BÀI TẬP THỰC HÀNH SỐ 4

Học phần: CSE485 - Công nghệ Web

Yêu cầu: Tạo một blog cá nhân đơn giản bằng cách sử dụng Laravel Framework. Sử dụng Bootstrap và kiến trúc mẫu Blade để thiết kế giao diện.

Một số đoạn code minh họa bên dưới có thể cần sửa đổi phù hợp với phiên bản của Bootstrap và Laravel mới nhất.

HƯỚNG DẪN

BƯỚC 1: Thiết lập dự án Laravel (phiên bản mới nhất: 10)

- Cài đặt PHP và composer (nếu chưa có).
- Cài đặt công cụ cài đặt Laravel qua composer:

composer global require laravel/installer

- Tạo dự án Laravel mới có tên tùy ý, chẳng hạn **blog**:

  laravel new blog

- Điều hướng vào thư mục dự án:

  cd blog

BƯỚC 2: Tạo và cấu hình CSDL

- Tạo CSDL có tên tùy ý, chẳng hạn, *laravel_blog*.
- Mở tệp ***.env*** trong thư mục gốc của dự án Laravel và cập nhật thông tin chi tiết kết nối cơ sở dữ liệu.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel_blog
DB_USERNAME=your_mysql_username
DB_PASSWORD=your_mysql_password
```

BƯỚC 3: Tạo Post model và migration

Trong Laravel, phần "model" của mẫu MVC (dạng số ít, ví dụ **Post**) thường tương ứng với bảng cơ sở dữ liệu (dạng số nhiều, ví dụ **posts**).

- Tạo mô hình cho Post và tệp migration (di chuyển) cho cơ sở dữ liệu:

php artisan make:model Post -m

- Mở tệp migration cho **Post** được thấy tại thư mục **database/migrations/** và cập nhật nội dung của hàm `up` để định nghĩa bảng trong CSDL. Ví dụ bảng Post cho CSDL như sau:

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

- Chạy migration (để ra lệnh cho thực thi các tệp migration nhằm tạo Bảng cho CSDL):

php artisan migrate

BƯỚC 4: Sử dụng Faker và Seeder

Faker là một thư viện PHP tạo dữ liệu giả cho bạn (tương tự như **mockaroo** đã được giới thiệu). Thật hữu ích khi bạn cần điền dữ liệu vào cơ sở dữ liệu cho mục đích thử nghiệm. Bạn có thể sử dụng nó kết hợp với các trình tạo cơ sở dữ liệu của Laravel, là các lớp chứa một phương thức để chèn dữ liệu vào cơ sở dữ liệu.

- Cài đặt thư viện Faker:

  composer require fzaninotto/faker

- Tạo một đối tượng Seeder:

  php artisan make:seeder PostsTableSeeder ☞ Lệnh này sẽ tạo một tệp mới **PostTableSeeder.php** nằm trong thư mục: **database/seeds**

- Mở tệp **PostTableSeeder.php** và sửa nội dung của phương thức run như sau:

Bạn sẽ tạo 50 bài đăng giả trong bảng bài đăng của mình. Điều này đặc biệt hữu ích khi bạn cần kiểm tra phân trang hoặc khi bạn cần mô phỏng cách ứng dụng hoạt động với nhiều dữ liệu hơn.

```
public function run()
{
    $faker = Faker\Factory::create();

    for ($i = 0; $i < 50; $i++) { // Creating 50 posts just for
example.
        App\Post::create([
            'title' => $faker->sentence(6, true), // Generates a random
sentence with 6 words.
            'body' => $faker->paragraphs(3, true), // Generates 3
random paragraphs.
        ]);
    }
}
```

- Đăng ký Seeder: Mở lớp **DatabaseSeeder** trong thư mục **database/seeds**, sửa nội dung của phương thức up:

```
public function run()
{
    $this->call(PostsTableSeeder::class);
}
```

- Chạy Seeder

php artisan db:seed

Laravel sẽ thực thi tất cả các seeder đã đăng ký. Nếu bạn muốn thực thi một seeder cụ thể, bạn có thể chỉ định nó bằng tùy chọn –class:

php artisan db:seed --class=PostsTableSeeder

Lưu ý: nếu bạn cần đặt lại cơ sở dữ liệu của mình và chạy lại tất cả các lần di chuyển và seeder, bạn có thể sử dụng lệnh migration:fresh theo sau là db:seed:

php artisan migrate:fresh

php artisan db:seed

BƯỚC 5: Tạo Post Controller

- Thực thi lệnh tạo PostController:

php artisan make:controller PostController --resource

Cờ --resource sẽ tạo bộ điều khiển với các phương thức xử lý tất cả các hoạt động CRUD cơ bản (Tạo, Đọc, Cập nhật, Xóa) dạng chuẩn.

BƯỚC 6: Định nghĩa Routes

- Mở tệp **web.php** trong thư mục **routes** và thêm vào các route cho Post:

Route::resource('posts', 'PostController');

Điều này sẽ tự động tạo các route cho hoạt động CRUD với PostController..

BƯỚC 7: Tạo các views với Blade và Bootstrap

Đối với blog đơn giản này, hãy tạo 3 views:

- **layout.app** để chứa cấu trúc tái sử dụng

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Your Blog</title>

    <!-- Fonts -->
    <link  href="https://fonts.googleapis.com/css?family=Nunito:200,600"
rel="stylesheet">

    <!-- Styles -->
    <link  href="{{  asset('css/app.css')  }}"  rel="stylesheet">  <!--
Bootstrap CSS -->

</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand" href="#">Blog</a>
        <div class="collapse navbar-collapse">
```

```
            <ul class="navbar-nav mr-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/posts">Posts</a>
                </li>
            </ul>
        </div>
    </nav>

    <main class="py-4">
        @yield('content')
    </main>

    <!-- Scripts -->
    <script src="{{ asset('js/app.js') }}"></script> <!-- Bootstrap JS -
->
</body>
</html>
```

- **index** để hiển thị tất cả các bài đăng

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

@section('content')
<div class="container">
    @foreach ($posts as $post)
    <div class="card mb-4">
        <div class="card-header">
            {{ $post->title }}
        </div>
        <div class="card-body">
            <p class="card-text">{{ $post->body }}</p>
        </div>
    </div>
    @endforeach
    {{ $posts->links() }}
</div>
@endsection
```

- **show** để hiển thị một bài đăng

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

@section('content')
<div class="container">
    <div class="card">
        <div class="card-header">
            {{ $post->title }}
        </div>
        <div class="card-body">
            <p class="card-text">{{ $post->body }}</p>
        </div>
    </div>
</div>
@endsection
```

- **create** để tạo một bài đăng mới.

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

@section('content')
<div class="container">
    <form method="POST" action="/posts">
        @csrf
        <div class="form-group">
            <label for="title">Title</label>
            <input    type="text"    class="form-control"    id="title"
name="title">
        </div>
        <div class="form-group">
            <label for="body">Body</label>
            <textarea    class="form-control"    id="body"    name="body"
rows="3"></textarea>
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
    </form>
</div>
```

```
@endsection
```

Tất cả các view này sẽ là các tệp **.blade.php**, sử dụng công cụ tạo khuôn mẫu Blade của Laravel

- Trong thư mục ***resources/views***, tạo một thư mục mới có tên ***posts***.
- Bên trong thư mục ***posts***, tạo ba tệp: index.blade.php, show.blade.php và create.blade.php
- Trong các tệp này, bạn có thể sử dụng cú pháp Blade và Bootstrap để tạo giao diện blog của mình

BƯỚC 8: Cập nhật các phương thức của PostController

Mở tệp **PostController.php** và cập nhật các phương thức xử lý dữ liệu và trả về view:

- **index()** để truy xuất và hiển thị tất cả các bài đăng.

```
public function index()
{
    // Retrieve all the posts
    $posts = Post::orderBy('created_at', 'desc')->paginate(10);

    // Pass the posts data to the view
    return view('posts.index', ['posts' => $posts]);
}
```

- **create()** để hiển thị biểu mẫu tạo bài đăng.

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

- **store(Request $request)** để xác thực và lưu dữ liệu bài đăng mới.

```
public function store(Request $request)
```

```
{
    // Validate the form data
    $validated = $request->validate([
        'title' => 'required|max:255',
        'body' => 'required',
    ]);

    // Create a new post instance and save it to the database
    $post = new Post;
    $post->title = $request->title;
    $post->body = $request->body;
    $post->save();

    // Redirect the user to the index view
    return redirect('/posts');
}
```

- **show($id)** để truy xuất và hiển thị một bài đăng

```
public function show($id)
{
    // Retrieve the post by its id
    $post = Post::findOrFail($id);

    // Pass the post data to the view
    return view('posts.show', ['post' => $post]);
}
```

BƯỚC 9: Chạy ứng dụng và kiểm tra kết quả

php artisan serve

Truy cập địa chỉ: http://localhost:8000/posts trên trình duyệt và thử kiểm tra các hoạt động CRUD

HẾT