ်းကြောင်းတွေကို transfer လုပ်ပေးတဲ့ အနေနဲ့ သုံးပါတယ်။
API Route file ရှိသလို web အတွက် route ခွဲထားနိုင်ပါတယ်။

Type of Router Method

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

Routing Category

Simple Routing

```
Route::get('/', function () {
    return 'Hello';
});
```

```
Route::get('/home', function () {
    return 'home';
});
```

Redirect Routing

```
Route::get('/userinfo', function () {
    return redirect('/');
});
```

View Routing

```
Route::get('/home', function () {
    return View("home.index");
});
```

Controller Routing

```
Route::get('home', [HomeController::class, 'homePage']);
```

Route with parameters

```
Route::get('/userinfo/{id}', function ($id) {
    return "User Info $id";
});
```

Optional Parameter

```
Route::get('/address/{postalCode?}', function ($code = 10101) {
    return "Postal Code is $code";
});
```

ကိုယ့် app မှာရှိတဲ့ route တွေကို ကြည့်ချင်တဲ့အခါ

Command: php artisan route:list

7) View Template Engine (blade)

Pure Php မှာ view တွေဆီ variable တွေ ပို့တဲ့အခါ php code တွေတဆင့်ခံရတာ၊ layout တွေကို condition တွေနဲ့ထုတ်ပြီး ပြတာ|common ဖြစ်တဲ့ layout တွေကို ခေါ်သုံးတဲ့အခါ စတဲ့ layout Management တွေအတွက် ပိုမိုလွယ်ကူနိုင်အောင် framework တွေနဲ့ Template Engine တွေကို တွဲသုံးကြပါတယ်။ Laravel မှာတော့ blade ဆိုတဲ့ powerful ဖြစ်တဲ့ Template Engine ကို သုံးကြပါတယ်။

Name ပေးတဲ့ rule ကတော့ file name ရဲ့နောက်မှာ [blade.php](#) နဲ့ရေးသားရပါတယ်။

■ Layouts Design

Html layout တခုကို common ထားပြီး ကျန်တဲ့ page တွေက ကိုယ်ပိုင် data တွေနဲ့ access လုပ်ပြီး သုံးချင်တဲ့အခါ layout format ချရပါတယ်။

```
Layout/app.blade.php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title> @yield('title')</title>
</head>
<body>
    <div>
        @yield('body')
    </div>
</body>
</html>
```

```
Home/home.blade.php
@extends('layout.app')
@section('title', 'Home')

@section('body')
    <h2>Home Page</h2>
    <p>Welcome From Our Page</p>
@endsection
```

@yield('name') ကတေသူသူနေရာမှာအစားထိုးနိုင်ဖို့နာမည်ပေးထားတဲ့ keywordဖြစ်ပါတယ်။

@extends ကတေသူ layout ချထားတဲ့ template ကိုခေါ်သုံးချင်တဲ့အခါ သုံးပါတယ်။

@section , @endsection ကတေသူ layout မှာရှိတဲ့ @yield ရဲ့ name တွေကိုခေါ် ပြီးပြန်အစားထိုးတဲ့အခါသုံးပါတယ်။

- text only or one line ဆို `@section('title','Home')`
- Other code or multi line ဆို `@section('name') //code for something @endsection`

■ Deliver parameter to Layout template

Route ကနေတဆင့် view ဆိုကို data တွေပို့တဲ့အခါ

Routes/web.php

```
Route::get('/userinfo', function () {  
    $info = [  
        'name' => 'john',  
        'age' => 23,  
        'married' => false  
    ];  
    return view('home.home', ['user' => $info]);  
});
```

```
@extends('layout.app')
```

```
@section('title','Home')
```

```
@section('body')  
<p>UserName is {{ $user['name'] }}</p>  
<p>Age is {{ $user['age'] }}</p>  
@if ( $user['married'] )  
    <p>Married</p>  
@else  
    <p>Single</p>  
@endif
```

```
@endsection
```

Conditional Flow တွေမှာဆိုရင် အောက်ပါtemplate Command တွေရှိပါတယ်။

- **@if @elseif @else**
 - **@unless @endunless**
 - **@isset @endisset**
 - **@empty @endempty**
 - **@switch(\$i)**
 - @case(1)**
First case...
 - @break**
 - @case(2)**
Second case...
 - @break**
 - @default**
Default case...
 - @endswitch**
-
- **@for (\$i = 0; \$i < 10; \$i++)**
The current value is {{ \$i }}
@endfor
 - **@foreach (\$users as \$user)**
<p>This is user {{ \$user->id }}</p>
@endforeach
 - **@while (true)**
<p>I'm looping forever.</p>
@endwhile
 - **@break(#condition)**
 - **@continue(#condition)**
 - **@forelse (\$users as \$user)**
{{ \$user->name }}
@empty
<p>No users</p>
@endforelse

8) Controller

Controller ကတေသ့ business Logic တွေကို အမိန်ထားရေးတဲ့ နေရာ တခုဖြစ်ပါတယ်။

Controller တခု တည်ဆောင်ဖို့အတွက် command ကတေသ့

Command: `php artisan make: controller Controller-name`

Example :

```
<?php  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
  
class HomeController extends Controller  
{  
}
```

စုပြီး Create လုပ်လိုက်တဲ့ အခါ အပေါ်ကပုံစံအတိုင်း Default တည်ဆောက်ပြီး သား တွေရပါလိမ့်မယ်။ Controller တခုတည်ဆောက်လိုက်တိုင်းမှာ Laravel ရဲ့ base controllers တွေပေါ်မှာ extends လုပ်ထားတာတွေရပါလိမ့်မယ်။ Controller ထဲမှာတော့ function တွေခဲ့ပြီး Business Logic တွေရေးသားနိုင်ပါတယ်။ function တွေခဲ့ပြီး controller တခုတည်းမှာပဲရေးသားနိုင်တဲ့ အတွက် code တွေကို ရှင်းလင်းစေပါတယ်။

Example:

```
Route::get('home', [HomeController::class, 'home']);  
Route::get('about', [HomeController::class, 'aboutUs']);
```

```
class HomeController extends Controller  
{  
    public function home()  
    {  
        return view('home.home');  
    }  
    public function aboutUs()  
    {  
        return view('about.about');  
    }  
}
```

Resource Controller

Controller တည်ဆောက်တဲ့အခါ ပိုပြီး powerful ဖြစ်စေတဲ့ controller တည်ဆောက်နည်းဖြစ်ပါတယ်။

Command: `php artisan make: controller Controller-name —resource`

ဒီလိုတည်ဆောက်လိုက်ရင်တော့ Controller မှာ function 7ခုကို Default တည်ဆောက်ပေးထားမှာဖြစ်ပါတယ်။

- **Index()** // url က get Method နဲ့ user/ လိုခေါ်လိုက်တဲ့အခါရောက်မယ့်function
- **Create()** url က user/create လိုခေါ်လိုက်တဲ့အခါရောက်မယ့်function
- **Store()** // url က post Method နဲ့ user/ လိုခေါ်လိုက်တဲ့အခါရောက်မယ့်function
- **Show()** // url က get Method နဲ့ user/{user}လိုခေါ်လိုက်တဲ့အခါရောက်မယ့်function
- **Edit()** // url က get Method နဲ့ user/{user}/edit လိုခေါ်လိုက်တဲ့အခါရောက်မယ့်function
- **Update()** // url က put Method နဲ့ user/{user}လိုခေါ်လိုက်တဲ့အခါရောက်မယ့်function
- **Destroy()** // url က delete Method နဲ့ user/{user}လိုခေါ်လိုက်တဲ့အခါရောက်မယ့်function

Route ကနေ လမ်းကြောင်းပေးတဲ့အခါဒီလိုရေးနိုင်ပါတယ်။

`Route::resource('user', UserController::class);`

9) Request & Response

Request

HTTP ကနေတဆင့် web app ထဲကိုပါလာတဲ့ request တွေကိုပြန်ဖမ်းတဲ့အခါ Request Class ကိုသုံးပါတယ်။ Laravel ရဲ့ service container ကနေတဆင့် ဝင်လာတဲ့ request တွေကို auto inject လုပ်ပြီးဖမ်းသွားမှာဖြစ်ပါတယ်။

HomeController.php

```
public function home(Request $request)
{
    //write here $request Methods
}
```

■ `$request->all();`

Request မှာပါလာသမျှအကုန် ထုတ်ပြတဲ့အခါ သုံးပါတယ်။

■ `$request->except('key');`

Request ထဲမှာပါလာတဲ့ parameter တွေထဲက ပေးလိုက်တဲ့ key ကလွှဲလိုကျန်တာ ကိုထုတ်ပြချင်တဲ့အခါ သုံးပါတယ်။ တခုထက်ပိုတဲ့အခါ [] ကိုသုံးပါတယ်။

■ `$request->only('key');`

Except နဲ့ ပြောင်းပြန်ဖြစ်ပါတယ်။ Request ထဲမှာပါလာတဲ့ parameter တွေထဲက ပေးလိုက်တဲ့ key ကိုသာထုတ်ပြပြီး ကျန်တာ ကိုထုတ်မပြချင်တဲ့အခါ သုံးပါတယ်။ တခုထက်ပိုတဲ့အခါ [] ကိုသုံးပါတယ်။

■ `$request->has('key');`

Request ထဲမှာပါလာတဲ့ parameter တွေထဲက ကိုယ်လိုချင်တဲ့ key ပါလား မပါလား စစ်ချင်တဲ့အခါ သုံးပါတယ်။ true နဲ့ false ကို return ပြန်ပေးပါတယ်။

■ `$request->input('key');`

Request မှာပါလာတဲ့ parameter တွေက key ပေးပြီး value ထုတ်ပြတဲ့အခါ သုံးပါတယ်။

■ `$request->input('key', 'default value');`
Request မှာပါလာတဲ့ parameter တွေထဲလိုချင်တဲ့ key ကပါလာခဲ့ရင်ယူပြီး
ပါမလာခဲ့ရင် default value ကို ထုတ်ပြတဲ့အခါ သုံးပါတယ်။

■ `$request->path();`
Domain နောက်က Path လမ်းကြောင်း ထုတ်ပြတဲ့အခါ သုံးပါတယ်။

■ `$request->url();`
Parameter မပါတဲ့ URL လမ်းကြောင်းကို ထုတ်ပြတဲ့အခါ သုံးပါတယ်။

■ `$request->fullUrl();`
URL အပြည့်အစုံ ကိုထုတ်ပြတဲ့အခါ သုံးပါတယ်။

■ `$request->method();`
Request လုပ်လာတဲ့ Method ကို ထုတ်ပြတဲ့အခါ သုံးပါတယ်။

■ `$request->isMethod('name');`
Request လုပ်လာတဲ့ Method ကို လိုချင်တဲ့ Method ဟုတ်မဟုတ်စစ်တဲ့အခါ
သုံးပါတယ်။ true နဲ့ false return ပြန်ပါတယ်။

■ `$request->ip();`
IP address ကို ထုတ်ပြတဲ့အခါ သုံးပါတယ်။

■ `$request->header('header-key');`
Request header ထဲမှာပါလာတဲ့ header value တွေကိုထုတ်ပြတဲ့အခါ သုံးပါတယ်။

■ `$request->is('home');`
Request လမ်းကြောင်းက သတ်မှတ်ထားတဲ့လမ်းကြောင်းဟုတ်မဟုတ်စစ်တဲ့အခါ
သုံးပါတယ်။

■ `$request->routess('home');`
Route မှာ နာမည်ပေးထားတဲ့ name ဟုတ်မဟုတ် စစ်တဲ့အခါ သုံးပါတယ်။

Response

Route ဆီရောက်လာတဲ့ request တွေကို response ပြန်ပေးဖို့အတွက် Response method ကိုအသုံးပြုပါတယ်။

String Response

```
Route::get('/', function () {
    return 'Hello World';
});
```

Array Response

```
Route::get('/', function () {
    return ["John", 25, "Yangon"];
});
```

Header Response

```
Route::get('/', function () {
    return response('<h1>Hello World</h1>', 200)
        ->header('Content-Type', 'text/html');
});
```

Cookie Response

Cookie('name','value',minutes)

```
Route::get('/', function () {
    return response('<h1>Hello World</h1>', 200)
        ->header('Content-Type', 'text/html')
        ->cookie('name', 'John', 0.5);
});
```

Json Response

```
Route::get('/', function () {
    return response()
        ->json([
            'name' => 'John',
            'address' => 'Yangon',
            'age' => 25
        ]);
});
```

Download Response

```
Route::get('/', function () {  
    return response()->download(public_path('test.txt'), 'New File');  
});
```

File Response

```
Route::get('/', function () {  
    return response()->file(public_path('test.pdf'));  
});
```

10) Model (Database)

Model မှာတော့ Data flow တွေကို control လုပ်ပေးပြီးတော့ database နဲ့ချိတ်ဆက်တဲ့အခါ CRUD တွေအတွက် အသုံးပြုပါတယ်။ Laravel ဟာ powerful ဖြစ်တဲ့ framework ဖြစ်တာကြောင့် less code နဲ့ database ရဲ့ CRUD Statements တွေကို ရေးသားနိုင်ပါတယ်။ Database နဲ့ချိတ်ဆက်ဖို့ Laravel ရဲ့ .env file မှာ setting ချိန်ရပါတယ်။

```
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=exbrain
15 DB_USERNAME=root
16 DB_PASSWORD=
17
```

DB_CONNECTION -> မှာတော့ mysql

DB_HOST -> ကတော့ localhost ဆို 127.0.0.1 သို့မဟုတ် domain

DB_PORT -> ကတော့ MySQL ဖွင့်ထားတဲ့ port

DB_DATABASE -> ချိတ်မယ့် database name

DB_USERNAME -> MySQL ရဲ့username

DB_PASSWORD -> MySQL ရဲ့password

Setting ချိန်ပြီးတဲ့အခါ default table တွေ တည်ဆောက်ဖို့ သို့မဟုတ် database နဲ့ connection ဖြစ်မဖြစ်စစ်နိုင်ဖို့ command ကို အသုံးပြုပါတယ်။

Command : php artisan migrate

Model without migration

Model တရာ့တည်ဆောက်တဲ့ Command ကတော့

Command : `php artisan make:model model-name`

ပုံမှန်အားဖြင့် Model နဲ့ database table တည်ဆောက်တဲ့အခါ လိုက်နာရမယ့် rule ကတော့ database table မှာ plural format ဖြစ်ပြီး Model single format ဖြစ်ရပါမယ်။

Example:

Database table name : customers

Model name : customer

သို့ပေမယ့် မတူတာတွေကိုပေးချင်တဲ့အခါ User model မှာ

`public $table = "name";`

မှာပေးလို့ရပါတယ်။

Model with -m (migration)

Model class တရာ့ကို migration(-m) နဲ့ တည်ဆောက်တဲ့အခါ

Command : `php artisan make:model model-name -m`

Database folder ဒေါက်ကmigration folder မှာ migration class တရာ့ကိုပါ
တည်ဆောက်ပေးပါတယ်။

ဒါclass ထဲမှာတော့ up function နဲ့ down function ဆိုပြီး ရှိပြီး up ကတော့ create table (table
မှာပါဝင်တဲ့ column တွေကို တည်ဆောက်ပေးတဲ့ function) နဲ့ down (table ရှိနေတဲ့အခါ drop
လုပ်ပေးတဲ့ function) တွေပါဝင်ပါတယ်။ပြ function မှာ လိုချင်တဲ့columnတွေကို
စီစဉ်ပြီးတဲ့အခါ migration command ကို run ပေးရပါတယ်။

Command : `php artisan migrate`

MySQL Database မှာ up function မှာရေးထားတဲ့ create table structure အတိုင်း
တည်ဆောက်ထားတာတွေရပါလိမ့်မယ်။

11) Eloquent ORM(Object Relation Mapping) And Query Builder

Query Builder

ပုံမှန် sql query statement တွေကို ရေးသားတဲ့နည်းဖြင့်ပါတယ်။ Eloquent Model နဲ့ယူဉ်ရင် ပိုမြန်မြန်ဆန်ဆန် data communication လုပ်နိုင်ပါတယ်။

```
use Illuminate\Support\Facades\DB;
```

```
DB::table('customers')
```

```
DB::select(query) & ->select('column')
```

1. DB::select('select * from customers where id = ?', [2]);
2. DB::select('select * from customers where id = :id', ['id' => 2]);
3. DB::table('customers')
 ->select('name')
 ->get();

```
->where('column','operator'=optional,value)
```

```
DB::table('customers')  
    ->where('id', 3)  
    ->get();
```

```
->orWhere('column','operator'=optional,value)
```

```
DB::table('customers')  
    ->where('name', 'mary')  
    ->orWhere('age', '>', 24)  
    ->get();
```

```
->whereBetween('column',[value])
```

```
DB::table('customers')  
    ->whereBetween('age', [20, 26])  
    ->get();
```

```
->distinct('column')
```

```
DB::table('customers')  
    ->distinct('name')  
    ->get();
```

```
->orderBy('column','option')
    DB::table('customers')
        ->orderBy('name', 'desc')
        ->get();
```

```
->first()
    DB::table('customers')
        ->first();
```

```
->find()
    DB::table('customers')
        ->find(1);
```

```
->insert([key=>value])
    DB::table('customers')
        ->insert([
            'name' => "John",
            'age' => 34
        ]);
```

```
->update
    DB::table('customers')
        ->where('id',4)
        ->insert([
            'name' => "John",
            'age' => 34
        ]);
```

Query Builder နဲ့ insert update လုပ်တဲ့အခါ create date ,update date ထွက်ရှိ time stamp ထည့်မပေးပါဘူး။ Manual ထည့်ပေးရပါတယ်။ Eloquent မှာတော့auto timestamp insert လုပ်ပေးပါတယ်။

```
->delete
    DB::table('customers')
        ->where('id',4)
        ->delete();
```

For More method

<https://laravel.com/docs/8.x/queries>

Eloquent Model

Laravel ရဲ့ powerful အဖြစ်ဆုံးတဲက တခုဖြစ်တဲ့ Eloquent ဟာ database နဲ့ communication လုပ်နိုင်တဲ့ object mapping ဖြစ်ပါတယ်။ query build လုပ်တဲ့နေရာမှာ

Less code နဲ့ database table တွေ ဆီ CRUD လုပ်နိုင်တဲ့ object mapping တခုဖြစ်ပါတယ်။

■ \$ModelClass::all()

```
Customer::all();  
//SELECT * FROM Customers
```

■ \$ModelClass::find(*id*)

```
Customer::find(1);
```

■ \$ModelClass::create(*array*)

```
Customer::create([  
    'name' => 'Linn',  
    'age' => 27  
]);
```

Array နဲ့ data pass လုပ်တဲ့အခါ Model မှာ data တွေကို approve ဖြစ်အောင်\$fillable မှာဖြည့်ပေးရပါတယ်။

```
public $fillable = ['name', 'age'];
```

■ \$model->save()

```
//call customer model  
$customer = new Customer();  
$customer->name = "Mary";  
$customer->age = 27;  
$customer->save();
```

■ \$ModelClass::get()

```
Customer::get();
```

■ \$ModelClass::first()

```
Customer::first();
```

■ \$ModelClass::select(['*column_name*',...])

```
Customer::select(['name', 'age'])->get();
```

- **\$ModelClass::where('condition')**
Customer::where('id', '=', 2)->get();
- **\$ModelClass::orderBy('column_name','desc')**
Customer::orderBy('name', 'desc')->get();
- **\$model->delete()**
\$customer = Customer::find(1);
\$customer->delete();
- **\$Model::whereIn('column_name',[value,...])**
Customer::whereIn('id', [1, 2])->get();

Chunk Retrieving

Database ထဲ big data တွေ ထောင်သောင်းမက ရှိနေတဲ့အခါ မျိုးတွေမှာ chunk method ကိုသုံးပြီး group တွေခွဲပြီးထုတ်ပေးပါတယ်။ ဥပမာ database ထဲမှာရှိတဲ့ user တွေခွဲဆို mail တွေပို့တဲ့အခါ chunk method ကိုသုံးပြီးပို့ရပါတယ်။

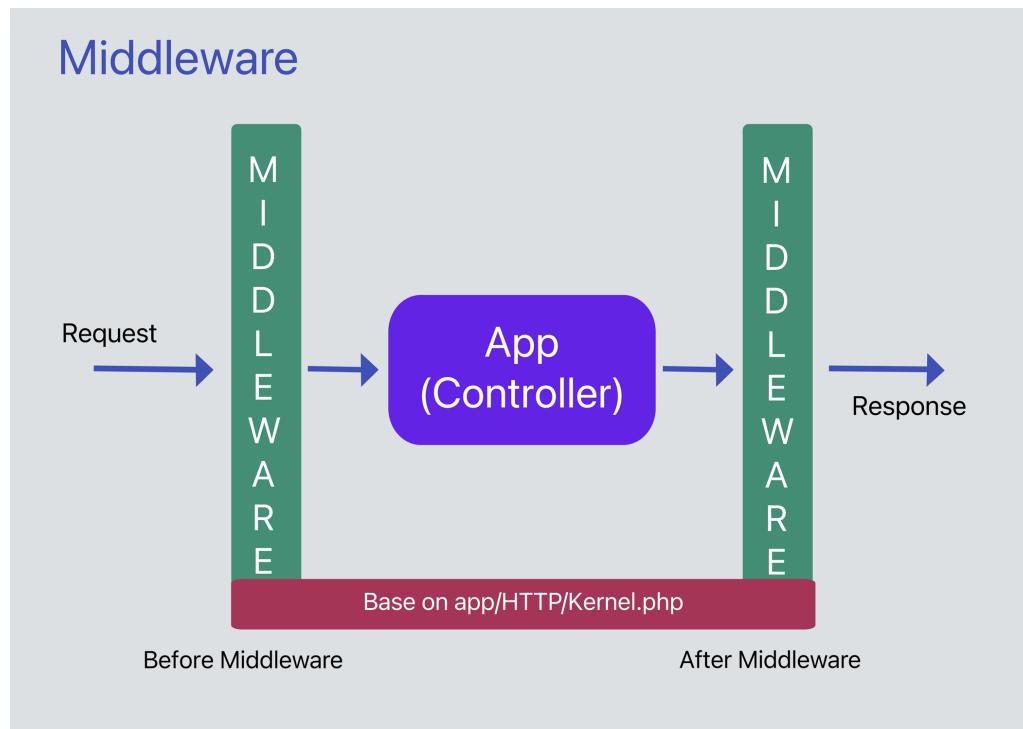
Example:

```
Customer::chunk(200, function ($customers) {
    foreach ($customers as $customer) {
        Echo $customer->name;
    }
});
```

Change CreateDate/UpdateDate Format

```
public function getCreatedAtAttribute($value)
{
    $date = Carbon::parse($value);
    return $date->format('d-m-Y');
}
```

12) Middleware



Middleware ကတေသာ app ထဲကို request တွေမရောက်ခင် response တွေထွက်မသွားခင် ကြားမှာ filter စစ်ပေးတဲ့ ကြားခံနယ်ပယ်တဲ့ဖြစ်ပါတယ်။ Security အရသော်လည်းကောင်း app ထဲ ဝင်ရောက်မလာခဲ့ လိုအပ်ချက်တွေကို စစ်ပေးတဲ့နေရာအတွက် ရှိနေတာဖြစ်ပါတယ်။ Middleware ထဲမှာတော့ authorization တွေ CSRF protection တွေ redirect တွေစဲတဲ့ Method တွေ base ခံထားပါတယ်။

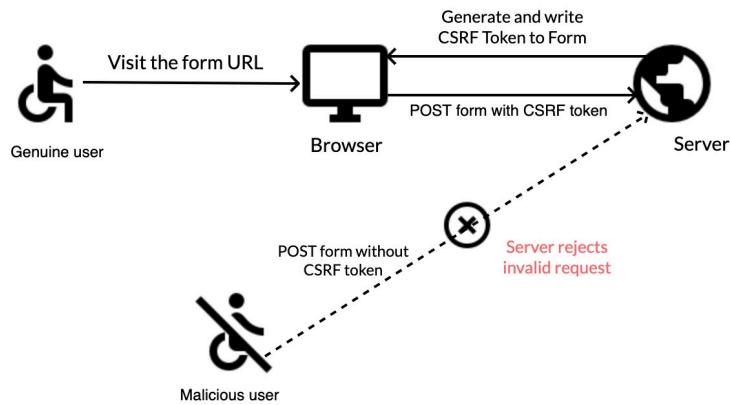
ကိုယ်ပိုင် Middleware လည်းတည်ဆောက်နိုင်ပါတယ်။

Command: `php artisan make:middleware middleware-name`

13) CSRF Protection

CSRF ဟာကိုယ့် App ထဲကို တခြား resource ကနေaccess မလုပ်နိုင်အောင် ကာကွယ်တဲ့အနေနဲ့
သုံးတဲ့technique တခုဖြစ်ပါတယ်။

ဥပမာ ကိုယ့် Laravel app ကနေ create လုပ်ထားတာမဟုတ်တဲ့ form request ကနေ route
တွေဆီ data တွေပိုလွှတ်တဲ့အခါ controllers တွေဆီမရောက်ခင် Middleware ကနေ block
လုပ်ပေးတာဖြစ်ပါတယ်။



Html form tag ရဲအောက်မှထည့်ပေးရပါတယ်။

```
<form action="path" method="post">  
@csrf
```

ဒီလိုရေးလိုက်ခြင်းအားဖြင့် Laravel run တဲ့အခါ input type hidden နဲ့ token value
တခုဖန်တီးပေးမှာဖြစ်ပါတယ်။ ဒီtoken ကို middleware ကတဆင့် စစ်ပြီး controller ထဲကို
ဝင်ခွင့်ပေးတာဖြစ်ပါတယ်။

တခြား resource တွေကနေ access လုပ်တဲ့ အခါ token တွေပါမလာတဲ့အတွက် app ဆိုကို
ဝင်ရောက်ခွင့်ရမှာမဟုတ်ပဲ denied ဖြစ်မှာဖြစ်ပါတယ်။

14) Session

HTTP protocol ဟာ stateless protocol ဖြစ်တဲ့ အတွက် request တွေဟာ ကိုယ်ပိုင်ရပ်တည်ပြီး client နဲ့ server ကား connection တခုပြီးသွားတဲ့ အခါ information တွေဟာ lost ဖြစ်သွားပါတယ်။ information တွေကို server မှာသိမ်းထားနိုင်ဖို့ session ကို သုံးတာဖြစ်ပါတယ်။ Session ကို page တိုင်းကနေခေါ်သုံးနိုင်ပါတယ်။ Laravel မှာလည်း session ပါဝင်ပါတယ်။

Laravel မှာတော့ session တွေကို သိမ်ဆည်းထားဖို့ driver မေ့မျှုးရှိပါတယ်။

- File
- Cookie
- Database
- Memcached/Redis
- Dynamodb
- Array

Store Session Data

```
$request->session()->put('key', 'value');  
OR  
session(['key' => 'value']);
```

Get Session Data

```
$value = $request->session()->get('key');  
OR  
session('key');
```

Get & Delete Session Data

```
$value = $request->session()->pull('key');  
OR  
$value = session()->pull('key');
```

Delete Session Data

```
$request->session()->forget('name');  
OR  
session()->forget('name');
```

Delete Specific Session or multiple key

```
$request->session()->forget(['key', 'value']);  
OR  
session()->forget(['key', 'value']);
```

Delete all Session Data

```
$request->session()->flush();
```

Check Session Data

```
if ($request->session()->has('users')) {  
    //  
}  
  
if ($request->session()->exists('users')) {  
    //  
}
```

15) Validation & Error Handling

Validation ဆိုတာကတော့ client ကနေ server ဆီ data တွေပို့လွတ်တဲ့အခါ data တွေဟာ valid ဖြစ်မှု တနည်းအားဖြင့် လိုချင်တဲ့ ပုံစံအနေအထားဖြစ်နေမှာ program ဟာ ကောင်းမွန်စွာ run နိုင်မှာဖြစ်ပါတယ်။ နောက်ပြီး attract လုပ်တဲ့အခါ data တွေဟာ မမျှော်လင့်ထားတဲ့ format တွေနဲ့ ဝင်လာနိုင်တဲ့အတွက် လိုအပ်တာတွေ တိတိကျကျ သတ်မှတ်ထားနိုင်ဖို့ server side မှာ rule တွေ သတ်မှတ်ထားရပါတယ်။ ဒါတွေကို validation လိုပေါ်ပါတယ်။

Validation ကို class သီးသန့် (rule သီးသန့်) ပြီး စစ်တန်း သီးသန့်မခဲ့ပဲစစ်တာ နှစ်မျိုးရှိပါတယ်။

Validation on request

```
$request->validate([
    'username' => 'required',
    'password' => 'required'
]);
```

Validation စစ်နှင့်တဲ့ Rules တွေကတော့

- Accepted
- Accepted If
- Active URL
- After (Date)
- After Or Equal (Date)
- Current Password
- Date
- Date Equals
- Date Format
- Declined
- Declined If
- Different
- Digits
- Digits Between
- Alpha
- Alpha Dash
- Alpha Numeric
- Array
- Bail
- Before (Date)
- Dimensions (Image Files)
- Distinct
- Email
- Ends With
- Enum
- Exclude
- Exclude If
- Exclude Unless
- Before Or Equal (Date)
- Between
- Boolean
- Confirmed
- Exclude Without
- Exists (Database)
- File
- Filled
- Greater Than
- Greater Than Or Equal
- Image (File)
- In

- In Array
- Integer
- IP Address
- JSON
- Less Than
- Nullable
- Numeric
- Password
- Present
- Prohibited
- Prohibited If
- Prohibited Unless
- Prohibits
- Unique (Database)
- URL
- Less Than Or Equal
- Max
- MIME Types
- MIME Type By File Extension
- Regular Expression
- Required
- Required If
- Required Unless
- Required With
- Required With All
- Required Without
- UUID
- Min
- Multiple Of
- Not In
- Not Regex
- Required Without All
- Same
- Size
- Sometimes
- Starts With
- String
- Timezone

Validation with standalone (class)

သီးခြား Validation class တစ်ညွှန်ခေါက်နှင့် Request Class တစ်အရင်တည်ဆောက်ရပါတယ်။

Command : `php artisan make:request Request-name`

တည်ဆောက်လိုက်တဲ့ Request Class ကို app/HTTP/Request folder အောင်မှာတွေ့နိုင်ပါတယ်။ Function အနေနဲ့ rules နဲ့ authorize ကို default တည်ဆောက်ပြီးသားတွေ့ရပါလိမ့်မယ်။ Rules ကတော့ validation စစ်တဲ့နေရာဖြစ်ပါတယ်။ authorize မှာတော့ return ကို true ပြောင်းပေးရပါမယ်။

```
public function rules()
{
    return [
        'username' => 'required',
        'password' => 'required'
    ];
}
```

Controller ရဲ့ validated လုပ်တဲ့နေရာမှာ parameter ကို Request Class မဟုတ်တော့ပဲ တည်ဆောက်ထားတဲ့ Class ကိုအစားထိုးပြီး အောက်ပါအတိုင်းရေးရပါတယ်။

```
public function home(ValidationRequest $request)
{
    $validated = $request->validated();
    ....
    ....
    ....
}
```

Show Validation Error in View

■ All error List

```
@if ($errors->any())
    @foreach ($errors->all() as $error)
        <li>{{ $error }}</li>
    @endforeach
@endif
```

■ Specific Error

```
@error('username')
    {{ $message }}
@enderror
```

Repopulation Form value

Validation မှာ valid မဖြစ်တဲ့အခါ request လုပ်တဲ့ form ဆီ error message နဲ့တူတူ return ပြန်လာပါတယ်။ ဒီအခါ မှာ အရင်က ထည့်လိုက်တဲ့ input box ထဲက value တွေပါ ပြန်ထုတ်ပြချင်တဲ့အခါ old method ကို သုံးနိုင်ပါတယ်။

```
<input type="text" name='username' value="{{ old('username') }}" />
```

16) Logging

ကိုယ့်ရဲ့ App တွေကို trace လိုင်တဲ့အခါ developer တွေဟာ များသောအားဖြင့် code တွေဖတ်ပြီး debugging လိုက်ကြတာများပါတယ်။ အဲလိုမဟုတဲ့ user ရဲ့ instructions တွေကို file တခုထဲမှာ သိမ်းဆည်းထားပြီး developer တွေ system မှာ error တတ်တဲ့ အခါ code တွေကို မပြင်ခင် log file ကို ကြည့်ခြင်းဖြင့် ဘယ် အပိုင်းမှာ error တတ်တယ်ဆိုတာ ခွဲ့ခြားသိနိုင်မှာဖြစ်ပါတယ်။ ဒါကြောင့် log ထုတ်ဖို့ဆိုတာ အရေးကြီးတဲ့ အစိတ်အပိုင်း တခုဖြစ်ပါတယ်။

`use Illuminate\Support\Facades\Log;`

Log Level အမျိုးအစား စမျိုးရှိပါတယ်။

`Log::emergency($message);
Log::alert($message);
Log::critical($message);
Log::error($message);`

`Log::warning($message);
Log::notice($message);
Log::info($message);
Log::debug($message);`

Example:

```
Log::critical("Home", [  
    "username is $username"  
]);
```

Log တွေကို storage/logs/laravel.log မှာတွေ့နိုင်ပါတယ်။

Custom Log file

Laravel.log ထဲမှာမဟုတ် ကိုယ်ပိုင် log တွေဖန်တီးနိုင်ပါတယ်။ Custom Log file တခုတည်ဆောက်တဲ့အခါ config/logging.php မှာ အောက်ပါအတိုင်းsetting ချိန်ပေးရပါတယ်။

```
'customlog' => [  
    'driver' => 'single',  
    'path' => storage_path('logs/custom.log'),  
    'level' => 'debug'  
]
```

ပြန်ခေါ်သံဃားတဲ့အခါ မှာတော့ channel method နဲ့ခေါ်သံဃားတဲ့အခါ ကိုယ်ပိုင် Log file ထဲမှာ log တွေကို တွေ့ရမှာဖြစ်ပါတယ်။

```
Log::channel('customlog')->info("Home", [  
    "username is $username"  
]);
```

17) HTTP Client

တခြား domain ဆိုက API နဲ့ data ယူတာ data ပိုတာတွေ စတဲ့ data transfer on http function တွေ အတွက် အသုံးပြုပါတယ်။

■ **Http::get('url')**

```
$response = Http::get('http://example.com');
```

■ **Http::post('url',[parameter])**

```
$response = Http::post('http://example.com/users', [
    'name' => 'John',
    'age' => 25
]);
```

■ **Http::withHeaders([header])->post('url',[parameter])**

```
$response = Http::withHeaders([
    'token' => 'fjgjfiwkrjw29349'
])->post('http://example.com/users', [
    'name' => 'Marry',
]);
```

■ **timeout(second)**

```
$response = Http::timeout(3)->get(...);
```

■ **retry(time,second)**

```
$response = Http::retry(3, 100)->post(...);
```

■ **Error Handle**

```
$response->successful(); // code >= 200 and < 300 ပြန်တဲ့အခါ
```

```
$response->failed(); // code >= 400 ပြန်လာတဲ့အခါ
```

```
$response->clientError(); // 400 Error ဆွဲ return ပြန်တဲ့အခါ
```

```
$response->serverError(); // 500 Error ဆွဲ return ပြန်တဲ့အခါ
```

18) Localization

Localization ဆိုတာကတော့ ကိုယ်ရဲ့ Application ထဲမှာ language တွေကိုသတ်မှတ်တာ သို့မဟုတ် language တွေ ထပ်ထည့်တာ । ပြောင်းလဲတာ စတဲ့ languages တွေကိုလိုသလို control လုပ်တဲ့ process ကိုဆိုလိုပါတယ်။

Localization ရဲ့ setting တွေကို config/app.php မှာတွေနိုင်ပါတယ်။

'locale' => 'en' // default language English

'fallback_locale' => 'en', //လက်ရှိlanguageerror တတ်ခဲ့တဲ့အခါပြမယ် language code

Language sources

Language ရဲ့ resource တွေကို resource/lang/en folder ထဲမှာတော့ default language ဖြစ်တဲ့ English language ရဲ့ file တွေတည်ရှိပါတယ်။

Translation Text

En folder နဲ့ Other Language အတွက် folder တွေကို တည်ဆောက်ပြီး message.php ကိုကိုယ်စီးပွားရန်ထိန်းများအတွက် အသုတေသနပါ။

Resources

```
Lang
|—en
|  |— message.php
|—jp
|  |— message.php
```

En/message.php

```
<?php
return [
    'welcome' => 'Welcome From Myanmar',
    'logout' => 'Logout'
];
```

Jp/message.php

```
<?php
return [
    'welcome' => 'ミャンマーからようこそ！',
    'logout' => 'ログアウト'
];
```

Using in View

Method 1.

```
 {{__('message.welcome')}}
```

Method 2.

```
@lang('message.welcome')
```

Replacing Parameters In Translation Strings

En/message.php

```
'hello' => 'Hello :username'
```

JP/message.php

```
'hello' => 'こんにちは :username'
```

In view

```
 {{ __('message.hello', ['username' => 'John']) }}
```

Pluralization

```
'order' => '{0} No order! |{1} :count order|[2,*] :count orders'
```

In view

```
<p class="display-6 ps-3">{{ trans_choice('message.order', 0) }}</p>
<p class="display-6 ps-3">{{ trans_choice('message.order', 1) }}</p>
<p class="display-6 ps-3">{{ trans_choice('message.order', 2) }}</p>
<p class="display-6 ps-3">{{ trans_choice('message.order', 10) }}</p>
```

Php file တည်ဆောက်ပြီးသုံးနိုင်ယုံဘမက json fileတွေနဲ့ပါ သုံးနိုင်ပါတယ်။

Configure Locale with route

```
use Illuminate\Support\Facades\App;
```

```
Route::get('/greeting/{locale}', function ($locale) {
    if (! in_array($locale, ['en', 'jp'])) {
        abort(400);
    }
});
```

```
App::setLocale($locale);
```

```
//  
});
```

19) File Storage Management

File တွေကို store လုပ်ဖို့ AWS Cloud server တွေနဲ့ ချိတ်ဆက်ပြီး file တွေ store လုပ်ဖို့
အတွက် Laravel မှာ file storage system ကို support ပေးထားပါတယ်။
Config/filesystem.php fileထဲမှာတော့ file store မယ့် နေရ တွေကို setting ချိတ်ထားတဲ့
Configure တွေပါရှိပါတယ်။

Store လုပ်နည်တဲ့ Disks တွေအနေနဲ့ကတော့ Local,ftp,sftp,s3 စတဲ့ disk တွေမှာ store
လုပ်နည်ပါတယ်။

File Upload

```
$request->hasFile('image') // upload တင်လိုက်တဲ့ file ရှိမရှိ  
$file = $request->file('image'); // upload တင်လိုက်တဲ့ file ကို access လုပ်  
$file->getClientOriginalName(); // file ရဲ့nameType  
$file->getClientOriginalExtension(); // file ရဲ့extension  
$file->getClientOriginalName(); // file ရဲ့original name
```

File Save in Storage

Store('filename') //pathကိုreturn ပြန်ပါတယ်

1. Env file ထဲမှာ local လိုသတ်မှတ်ထားတဲ့အခါ file ကို storage /app အောက်မှာ folder တည်ဆောက်ပြီး store လုပ်ထားမှာဖြစ်ပါတယ်။
2. Public လို သတ်မှတ်ထားခဲ့ရင်တော့ storage/app/public အောက်မှ folder တည်ဆောင်ပြီး store လုပ်ထားမှာ ဖြစ်ပါတယ်။
`$file->store('uploadedFiles');`

StoreAs('filename', 'filename') //pathကိုreturn ပြန်ပါတယ်

File nameကို လိုချင်တဲ့နာမည်အတိုင်းပေးချင်တဲ့အခါ သုံးပါတယ်။

```
$file->storeAs('file', 'profile' . $file->getClientOriginalExtension());
```

Helper Class

Storage::disk('storepath')->put('filename' , \$file) //pathကိုreturnပြန်ပါတယ်

```
Storage::disk('local')->put('files' , $file);
```

Get Storage URL

```
Storage::url($file)
```

Store ထားတဲ့ file တွေရဲ့ပတ်လမ်းကြောင်းအပြည့်အစုံ

```
$storepath = $file->storeAs('uploadedFiles' , 'filename.' . $file->getClientOriginalExtension());
```

```
Echo Storage::url($storepath);
```

Magic Link

file တွေကို download လုပ်တာဖြစ်စေ fronted မှာ link ချိတ်တာဖြစ်စေ | Css javascript စတဲ့ file တွေ ခေါ်သုံးတဲ့အခါ ဖြစ်စေ app/public folder အောက်မှာရှိတဲ့ file တွေကို default အနေနဲ့များသောအားဖြင့် link ချိတ်ကြပါတယ်။
store လုပ်ထားတဲ့ file တွေကို လည်း public အောက်မှ ပြောင်းထားပြီး လိုသလို ခေါ်သုံးချင်တဲ့အခါ Magic link ကိုသုံးနိုင်ပါတယ်။

Command: `php artisan storage:link`

ဒီလိုရေးလိုက်ခြင်းအားဖြင့် app/public folder အောက်မှ storage shortcut folder တစ္ဆေတည်ဆောက်ပေးပြီး original Store link ဖြစ်တဲ့ storage ရဲ့ပတ်လမ်းကြောင်းနဲ့ ချိတ်ဆက်ပေးမှာဖြစ်ပါတယ်။ ဒါကြောင့် app/public မှာ ခေါ်သလို ခေါ်သုံးနိုင်ပါတယ်။

20) Sending Mail

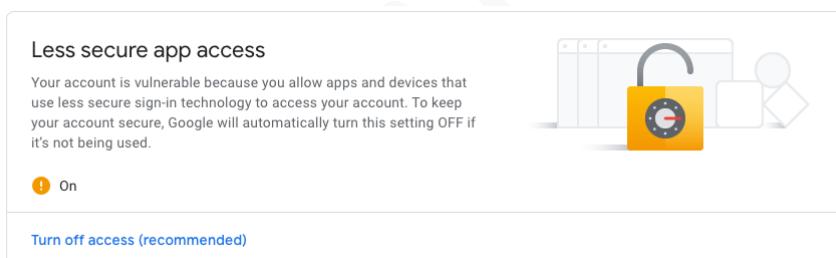
Business App တွေအတွက် Laravel မှာ mail ပို့စ်တဲ့ features တွေပါဝင်ပါတယ်။ config/mail.php မှာတော့ support ပေးတဲ့ driver တွေအနေနဲ့ကတော့ SMTP, mailgun,mandrill စဲတဲ့ driver တွေကို default support ပေးထားပါတယ်။

Setting Up Env File (For Gmail)

Gmail setting အတွက် .env file မှာ အောက်ပါအတိုင်း setting ချိန်ရပါတယ်။

```
MAIL_MAILER=smtp  
MAIL_HOST=smtp.googlemail.com  
MAIL_PORT=465  
MAIL_USERNAME=testing123@gmail.com  
MAIL_PASSWORD=testing123  
MAIL_ENCRYPTION=ssl  
MAIL_FROM_ADDRESS=testing123@gmail.com  
MAIL_FROM_NAME="${APP_NAME}"
```

Google mail security မှာလည်း Less secure app access မှာ turn on ထားပေးဖို့လိုပါတယ်။



Email ပို့တဲ့အခါ template နဲ့ပို့နို့အတွက် template view တွေကို တည်ဆောက်ရပါတယ်။

Command: `php artisan make:mail Template-name`

တည်ဆောက်လိုက်တဲ့ mail ထဲမှာတော့ default အနေနဲ့ constructor နဲ့ build function ပါဝင်ပါတယ်။ build function ကတော့ temple view ကို return ပြန်ပေးဖို့ အသုံးပြုပါတယ်။ အဲဒေသတွက်ကြောင့် mail template view တွေကိုပါ တည်ဆောက်ပေးဖို့လိုပါတယ်။

Sending Email

Mail::to('mail-address')->send(template)

Example:

Controller

```
$mail = [
    'title' => 'Mail Send From Laravel Framework',
    'body' => 'Mail testing form laravel framework'
];
```

```
Mail::to("receiver@gmail.com")->send(new SendMail($mail));

return "Mail Send Successfully!";
```

တည်ဆောက်ထားတဲ့ mail template ခဲ့ build function ထဲမှာတော့

```
public function build()
{
    return $this->subject('Mail Testing')
        ->view('mail.template')
        ->with([
            'title' => $this->mail['title'],
            'text' => $this->mail['body']
        ]);
}
```

Template view မှာတော့

```
@extends('layout')

@section('body')
    <h2>{{ $mail['title'] }}</h2>
    <p>{{ $mail['body'] }}</p>
    <p>From Exbrain</p>
@endsection
```

ဒီလိုဆိုရင် receiver@gmail.com ဆိုတဲ့ mail address ထဲကို mail ပိုပေးမှာဖြစ်ပါတယ်။

Multi mail recipients sending

```
$emails = [test1@gmail.com', 'test2@gmail.com','test3@gmail.com'];
Mail::send(
    'mail.template',
    [
        'title' => $mail['title'],
        'text' => $mail['body']
    ],
    function ($message) use ($emails) {
        $message->to($emails)->subject('Mail Testing');
    }
);
```

BCC,CC

```
Mail::to($request->user())
->cc($moreUsers)
->bcc($evenMoreUsers)
->send(template);
```

Attach File

Mailပိုတဲ့အခါ image တွေကို attach တဲ့ချင်တာ pdf ,excel စတဲ့ file တွေကို mail နဲ့အတူတူ attach တဲ့ချင်တဲ့အခါ သုံးပါတယ်။

Attach('path',[option])

Example:

```
->attach(public_path('image/testing.jpg'), [
    'as' => 'MailAttach.jpg',
    'mime' => 'image/jpeg',
])
```

Inline Attachments

Mailပိုတဲ့အခါ mail ထဲမှာတင်ပုံတွေကို ထည့်ပြီ ပိုလိုက်ချင်တဲ့အခါ။ Inline Attachment ကိုသုံးပါတယ်။

Example:

```

```

21) Factory & Seeding

Dummy data တွေကို database ထဲထည့်ချင်တဲ့အခါ Laravel မှာပါတဲ့ factory ကို
သုံးဖို့ရပါတယ်။

Factory:

Command: `php artisan make:factory Factory-Name`

တည်ဆောက်လိုက်တဲ့ file ဟာ database/factory အောက်ရောက်နေမှာဖြစ်ပါတယ်။

Model နဲ့ချိတ်ဆက်ပေးနိုင်ဖို့အတွက်

`protected $model = Model::class;`

မှာ ချိတ်ဆက်ချင်တဲ့ Model class ကိုရေးပေးရပါတယ်။

Factory တခုကို တည်ဆောက်လိုက်တဲ့အခါ default ပါလာတဲ့ function ကတော့ definition
function ဖြစ်ပြီးသူထဲမှာတော့ faker class ကို အသုံးပြုပြီး fake data (dummy data)
တွေကို database column အလိုက် သတ်မှတ်ပေးရပါတယ်။

What is faker

Faker ဆုတေသနတော့ php library တခုဖြစ်ပြီး fake data တွေကို generate
လုပ်ပေးတာဖြစ်ပါတယ်။

Declaration

`$faker = Factory::create();`

အသုံးများတဲ့ method တွေကတော့

`$this->faker->name()`

`$this->faker->address()`

`$this->faker->unique()->email()`

`$this->faker->phoneNumber()`

`$this->faker->paragraph()`

`$this->faker->numerify('##')`

`$this->faker->text()`

More detail

<https://github.com/FakerPHP/Faker>

Using in definition

Example:

```

public function definition()
{
    return [
        'name' => $this->faker->name(),
        'email' => $this->faker->unique()->safeEmail(),
        'age' => $this->faker->numerify('##'),
        'address' => $this->faker->address()
    ];
}

```

တည်ဆောက်ထားတဲ့ factory class ကို ခေါ်ပြီး dummy data တွေ လိုအပ်သလောက်
ထည့်နိုင်ဖို့ Seeder Class ကို သုံးရပါတယ်။

Run with Default Seeder

Default ပါတဲ့ DBSeeder Class ကိုပဲခေါ်သုံးပြီး db ထဲကို fake data တွေထည့်ချင်တဲ့အခါ

```

public function run()
{
    Model::factory(count)->create();
}

```

Command: artisan db:seed

Run with Custom Seeder

ကိုယ်ပိုင်seeder class ကို တည်ဆောက်ပြီး factory classကိုခေါ်သုံးချင်တဲ့အခါ

Command: artisan make:seeder Seeder-name

Run function မှာ factory ကို ခေါ်ပြီး

Command: artisan db:seed —class=SeederClass

22) Pagination

အရည်အတွက်များတဲ့ Data တွေကို display view ပြတဲ့အခါ အကုန်မပြချင်ပဲ limit တစူသတ်မှတ်ပြီး pagination လုပ်ပြီး ပြတဲ့အခါ ကြည့်ရတာပိုလွယ်ဖော်တယ်။

Paginate(count)

```
$customer = DB::table('customers')  
    ->paginate(5);
```

In view

```
{{ $customers->links() }}
```

Bootstrap ရဲ့ pagination design ကိုယူသုံးမှာဖြစ်တာကြောင့်

app/Provider/AppServiceProvider.php မှတော့ အောက်ပါအတိုင်း ထည့်ပေးရပါတယ်။
use Illuminate\Pagination\Paginator;

```
public function boot()  
{  
    Paginator::useBootstrap();  
}
```

#	First	Email	Age	Address
276	Prof. Jordyn Stanton PhD	veum.sibyl@example.net	05	673 Rempel Light Suite 598 Keelyshire, CT 75196-5196
277	Harrison Muller	vkeebler@example.com	40	884 Hoeger Locks Suite 403 East Elnor, CO 86612-1426
278	Dr. Elsie Rolfson	spencer14@example.com	27	217 Hammes Prairie Heathcoteland, IA 82302
279	Mr. Ezra Kohler III	stark.anne@example.org	77	89052 Abe Mountains Borerfort, NH 80424-1389
280	Nick Hickle	ssawayn@example.com	66	12473 Muller Parkway Suite 732 Rippington, KY 67105-5099
281	Fabiola Bogisich Sr.	tsimonis@example.org	46	954 Sauer Light Suite 960 Lake Pink, AL 32733-5080
282	Daisha Bechtelar	vmarks@example.net	77	1930 Ally Mill Mayertberg, WV 07418-0287
283	Dangelo Pagac III	leuschke.sadye@example.org	88	951 Rolfson Lakes Crystalmouth, MD 26101-3948
284	Arnold Hudson	santiago03@example.org	41	549 Jenkins Corners New Miller, ME 63350
285	Amely Block	cornelius.fay@example.net	05	74314 Rath Islands Llewellynmouth, ID 57338
286	Mr. Issac McGlynn	barbara.wintheiser@example.org	00	1423 Earl Knolls Suite 393 East Jacksonfort, PA 05694-8087
287	Henderson Bruen	keeling.cody@example.com	37	82776 Andy Orchard East Jeanettefort, NJ 67432
288	Ms. Alfreda Hudson DDS	ryan.destany@example.com	04	636 Bartell Tunnel Port Major, IN 60149
289	Asha Quitzon	medhurst.christiana@example.net	44	97316 Hiram Grove Apt. 106 Mackstad, OH 29149
290	Mr. Percy Heller	weston44@example.com	50	35344 Little Ferry Suite 652 Port Johnnie, SC 32189-9297

< 1 2 3 4 5 6 7 >

23) Eloquent Relationship & Serialization

Database Table နှစ်ခု ချိတ်ဆက်တဲ့အခါ တခုနဲ့တခုမြို့ခိုင်နေတဲ့ပုံစံဖြစ်ပါတယ်။

Database Foreign Key

Foreign key ကတေသ့ တခြား table တခုမှ primary key ဖြစ်နေတဲ့ column field ကို reference ယူထားတဲ့ key ဖြစ်ပါတယ်။

Advantage of Foreign Key

orphan record တွေမဖြစ်အောင် ကာကွယ်ပေးပါတယ်။

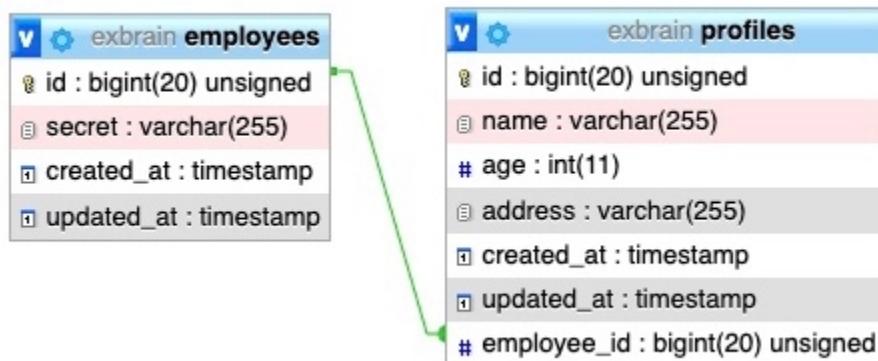
Disadvantage of Foreign Key

Database CPU Loading

Note>> Restrict,Cascade,Set null

Must be order

One to One Relationship



Employee table နဲ့ Profile table ရှိတယ်လိုလိုကြပါစိုး။

Employee ဆိတာ Profile တခုတည်းရှိသလို Profile ဆိတာလည်း Employee တယောက်ပဲရှိပါတယ်။ ဒီလိုမျိုး ဆက်ဆက်နေတဲ့ relationship ကို one to one relationship လိုခေါ်ပါတယ်။

Create Employee Table (Migration)

```
Schema::create('employees', function (Blueprint $table) {  
    $table->id();  
    $table->string('secret');  
    $table->timestamps();  
});
```

Create Profile Table (Migration)

```
Schema::create('profiles', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->integer('age');
    $table->string('address');
    $table->timestamps();
    $table->unsignedBigInteger('employee_id');
    $table->foreign('employee_id')->references('id')->on('employees')
        ->onDelete('cascade');
});
```

Profile Model

Profile မှာ Employee ရဲ့ ID နဲ့ချိတ်ဆက်ထားတောက်ပွဲတဲ့ Profile data ဟာ Employee ရှိမှ တည်ဆောက်လို့ရတောက်ပွဲတဲ့ belongTo နဲ့ Employee Model ကို ချိတ်ဆက်ပေးရပါတယ်။

```
public function employee()
{
    return $this->belongsTo('App\Models\Employee');
```

Employee Model

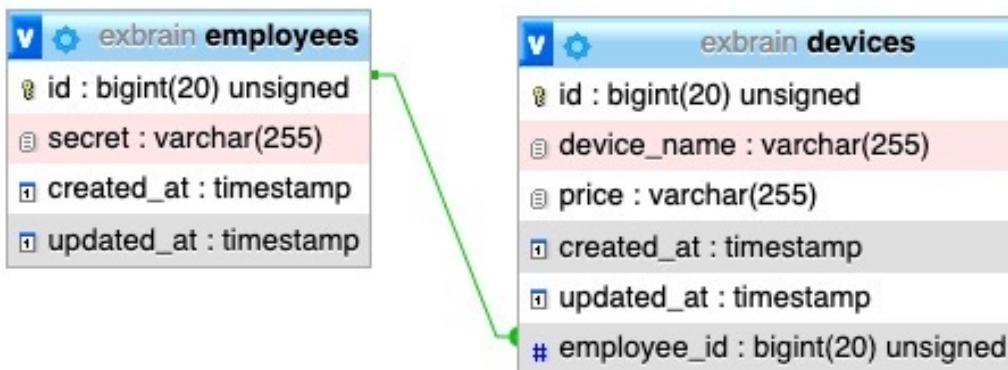
Employee မှာ Profile တစ်ခုပဲရှိတောက်ပွဲတဲ့ hasOne နဲ့ Profile ကို ချိတ်ပေးရပါတယ်။

```
public function profile()
{
    return $this->hasOne('App\Models\Profile');
```

Inserting Query with eloquent

```
//insert first employee  
$emp = new Employee();  
$emp->secret = "I am fat";  
$emp->save();  
  
//prepare insert data  
$profile = new Profile();  
$profile->name = "John";  
$profile->age = 27;  
$profile->address = "Yangon Thingangyun";  
  
//save profile and add employee id in foreign key  
$emp->profile()->save($profile);
```

One to Many Relationship



Employee table နဲ့ device table ရှိတဲ့အခါ employee တယောက်မှာ အသုံးပြုတဲ့ device ဟာ တခုထက်မက ပိုရှိနိုင်ပါတယ်။ဒီလိုမျိုး relationship ကို one to many relationship လို သတ်မှတ်ပါတယ်။

Device Model

Device မှာတော့ Employee ကို belongsTo နဲ့ချိတ်ပေးရပါတယ်။

Employee Model

Employee မှာ Device တခုထက်ပိုပြီးရှိနိုင်တော့ကြောင့် hasMany နဲ့ Device ကို ချိတ်ပေးရပါတယ်။

```
public function device()
{
    return $this->hasMany('App\Models\Device');
```

- 24) Encryption & Hashing**
- 25) Laravel Mix**
- 26) Authentication & Authorization**
- 27) API**