

Laporan Modul 6: Model dan Laravel Eloquent

Mata Kuliah: Workshop Web Lanjut

Nama: Hamizan Putra Zulia **NIM:** 2024573010013 **Kelas:** TI-2C

Abstrak

Laporan praktikum ini membahas tentang penerapan Model dan Laravel Eloquent ORM dalam pembuatan aplikasi berbasis web menggunakan framework Laravel 12. Tujuan dari praktikum ini adalah untuk memahami bagaimana cara menghubungkan aplikasi dengan database menggunakan model, bagaimana Eloquent ORM bekerja dalam menangani proses CRUD (Create, Read, Update, Delete), dan bagaimana penggunaan DTO (Data Transfer Object) dapat membantu menjaga struktur kode agar tetap bersih dan mudah dikembangkan. Selain itu, laporan ini juga mencakup implementasi aplikasi sederhana seperti Todo App yang menggunakan database MySQL dengan konsep Eloquent ORM. Dengan adanya praktikum ini, diharapkan mahasiswa mampu memahami cara kerja model di Laravel serta dapat membangun aplikasi yang efisien, terstruktur, dan mudah dipelihara.

1. Dasar Teori

Pengertian MVC (Model, View, Controller)

MVC adalah sebuah pendekatan perangkat lunak yang memisahkan aplikasi logika dari presentasi. MVC memisahkan aplikasi berdasarkan komponen-komponen aplikasi, seperti : manipulasi data, controller, dan user interface.

Model, Model mewakili struktur data. Biasanya model berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaruan data dan lain-lain.

View, View adalah bagian yang mengatur tampilan ke pengguna. Bisa dikatakan berupa halaman web.

Controller, Controller merupakan bagian yang menjembatani model dan view.

Contoh MVC pada Laravel

Pertama, kita buat controller dengan perintah artisan dengan nama controller BelajarController.

Menggunakan Data Transfer Objects (DTO) di Laravel untuk Arsitektur yang Bersih dan Skalabel

Dalam aplikasi Laravel, penting untuk menjaga kode tetap mudah dibaca, mudah dipelihara, dan skalabel. Salah satu pendekatan yang dapat Anda lakukan untuk meningkatkan struktur kode adalah dengan menggunakan Objek Transfer Data (DTO). Teknik ini membantu memisahkan masalah dengan menciptakan objek yang hanya membawa data antar lapisan aplikasi yang berbeda.

Mengapa Menggunakan DTO?

DTO menawarkan beberapa keuntungan:

Pemisahan Permasalahan : Membantu menjaga agar berbagai lapisan aplikasi Anda terpisah dengan baik.

Kejelasan Kode : Dengan mengirimkan objek sederhana yang hanya berisi data, Anda meningkatkan keterbacaan dan pemeliharaan kode Anda.

Skalabilitas : DTO membuat kode Anda lebih mudah diuji dan diperluas tanpa menduplikasi logika.

Apa itu Eloquent ORM? Eloquent ORM adalah fitur dari Laravel yang memungkinkan developer untuk dapat menggunakan dan memanipulasi data yang ada di dalam database dengan PHP objects dan model terkait.

Sebagai Object-Relational Mapping (ORM) yang disediakan oleh Framework Laravel, Eloquent memiliki fungsi query SQL yang memungkinkan untuk mengelola data dalam database tanpa harus menuliskannya kembali secara manual.

ORM sendiri mengacu pada suatu metode pemrograman yang menghubungkan database relasional dengan kode pemrograman yang berorientasi sebagai objek.

Hal ini memungkinkan interaksi antara database dan objek menjadi mudah, sehingga developer dapat memanfaatkan kode saat bekerja alih-alih menulis SQL secara manual.

Model atau kode pemrograman yang berbentuk objek tersebut kemudian dipergunakan untuk melakukan operasi data seperti CRUD (Create, Read, Update, Delete) secara efisien dan mudah dalam database.

2. Langkah-Langkah Praktikum

2.1 Praktikum 1 - Menggunakan Model untuk Binding Form dan Display

- Langkah-langkah:

1. Buat proyek baru dan masuk kedalamnya

```
Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013/projects (main)
$ laravel new model-app
```

2. Buat model data sederhana (POCO)

```
Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013/projects (main)
$ mkdir app/ViewModels
```

3. Masuk ke direktori app\ViewModels dan isi:

```
<?php
namespace App\ViewModels;

class ProductViewModel
{
    public string $name;
    public float $price;
    public float $description;

    public function __construct(string $name = '', float $price = 0, string $description = '')
    {
        $this->name = $name;
        $this->price = $price;
        $this->description = $description;
    }

    public static function fromRequest(array $data): self
    {
        return new self(
            $data['name'] ?? '',
            (float)($data['price'] ?? 0),
            $data['description'] ?? ''
        );
    }
}
```

4. Buat controller productController

```
Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573
010013/projects/dto-app (main)
$ php artisan make:controller ProductController
```

5. Edit file ProductController.php dengan mengisi:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\DTO\ProductDTO;
use App\Services\ProductService;

class ProductController extends Controller
{
    public function create() {
        return view('product.create');
    }

    public function result(Request $request) {
        $dto = ProductDTO::fromRequest($request->all());
        $service = new ProductService();
        $product = $service->display($dto);
        return view('product.result', compact('product'));
    }
}
```

6. Definiskan route/web.php:

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ProductController;

Route::get('/product/create', [ProductController::class, 'create'])->name('product.create');
Route::get('/product/result', [ProductController::class, 'result'])->name('product.result');
```

7. Buat direktori resources/views/product

8. Buat dua file bernama create.blade.php dan result.blade.php dan isikan create.blade.php :

```
1  @extends('layouts.app')
2
3  @section('title', 'Buat Task Baru')
4
5  @section('content')
6      <h2>Buat Task Baru</h2>
7
8      <form action="{{ route('todos.store') }}" method="POST" class="mt-3">
9          @csrf
10         <div class="mb-3">
11             <label for="task" class="form-label">Nama Task</label>
12             <input type="text" name="task" id="task" class="form-control" required>
13         </div>
14         <button type="submit" class="btn btn-success">Tambah Task</button>
15         <a href="{{ route('todos.index') }}" class="btn btn-secondary">Kembali ke Daftar</a>
16     </form>
17
18 @endsection
```

result.blade.php

```

1  <!DOCTYPE html>
2  <html>
3
4  <head>
5    <title>Product Result</title>
6    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
7  </head>
8
9  <body class="container py-5">
10   <h2>Submitted Product Details</h2>
11   <ul class="list-group">
12     <li class="list-group-item"><strong>Name:</strong> {{ $product->name }}</li>
13     <li class="list-group-item"><strong>Price:</strong> ${{ number_format($product->price, 2) }}</li>
14     <li class="list-group-item"><strong>Description:</strong> {{ $product->description }}</li>
15   </ul>
16   <a href="{{ route('product.create') }}" class="btn btn-link mt-3">Submit Another Product</a>
17 </body>
18
19 </html>

```

9. Dan web sudah selesai, sudah bisa dijalankan menggunakan 'php artisan serve'

2.2 Praktikum 2 - Menggunakan DTO (Data Transfer Object)

- Langkah-langkah:

1. Buat proyek baru dan masuk ke dalam proyek nya

```

Administrator@WIN-L8SM4Q0J2T 10013/projects/role-lab (main)
$ laravel new dto-app

```

2. Buat direktori baru

3. Buat dan masuk ke file app/DTO/ProductDTO.php dan isi:

```

Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013 (main)
● $ cd projects/

Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013/projects
(main)
○ $ laravel new dto-app

```

4. Lalu buat folder baru app/Services/ProductServices.php dan isikan:

```
<?php

namespace App\Services;

use App\DTO\ProductDTO;

class ProductService {
    public function display(ProductDTO $product): array {
        return [
            'name' => $product->name,
            'price' => $product->price,
            'description' => $product->description,
        ];
    }
}
```

5. buat controller baru dengan nama ProductController

```
Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573
010013/projects/dto-app (main)
$ php artisan make:controller ProductController
```

6. Masuk dan edit file ProductController.php

```
<?php

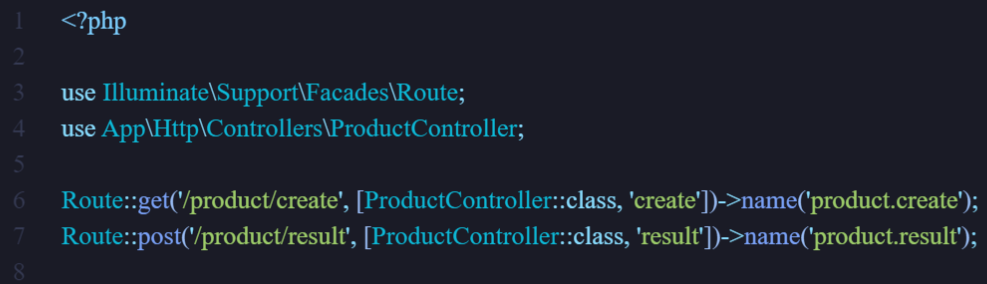
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\DTO\ProductDTO;
use App\Services\ProductService;

class ProductController extends Controller
{
    public function create() {
        return view('product.create');
    }

    public function result(Request $request) {
        $dto = ProductDTO::fromRequest($request->all());
        $service = new ProductService();
        $product = $service->display($dto);
        return view('product.result', compact('product'));
    }
}
```

7. Edit file routes/web.php

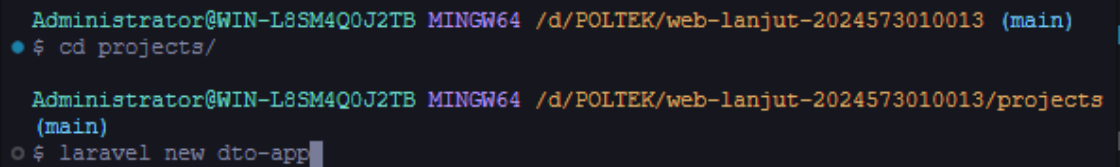


```
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\ProductController;
5
6 Route::get('/product/create', [ProductController::class, 'create'])->name('product.create');
7 Route::post('/product/result', [ProductController::class, 'result'])->name('product.result');
8
```

8. Buat direktori product dalam resources/views

9. Buat 2 file view dalam folder product

■ create.blade.php



```
Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013 (main)
● $ cd projects/

Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013/projects
(main)
○ $ laravel new dto-app
```

■ result.blade.php

```

1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <title>Product Result</title>
6      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
7  </head>
8
9  <body class="container py-5">
10     <div class="row justify-content-center">
11         <div class="col-md-6">
12             <h2 class="mb-4">Product DTO Result</h2>
13             <div class="card">
14                 <div class="card-header">
15                     <h5 class="card-title mb-0">Product Details</h5>
16                 </div>
17                 <ul class="list-group list-group-flush">
18                     <li class="list-group-item">
19                         <strong>Name:</strong> {{ $product['name'] }}
20                     </li>
21                     <li class="list-group-item">
22                         <strong>Price:</strong> ${{ number_format($product['price'], 2) }}
23                     </li>
24                     <li class="list-group-item">
25                         <strong>Description:</strong> {{ $product['description'] }}
26                     </li>
27                 </ul>
28             </div>
29             <a href="{{ route('product.create') }}" class="btn btn-secondary mt-3">Submit Another Product</a>
30         </div>
31     </div>
32 </body>
33
34 </html>

```

10. Sudah selesai, dan jalankan web menggunakan 'php artisan serve'

2.3 Praktikum 3 - Membangun Aplikasi Web Todo Sederhana dengan Laravel 12, Eloquent ORM, dan MySQL

• Langkah-langkah:

1. Buat projek baru dan masuk ke dalam projek nya

```

Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573
010013/projects (main)
$ laravel new todo-app-mysql

```

2. Install dependensi MySQL

```
23 DB_CONNECTION=mysql
24 DB_HOST=127.0.0.1
25 DB_PORT=3306
26 DB_DATABASE=todo_app
27 DB_USERNAME=root
28 DB_PASSWORD=
```

PROBLEMS 2 OUTPUT TERMINAL ... mysql - projects + - □ □ ... ^ x

Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013/projects (main)

○ \$ mysql -uroot

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database todo_app;
Query OK, 1 row affected (0.06 sec)

3. Konfigurasi bagian .env

```
23 DB_CONNECTION=mysql
24 DB_HOST=127.0.0.1
25 DB_PORT=3306
26 DB_DATABASE=todo_app
27 DB_USERNAME=root
28 DB_PASSWORD=
```

PROBLEMS 2 OUTPUT TERMINAL ... mysql - projects + - □ □ ... ^ x

Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013/projects (main)

○ \$ mysql -uroot

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database todo_app;
Query OK, 1 row affected (0.06 sec)

4. Buat migration todos

```
Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013/projects/todo-app-mysql (main)
$ php artisan make:migration create_todos_table
```

5. isi file create_todos_tables yang baru dibuat dengan

```
Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013/projects/todo-app-mysql (main)
$ php artisan make:migration create_todos_table
```

6. Jalankan 'php artisan migrate'

7. Jalankan perintah untuk membuat seeder dengan cara:

```
Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013/projects/todo-app-mysql (main)
$ php artisan make:seeder TodoSeeder

INFO Seeder [D:\POLTEK\web-lanjut-2024573010013\projects\todo-app-mysql\database\seeders\TodoSeeder.php] created successfully.
```

8. Dalam file seeder yg baru dibuat isikan:

```
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8  use Carbon\Carbon;
9
10 class TodoSeeder extends Seeder
11 {
12     public function run()
13     {
14         DB::table('todos')->insert([
15             [
16                 'task' => 'Belanja bahan makanan',
17                 'completed' => false,
18                 'created_at' => Carbon::now(),
19                 'updated_at' => Carbon::now()
20             ],
21             [
22                 'task' => 'Beli buah-buahan',
23                 'completed' => false,
24                 'created_at' => Carbon::now(),
25                 'updated_at' => Carbon::now()
26             ],
27             [
28                 'task' => 'Selesaikan proyek Laravel',
29                 'completed' => true,
30                 'created_at' => Carbon::now(),
31                 'updated_at' => Carbon::now()
32             ],
33         ]);
34     }
35 }
```

9. Jalankan perintah 'php artisan db:seed --class=TodoSeeder'

10. Buat model bernama Todo

```
Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573010013/projects/todo-app-mysql (main)
$ php artisan make:model Todo
```

11. Isikan file yang baru ditambahkan dengan:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Todo extends Model
{
    use HasFactory;

    protected $fillable = ['task', 'completed'];
}
```

12. Jalankan perintah untuk membaut TodoController:

```
Administrator@WIN-L8SM4Q0J2TB MINGW64 /d/POLTEK/web-lanjut-2024573
010013/projects/todo-app-mysql (main)
$ php artisan make:controller TodoController
```

13. Isi kode ini untuk file TodoController.php:

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\Todo;
7
8 class TodoController extends Controller
9 {
10     public function index()
11     {
12         $todos = Todo::all();
13         return view('todos.index', compact('todos'));
14     }
15
16     public function create()
17     {
18         return view('todos.create');
19     }
20
21     public function store(Request $request)
22     {
23         $request->validate(['task' => 'required|string']);
24         Todo::create(['task' => $request->task]);
25         return redirect()->route('todos.index')->with('success', 'Task added successfully!');
26     }
27
28     public function show(Todo $todo)
29     {
30         return view('todos.show', compact('todo'));
31     }
32 }
```

```
33     public function edit(Todo $todo)
34     {
35         return view('todos.edit', compact('todo'));
36     }
37
38     public function update(Request $request, Todo $todo)
39     {
40         $request->validate(['task' => 'required|string']);
41         $todo->update(['task' => $request->task]);
42         return redirect()->route('todos.index')->with('success', 'Task updated successfully!');
43     }
44
45     public function destroy(Todo $todo)
46     {
47         $todo->delete();
48         return redirect()->route('todos.index')->with('success', 'Task deleted successfully!');
49     }
50 }
51
```

14. Masuk dan edit routes/web.php dengan:

```
objects > todo-app-mysql > routes > web.php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\TodoController;

Route::get('/', [TodoController::class, 'index'])->name('todos.index');
Route::get('/todos/create', [TodoController::class, 'create'])->name('todos.create');
Route::post('/todos', [TodoController::class, 'store'])->name('todos.store');
Route::get('/todos/{todo}', [TodoController::class, 'show'])->name('todos.show');
Route::get('/todos/{todo}/edit', [TodoController::class, 'edit'])->name('todos.edit');
Route::patch('/todos/{todo}', [TodoController::class, 'update'])->name('todos.update');
Route::delete('/todos/{todo}', [TodoController::class, 'destroy'])->name('todos.destroy');
```

15. Buat foler layouts di resources/views dan buat file app.blade.php lalu isikan:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>@yield('title', 'Todo App')</title>
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
9 </head>
10
11 <body class="container mt-4">
12
13   <h1 class="text-center mb-4">Laravel 12 Todo App</h1>
14
15   @if(session('success'))
16   <div class="alert alert-success">{{ session('success') }}</div>
17   @endif
18
19   <nav class="mb-3">
20     <a href="{{ route('todos.index') }}" class="btn btn-primary">Todo List</a>
21     <a href="{{ route('todos.create') }}" class="btn btn-success">Add New Task</a>
22   </nav>
23
24   @yield('content')
25
26 </body>
27
28 </html>
```

16. Buat file didalam resources/views/todos/index.blade.php dan isikan

```
1  @extends('layouts.app')
2
3  @section('title', 'Daftar Todo')
4
5  @section('content')
6  <h2>Daftar Todo</h2>
7
8  <ul class="list-group">
9      @foreach($todos as $todo)
10     <li class="list-group-item d-flex justify-content-between align-items-center">
11         {{ $todo->task }}
12         <div>
13             <form action="{{ route('todos.show', $todo->id) }}" method="GET" class="d-inline">
14                 <button type="submit" class="btn btn-info btn-sm">Detail</button>
15             </form>
16             <form action="{{ route('todos.edit', $todo->id) }}" method="GET" class="d-inline">
17                 <button type="submit" class="btn btn-warning btn-sm">Edit</button>
18             </form>
19             <form action="{{ route('todos.destroy', $todo->id) }}" method="POST" class="d-inline">
20                 @csrf
21                 @method('DELETE')
22                 <button class="btn btn-danger btn-sm">Hapus</button>
23             </form>
24         </div>
25     </li>
26 @endforeach
27 </ul>
28 @endsection
```

17. Buat file didalam resources/views/todos/create.blade.php dan isikan

```
1  @extends('layouts.app')
2
3  @section('title', 'Buat Task Baru')
4
5  @section('content')
6  <h2>Buat Task Baru</h2>
7
8  <form action="{{ route('todos.store') }}" method="POST" class="mt-3">
9      @csrf
10     <div class="mb-3">
11         <label for="task" class="form-label">Nama Task</label>
12         <input type="text" name="task" id="task" class="form-control" required>
13     </div>
14     <button type="submit" class="btn btn-success">Tambah Task</button>
15     <a href="{{ route('todos.index') }}" class="btn btn-secondary">Kembali ke Daftar</a>
16 </form>
17
18 @endsection
```

18. Buat file didalam resources/views/todos/edit.blade.php dan isikan

```
1  @extends('layouts.app')
2  @section('title', 'Edit Task')
3
4  @section('content')
5  <h2>Edit Task</h2>
6
7  <form action="{{ route('todos.update', $todo->id) }}" method="POST" class="mt-3">
8      @csrf
9      @method('PATCH')
10     <div class="mb-3">
11         <label for="task" class="form-label">Nama Task</label>
12         <input type="text" name="task" id="task" class="form-control" value="{{ $todo->task }}" required>
13     </div>
14     <button type="submit" class="btn btn-warning">Update Task</button>
15     <a href="{{ route('todos.index') }}" class="btn btn-secondary">Kembali ke Daftar</a>
16 </form>
17
18 @endsection
```

19. Buat file didalam resources/views/todos/show.blade.php dan isikan

```
1 @extends('layouts.app')
2 @section('title', 'Detail Task')
3 @section('content')
4 <h2>Detail Task</h2>
5
6 <div class="card mt-3">
7     <div class="card-body">
8         <h5 class="card-title">{{ $todo->task }}</h5>
9         <p class="card-text">Status: {{ $todo->completed ? 'Selesai' : 'Belum Selesai' }}</p>
10        <a href="{{ route('todos.edit', $todo->id) }}" class="btn btn-warning">Edit</a>
11        <a href="{{ route('todos.index') }}" class="btn btn-secondary">Kembali ke Daftar</a>
12    </div>
13 </div>
14 @endsection
```

20. Dan Web app sudah selesai, sudah bisa jalankan 'php artisan serve'

3. Hasil dan Pembahasan

1. Praktikum 1 – Menggunakan Model untuk Binding Form dan Display

Pada praktikum pertama, dibuat sebuah model sederhana untuk menampilkan dan menerima data dari form. Model ini berfungsi sebagai penghubung antara form input pengguna dan tampilan hasil (view). Controller bertugas menerima input dari form, lalu mengirimkannya ke view menggunakan model yang telah dibuat. Proses ini menunjukkan bagaimana model di Laravel dapat digunakan untuk mengatur dan mengelola data secara efisien tanpa harus terhubung langsung ke database. Dari hasil percobaan, data yang dikirim melalui form berhasil ditampilkan kembali di halaman hasil, yang berarti komunikasi antara controller, model, dan view berjalan dengan baik.

2. Praktikum 2 – Menggunakan DTO (Data Transfer Object)

Pada bagian ini, dilakukan implementasi DTO (Data Transfer Object) untuk memisahkan logika bisnis dari logika data. DTO membantu agar data yang dikirim antar bagian aplikasi tetap rapi dan tidak bercampur dengan kode logika lainnya. Data yang dikirim melalui form disimpan di dalam objek DTO, lalu diteruskan ke service untuk diproses lebih lanjut. Pendekatan ini membuat kode lebih mudah dibaca, lebih aman dari kesalahan manipulasi data, serta lebih mudah di-maintain dan dikembangkan di masa depan. Dari hasil praktikum, data dari DTO berhasil ditampilkan dan diproses di controller, membuktikan bahwa DTO berfungsi dengan baik sebagai perantara data antar lapisan aplikasi.

3. Praktikum 3 – Membangun Aplikasi Web Todo Sederhana dengan Laravel 12, Eloquent ORM, dan MySQL

Praktikum ini merupakan penerapan langsung konsep Eloquent ORM. Dibuat aplikasi Todo List yang dapat melakukan operasi CRUD (Create, Read, Update, Delete) menggunakan database MySQL. Langkah pertama

adalah membuat migrasi dan model Todo, kemudian membuat controller yang menangani logika untuk menampilkan, menambah, mengedit, dan menghapus data todo. Setelah itu dibuat beberapa file tampilan seperti `index.blade.php`, `create.blade.php`, `edit.blade.php`, dan `show.blade.php`. Semua file ini saling terhubung melalui route dan controller, yang mengatur bagaimana data dari database ditampilkan dan diolah. Hasil dari percobaan menunjukkan bahwa semua fitur CRUD bekerja dengan baik — data yang disimpan bisa muncul di daftar todo, dapat diedit dan dihapus dengan lancar. Dengan menggunakan Eloquent ORM, proses interaksi dengan database menjadi lebih sederhana dan ringkas, tanpa perlu menulis query SQL secara manual.

Secara keseluruhan, hasil dari ketiga praktikum ini menunjukkan bahwa Laravel Eloquent ORM sangat memudahkan pengelolaan data, dan DTO membantu menjaga struktur kode tetap rapi dan mudah dikembangkan.

4. Kesimpulan

Dari hasil praktikum Modul 6 ini dapat disimpulkan bahwa:

Model di Laravel berfungsi sebagai penghubung antara aplikasi dan database, serta menjadi tempat utama dalam mengatur logika data.

Eloquent ORM mempermudah proses interaksi dengan database karena memungkinkan developer untuk melakukan operasi CRUD menggunakan kode PHP berorientasi objek tanpa menulis SQL secara langsung.

DTO (Data Transfer Object) sangat berguna untuk menjaga agar kode tetap terstruktur, bersih, dan mudah dikembangkan dengan memisahkan data dari logika bisnis.

Penerapan Eloquent dan DTO menjadikan aplikasi lebih efisien, aman, serta mudah untuk di-maintain.

Melalui pembuatan aplikasi Todo List sederhana, dapat dibuktikan bahwa Eloquent ORM bekerja dengan sangat baik dalam mengelola data dari database MySQL secara otomatis dan efisien.

Secara keseluruhan, modul ini memberikan pemahaman yang lebih dalam tentang bagaimana Laravel mengelola data menggunakan model dan ORM, serta pentingnya struktur kode yang baik melalui penerapan DTO dalam pengembangan aplikasi modern.

5. Referensi

- Pengertian MVC - <https://medium.com/@kevinffa0107/pengertian-mvc-model-view-controller-pada-framework-laravel-20f261ccf233>
- Laravel DTO - https://dev-to.translate.googleusercontent.com/using-data-transfer-objects-dto-in-laravel-for-a-clean-and-scalable-architecture-172o?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=tc
- Eloquent ORM - <https://www.sekawanmedia.co.id/blog/eloquent-orm/>