TUGAS TEORI PEMROGAMAN BERORIENTASI OBJEK (PBO)

Nama : Fajar Mirza Hanif

NIM : 4342401061

Kelas : TRPL 2C PAGI

Soal :

1. Jelaskan konsep Model dalam arsitektur MVC (Model-View-Controller) di Laravel. Apa peran utama dari Model dalam arsitektur ini dan bagaimana Model berinteraksi dengan View dan Controller?

2. Bagaimana Eloquent ORM di Laravel mempermudah interaksi dengan basis data dibandingkan dengan metode query tradisional? Jelaskan dengan contoh bagaimana Eloquent ORM dapat digunakan untuk mengambil data dari tabel basis data.

3. Sebutkan dan jelaskan beberapa fitur utama dari Eloquent ORM yang tidak ditemukan pada metode query builder atau PDO biasa. Bagaimana fitur-fitur ini meningkatkan produktivitas web developer?

4. Analisis keuntungan dan kerugian menggunakan Eloquent ORM di Laravel. Apakah ada situasi tertentu di mana menggunakan Eloquent ORM tidak direkomendasikan? Jelaskan dengan argumen yang mendukung pendapat Anda.

Jawaban :

1. Menurut pola MVC (Model-View-Controller), pada bagian Model adalah tempat menyimpan logika yang berhubungan dengan data. Dari bagian tersebut terdapat peran penting sebagai berikut :

- Mewakili tabel dalam database. Contoh, model `User` mewakili tabel `users`.
- Mengatur interaksi dengan database seperti query, relasi, insert, update, delete, dan validasi logika bisnis.
- Menyediakan abstraksi data sehingga controller dan view tidak perlu tahu detail teknis database.

Bagaimana Model berinteraksi dengan View dan Controller?

Pertama, *Controller* bertugas menerima permintaan dari user (request), lalu menggunakan *Model* untuk mengambil data yang diperlukan. Lalu, setelah data siap, *Controller* mengirim data ke *View*. *View* hanya bertugas menampilkan data sehingga tidak langsung berhubungan dengan database dan model.

Contoh :

```
public function index() {
   $users = User::all(); // ambil semua data user lewat model
   return view('users.index', compact('users')); // kirim ke tampilan
}
```

2. Eloquent ORM merupakan Object-Relational Mapping (ORM) di Laravel yang memungkinkan kita berinteraksi dengan database menggunakan objek dan relasi antar objek, bukan perintah SQL secara langsung.

Contoh perbandingan:

Kalau pakai SQL biasa:

```
php
SalinEdit
$pdo = DB::connection()->getPdo();
$stmt = $pdo->prepare("SELECT * FROM users WHERE id = ?");
$stmt->execute([1]);
$user = $stmt->fetch();
```

Kalau pakai **Eloquent**:

```
php
SalinEdit
$user = User::find(1);
```

Lebih singkat, kan?

Contoh lain:

```
php
SalinEdit
$users = User::where('status', 'active')->orderBy('name')->get();
```

Tanpa perlu tulis query panjang-panjang, kita bisa langsung ambil data yang kita butuhkan. Lebih jelas, lebih aman (otomatis terhindar dari SQL Injection), dan lebih enak dibaca.

3. Berikut fitur andalan Eloquent yang tidak ada di Query Builder atau PDO:

a. Relasi Antar Tabel

Contoh, 1 user punya banyak post:

```
php
SalinEdit
// di model User.php
public function posts() {
    return $this->hasMany(Post::class);
}

// akses dari controller
$user = User::find(1);
$posts = $user->posts; // langsung dapet semua postingan user itu
```

b. Eager Loading

Menghindari query berulang-ulang (N+1 problem):

```
php
SalinEdit
$users = User::with('posts')->get(); // ambil semua user + postingannya
sekaligus
```

c. Mass Assignment

Langsung isi banyak kolom sekaligus:

```
php
SalinEdit
User::create([
    'name' => 'Ali',
    'email' => 'ali@example.com',
]);
```

d. Accessor & Mutator

Bisa ubah data sebelum ditampilkan atau sebelum disimpan:

```
php
SalinEdit
```

```
public function getNameAttribute($value) {
    return strtoupper($value); // Nama selalu ditampilkan dalam huruf besar
}
```

e. Observer dan Event

Misalnya ingin mencatat log setiap kali user baru dibuat:

```
php
SalinEdit
User::creating(function ($user) {
    Log::info('User baru dibuat: ' . $user->email);
});
```

Semua fitur ini nggak ada di query builder atau PDO dan bikin kerja jadi jauh lebih efisien.

4. Keuntungan Menggunakan Eloquent ORM :

A. Sintaks mudah dibaca, kode lebih sederhana daripada SQL biasa.
B. Kode menjadi rapi, dikelola oleh model.
C. CRUD bisa dilakukan dengan cepat.
D. Integrasi bagus seluruh bagian Laravel.

Kerugian :

A. Kurang optimal untuk query kompleks, focus pada kemudahan SQL.
B. Overheads pada memori dan objek, setiap baris data diambil lewat Eloquent dibungkus sebagai objek. Jika ambil ribuan baris, bisa memperlambat memori dengan cepat.
C. Kurang fleksibel untuk akses sata khusus. Untuk beberapa kebutuhan seperti query analitik (aggregate berat, CTE, window function), Eloquent masih terbatas.
D. Magic Behind the Scene bisa menyesatkan. Misal eager loading yang lupa ditulis bisa memicu N+1 query problem yang sangat memperlambat sistem.

Kapan Eloquent Tidak Disarankan?

1. **Sistem Real-time, High Throughput (misalnya: Analytics Engine, Event Processor)** Sistem ini sering membutuhkan query khusus yang dioptimalkan sampai ke indeks dan raw SQL.

2. **Migrasi atau Integrasi dengan Sistem Legacy**
   Jika sistem lama punya struktur tabel tidak konvensional atau banyak anomali data, Eloquent bisa sulit diterapkan karena analogi dengan struktur rapi.
3. **Aplikasi dengan Fokus pada Performa Ekstrem**
   Misalnya: e-commerce besar dengan ratusan ribu transaksi per jam. Developer akan cenderung memakai raw SQL dan database tuning langsung.