|  |  |
| --- | --- |
| **Materi** | **Nilai** |
| Laravel Tgl 12-14 Jan SMST6 (1 playlist) | 95 |

**Saya Sudah Belajar dan Mengerti dan Saya BISA**

1. Melakukan installasi composer dan xampp untuk membuat sebuah project Laravel & tailwind-css.

2. Mengecek composer apakah sudah terinstall dan tersambung dengan xampp nya dengan cara mengetik cmd di lokasi htdocs kemudian jika cmd sudah muncul ketikkan composer. Apabila composer sudah terinstall akan muncul tulisan composer yang besar dan keterangan yang banyak dibawahnya.

3. Installasi Laravel apabila sudah ada composer dengan cara mengetikkan composer create-project laravel/laravel laravel-resto yang berarti dalam composer akan dibuat sebuah project yaitu Laravel dengan nama projectnya yaitu Laravel-resto. Jika pembuatan projek ini berhasil akan muncul keterangan Application key set successfully di paling bawah dari proses installasi melalui cmd. Jika dicek di dalam htdocs akan muncul folder baru beserta isi file-filenya yang bernama Laravel-resto (sesuai yang dibuat di cmd).

4. Untuk memulai sebuah project, di dalam vs-code buka terminal kemudian ketik php artisan serve untuk menyalakan server dari laravelnya. Jika sudah akan muncul Starting Laravel development server: <http://127.0.0.1:8000> dan link tersebut bisa dibuka di browser.

5. Melakukan installasi nodejs tetapi jika sudah bisa dicek melalui terminal vscode dengan mengetik node -v. apabila nodejs sudah terinstall maka setelah mengklik enter akan muncul versi dari nodejs yang terinstall.

6. Menginstall npm dengan cara mengetikkan npm install di terminal. Jika npm sudah terinstall selanjutnya install tailwindcss melalui npm dengan mengetikkan npm install -D tailwindcss.

7. Membuat tailwindcss config file dengan mengetikkan npx tailwindcss init di dalam terminalnya. File tersebut bernama tailwind.css.js

8. Menambahkan

@tailwind base;

@tailwind components;

@tailwind utilities;

Ke dalam file app.css di dalam folder resources->css

9. Menambahkan

require('tailwindcss'),

di dalam mix.js di bagian kurung staplesnya. Code ini berada di file webpack.mix.js

10. Mengetikkan npm run watch di dalam terminal untuk merunning terminal dan melihat semua css serta javascript yang relevant untuk dipakai. Jika proses sukses akan muncul webpack compiled successfully di akhir code. Di tengahnya akan ada table file css dan js yang relevant digunakan.

11. Menghapus semua isi dari file welcome.blade.php untuk diganti dengan code html yang baru. Di dalam html baru yang sudah dibuat bisa ditambahkan

<link href="/css/app.css" rel="stylesheet">

Untuk memberi css yang diambil dari app.css

12. Karena tailwindcss yang digunakan versi baru, karena itu di dalam file tailwind.config.js pada bagian content diisi dengan

"./resources/views/\*.blade.php",

"./resources/views/admin/\*.blade.php",

"./resources/views/template/\*.blade.php",

Agar css dapat digunakan di dalam file htmlnya, jika tidak disertakan code tersebut maka css tidak dapat berfungsi semestinya. Setelah mengetikkan code tsb kemudian mengetikkan npm run dev di dalam terminal untuk merun ulang tailwindnya dan ketik php artisan serve untuk menyalakan server laravelnya lagi.

13. Menggunakan route untuk mengatur tampilan website sesuai dengan linknya, misalkan jika linknya ada tambahan /login maka dia akan masuk ke dalam halaman login. Contoh penulisan routenya seperti ini

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

    return view ('login');

});

Yang apabila penulisan route nya atau tambahan linknya /login maka dia akan menampilkan view dari page login. Bisa juga return view nya diganti seperti ini apabila masuk ke dalam sebuah folder

return view ('admin.login');

tanda titik ditengah-tengah admin dan login bisa juga diartikan sebagai pengganti tanda /

14. di dalam route terdapat 2 bagian pada tanda kurung getnya () yaitu di sebelah kiri adalah yang memanggil dan disebelah kanan adalah yang dipanggil.

15. Membuat file html untuk dijadikan template dari file lain. Caranya dengan membuat folder baru bernama template di dalam views kemudian diisi dengan file bernama layout.blade.php. Di dalam file ini dibuatkan sebuah html baru yang disambungkan dengan css dari app.css public. Kemudian agar file ini bisa digunakan oleh file lain, beri @yield lalu namanya, contoh

@yield('content')

Dengan begitu file ini dapat digunakan sebagai template di dalam file lain, untuk penggunaannya di dalam file lain yang dimaksud diberi code

@extends('template.layout')

Untuk mencari tempat file template nya yaitu di folder template bagian file layout kemudian

@section('content')

Untuk mengambil bagian yang ada @yieldnya dan sesuai dengan nama yang ada di dalam kurungnya yaitu @yield(‘content’). Terakhir untuk menutup section maka setelah code html diberi tambahan

@endsection

16. Controller berfungsi untuk menerima sebuah perintah dari routes, jadi dari dalam route untuk mengakses views harus melalui controller terlebih dahulu. Caranya dengan membuat sebuah controller yaitu ketik php artisan make:controller nama\_controller lalu enter di dalam terminal. Jika sudah bisa dicek di dalam app\http\controller\nama\_controller. Buka controller tersebut kemudian isi dengan sebuah function baru misalnya index, index ini berisi pemanggilan ke file views nya yaitu return view(‘home’); jadi apabila ada yang memanggil controller dengan menyertakan nama functionnya yaitu index maka yang akan tampil adalah isi dari function index yaitu memanggil view home.

17. Untuk memberi sebuah perintah terhadap controller melalui routes caranya dengan mengetikkan

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

Jadi disebelah kiri adalah yang memanggil berarti di linknya langsung ketika dibuka dia akan menjalankan class HomeController pada bagian function index.

18. Menggunakan php artisan migration untuk menambahkan sebuah table di dalam phpMyAdmin. Caranya dengan membuat database terlebih dahulu di dalam phpMyAdmin misalkan Laravel-resto. Kemudian di dalam .env sambungkan dulu projectnya dengan database yang sudah dibuat dengan cara mengisi DB\_DATABASE= dengan nama database yang dituju. Kemudian di dalam terminalnya buat sebuah table dengan mengetikkan php artisan make:migration nama\_table lalu enter dan tunggu sampai proses selesai. Jika berhasil maka file baru akan muncul di dalam folder database\migrations.

19. Apabila sebuah migration baru berhasil dibuat, selanjutnya adalah proses mengatur nama table dan kolom-kolom apa saja yang akan digunakan di dalam table tersebut melalui file migrationnya. Pada bagian function up isi dengan code

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

            $table->bigIncrements('idkategori');

            $table->string('kategori');

            $table->timestamps();

        });

Yang berarti membuat sebuah table dengan nama kategoris kemudian isi table tersebut dengan kolom yang pertama yaitu idkategori bertype bigIncrements, kolom kedua kategori dengan type string, dan yang terakhir adalah kolom yang harus ada yaitu waktu pembuatannya. Kemudian isi juga function down dengan code

Schema::dropIfExists('kategoris');

Yang di dalam kurung sesuaikan dengan nama table yang akan dibuat. Kemudian finalisasi atau proses upload table beserta kolom tersebut dengan cara mengetikkan php artisan migrate atau upload semua migration yang telah dibuat. Apabila table yang telah dibuat menggunakan migration terlanjur masuk ke dalam database tetapi ingin dibatalkan caranya ketik php artisan migrate:rollback atau kembalikan data migrationnya.

20. Menggunakan seeder untuk mengisi kolom dari table yang telah dibuat melalui Laravel, caranya dengan membuat seeder terlebih dahulu. Di dalam terminal ketikkan php artisan make:seeder NamatableSeeder lalu enter. Setelah itu buka file seeder yang telah dibuat di dalam folder database\seeders. Di dalam file seeder ini terdapat function run lalu isi dengan code

DB::table('kategoris')->insert([

            'kategori' => 'Makanan',

        ]);

Yang berarti seeder atau penambahan data ke kolom ini tujuannya ke table kategoris di bagian kolom kategori isi dengan string ‘Makanan’. Agar makanan ini bisa masuk ke dalam kolom kategori table kategoris caranya dengan menambahkan code

$this->call([

            KategoriSeeder::class,

        ]);

Di dalam DatabaseSeeder agar memanggil KategoriSeeder yang sudah disetting untuk memasukkan data ke dalam table kategori. Kemudian finalisasinya dengan mengetikkan php artisan db:seed di dalam terminal kemudian enter. Jika berhasil maka di dalam phpMyAdmin pada bagian table kategoris, kolom kategori, akan bertambah 1 data yaitu “Makanan”.

21. Membuat sebuah model dengan mengetikkan php artisan make:model namaModel di dalam terminal kemudian filenya bisa dilihat di dalam app\models. Di dalam file ini terdapat sebuah class yang dapat diisi untuk tujuan table serta kolom yang akan dipakai model ini untuk menyambungkannya dengan database. Code untuk menentukan tablenya adalah protected $table = ‘namatable; sedangkan untuk menentukan kolomnya adalah protected $fillable = [‘namakolom’];

22. Jika sebelumnya input sebuah data berjumlah 1 saja ke dalam kolom sebuah table menggunakan setting melalui seeder langsung, maka untuk menginputkan data lebih dari 1 di dalam sebuah table menggunakan factory. Caranya buat sebuah file factory terlebih dahulu yaitu dengan mengetikkan

php artisan make:factory NamaModelFactory

di dalam terminalnya. Kemudian jika factory sudah berhasil dibuat, di dalam function definition lalu ada return []; isi dengan code

'kategori' => $this->faker->name(),

Yang berarti untuk kolom kategori akan diisikan dengan data faker atau data palsu dengan tipe data palsu sebuah nama. Kemudian di file seedernya bagian run diisi dengan

NamaModel::factory(5)->create();

Yang berarti dari model tersebut (model yang sudah memiliki factory) dijalankan factorynya sebanyak 5 data untuk dibuat atau ditambahkan ke dalam tablenya.

23. Mengambil data menggunakan view. Caranya dengan mengaturnya di dalam controller terlebih dahulu, karena berhubungan dengan database maka import atau tambahkan model dulu di dalam controllernya

use App\Models\KategoriModel;

kemudian di dalam function index nya isi dengan code

$kategoris = KategoriModel :: all();

return view ('home', compact('kategoris'));

yang berarti $kategori sebagai variable yang menyimpan data yang diambil dari KategoriModel dimana model ini tersambung dengan table kategoris, untuk data yang diambil adalah semua atau all. Kemudian return view ke home atau buka view home dengan dikirim ‘kategoris’ untuk bisa digunakan di home. Di dalam view home nya tambahkan code

@foreach ($kategoris as $isi)

    <li> {{$isi->kategori}} </li>

@endforeach

Yang berarti menggunakan foreach ini dia mengambil data yang dikirim ke file home ini yaitu $kategori untuk dimasukkan ke dalam $isi, kemudian $isi menampilkan data yang diambil dari kolom kategori untuk ditampilkan di dalam li. Maka hasilnya akan muncul seperti list (li) dengan data-data yang diambil dari table kategoris bagian kolom kategori.

24. Memindahkan data request yang dimasukkan dari halaman satu ke halaman lain menggunakan form. Caranya buat sebuah form terlebih dahulu dengan method post seperti ini

<form action="/store" method="post">

        @csrf

        <input class="border" type="text" name="kategori">

        <input type="submit" name="btn" value="simpan">

</form>

Jangan lupa diberi @scrf agar proses post berhasil. Kemudian karena form ini memiliki action /store artinya apabila sudah melakukan input kemudian klik button dia akan menambahkan linknya dengan /store. Agar /store ini berfungsi, maka di dalam router web.php ditambahkan setting untuk router /store seperti ini

Route::post('/store', [HomeController::class,'store']);

Yang berarti dia akan menjalankan class HomeController bagian function store, di dalam HomeController bagian function store nya berisi seperti ini

public function store(Request $request)

    {

        echo $request->kategori;

    }

Jadi apabila function ini dijalankan maka dia harus menerima data request yang dikirim melalui form untuk dimasukkan ke dalam $request yang kemudian $request akan ditampilkan di halaman store ini menggunakan echo. Data $request yang dipilih sesuai dengan nama input yang ingin ditampilkan datanya yaitu input name kategori.

25. Insert data menggunakan form untuk dimasukkan ke dalam database dengan cara yang sama seperti nomer 24 namun function store di dalam HomeControllernya diubah menjadi seperti ini

$kategori = new KategoriModel();

$kategori->kategori = $request->kategori;

$kategori->save();

Yang berarti $kategori menyimpan atau membuat data baru untuk dimasukkan ke dalam table yang ada di KategoriModel kemudian $kategori ini juga memilih kolom mana yang akan dia beri data baru yaitu kolom kategori karena $kategori->kategori, kolom kategori ini akan diisi dengan $request atau data yang diambil dari form dengan nama input kategori. Kemudian $kategori menjalankan function save agar data yang diinputkan tersimpan dan masuk ke dalam databasenya.

26. Melakukan sebuah update data yang ada di dalam database melalui Laravel, caranya dengan menambahkan salah satu idkategori yang akan dicoba untuk diupdate ke dalam action dari formnya seperti ini misalnya /8 atau id 8 jadi actionnya /store/8. Kemudian mengatur routernya

Route::post('/store/{id}', [HomeController::class,'update']);

Yang berarti apabila linknya /store/{id} atau angka berapapun yang merupakan idkategori dia akan menjalankan HomeController function update. Di dalam function ini berisi

public function update($id, Request $request)

    {

        $kategoris = [

            'kategori' => $request->kategori,

        ];

        KategoriModel::where('idkategori', $id)

        ->update($kategoris);

        return redirect("/");

    }

Yang berarti function ini menerima $id sesuai dengan angka pada linknya missal /8 maka idnya 8 dan menerima data Request untuk dimasukkan ke dalam variable $request. Kemudian terdapat $kategoris yang menyimpan data untuk mengisi kolom ‘kategori’ dengan data $request yang diambil dari input bernama kategori. Kemudian function ini menjalankan KategoriModel untuk mencocokkan data dengan idkategori yang sama dengan $id yang didapatkan oleh function ini. Jika data tsb sudah ditemukan maka data tsb akan diupdate dengan $kategoris (data request/data yang dimasukkan di dalam inputan formnya). Kemudian ada return redirect(“/”) yang berfungsi untuk kembali pada halaman ini meskipun telah melakukan submit pada form (jadi tidak pindah halaman, seperti langsung terrefresh).

27. Menghapus data yang ada di database menggunakan Laravel. Caranya dengan menambahkan tulisan hapus yang dapat diklik disetiap data yang sudah tampil di browser seperti ini

-- <a href="/store/{{$isi->idkategori}}">Hapus</a>

Jadi aka nada tulisan hapus disamping kanan dari data yang tampil di browser kemudian apabila Hapus ini diklik maka dia akan menjadikan linknya ada tambahann /store/idkategori sesuai dengan data yang dipilih untuk dihapus, mengapa $isi->idkategori karena sebelumnya ada $isi yang merupakan variable yang mengambil data database, jadi jika $isi->idkategori sama saja ambil data idkategorinya. Kemudian jika berhasil mengambil idkategori, pada routernya ditambahkan

Route::get('/store/{id}', [HomeController::class,'destroy']);

Karena dalam hapus ini menggunakan method get maka dirouternya apabila linknya /store/{id} dengan method get dia akan menjalankan HomeController bagian function destroy. Karena dari router ini sudah mengambil data idkategori yang akan dihapus, maka function destroy dapat menerima $id tersebut seperti ini

public function destroy($id)

    {

        KategoriModel::where('idkategori', $id)->delete();

        return redirect("/");

    }

Function destroy sudah mendapatkan $id yang berupa idkategori, kemudian function ini menjalankan code yaitu pencocokkan data yang ada didatabase bagian idkategori dengan $id yang didapat kemudian jika sudah mendapat data yang dicari, data tsb akan didelete atau dihapus.

28. Memberikan validasi untuk inputan yang ada di form. Caranya dengan menambahkan code

$this->validate($request, [

            'kategori' => 'required | max:30',

        ]);

Di dalam function yang digunakan untuk insert data yaitu store, arti dari code tersebut adalah data yang diinputkan atau $request dari inputan kategori harus terisi dan maksimal 30 karakter. Kemudian agar user tau apabila inputannya kosong dan belum diisi sedangkan syaratnya harus terisi, maka dapat ditambahkan sebuah message error di atas inputannya seperti ini

@error('kategori')

    <p>{{$message}}</p>

@enderror

Dengan begitu apabila inputan tidak sesuai dengan validate maka dia akan memunculkan pesan errornya.

29. Memindahkan atau memasukkan file yang dipilih dari form ke dalam folder images. Caranya dengan memberikan form input file terlebih dahulu kemudian ketika disimpan dia akan menjalankan function store seperti biasa dengan code

$name = $request->file('image')->getClientOriginalName();

$request->image->move(public\_path('images'),$name);

dd('sukses');

jadi $name adalah variable yang menyimpan data dari inputan atau request file bernama image dan diambil nama dari filenya juga. Kemudian gambar inputan ini tadi dipindahkan ke dalam folder images yang ada di folder public. Apabila proses ini berhasil maka setelah klik simpan akan muncul pesan sukses.

30. Menambahkan beberapa theme di dalam config tailwind-css agar tampilan web jadi lebih tertata dan rapi sesuai dengan settingnya.

31. Membuat model all Laravel, jadi mengetikkan 1 code untuk banyak file. Maksudnya apabila pembuatan sebuah model diberi keterangan untuk semua atau -a/--all maka tidak hanya file model yang dibuat tetapi sekaligus dengan controller, seeder, migrations, dll.

**Saya Belum Mengerti**

1.