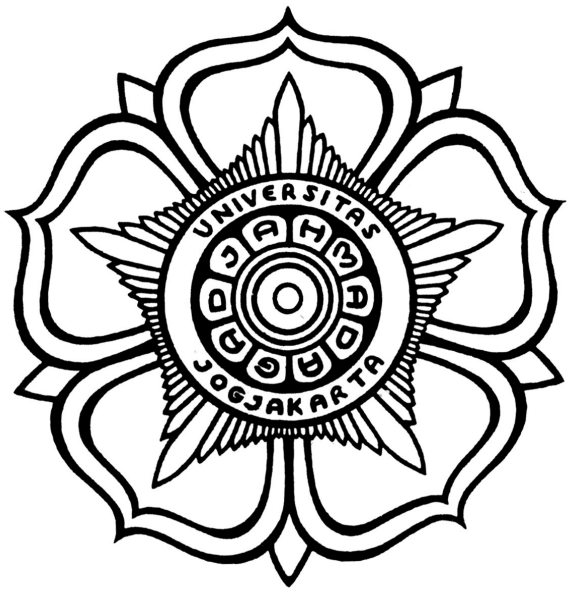
LAPORAN PRAKTIKUM PEMROGRAMAN WEB 2

PERTEMUAN 5

Laravel Controller, Model, dan Migration



NAMA : Aliif Arief Maulana

NIM : 21/479029/SV/19418

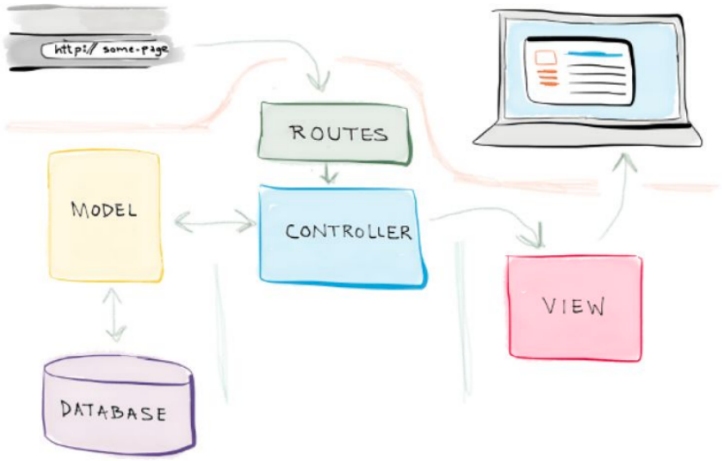
Dosen Pengampu : Achmad Choirudin Emcha, S.Kom., M.Eng.

PROGRAM STUDI D-IV TEKNOLOGI REKAYASA PERANGKAT LUNAK
DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA
SEKOLAH VOKASI
UNIVERSITAS GADJAH MADA
YOGYAKARTA

2022

Pendahuluan

Pada praktikum kali ini kita membahas tentang dasar dari Controller, Models, dan Migration yang merupakan elemen penting dalam framework Laravel yang memiliki konsep MVC(model, view, controller).



Pembahasan

1. **Controller**

   Controller adalah salah satu komponen inti dari MVC yang berfungsi sebagai penghubung antara request user (View) ke model yang nantinya akan dikembalikan lagi ke View dalam bentuk response. Pada controller akan banyak berisi logika – logika dalam menyusun suatu fungsi tertentu aau biasa disebut dengan bussiness process. Contoh sederhananya adalah aktivitas CRUD (Create, Read, Update, Delete) yang prosesnya berjalan di dalam Controller.

   Dalam framework laravel kita dapat membuat controller dengan dua cara yang pertama adalah dengan membuatnya manual pada folder App/Http/Contollers misal PostController.php dan buat nama class nya sesua dengan nama filenya lalu classnya diexntends dengan class Controller.

   Cara pertama yang manual itu terlalu rumit karena itu Laravel dapat mengotomatisasinya dengan perintah artisan dalam membuat sebuah controller contohnya kita akan membuat PostController maka ketikan perintah artisan seperti kode diabawah ini pada terminal:

```
php artisan make:controller PostController
```

Atau

```
php artisan make:controller PostController --resource
```

Kedua perintah diatas memiliki perbedaan yaitu tambahan -–resource yang berfungsi untuk men-generate berbagai fungsi untuk pembuatan CRUD (Create, Read, Update, dan Delete) pada Database. Jika Controller tidak digunakan untuk akses Database atau Model, opsi tersebut tidak perlu disertakan.

Setelah membuat controller tentu kita perlu membuat route nya sebagai endpointnya, sekarang kita pergi ke folder routes pada file web.php kita dapat membuat beberapa route nya untuk melakukan CRUD untuk PostController seperti kode dibawah ini:

```
Route::get('/post','PostController@post');
Route::get('/post/create','PostController@create');
Route::post('/post','PostController@store');
Route::get('/post/{id}','PostController@show');
Route::get('/post/{id}/edit','PostController@edit');
Route::put('/post/{id}','PostController@update');
Route::delete('/post/{id}','PostController@destroy');
```

Daripada membuat route yang banyak seperti kode diatas kita dapat menyingkatkannya dengan menuliskan kode seperti dibawah ini:

```
Route::resource('posts', 'App\Http\Controllers\PostController');
```

Dengan menggunakan kode yang lebih pendek seperti diatas namun dapat memiliki fungsi yang sama saja.

2. **Model dan Migration**

Model merupakan salah satu dari bagian MVC yang akan berkomunikasi dengan database. Model yang sudah terhubung ke database akan digunakan/dipanggil via Controller sebagaimana konsep MVC itu berjalan.

Migration merupakan salah satu fitur Laravel yang berfungsi seperti version control untuk database. Melalui fitur ini sebuah team pengembangan web development akan dapat bekerja dalam team untuk mengelola dan modifikasi skema basis data aplikasi.

Migration biasanya dipasangkan dengan Schema Builder dari Laravel untuk dengan mudah membangun skema basis data aplikasi Anda. Facade Skema Laravel menyediakan dukungan untuk membuat dan memanipulasi tabel di semua sistem basis data yang didukung Laravel.

Pertama buat database dahulu samakan nama database dengan config env di laravelnya lalu setelah itu buat modelnya dengan menggunakan perintah artisan seperti diabawah ini:

```
php artisan make:model Post --migration
```

Setelah itu model akan otomatis di generate laravel dan ada di app/Models/Post.php dan file migration akan ada di folder database, setelah itu kita kustomisasi file migrationnya sesuai dengan kebutuhan schema database yang kita inginkan kali ini pada praktikum file migationnya seperti kode diabawah ini:

```
public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string('title', 200);
            $table->mediumText('description');
            $table->timestamps();
        });
    }
```

Pada file migration function up digunakan untuk menggenerate table daatabase sedangkan function down untuk menghapus table, pastikan database nya sudah dijalankan, database sudah dibuat dan dikonfigurasikan sebagaimana mestinya agar tidak error, lalu setelah itu jalankan perintah artisan seperti dibawah ini:

```
$ php artisan migrate
```

Setelah itu laravel akan otomatis membuatkan table - tabel yang sudah kita tentukan tadi di database yang sudah kita tentukan juga.

3. **CRUD (create, read, update, delete)**

Pada praktikum kali ini kita akan membuat proses CRUD dengan laravel dan MySQL, pertama kita akan membuat beberapa data dummy dulu langsung dari database insert datanya setelah sudah kita akan membuat function untuk menampilkan data dari database ke tampilan website, untuk melakukan maka kita harus pergi ke Controller tepatnya pada PostController.php lalu didalam class nya kita akan membuat logikanya pada function index, berikut kode untuk menampilkan semua data ke dalam webite:

```php
public function index()
{
    $data = array(
        'id' => "posts",
        'posts' => Post::all()
    );
    return view('posts.index')->with($data);
}
```

Dapat dilihat dari kode diatas kita membuat varibel data yang berisi array yang memiliki key id dengan dilai posts dan key posts dengan isi array hasil query ke database yang tentunya berisi array juga sebagai informasi dalam framework laravel untuk melakukan query database kita menggunakan ORM (Object Relational Mapping) Eloquent secara default yang sudah disediakan dengan laravel sehingga penulisan kode php menjadi lebih rapih karena meminimalisir syntax raw SQL.

Fungsi index akan mengembalikan view yaitu file index.blade.php pada folder posts yang terletak didalam folder views selain itu juga mengirimkan datanya ke dalam file bladenya, berikut kode file blade untuk menampilkan datanya:

```
@extends('layouts.base')

@section('content')
    <div class="jumbotron jumbotron-fluid">
        <div class="container">
            <h1>Blog Post</h1>
            @if (count($posts)>0)
                @foreach ($posts as $p)
                    <div class="well">
                        <h3><a href="/posts/{{$p->id}}">{{$p->title}}</a></h3>
                        <small>Tanggal: {{$p->created_at}}</small>
                    </div>
                @endforeach
            @else
                <p>Belum ada post</p>
            @endif
        </div>
    </div>
@endsection
```

Setelah kita bisa menampilkan list datanya maka saatnya kita membuat fungsi untuk menampilkan datanya satu persatu namun secara lebih detail lagi, untuk itu kita memodifikasi fungsi show dalam file PostController.php seperti dibawah ini:

```
public function show($id)
{
    $data = array(
        'id' => "posts",
        'posts' => Post::find($id)
    );
    return view('posts.show')->with($data);
}
```

Dari kode diatas dapat dilihat kita menggunakan fungsi find dalam eloquent untuk melakukan query ke database dengan parameter id yang merupakan primary key untuk mendapatkan data lebih detailt sesuai dengan id nya lalu fungsi juga akan mengembalikan views dan mengirimkan data ke dalam file blade nya juga, lalu buat file show.blade.php dalam folder views/posts seperti dibawah ini:

```
@extends('layouts.base')
@section('content')
    <div class="jumbotron jumbotron-fluid">
        <div class="container">
            <h1>{{$posts->title}}</h1>
            <small>Tanggal: {{$posts->created_at}}</small>
            <p>{{$posts->description}}</p>
        </div>
    </div>
@endsection
```

Setelah itu kita akan membuat fungsi untuk menambahkan data lewat tampilan website yaitu dengan menggunakan form untuk membuat fitur tambah data maka pertama kita modifikasi fungsi create pada PostController kembalikan fungsi dengan views blade tampilan form untuk menambahkan datanya lalu buat file blade form untuk mengissi data yang ingin ditambahkan ke database lalu modifikasi fungsi store dalam PostController untuk menerima request http post dari form untuk menyimpannya ke dalam database berikut kodenya:

```
@extends('layouts.base')
@section('content')
    <div class="jumbotron jumbotron-fluid">
        <div class="container">
            <h1>Add Blog Post</h1>
            <form action="{{route('posts.store')}}" method="POST">
                {{ csrf_field() }}
                <div class="form-group">
                    <label for="title">Title</label>
                    <input type="text" name="title" id="title" class="form-control">
                </div>
                <div class="form-group">
                    <label for="description">Description</label>
                    <textarea type="text" name="description" id="description" class="form-control" rows="5">
</textarea>
                </div>
                <button type="submit" class="btn btn-primary">Submit</button>
            </form>
        </div>
    </div>
@endsection
```

```
public function create()
{
    return view('posts.create');
}
```

```php
public function store(Request $request)
{
    $post = new Post;
    $post->title = $request->input('title');
    $post->description = $request->input('description');
    $post->save();
}
```

Setelah itu kita dapat menambahkan data baru lewat form yang ada dalam permalink localhost/posts/create.

Tugas:

Dalam praktikum kali ini kita diberikan tugas untuk membuat website implementasi dari pelajaran yang sudah kita pelajari oleh karena itu saya akan melakukan improvisasi dari tugas yang dikerjakan selama praktikum yaitu dengan menambahkan fitur untuk update dan delete data lewat tampilan websitenya.

Jadi pertama saya membuat fitur untuk melakukan update tiap data nya pertama ke file PostController edit fungsi edit dan update, fungsi edit berfungsi untuk menampilkan views formnya dan juga megirim data nya sebelum diedit lalu fungsi update akan berfungsi untuk menangkap request http PUT dari form edit data lalu akan menyimpan data terbarunya ke dalam database berikut kodenya:

```php
public function edit($id)
{
    $data = array(
        'id' => "posts",
        'posts' => Post::find($id)
    );
    return view('posts.edit')->with($data);
}
```

```php
public function update(Request $request, $id)
{
    $post = Post::find($id);
    $post->title = $request->input('title');
    $post->description = $request->input('description');
    $post->save();
}
```

```
@extends('layouts.base')
@section('content')
    <div class="jumbotron jumbotron-fluid">
        <div class="container">
            <h1>Edit Blog Post</h1>
            <form action="{{route('posts.update', $posts->id)}}" method="POST">
                @csrf
                @method('PUT')
                <div class="form-group">
                    <label for="title">Title</label>
                    <input type="text" name="title" id="title" class="form-control" value="{{$posts->title}}">
                </div>
                <div class="form-group">
                    <label for="description">Description</label>
                    <textarea type="text" name="description" id="description" class="form-control" rows="5">{{$posts-
>description}}</textarea>
                </div>
                <button type="submit" class="btn btn-primary">Submit</button>
            </form>
        </div>
    </div>
@endsection
```

Setelah itu kita dapat mengakses endpoint url localhost/posts/{id}/edit untuk melakukan edit tiap item data dalam database.

Setelah fitur edit data selessai dibuat saya membuat fungsi untuk melakukan delete tiap item datanya yaitu dengan memodifikasi fungsi destroy dalam PostController dengan parameter id dan fungsi delete dalam Eloquent kita dapat menghapus tiap item datanya berikut kodenya:

```php
public function destroy($id)
{
    $post = Post::find($id);
    $post->delete();
}
```

Selain itu kita akan mengedit file show.blade.php dengan menambahkan tombol edit dan delete berikut kodenya:

```
@extends('layouts.base')
@section('content')
    <div class="jumbotron jumbotron-fluid">
        <div class="container">
            <h1>{{$posts->title}}</h1>
            <small>Tanggal: {{$posts->created_at}}</small>
            <p>{{$posts->description}}</p>
            <div class="d-flex">
                <a class="btn btn-primary" style="margin-right: 10px" href="{{$posts->id}}/edit">Edit</a>
                <form action="{{route('posts.destroy', $posts->id)}}" method="POST">
                    @csrf
                    @method('DELETE')
                    <button type="submit" class="btn btn-danger">Delete</button>
                </form>
            </div>
        </div>
    </div>
@endsection
```

Setelah semua jadi maka website yang sudah memiliki fungsi CRUD secara sempurna karena sudah dapat menambah data (create), menampilkan data (read), edit data (update), destroy data (delete) sehingga selesai sudah website yang kita buat.

Untuk file project secara keseluruhan sudah saya upload di repository GitHub dengan nama branch week5, berikut link repository filenya:

https://github.com/aliifam/ppw2/tree/week5

Kesimpulan:

Setalah mengikuti dan mengerjakan tugas pada praktikum pekan kelima perkuliahan ini alhamdulillah saya semakin paham bagaimana framework laravel ini bekerja secara garis besar terutama konsep controller, model, dan migration.

Daftar Pustaka:

https://laravel.com/docs/9.x/

https://informatika.uc.ac.id/id/2019/10/laravel-model-2

https://techvblogs.com/blog/laravel-9-crud-application-tutorial-with-example