

1. Lý thuyết Cốt lõi (Khái niệm)

- **ORM là gì?** (Object-Relational Mapping). Đây là một kỹ thuật "ánh xạ" (mapping) một bảng (table) trong CSDL (ví dụ: bảng users) thành một "Đối tượng" (Object) trong code (ví dụ: class User của PHP).
- **Eloquent (8.1):** Là ORM của Laravel.
 - **Model:** Mỗi bảng trong CSDL của bạn sẽ có một file Model tương ứng trong thư mục app/Models/. Ví dụ: Bảng sinhvien \rightarrow Model app/Models/SinhVien.php.
 - **Quy ước (Convention):** Eloquent rất thông minh. Nếu bạn đặt tên Model là SinhVien (số ít, viết hoa chữ cái đầu), nó sẽ tự động hiểu rằng Model này làm việc với bảng sinhvien (số nhiều, viết thường).
- **Migrations (8.2):** Là "hệ thống quản lý phiên bản" cho CSDL của bạn. Thay vì tạo bảng bằng tay trong phpMyAdmin, bạn sẽ viết code PHP (trong thư mục database/migrations/) để định nghĩa cấu trúc bảng.
 - Lệnh tạo: php artisan make:model TenModel -m (cờ -m sẽ tạo luôn file migration).
 - Lệnh chạy: php artisan migrate (để tạo bảng trong CSDL).
- **Truy vấn (8.3, 8.4):** Đây là điều tuyệt vời nhất.
 - Thay vì PDO (Chương 4):

```
$sql = "SELECT * FROM sinhvien";
```

- Dùng Eloquent (Chương 8):

```
$all_students = SinhVien::all();
```

- Thay vì INSERT (Chương 4):

```
$sql = "INSERT INTO sinhvien (ten, email) VALUES (?, ?)";
```

- Dùng Eloquent (Chương 8):

```
SinhVien::create(['ten_sinh_vien' => $ten, 'email' => $email]);
```

2. Nhiệm vụ Thực hành (BẮT BUỘC)

Kịch bản: Chúng ta sẽ "nâng cấp" PHT Quản lý sinh viên (từ Chương 4 C 5) lên chuẩn

A. Thiết lập Database & Migration (8.2)

1. **// TODO 1:** Mở file .env ở thư mục gốc dự án (cse485_chapter6).
2. **// TODO 2:** Sửa các thông số CSDL. Tạo một CSDL mới trong phpMyAdmin tên là cse485_laravel (hoặc tên khác tùy bạn).

Đoạn mã

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=cse485_laravel # Tên CSDL bạn vừa tạo
DB_USERNAME=root          # User XAMPP
DB_PASSWORD=              # Pass XAMPP (rỗng)
```

3. **// TODO 3:** Mở Terminal, cd vào dự án, chạy lệnh Artisan sau để tạo Model SinhVien VÀ file migration (-m):

Bash

php artisan make:model SinhVien -m

4. **// TODO 4:** Mở file migration vừa được tạo trong:

database/migrations/..._create_sinh_viens_table.php (tên file có thêm timestamp).

5. **// TODO 5:** Sửa hàm up() để định nghĩa cấu trúc bảng (thêm 2 cột ten_sinh_vien và email):

PHP

```
public function up(): void
{
    Schema::create('sinh_viens', function (Blueprint $table) {
        $table->id(); // Tự động tạo cột 'id' (bigint, auto-increment, primary
key)

        // TODO 5: Thêm 2 dòng này
        $table->string('ten_sinh_vien', 255);
        $table->string('email', 255)->unique(); // unique() là ràng buộc duy
nhất

        $table->timestamps(); // Tự động tạo 2 cột 'created_at' và 'updated_at'
    });
}
```

6. **// TODO 6:** Chạy lệnh migrate để tạo bảng trong CSDL:

Bash

php artisan migrate

(Vào phpMyAdmin kiểm tra CSDL cse485_laravel, bạn sẽ thấy bảng sinh_viens xuất hiện).

B. Cập nhật Model G Controller (8.3, 8.4)

1. **// TODO 7:** Mở file Model app/Models/SinhVien.php.
2. **// TODO 8:** Để cho phép ::create() (Mass Assignment), chúng ta phải khai báo use HasFactory; và các trường được phép điền vào mảng \$fillable:

PHP

```
<?php
namespace App\Models;

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

class SinhVien extends Model
{
    use HasFactory;

    // TODO 8: Thêm mảng $fillable
    protected $fillable = [
        'ten_sinh_vien',
        'email',
    ];
}
```

3. **// TODO 9:** (Tùy chọn) Tạo 1 Controller mới cho Sinh Viên:

Bash

php artisan make:controller SinhVienController

4. **// TODO 10:** Mở app/Http/Controllers/SinhVienController.php và **use** Model SinhVien ở đầu file. Sau đó tạo 2 phương thức index() (để hiển thị) và store() (để lưu):

PHP

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;

// TODO 10: Import Model SinhVien
use App\Models\SinhVien;
```

```

class SinhVienController extends Controller
{
    // Phffơng thức index() (SELECT)
    public function index()
    {
        // TODO 11: Dùng Eloquent ::all() để lấy toàn bộ sinh viên
        // Gợi ý: $danhsachSV = SinhVien::all();

        // TODO 12: Trả về 1 view 'sinhvien.list' và truyền $danhsachSV
        // Gợi ý: return view('...', compact('...'));
    }

    // Phffơng thức store() (INSERT)
    public function store(Request $request)
    {
        // TODO 13: Lấy toàn bộ dữ liệu từ form
        // Gợi ý: $data = $request->all();

        // TODO 14: Dùng Eloquent ::create() để lffu vào CSDL
        // (Lffu ý: tên input trong form phải khớp với $fillable và tên cột)
        // Gợi ý: SinhVien::create($data);

        // TODO 15: Chuyển hffớng về trang danh sách
        // Gợi ý: return redirect()->route('sinhvien.index');
    }
}

```

5. **// TODO 16:** Mở routes/web.php, import SinhVienController và tạo 2 route (1 GET, 1 POST):

PHP

```
// Gợi ý:
// use App\Http\Controllers\SinhVienController;
// Route::get('/sinhvien', [SinhVienController::class, 'index'])->name('sinhvien.index');
// Route::post('/sinhvien', [SinhVienController::class, 'store'])->name('sinhvien.store');
```


6. **// TODO 17:** Tạo file View resources/views/sinhvien/list.blade.php (tạo thư mục sinhvien nếu chưa có). Copy/paste code từ PHT Chương 7 (layout, form, table) và sửa lại:
- Form: action="{{ route('sinhvien.store') }}" và method="POST". Thêm @csrf (Bắt buộc cho form POST của Laravel - sẽ học ở Chương 9). Input name phải là ten_sinh_vien và email.
 - Table: Dùng @foreach(\$danhSachSV as \$sv) và {{ \$sv->ten_sinh_vien }}.

3. Yêu cầu Bằng chứng (Proof of Work)

Bạn phải nộp lại 3 bằng chứng sau:

A. Code đã hoàn thiện:

1. Dán (paste) code hàm up() trong file Migration (..._create_sinh_viens_table.php).

```
public function up(): void
{
    Schema::create('sinh_viens', function (Blueprint
$table) {
        $table->id();
        $table->string('ten_sinh_vien', 255);
        $table->string('email', 255)->unique();
        $table->timestamps();
    });
}
```

2. Dán (paste) toàn bộ code file app/Models/SinhVien.php.

```
<?php

namespace App\Models;
```

```

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class SinhVien extends Model
{
    use HasFactory;

    protected $fillable = [
        'ten_sinh_vien',
        'email',
    ];
}

```

3. Dán (paste) toàn bộ code file app/Http/Controllers/SinhVienController.php.

```

<?php

namespace App\Http\Controllers;

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

class SinhVienController extends Controller
{
    // Phương thức index() (SELECT)

```

```

    public function index()
    {
        $danhsachSV = SinhVien::all();

        return view('sinhvien.list',
compact('danhsachSV'));
    }

    // Phương thức store() (INSERT)
    public function store(Request $request)
    {
        $data = $request->all();

        SinhVien::create($data);

        return redirect()->route('sinhvien.index');
    }
}

```

4. Dán (paste) code 2 route trong routes/web.php.

```

<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\PageController;
use App\Http\Controllers\SinhVienController;

// Route cho Chapter 6-7

Route::get('/', [PageController::class, 'showHomepage']);

```

```
Route::get('/about', [PageController::class,
'showHomepage']);

// Route cho Chapter 8 - Quản lý sinh viên

Route::get('/sinhvien', [SinhVienController::class,
'index'])->name('sinhvien.index');

Route::post('/sinhvien', [SinhVienController::class,
'store'])->name('sinhvien.store');
```

B. Ảnh chụp màn hình Kết quả (3 ẢNH):

1. **Ảnh 1 (Terminal):** Chụp màn hình Terminal sau khi chạy php artisan make:model SinhVien -m VÀ php artisan migrate thành công.

```
PS D:\xampp82\htdocs\cse485> cd cse485_chapter6
PS D:\xampp82\htdocs\cse485\cse485_chapter6> php artisan make:model SinhVien -m

INFO Model [D:\xampp82\htdocs\cse485\cse485_chapter6\app\Models\SinhVien.php] created successfully.

INFO Migration [D:\xampp82\htdocs\cse485\cse485_chapter6\database\migrations\2025_12_26_105724_create_sinh_viens_table.php] created successfully.

PS D:\xampp82\htdocs\cse485\cse485_chapter6> php artisan migrate

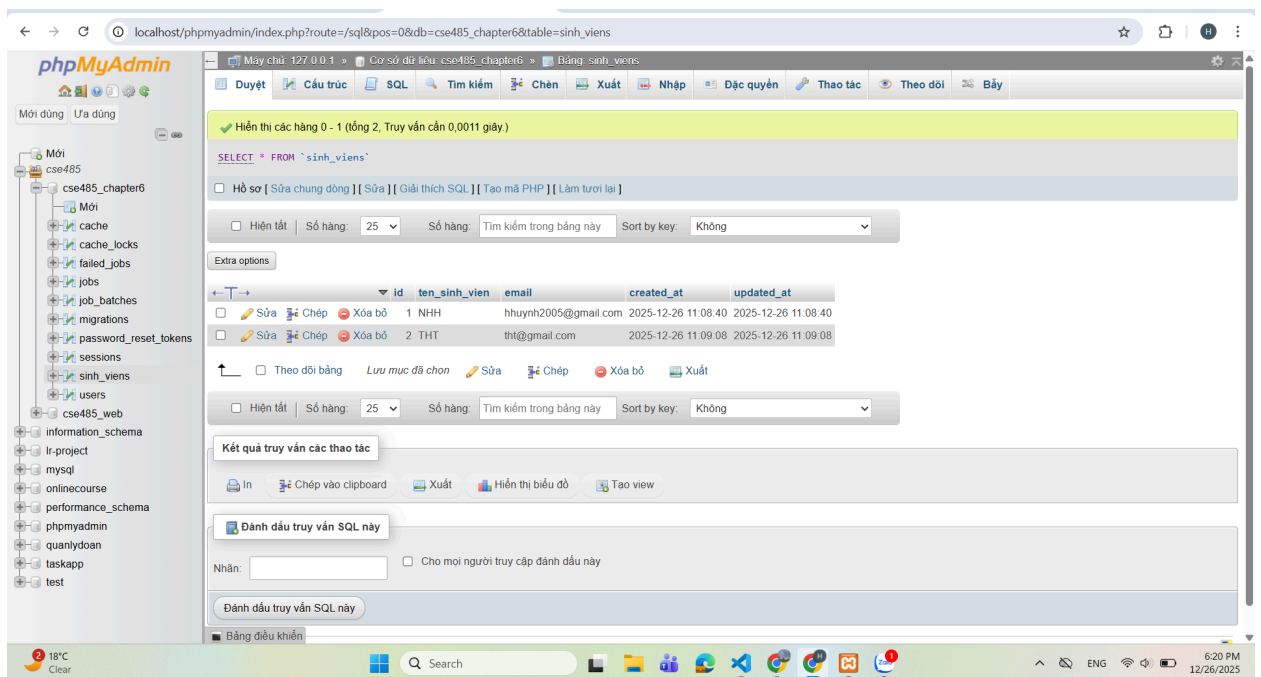
INFO Preparing database.

Creating migration table ..... 343.88ms DONE

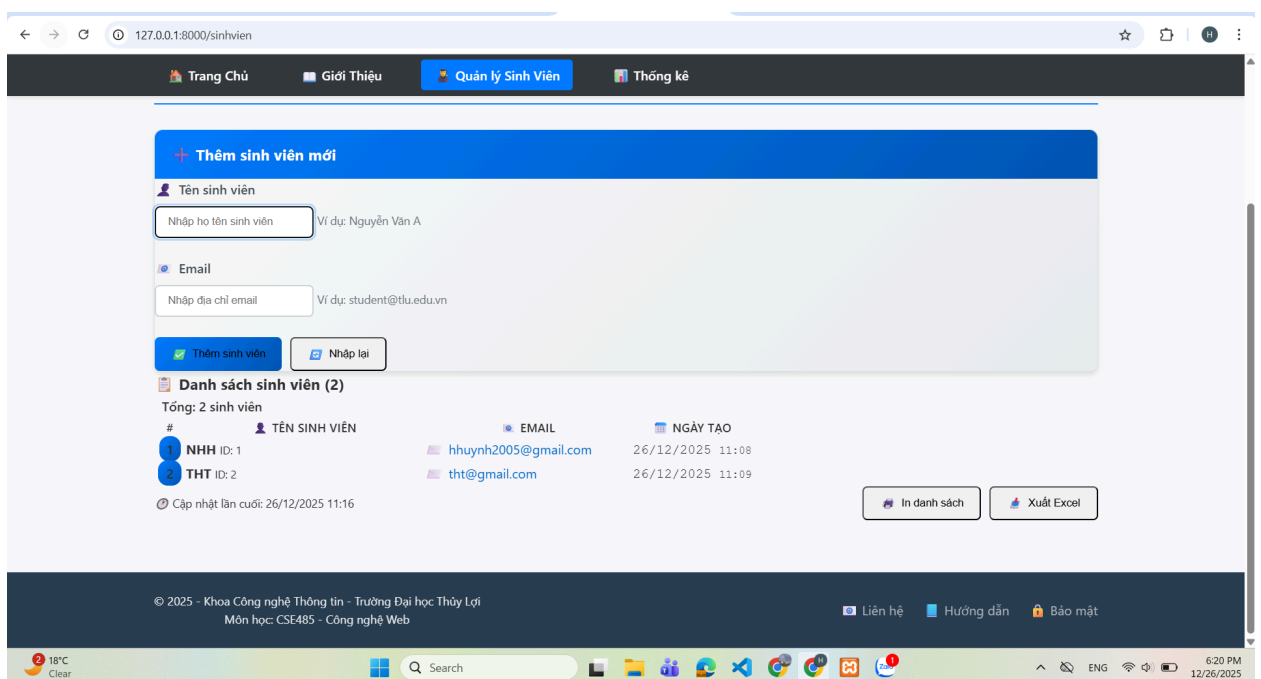
INFO Running migrations.

0001_01_01_000000_create_users_table ..... 405.29ms DONE
0001_01_01_000001_create_cache_table ..... 36.36ms DONE
0001_01_01_000002_create_jobs_table ..... 153.95ms DONE
2025_12_26_105724_create_sinh_viens_table ..... 45.56ms DONE
```

2. **Ảnh 2 (phpMyAdmin):** Chụp màn hình tab "Structure" (Cấu trúc) của bảng sinh_viens trong CSDL, cho thấy rõ các cột id, ten_sinh_vien, email, created_at.



3. **Ảnh 3 (Trình duyệt Web):** Chụp ảnh màn hình trang /sinhvien, sau khi đã dùng form thêm 1-2 sinh viên (chứng minh ::create() và ::all() đều hoạt động).



(Dán Code A1, A2, A3, A4 và 3 Ảnh B1, B2, B3 của bạn vào đây)

4. Câu hỏi Phản biện (Bắt buộc)

Sau khi hoàn thành Phần 2 C 3, hãy đặt 01 câu hỏi tư duy.

(Gợi ý: "Hãy so sánh PHT này (Eloquent) với PHT Chương 4 (PDO). Lợi ích lớn nhất của Eloquent là gì? Khi nào (trường hợp nào) thì dùng Eloquent sẽ tiện lợi hơn, và khi nào thì dùng PDO/SQL thuần có thể sẽ tốt hơn?").

Câu hỏi của tôi là: Khi dùng `SinhVien::all()` trong một hệ thống có 50,000 sinh viên, Eloquent sẽ load toàn bộ dữ liệu vào memory. So với PDO có thể dùng `fetch()` để lấy từng dòng, Eloquent có giải pháp gì cho việc xử lý dữ liệu lớn? Liệu rằng khi load toàn bộ dữ liệu lớn như vậy thì hiệu suất sẽ như thế nào?

5. 🏆 Kết nối Đánh giá (Rất quan trọng)

Eloquent ORM (Chương 8) là "trái tim" xử lý dữ liệu của Laravel. Kỹ năng `make:model`, `migrate`, và sử dụng các phương thức `::all()`, `::find()`, `::create()` là **tối quan trọng**.

Đây là kiến thức **trọng tâm** của cả **Bài tập trên lớp (Phần Laravel)** (20%)² và **Bài tập lớn theo nhóm (50%)**³. Nếu không nắm vững Eloquent, bạn không thể xây dựng bất kỳ chức năng nào cho dự án của mình.