# Modul Ajar: Design Pattern MVC pada Laravel

## Pengenalan Laravel dan Design Pattern

Laravel adalah salah satu framework PHP yang paling populer. Salah satu alasan utamanya adalah penerapan prinsip **design pattern** yang baik, termasuk **Model-View-Controller (MVC)**. MVC adalah pola arsitektur yang memisahkan aplikasi menjadi tiga komponen utama: Model, View, dan Controller. Dengan memisahkan tanggung jawab, pengembangan aplikasi menjadi lebih terstruktur dan mudah untuk dipelihara.

## Apa itu MVC?

- **Model**: Bagian ini bertanggung jawab untuk mengelola data dan logika bisnis. Model berinteraksi dengan database dan menangani semua operasi yang berkaitan dengan data, seperti mengambil data, menyimpan data, dan memproses data. Dalam Laravel, model biasanya merupakan representasi dari tabel database.
- **View**: Ini adalah antarmuka pengguna yang menampilkan data kepada pengguna. View bertugas untuk menyajikan informasi yang diterima dari controller dalam bentuk yang dapat dimengerti oleh pengguna. Di Laravel, view biasanya menggunakan Blade, templating engine yang memungkinkan penulisan sintaksis PHP di dalam HTML.
- **Controller**: Komponen ini bertanggung jawab untuk mengendalikan alur aplikasi. Controller menerima input dari pengguna melalui view, berinteraksi dengan model untuk mengambil atau memproses data, dan mengembalikan hasilnya ke view. Controller bertindak sebagai penghubung antara model dan view.

## Struktur Direktori MVC dalam Laravel

Dalam proyek Laravel, struktur direktori mengikuti pola MVC dengan jelas:

- **app/Models**: Tempat model disimpan. Setiap model biasanya terkait dengan tabel tertentu dalam database.
- **app/Http/Controllers**: Tempat controller disimpan. Controller akan mengelola alur aplikasi dan logika bisnis.
- **resources/views**: Tempat file view disimpan. File-file ini bertanggung jawab untuk menampilkan data kepada pengguna.

## Contoh Implementasi MVC di Laravel

Untuk memberikan gambaran yang lebih jelas tentang cara kerja MVC di Laravel, berikut adalah contoh sederhana tentang pengelolaan data pengguna.

### 1. Membuat Model

Untuk membuat model, kita dapat menggunakan Artisan command:

```
bash

php artisan make:model User
```

Model `User` akan terletak di `app/Models/User.php`:

```
php

namespace App\Models;

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

class User extends Model
{
    use HasFactory;

    protected $fillable = ['name', 'email', 'password'];
}
```

Model ini menggunakan **Eloquent ORM** yang memudahkan kita dalam berinteraksi dengan database.

### 2. Membuat Controller

Selanjutnya, kita buat controller untuk mengelola logika terkait pengguna:

```
bash

php artisan make:controller UserController
```

Controller `UserController` akan terletak di `app/Http/Controllers/UserController.php`:

```
php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;

class UserController extends Controller
{
    public function index()
    {
        $users = User::all();
        return view('users.index', compact('users'));
    }

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

```
    public function store(Request $request)
    {
        User::create($request->all());
        return redirect()->route('users.index');
    }
}
```

Controller ini memiliki beberapa metode:

- **index()**: Mengambil semua data pengguna dan mengirimkannya ke view.
- **create()**: Menampilkan form untuk menambahkan pengguna baru.
- **store()**: Menyimpan data pengguna baru ke dalam database.

### 3. Membuat View

Kita juga perlu membuat view untuk menampilkan data pengguna. View ini akan berada di `resources/views/users/index.blade.php`:

```
blade

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Users</title>
</head>
<body>
    <h1>List of Users</h1>
    <a href="{{ route('users.create') }}">Add New User</a>
    <ul>
        @foreach ($users as $user)
            <li>{{ $user->name }} ({{ $user->email }})</li>
        @endforeach
    </ul>
</body>
</html>
```

Dan untuk view form pembuatan pengguna baru di `resources/views/users/create.blade.php`:

```
blade

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Create User</title>
</head>
<body>
```

```
    <h1>Create New User</h1>
    <form action="{{ route('users.store') }}" method="POST">
        @csrf
        <input type="text" name="name" placeholder="Name" required>
        <input type="email" name="email" placeholder="Email" required>
        <input type="password" name="password" placeholder="Password"
required>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

## 4. Menentukan Rute

Akhirnya, kita perlu mendefinisikan rute untuk controller di `routes/web.php`:

```
php

use App\Http\Controllers\UserController;

Route::resource('users', UserController::class);
```

Dengan `Route::resource`, kita sudah mendefinisikan semua rute yang dibutuhkan untuk operasi CRUD.

## Kesimpulan

MVC adalah pola arsitektur yang kuat yang membantu memisahkan logika aplikasi, membuat kode lebih terstruktur dan mudah dipelihara. Laravel menerapkan pola ini dengan sangat baik melalui Eloquent ORM untuk model, Blade untuk view, dan controller yang memproses logika aplikasi.

Dengan memahami MVC, Anda akan lebih mudah dalam membangun aplikasi yang kompleks dan terstruktur menggunakan Laravel. Selamat belajar!

# Modul Ajar: Dasar Login dengan Laravel

## Pengenalan

Salah satu fitur penting dalam aplikasi web adalah autentikasi pengguna. Laravel menyediakan berbagai alat untuk mengelola proses autentikasi dengan mudah dan efisien. Modul ini akan membahas cara dasar untuk membuat sistem login di Laravel.

## 1. Menginstal Laravel

Jika Anda belum menginstal Laravel, Anda dapat melakukannya dengan menjalankan perintah berikut:

```
bash

composer create-project --prefer-dist laravel/laravel myapp
```

Gantilah `myapp` dengan nama proyek Anda.

## 2. Mengkonfigurasi Database

Sebelum melanjutkan, Anda perlu mengonfigurasi koneksi database di file `.env`. Pastikan Anda sudah membuat database di server Anda, lalu ubah file `.env` seperti berikut:

```
plaintext

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=nama_database
DB_USERNAME=username_database
DB_PASSWORD=password_database
```

## 3. Menginstal Laravel Breeze

Untuk scaffolding autentikasi yang sederhana, kita akan menggunakan Laravel Breeze. Instal Breeze dengan menjalankan perintah berikut:

```
bash

composer require laravel/breeze --dev
```

Setelah itu, jalankan perintah berikut untuk menginstal Breeze:

```
bash

php artisan breeze:install
```

Kemudian jalankan migrasi untuk membuat tabel pengguna:

```
bash

php artisan migrate
```

## 4. Mengatur Rute

Laravel Breeze secara otomatis mengatur rute untuk login, register, dan reset password. Anda bisa menemukan rute ini di `routes/web.php`. Rute yang dihasilkan termasuk:

- `/login`: untuk halaman login
- `/register`: untuk halaman pendaftaran
- `/dashboard`: halaman yang hanya bisa diakses oleh pengguna yang sudah login

## 5. Mengakses Halaman Login

Sekarang Anda bisa mengakses halaman login melalui URL `http://localhost:8000/login`. Untuk menjalankan server lokal, gunakan perintah:

```
bash

php artisan serve
```

## 6. Proses Login

Ketika pengguna mengisi form login dan mengirimkan data, Laravel akan memverifikasi kredensial dengan menggunakan controller `Auth\AuthenticatedSessionController`. Jika kredensial valid, pengguna akan diarahkan ke halaman dashboard.

Berikut adalah contoh bagaimana form login terlihat di `resources/views/auth/login.blade.php`:

```
blade

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
</head>
<body>
    <h1>Login</h1>
    <form method="POST" action="{{ route('login') }}">
        @csrf
        <div>
            <label for="email">Email</label>
            <input type="email" name="email" required autofocus>
```

```
        </div>
        <div>
            <label for="password">Password</label>
            <input type="password" name="password" required>
        </div>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

## 7. Menangani Login yang Gagal

Jika pengguna memasukkan kredensial yang salah, Laravel akan menampilkan pesan kesalahan. Anda dapat mengubah pesan ini dengan memodifikasi file di `resources/lang/en/auth.php`.

## 8. Logout

Untuk logout, Laravel juga menyediakan rute dan controller yang sudah siap pakai. Pengguna dapat logout dengan mengunjungi URL `http://localhost:8000/logout`, dan Laravel akan menangani sesi pengguna.

## 9. Mengamankan Rute

Anda dapat mengamankan rute tertentu agar hanya bisa diakses oleh pengguna yang sudah login dengan menggunakan middleware `auth`. Contoh:

```
php

Route::get('/dashboard', function () {
    return view('dashboard');
})->middleware('auth');
```

## 10. Menambahkan Middleware

Jika Anda ingin membuat middleware khusus untuk memeriksa peran pengguna, Anda dapat menggunakan perintah Artisan untuk membuat middleware baru:

```
bash

php artisan make:middleware CheckRole
```

Di dalam middleware ini, Anda dapat memeriksa apakah pengguna memiliki peran tertentu dan mengarahkan pengguna jika tidak.

## Kesimpulan

Modul ini memberikan gambaran dasar tentang bagaimana mengimplementasikan sistem login di Laravel. Dengan menggunakan Laravel Breeze, proses pembuatan sistem autentikasi menjadi

sangat sederhana dan cepat. Anda dapat memodifikasi dan menyesuaikan autentikasi ini sesuai dengan kebutuhan aplikasi Anda.