ScopeTreasure Courses

LARAVEL COURSES · LESSON LV.2 · CODE SCOPE SERIES

Laravel Routing + Controllers

How a request travels: routes/web.php, controller classes, route parameters and named routes — the traffic system of every Laravel app.

Introduction

A visitor types an address — `school-app.test/students` — and somehow the right page appears. Who decides which code answers which address? In Laravel that decision lives in **routes**: one line per address, written in `routes/web.php`. And the code that answers is usually not squeezed into the route line itself — it lives in a **controller**, a tidy class with one method per job.

Think of routes as the **address book** of your app and controllers as the **office staff**: the address book says "marks questions go to the Marks Officer", and the officer does the actual work. Keep this split and your app stays readable when it grows from 5 addresses to 500.

(Roman Urdu: Route pata batata hai, controller kaam karta hai — pata aur kaam alag rakhna hi saaf app ki pehchan hai.)

Key points

- **All web routes live in `routes/web.php`:** every address your site answers is a line there. Nothing is reachable unless a route points to it.
- **A route names a verb + address + answer:** `Route::get('/about', ...)` answers GET requests (normal page visits). Later lessons add `Route::post` for form submissions.
- **Controllers are classes with job-methods:** `php artisan make:controller StudentController` creates `app/Http/Controllers/StudentController.php`. Methods like `index()` (list), `show()` (one item) each handle one job.
- **The modern pointer syntax:** `Route::get('/students', [StudentController::class, 'index']);` — read it as "when /students is visited, run the index method of StudentController".
- **Route parameters capture pieces of the address:** `/students/{id}` catches the number in `/students/5` and hands it to your method as `$id`. A parameter is a hole in the address with a name.
  (Roman Urdu: {id} address ke andar aik khali jagah hai — jo number wahan aaye ga, wohi aap ke method ko mile ga.)
- **Named routes end address-typo bugs:** `->name('students.show')` gives the route a permanent nickname. Links are then built with `route('students.show', 5)` — if the address ever changes, every link updates itself.
- **`php artisan route:list` is your X-ray:** it prints every route the app knows — verb, address, name, and which controller method answers. When a page 404s, this list is the first place to look.

Code examples

Example 1 — Your first routes: two pages, no controller yet

```
// routes/web.php
Route::get('/', function () {
    return view('welcome');
});

Route::get('/about', function () {
    return view('about');
});
```

What it does: Tiny routes can answer with a small function (a *closure*) that returns a view. Visit `/` and the welcome view opens; visit `/about` and the about view opens. This is perfect for two or three simple pages — and the wrong tool for a whole system, which is why controllers exist.

What you will see: `php artisan route:list` shows both addresses registered:

```
  GET|HEAD   / ..............
  GET|HEAD   about ............
```

Example 2 — A real controller: Artisan builds it, the route points to it

```
php artisan make:controller StudentController
```

```
// app/Http/Controllers/StudentController.php
namespace App\Http\Controllers;

class StudentController extends Controller
{
    public function index()
    {
        return view('students.index');
    }

    public function show($id)
    {
        return "Showing student number " . $id;
    }
}
```

```
// routes/web.php
use App\Http\Controllers\StudentController;

Route::get('/students', [StudentController::class, 'index']);
Route::get('/students/{id}', [StudentController::class, 'show']);
```

What it does: Artisan creates the controller file with the class ready. Each public method is one job. The routes stay one line each — the address book just points at the staff. The `{id}` parameter flows into `show($id)` automatically.

What you will see: `/students` opens the students page; `/students/5` prints `Showing student number 5`; `/students/42` prints `Showing student number 42` — one method, endless addresses.

Example 3 — Named routes: links that build themselves

```
// routes/web.php
Route::get('/students', [StudentController::class, 'index'])->name('students.index');
Route::get('/students/{id}', [StudentController::class, 'show'])->name('students.show');
```

```
{{-- inside any Blade view --}}
<a href="{{ route('students.index') }}">All students</a>
<a href="{{ route('students.show', 5) }}">Student 5</a>
```

What it does: Each route gets a nickname. Blade's `route()` helper looks the nickname up and prints the real address. Rename `/students` to `/learners` next month and you change ONE line in web.php — every link in every page quietly follows.

What you will see: the page source contains real addresses — `<a href="http://127.0.0.1:8000/students">` and `<a href="http://127.0.0.1:8000/students/5">` — generated by Laravel, never typed by you.

Quick recap

- Routes are the address book (`routes/web.php`); controllers are the staff (one method per job).
- Point at a controller with `[StudentController::class, 'index']` — never grow logic inside route lines.
- `{id}` parameters capture address pieces; named routes + `route()` make links typo-proof.
- `php artisan route:list` shows the whole traffic system at a glance.

Common mistakes

1. **Writing big logic inside route closures** — a 40-line function in web.php becomes unreadable and cannot be reused. Fix: the moment a route does real work, move it into a controller method.

2. **Forgetting the `use` line** — pointing at `StudentController::class` without importing it gives a "class not found" error. Fix: add `use App\Http\Controllers\StudentController;` at the top of web.php (your editor usually offers to add it).
3. **Typing addresses by hand in links** — `<a href="/studnets">` typos 404 silently. Fix: name every route and build every link with `route('name')` — a wrong nickname errors loudly at once.
4. **Route order traps** — `/students/{id}` written BEFORE `/students/create` swallows "create" as an id. Fix: put fixed addresses above parameter addresses (LV.6's resource routes handle this for you).

Practice

1. Add routes and a `PageController` with three methods — `home`, `about`, `contact` — each returning its own simple view, all with named routes. Verify with `php artisan route:list` and link the three pages to each other using only `route()`.
2. Make a `/teachers/{name}` route whose method prints the name from the address. Visit it with three different names, then explain in one line: why did you not need three routes?

Practice quiz

1. Where do all web routes live, and what happens to an address with no route?

- In the controller files, and it still works
- In routes/web.php, every address your site answers is a line there; nothing is reachable unless a route points to it
- In .env
- In the views folder

Answer: In routes/web.php, every address your site answers is a line there; nothing is reachable unless a route points to it

2. What does Route::get('/students/{id}', ...) hand to your method?

- Nothing, parameters are only for display
- The piece of the address caught by {id}, for example the number in /students/5, handed to your method as $id
- The whole URL as a string
- A database row automatically

Answer: The piece of the address caught by {id}, for example the number in /students/5, handed to your method as $id

3. Why name a route and build links with route('students.show', 5)?

- It is only a style habit
- Named routes end address-typo bugs: if the address ever changes, every link built with route() updates itself, while a hand-typed /studnets 404s silently

- Named routes are faster
- Hand-typed links are forbidden by PHP

Answer: Named routes end address-typo bugs: if the address ever changes, every link built with route() updates itself, while a hand-typed /studnets 404s silently

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/laravel-routing-controllers · More free tools and notes at https://scopetreasure.com