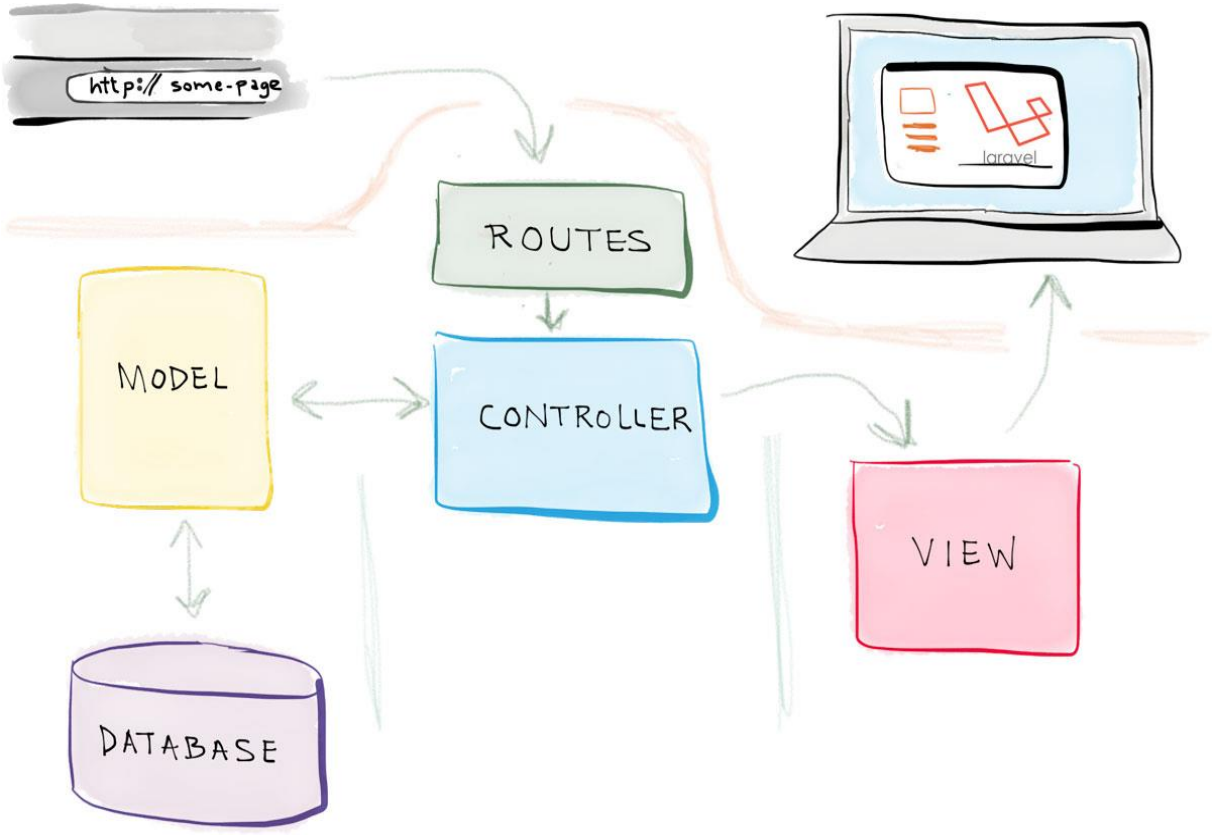
Mô hình MVC

1. Giới thiệu

Các ứng dụng Laravel theo mô hình **Model-View-Controller** truyền thống gồm có:

- **Các controller** xử lý các request của người dùng và truy xuất dữ liệu, bằng cách tận dụng các Model.
- **Các model** tương tác với database và truy xuất các thông tin từ đối tượng.
- **Các view** để hiển thị các trang.

Ngoài ra, **các route** được sử dụng để ánh xạ các URL tới các hành động được chỉ định trong controller, như hình dưới đây:



Vòng đời request trong ứng dụng Laravel

- Một request được tạo – khi người dùng nhập một URL liên kết với ứng dụng.
- Một **route** đã được liên kết với URL ánh xạ URL tới một hành động trong controller.
- **Controller** sử dụng các **model** cần thiết để trích xuất thông tin từ database, và sau đó truyền dữ liệu tới view.
- Và **view** hiển trị trang.

2. Xây dựng ứng dụng mẫu

Xây dựng một ứng dụng Laravel với tất cả các thành phần MVC (Model, View và Controller). Ví dụ, xây dựng một ứng dụng quản lý cars (xe hơi). Tạo một ứng dụng laravel với tên MVC.

Model

Đầu tiên, tạo một model, đại diện cho Car. Laravel có một giao diện dòng lệnh **Artisan CLI**, cung cấp một loạt các lệnh hữu ích để xây dựng ứng dụng.

Hãy mở command line (và kết nối tới máy ảo Homestead, nếu sử dụng Laravel Homestead), di chuyển tới thư mục của ứng dụng, và chạy lệnh sau để tạo một Car model mới:

```
php artisan make:model Car --migration
```

Tất cả các model được lưu trữ trong thư mục `app/Models`, vì vậy lệnh trên sẽ tạo ra một file tin model `app/Models/Car.php` với nội dung như sau:

```
<?php

namespace App\Models;

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

class Car extends Model
{
    use HasFactory;
}
```

Vì chức năng model có sẵn của Laravel, chỉ tạo một class model rỗng, Laravel sẽ giả sử rằng model này đã liên kết với một bảng `cars` trong cơ sở dữ liệu.

Và bằng việc cung cấp tùy chọn `–migration` khi tạo model, Laravel cũng tạo một tập tin **database migration** để tạo bảng `cars`. Tập tin migration được đặt tại `database/migrations/[timestamp]_create_cars_table.php` và có nội dung như sau:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
class CreateCarsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('cars', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('cars');
    }
}
```

Hoàn thành tập tin migration. Thêm vài cột, ví dụ là: make, model, và production date:

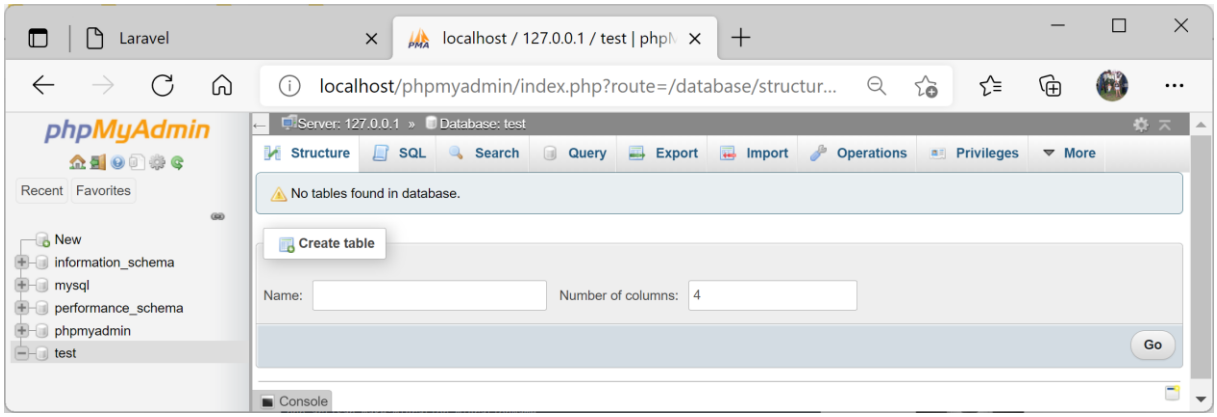
```
    public function up()
    {
        Schema::create('cars', function (Blueprint $table) {
            $table->id();
            $table->string('make');
            $table->string('model');
            $table->date('produced_on');
            $table->timestamps();
        });
    }
```

Và sau đó bạn có thể chạy migration để tạo bảng `cars` sử dụng lệnh sau:

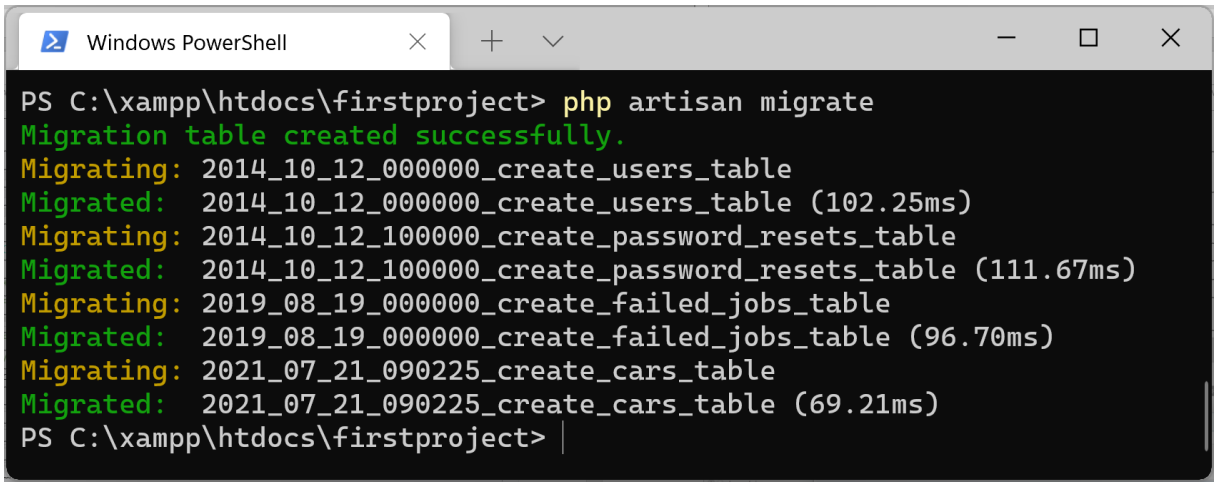
Lưu ý: Start MySQL trong XAMPP, các thông số cấu hình trong file .env. Ví dụ:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
```

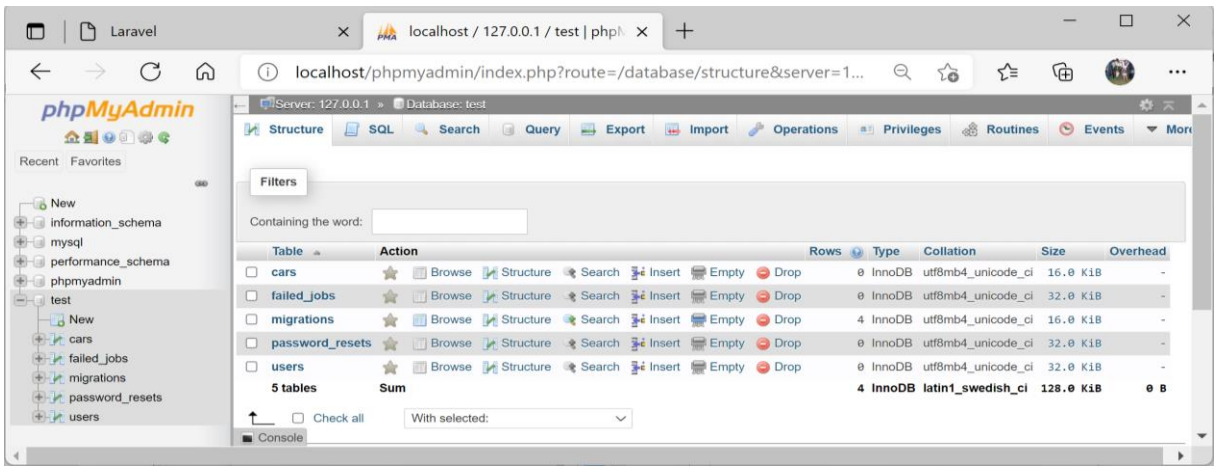
```
DB_PORT=3306
DB_DATABASE=test
DB_USERNAME=root
DB_PASSWORD=
```



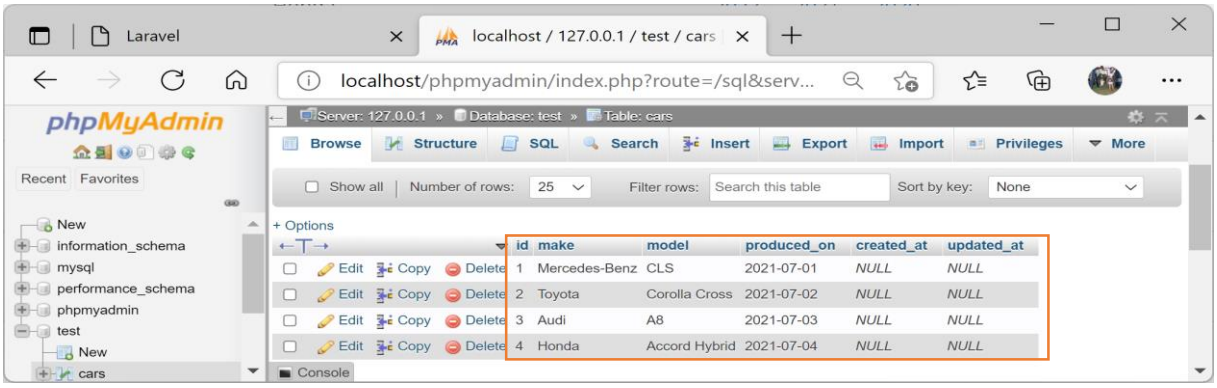

```
php artisan migrate
```



Kết quả:



Nhập dữ liệu mẫu:



Controller

Trong Laravel, một kiểu đối tượng (ở đây là Car), được gọi là một **resource**.

Có thể tạo một **resource controller** – một controller xử lý tất cả các request liên quan tới một resouce, sử dụng lệnh:

```
php artisan make:controller CarController --resource
```

Lệnh này sẽ tạo ra controller `app/Http/Controllers/CarController.php` với nội dung:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Car;

class CarController extends Controller
{
    // Display a listing of the resource.
    public function index()
    {
        //
    }
    // Show the form for creating a new resource.
    public function create()
    {
        //
    }
    // Store a newly created resource in storage.
    public function store(Request $request)
    {
        //
    }
```

```
    // Display the specified resource.
    public function show($id)
    {
        $car = Car::find($id);
        return view('cars.show', array('car' => $car));
    }

    // Show the form for editing the specified resource.
    public function edit($id)
    {
        //
    }
    // Update the specified resource in storage.
    public function update(Request $request, $id)
    {
        //
    }
    // Remove the specified resource from storage.
    public function destroy($id)
    {
        //
    }
}
```

Lưu ý: Với tham số "--resource" nó tự động tạo một controller với tất cả các hành động CRUD (Create, Retrieve/Read, Update, Delete) thường gặp.

Bây giờ định nghĩa các route để liên kết các URL với tất cả các hành động trong controller này.

Route

Một cách phổ biến là bạn có thể định nghĩa một **resouce route** duy nhất, nó sẽ tạo ra các route cho tất cả các hành động trong resouce controller.

Trong tập tin cấu hình các route "`routes/web.php`", định nghĩa Car resource route như sau:

```
<?php

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

Route::resource('cars', CarController::class);
```

Định nghĩa route ở trên sẽ xác định tất cả các route liên quan tới Car resource:

Request Type	Path	Action	Route Name
GET	`/cars`	`index`	cars.index
GET	`/cars/create`	`create`	cars.create
POST	`/cars`	`store`	cars.store
GET	`/cars/{car}`	`show`	cars.show
GET	`/cars/{car}/edit`	`edit`	cars.edit
PUT/PATCH	`/cars/{car}`	`update`	cars.update
DELETE	`/cars/{car}`	`destroy`	cars.destroy

Phương thức show của controller

Như đã thấy trong bảng route ở phần trước, trang Show Car sẽ có URL tương ứng là http://localhost:8000/cars/{car}. Trong trường hợp này, {car} sẽ là `id` của một đối tượng car trong database.

Vì thế, URL để xem chiếc xe có `id` là `1` sẽ như thế này http://localhost:8000/cars/1

Để triển khai trang Show Car, trong hành động `show` của controller, chúng ta cần:

- Sử dụng `Car` model để truy xuất đối tượng Car được chỉ định từ database.
- Tải một view cho trang Show Car, và truyền tới nó đối tượng Car lấy từ database.

Đầu tiên, để truy cập tới Car model trong controller, thêm một câu lệnh `use` ở phía trên class controller:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Car;
```

Sau đó, hoàn thành hành động `show` với đoạn code sau:

```
    public function show($id)
    {
        $car = Car::find($id);
        return view('cars.show', array('car' => $car));
```

```
    }
```

Khi nhập URL http://localhost:8000/cars/1, Laravel sẽ lấy 1 trong URL thông qua biến $id trong hàm show, như ở trên. Và truy xuất đối tượng car sử dụng model Car bằng cách gọi hàm Car::find với tham số $id. View sẽ được tải sau đó sử dụng hàm view với tham số là tên của view (sẽ tạo ở phần tiếp theo) và một mảng dữ liệu được cung cấp cho view.

View

Các tập tin view của Laravel được lưu trữ trong thư mục resoureces/views. Và có thể được tổ chức vào các thư mục con trong thư mục này.

Trước đó chúng ta đã truyền vào hàm view một view với tên cars.show. Điều đó nói với Laravel tìm một tập tin view có tên là show.blade.php được lưu trữ trong một thư mục là resources/views.

Các tập tin view của Laravel sử dụng Blade templating engine, và vì thế nó có phần mở rộng là .blade.php

Để hoàn thành trang Show Car, cần tạo view resources/views/cars/show.blade.php với nội dung như sau:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Car {{ $car->id }}</title>
    </head>
    <body>
        <h1>Car {{ $car->id }}</h1>
        <ul>
            <li>Hãng xe: {{ $car->make }}</li>
            <li>Mẫu xe: {{ $car->model }}</li>
            <li>Năm sản xuất: {{ $car->produced_on }}</li>
        </ul>
    </body>
</html>
```

Vì đã truyền đối tượng Car cho view, trong phương thức show của controller, với mảng có key là car, nên có thể truy cập nó trong view thông qua một biến cùng tên $car.

Các đối tượng được truy xuất thông qua một model là các instance của class model đó. Và các giá trị của đối tượng Car có thể truy cập sử dụng các tên giống như tên các cột trong bảng

`cars`. Cuối cùng sử dụng cú pháp của Blade để hiển thị thông tin.

Ví dụ để hiển thị giá trị make của car:

```
{{ $car->make }}
```

Kết quả: