# Struktur folder routes dan view

Routes/web.php

Route::get('/', function () {

    return view('welcome');

});

Sintax diatas berarti url(‘/’) memangil resource (welcome.blade.php)

Mencoba route dengan mengembalka nilai :

Route::get('/', function () {

    return view('welcome');

});

Route::get('/about', function () {

    return 'Andri Firmansyah Putra';

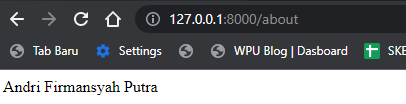
});

Route::get('/pekerjaan', function () {

    return 'Mahasiswa';

});

Coba kita lihat pada brow ser untuk memangilnya :



jika memangil sebuah view maka gunakan (buat file about.blade.php)

Route::get('/about', function () {

    return view('about');

});

Menggunakn foilder public :

* Menerapkan ccss, maka buat file css pada folder public lalu file dalam flder views tingal memangi nama filenya saja

## Mengirimkan sebuah data pada route :

Route::get('/about', function () {

    return view('about', [

        'nama' => 'Andri Firmansyah Putra',

        'alamat' => 'Mrisi Tirtinirmolo Kasihan Bantul',

        'img' => 'imgs/allTeam.jpeg'

    ]);

});

pada view :

<h3><?= $nama ?></h3>

<h4>{{ $alamat }}</h4>

<img src="<?= $img ?>" alt="<?= $img ?>">

# Blade Templatering engine

Supaya kita bias menggunakan template/ bagian html secara berulang jika bemiliki latar halaman yang sama kita dapat memanfaatkan blade templating.

* Kita akan memindahkan beberapa komponen htmlv: header , komponen html lain pada [home.blade.php](http://home.blade.php)
* Buat folder **view/layout/main.blade.php** untuk menyimpan
* Kita copy dari [home.blade.html](http://home.blade.html) ke kedalam layout/main.html lalu hapus isi utama halaman (container halaman) bagian yang berbeda dari tiap tiap halaman.

<div class="container mt-3">

        @yield('container')

    </div>

* Yield nanti akan dapat digantikan dengan tuisan lain
* Sekarang kita dapat menghapus html lain yang ada pada halaman lain (cth: about.blade.) dan hanya penrlu mengisikan isi dari layout utama yg sudah dibuat , sekarang bagaimana cara agar dapat menggunakan  **html utama**  yang menggunakan yield
* Membuat link(**extends**) dan memberikan bagian yg bias diid(**section**)

@extends('layouts.main')

@section('container')

    <div class="container mt-3">

        <h1>Halaman Home</h1>

    </div>

@endsection

## Memasukkan halaman html secara langsung

* Kita akan memisahkan bagian navigasi pada halaman layout utama(layout/main.blade.php)
* Membuat folder baru dan membuat file navigasi yang didalam sudah ada sintax” untuk membuat navigation
* Melakukan include ke file navigasi (dalam file utama) menggunakan include
* @include('partials.navbar')

## Rouute parameter

-pada web.php

Route::get('posts/{slug}', function ($slug) {

    $new\_post = [];

    foreach ($blog\_posts as $post) {

        if ($post['slug'] === $slug) {

            $new\_post = $post;

        }

    }

    // return $slug;

    return view('post', $new\_post);

});

* Pada html

# Model post Controller

Membuat model : **php artisan make: model nama\_model**

* **Jika sudah memilkikitable dapatjugauntukmembuatmigraton nya**

## Latihan Modeldengan data static

* B- membuat model (akan disavepada app/model)

**Php artisan make:model post**

* Lalu kita isi dan beberapa kita hapus (karena masih belum digunakan)

<?php

namespace App\Models;

class Post

{

    private static $blog\_post =  [

        [

            'title' => 'Judul Post Pertama',

            'slug' => 'judul-post-pertama',

            'author' => 'Andri Firmansyah Putra',

            'body' => 'Lorem ipsum dolor sit amet consectetur adipisicing elit. Ad Nemo sunt temporibus illo odit alias deleniti, officia explicabo quis? Consequuntur tempora in totam.'

        ], [

            'title' => 'Judul Post Kedua',

            'slug' => 'judul-post-kedua',

            'author' => 'Agung Jiwandono',

            'body' => 'Lorem ipsum dolor sit amet consectetur adipisicing elit. Ad voluptatum itaque doloribus optio, obcaecati culpa sit, sint architecto suscipit nemo, fugit incidunt totam.'

        ],

    ];

}

* Ceritanya nanti kita akan mengambil data dengan method all pada model, maka tambahkan function pada post model
* Berikut kode pada route :

Route::get('/blog', function () {

    return view('posts', [

        'title' => 'Blog Posts',

        'posts' => Post::all()

    ]);

});

**Kode pada** model :

## Membuat pencarian post dengan model

Route::get('posts/{slug}', function ($slug) {

    $new\_post = Post::find($slug);

    // return $slug;

    return view('post', $new\_post);

});

Function model yang dibuat

public static function find($slug)

    {

        $posts = self::$blog\_post;

        $new\_post = [];

        foreach ($posts as $post) {

            if ($post['slug'] === $slug) {

                $new\_post = $post;

            }

        }

        return $new\_post;

    }

## Menggunakan Array Collection :

Tinggal bungkus menggunakan collect :

$collect  = collect([1,2,3]);

public static function all()

    {

        // return self::$blog\_post; 5.5 tidak digunakan karena menggunakan collection

        return collect(self::$blog\_post);

    }

Mengubah sintax didalam post::find

$posts = static::all();

        $new\_post = $posts->firstWhere('slug', $slug);

        return $new\_post;

## Menggunakan Controller

Php artisan make:controller name

use Illuminate\Http\Request;

class PostController extends Controller

{

}

* Kita akan mengunah halaman post agar dapat menggunakan controller

use App\Models\Post;

class PostController extends Controller

{

    public function index()

    {

        return view('posts', [

            'title' => 'Blog Posts',

            'posts' => Post::all()

        ]);

    }

}

Route yang digunakan menjadi :

Route::get('/blog', [PostController::class, 'index']);

# 6 Database Migration dan Eloquent

File .env (file konfigurasi)

-didalamlaravel terdapat table table yang sudah disediakan dan dapat kita gunakan dengan melakukan :

**Php artisan migrate**

**File tersebut ada karena ada data** php yang kita sebut dengan migrasi pada setiap table yang telah dibuat, maka dari itu sekarang kita akan membuat data base dan mengkonfogurasi pada laravel pada file (**.env**)

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=wpu\_blog

DB\_USERNAME=root

DB\_PASSWORD=

**Lalu jalankan sintax migrate diatas pada terminal**

## Melakukanroleback pada migrasi

**Php artisan migrate:rollback ::** sintax ini digunakan untuk

## Melakukan hapus dan migrate sekaligs

Php artisan migrate:fresh

## Eloquent ORM

Digunakan untuk kita dapat menggunakan database. Sebuah fungsi untuk memetakan object untuk dapat berinmteaksi dengan database. Perantara (model) ke dalam database.

#CTT : setiap model pasti ada file migrationnya.

### File Model

#model digunakan untuk referensi table didalamdatabase

Pada protected adalah file yang boileh diisi

protected $fillable = [

        'name',

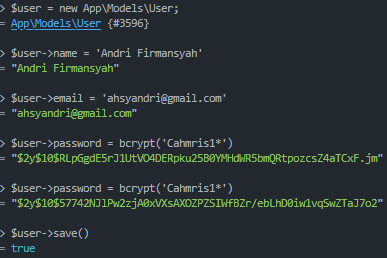
        'email',

        'password',

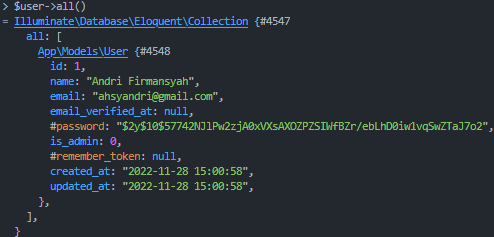
    ];

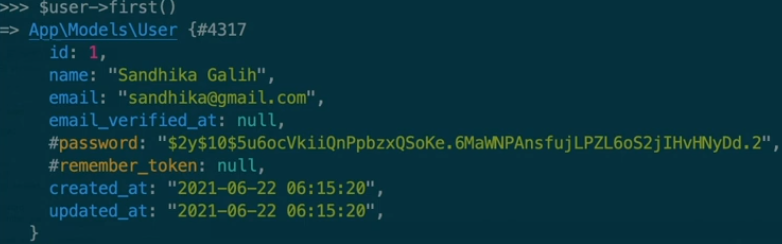
### Menggunakan artisan tinker

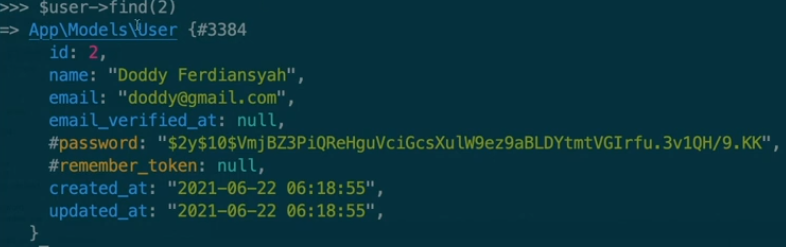
Menggunakan fitur ini untuk mengakses model dan mengisi database;

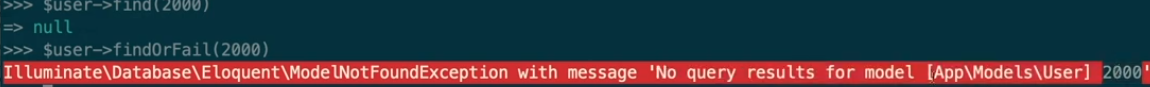


### Sintaxmilix eloquent







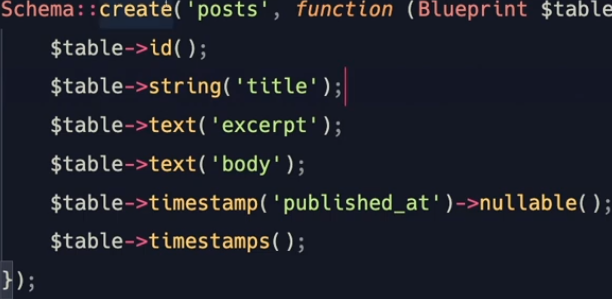


# 7 Membuat Model Post

## Membuat model post

-php artisan make:model –m Post (Membuat model dan file migrasi)

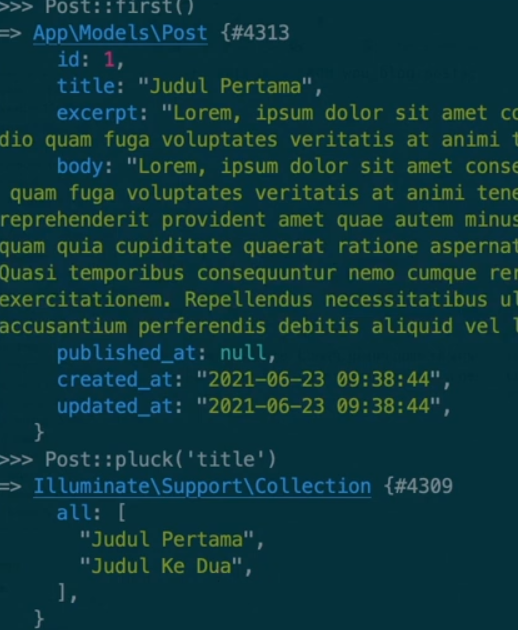
- memebnuhi file migratuuin posts



**#sintax $table->timestamps (merupakanmethod untuk menuliskan) delete at dan update add**

**-**agar dapat menambah menjadi table sekarang klakukan

**Php artisan migrate:fresh**

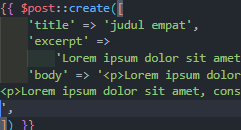
****

### Menampilkan post dengan format html

<p> {!! $post->body !!} </p>

### Cara lain menambahkan file data kedallam database dengan tinker

* Menggunakan method create



* Namun sebelum dijalankan akan muncul error masassigment

## Menghadapi error mass assignment

kita perlu untuk mendeskripsikan file diperbolehkan masuk kedalam database kedalam file model Post

protected $fillable = ['title', 'excerpt', 'body']; (ini atau yang bawah)

    protected $guard = ['id'];

## MassAssigment Update





## Orute model binding

Ini akan mengh-skip query dan akan langsungmengembalikan datanya, sebelumnya kita menggunakan id namun kita gunakan didalamfungsi yaitu tipedatanya ($nama \_model dan variable sesuai dengan route)



### Mengubah dengan model binding

Route::get('posts/{id}', [PostController::class, 'show']);

Menjadi

Route::get('posts/post}', [PostController::class, 'show']);

Pada Controller

public function show($id)

    {

        $new\_post = Post::find($id);

        return view('post', [

            'title' => 'Single post',

            'post' => $new\_post

        ]);

    }

Menjadi

public function show(Post $post) {

        return view('post', [

            'title' => 'Single post',

            'post' => $post

        ]);

    }

## Menambahkan slug

* Mengedit filemigrasi

Schema::create('posts', function (Blueprint $table) {

            $table->id();

            $table->string('title');

            $table->string('slug')->unique();

            $table->text('excerpt');

            $table->text('body');

            $table->timestamp('published\_at')->nullable();

            $table->timestamps();

        });

* Lakukan artisan migrate:fresh
* Sekarang datra kita sudah huilang
* Lakukan penambahan data

### Menganti pencarian tidak lagi dengan id tapi slug

* Melakukan pencarian dengan slug

<a href="/posts/{{ $post->id }}">{{ $post->title }}</a>

* Diganti
* <a href="/posts/{{ $post->slug }}">{{ $post->title }}</a>
* Mengubah route

Route::get('posts/{post:slug}', [PostController::class, 'show']);