# LARAVEL 8

## Instalasi & Documentation

- Documentataion : https://laravel.com/docs/
- Normal Install : composer create-project laravel/laravel example-app
- Global Install : laravel new example-app

## Tips atau Sering Digunakan

- blade tidak menggunakan titik koma ;
- php echo : {{ $nama }}
- periksa : @dump(); atau @dd();
- @crsf : hash dibawah tag <form> pembuka
- mt_rand(2, 8) : membuat bilangan random mt_rand(minimal, maksimal)
- Str::limit(strip_tags($data->body), 75, ' ...') : memotong kata sebanyak 75
- jika membuat controller menggunakan artisan make C+M Res, route bisa disingkat :

```
use App\Http\Controllers\StudentsController;
Route::resource('students', StudentsController::class);
```

## Php artisan

- php artisan db:seed : mengirim data seeder
- php artisan migrate:fresh –seed : mengirim data seeder & factory

## Artisan make

- controller : $ php artisan make:controller <namaController>
- model : $ php artisan make:model <nama>
- factory : $ php artisan make:factory <namaFactory>
- C Res : $ php artisan make:controller <namaController> --resource
- C+M Res : $ php artisan make:controller <namaController> -r -m <namaModel>
- M+ mfs : $ php artisan make:model <nama> -mfs

## Artisan Migrate (migration)

- migrate : $ php artisan migrate
- rollback : $ php artisan migrate:rollback
- rollback + migrate : $ php artisan migrate:fresh
- migrate + seeder : $ php artisan migrate:fresh --seed

## Struktur Folder MVC, Routes dan View

- routes : routes/web.php : file utama yg diakses
- controller : app/Http/Controllers
- model : app/Models
- view : resources/Views
- assets : public/assets : folder assets buat manual
- .env : konfigurasi url, database, dll

## Blade Templating Engine

- membuat templates : views/layouts/main.blade.php

```
<head>
```

```
    <title>@yield('title')</title>
</head>
<body>
    @include('partials.navbar')
    <div class="container m-4">
        @yield('container')
    </div>
</body>
```

- partials (opsioanal)      : views/partials/navbar.blade.php

- penggunaan di view     : views/home.blade.php

```
@extends('layouts.main')
@section('title', 'Laravel 8 | Home')
@section('container')
    {{-- isi konten --}}
@endsection
```

## Tinker

- Menambah data ke table menggunakan tinker di cmd

1. jalankan tinker : php artisan tinker
2. buat variable. ada 2 cara contoh :

```
>>> $user = new App\Models\User;
>>> $user = new User;
```

3. isi attribut table yg fillable. contoh :

```
>>> $user->name = 'Mochamad Ihsan Saepulloh';
>>> $user->email ='mihsansaepulloh9@gmail.com';
>>> $user->password = bcrypt('09072002');
```

4. simpan data diatas ke table

```
>>> $user->save()
```

5. memeriksa data table

```
>>> $user->all()
```

## ▪ Seeder, Factory & Faker

- Merubah lokasi data faker :

1. ubah config : config/app.php

```
'faker_locale' => env('FAKER_LOCALE', 'en_US'),
```

2. tambahkan baris dibawah di file .env

```
FAKER_LOCALE=id_ID
```

- Membuat seeder + factory

1. buat factory di cmd :

```
php artisan make:factory <namaFactory>
```

2. database/seeders/DatabaseSeeder.php

```
public function run()
{
    User::factory(3)->create();

    // manual tanpa factory
    Category::create([
        'name'  => 'Web Programming',
        'slug'  => 'web-programming'
    ]);

    Post::factory(20)->create();
}
```

3. contoh factory User : database/factories/UserFactory.php

```
public function definition()
{
    return [
        'name' => $this->faker->name(),
        'username' => $this->faker->unique()->userName(),
        'email' => $this->faker->unique()->safeEmail(),
        'email_verified_at' => now(),
        'password' => '1234..'
        'remember_token' => Str::random(10),
    ];
}
```

- **N+1 Problem (Query Berlebihan)**

- Untuk periksa jumlah query gunakan clockwork, cara instal :
  1. onProject : pasang library clockwork lewat cmd

     ```
     composer require itsgoingd/clockwork
     ```

  2. onChrome : instal ekstensi clockwork by itsgoingd

- Untuk mencegah multi query ada 3 cara :
  1. Jika di controllers/PostController.php : tambahkan with()

     ```
     public function index()
     {
         return view('posts',[
             "title" => "All Posts",
             // with('tabelRelasi1, dst..')
             "posts" => Post::with(['author', 'category'])->latest()->get()
         ]);
     }
     ```

  2. Jika di Web.php : untuk yg route model binding tambahkan load()

     ```
     Route::get('/authors/{author:username}', function(User $author) {
         return view('posts', [
             'title' =>  "Post by Author : $author->name",
             'posts' => $author->posts->load('author', 'category')
         ]);
     });
     ```

  3. Jika di models/Post.php : tambahkan protected $with

     ```
     protected $guarded = ['id'];
     protected $with = ['author', 'category'];
     ```

- **Searching**
  - controller/PostController.php : memanggil scope filter (function scopeFilter di model)

```php
return view('posts',[
    "title" => "All Posts" . $title,
    'active' => 'posts',
    // Tambahkan filter()
    "posts" => Post::latest()->filter(request(['search', 'category', 'author']))->get()
]);
```

  - model/Post.php :

```php
// fitur searching
public function scopeFilter($query, array $filters)
{
    // pengganti : isset( $filters['search'] ? $filters['search'] : false )
    // menjadi   : $filters['search'] ?? false
    $query->when($filters['search'] ?? false, function($query, $search) {
        return $query->where('title', 'like', '%' . $search . '%')
                     ->orWhere('body', 'like', '%' . $search . '%');
    });

    // menggunakan normal function
    $query->when($filters['category'] ?? false, function($query, $category) {
        return $query->whereHas('category', function($query) use ($category) {
            $query->where('slug', $category);
        });
    });

    // menggunakan arrow function (fn(..) => ..)
    $query->when($filters['author'] ?? false, fn($query, $author) =>
        $query->whereHas('author', fn($query) =>
            $query->where('username', $author)
        )
    );
}
```

  - view/posts.blade.php : menambah pengkodisian dan ubah link href author/category

```blade
{{-- tambahkan pengkondisian dibawah setelah form pembuka --}}
<form action="/posts">
    {{-- Searching berdasarkan keyword dan category/author --}}
    @if (request('category'))
    <input type="hidden" name="category" value="{{ request('category') }}">
    @endif
    @if (request('author'))
    <input type="hidden" name="author" value="{{ request('author') }}">
    @endif
    ...
</form>
```

```blade
{{-- ubah link menjadi href="/posts?category={{ $posts[0]->category->slug }}" --}}
<a href="/posts?category={{ $posts[0]->category->slug  }}" class="text-
decoration-none">
    {{ $posts[0]->category->name }}
</a> {{ $posts[0]->created_at->diffForHumans() }}
```

- **Pagination**
  - Controller :

```
"posts" => Post::latest()->filter(request(['search', 'category',
'author']))->paginate(7)->withQueryString()
```

  - View :

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

  - Http/Providers/AppServiceProviders.php : Ubah style ke bootstrap

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

- **Is Active**

  Sumber: modul-laravel kuliah

```
<li class="{{ request()->is('/') ? 'active' : '' }}"><span>Home</span></li>
<li class="{{ request()->is('user') ? 'active' : '' }}"><span>User</span></li>
<li class="{{ request()->is('siswa') ? 'active' : '' }}"><span>Siswa</span></li>
<li class="{{ request()->is('guru') ? 'active' : '' }}"><span>Guru</span></li>
```

  Sumber: comment YT

```
<a class="{{ request()->segment(1) == 'Home' ? 'active' : '' }}">Home</a>
```

## ▪ Auth Login dan Register

- Route

```
Route::get('/login', [LoginController::class, 'index']));
Route::post('/login', [LoginController::class, 'authenticate']);
Route::post('/logout', [LoginController::class, 'logout']);

Route::get('/register', [RegisterController::class, 'index']);
Route::post('/register', [RegisterController::class, 'store']);

Route::get('/dashboard', [DashboardController::class, 'index']);
```

- Register Controller

```
class RegisterController extends Controller
{
    public function index()
    {
        return view('register.index', [
            'title' => 'Register',
            'active' => 'register'
        ]);
    }

    public function store(Request $request)
    {
        $validateData = $request->validate([
            'name'      => 'required|max:255',
            'username'  => 'required|min:3|max:255|unique:users',
            'email'     => 'required|email:dns|unique:users',
            'password'  => 'required|min:4|max:225',
        ]);

        // enkripsi password
        $validateData['password'] = bcrypt($validateData['password']);

        // kirim ke database
        User::create($validateData);

        // flash dan redirect
        return redirect('/login')->with('success', ' Daftar berhasil.');
    }
}
```

- LoginController + Logout

```
class LoginController extends Controller
{
    public function index()
    {
        return view('login.index');
    }

    public function authenticate(Request $request)
    {
        $credentials = $request->validate([
            'email'      => 'required|email',
            'password'   => 'required',
        ]);

        if(Auth::attempt($credentials)) {
            $request->session()->regenerate();
            return redirect()->intended('/dashboard');
        }

        return back()->with('failed', 'Login Failed!');
    }

    public function logout()
    {
        Auth::logout();
        request()->session()->invalidate();
        request()->session()->regenerateToken();
        return redirect('/login')->with('success', 'Berhasil Logout!');
    }
}
```

- LoginView

```
<form action="/login" method="POST">
@csrf

    <div class="form-floating">
        <input type="email" name="email" value="{{ old('email') }}">
        <label for="email">Email</label>
    </div>
    @error('email') <small>{{ $message }}</small> @enderror

    <div class="form-floating">
        <input type="pass" name="pass" value="{{ old('pass') }}">
        <label for="pass">Password</label>
    </div>
    @error('pass') <small>{{ $message }}</small> @enderror

    <button type="submit">Login</button>
</form>
<small>Not registered? <a href="/register">Register Now!</a></small>
```

- LogoutView

```
<form action="/logout" method="post">
@csrf
    <button type="submit" class="dropdown-item">Logout</button>
</form>
```

## ▪ Middleware

- Merubah route redirect : app/Providers/RouteServiceProviders.php
- Menambah flashdata harus login : app/Http/Middleware/Authenticate.php
- Menambahkan di hak akses user dalam route dan mendefiniskan name route

```
Route::get('/login', [LoginController::class, 'index'])->name('login')-
>middleware('guest');
Route::get('/register', [RegisterController::class, 'index'])->middleware('guest');
Route::get('/dashboard', [DashboardController::class, 'index'])->middleware('auth');
```

- Memberi kondisi apakah user sudah login atau belum di view

```
@auth
    {{-- Kondisi jika sudah login --}}
@else
    {{-- Kondisi jika belum login --}}
@endauth
```

- Menampilkan data user yg login di view

```
Welcome back, {{ auth()->user()->name }} {{-- auth() -> table -> field --}}
```

- **CRUD**

  Disarankan jika membuat controller crud, gunakan controller -r -m (resource dan model). Agar route hanya menuliskan code :

```
Route::resource('/dashboard/posts', DashboardPostController::class)-
>middleware('auth');
```

  Jika ingin merubah default key route model binding, pada modelnya tambahkan function :

```
// merubah default key route model binding dari id ke yg diinginkan
public function getRouteKeyName()
{
    return 'slug';
}
```

- **Index (getAll)**
  - Route : menggunakan Route::resource seperti contoh diatas.
  - Controller :

```
public function index()
{
    return view('dashboard.posts.index', [
        'posts' => Post::where('user_id', auth()->user()->id)->get()
    ]);
}
```

  - View : kita hanya tinggal foreach data $posts diatas.

- **Show (getBy)**
  - Route : menggunakan Route::resource seperti contoh diatas.
  - Link di view :

```
<a href="/dashboard/posts/{{ $post->slug }}"> detail </a>
```

  - Controller :

```
public function show(Post $post)
{
    return view('dashboard.posts.show', [
        'post' => $post
    ]);
}
```

- **Create (viewInsert)**
  - Route : menggunakan Route::resource seperti contoh diatas.
  - Link di view :

```
<a href="/dashboard/posts/create"> Create New Post </a>
```

  - Controller :

```
public function create()
{
    return view('dashboard.posts.create', [
        'categories' => Category::all()
    ]);
}
```

- **Store (insertData)**
  - Route : menggunakan Route::resource seperti contoh diatas.
  - Form action di view :

```
<form action="/dashboard/posts" method="POST">
@csrf
    {{-- field / input --}}
<form>
```

  - Controller :

```
public function store(Request $request)
{
    $validatedData = $request->validate([
        'title' => 'required|max:255',
        'slug' => 'required|unique:posts',
        'category_id' => 'required',
        'body' => 'required',
    ]);

    $validatedData['user_id'] = auth()->user()->id;
    $validatedData['excerpt'] = Str::limit(strip_tags($request->body), 200,
'...');

    Post::create($validatedData);

    return redirect('/dashboard/posts')->with('success', 'Post berhasil
ditambah');
}
```

## Edit (viewUpdate)

- Route : menggunakan Route::resource seperti contoh diatas.
- Link di view :

```
<a href="/dashboard/posts/{{ $post->slug }}/edit" > EDIT </a>
```

- Controller :

```
public function edit(Post $post)
{
    return view('dashboard.posts.edit', [
        'post' => $post,
        'categories' => Category::all()
    ]);
}
```

## Update (updateData)

- Form action di view :

```
<form action="/dashboard/posts/{{ $post->slug }}" method="POST">
@method('put')
@csrf
    {{-- field / input --}}
<form>
```

- Controller :

```
public function update(Request $request, Post $post)
{
    $validatedData = $request->validate([
        'title' => 'required|max:255',
        'category_id' => 'required',
        'body' => 'required',
    ]);

    if($request->slug != $post->slug) {
        $validatedData['slug'] = 'required|unique:posts';
    }

    $validatedData['user_id'] = auth()->user()->id;
    $validatedData['excerpt'] = Str::limit(strip_tags($request->body), 200,
'...');

    Post::where('id', $post->id)->update($validatedData);

    return redirect('/dashboard/posts')->with('success', 'Post berhasil
diubah');
}
```

## Destroy (deleteData)

- Route : menggunakan Route::resource seperti contoh diatas.
- Form action di view (mengelabui method menjadi delete) :

```
<form action="/dashboard/posts/{{ $post->slug }}" method="post">
    @method('delete')
    @csrf
    <button class="badge bg-danger border-0" onclick="return 
confirm('Yakin?')"><span data-feather="x-circle"></span></button>
</form>
```

- Controller :

```
public function destroy(Post $post)
{
    Post::destroy($post->id);
    return redirect('/dashboard/posts')->with('success', 'berhasil dihaps');
}
```