ScopeTreasure Courses

LARAVEL COURSES · LESSON LV.1 · CODE SCOPE SERIES

PHP + Laravel Kya Hai?

PHP in one honest glimpse, why Laravel is famous, the MVC map every lesson follows — plus setup: Composer, your first project and artisan serve.

Introduction

Every website you love has two halves. The half you see — pages, buttons, colours — is the **front-end** (your HTML, CSS and JavaScript series). The half you never see — checking passwords, saving students, sending email — is the **back-end**, and it runs on a server. This series builds the back-end with **PHP** and its most famous framework, **Laravel**.

PHP is a language made for the web: it runs on the server, talks to databases easily, and powers a huge share of the internet (WordPress itself is PHP). **Laravel** is a framework — a ready-made skeleton and toolbox — built on PHP. Instead of writing every lock, door and pipe yourself, Laravel hands you tested parts: routing, login, database tools, email, queues. That is why companies hire Laravel developers and why job boards are full of Laravel posts: one developer with Laravel can ship what used to take a whole team.

In this lesson you will taste plain PHP, learn the **MVC map** that every Laravel app follows, and set up the series project on your own machine.

(Roman Urdu: PHP server ki zubaan hai, aur Laravel us ka tayyar dahncha — aap sirf apni app ki kahani likhte hain, baaqi saamaan framework deta hai.)

Key points

- **PHP runs on the server:** the browser never sees your PHP code. It sends a request, PHP thinks and answers with plain HTML. Files end in `.php`, and PHP code lives between `<?php` and `?>`.
- **A framework is a skeleton + toolbox:** Laravel decides where files live and gives you ready systems (routes, views, database, login, email) so you build features, not plumbing.
- **Why Laravel wins jobs:** it is fast to build with, its code reads almost like English, its documentation is famous, and a giant community means every problem you hit, someone has already solved.
  (Roman Urdu: Laravel ki sab se bari taqat us ki community aur saaf code hai — ghalti ho to hal hazir milta hai.)
- **MVC is the map: Model** = your data and its rules (the database's representative). **View** = what the user sees (Blade pages). **Controller** = the manager in the middle — it takes the request, asks the Model, and hands data to the View.
- **Composer is the delivery service:** PHP's package manager. Laravel itself, and almost every add-on, arrives through Composer. One command, `composer`, must work on your machine before anything else.
- **Artisan is Laravel's command-line helper:** `php artisan serve` starts your app, `php artisan make:...` builds files for you, `php artisan migrate` prepares the database. You will type Artisan commands in every lesson.

- **Setup is a one-time ceremony:** install PHP + Composer once, create the project once — after that, every lesson starts with `php artisan serve` and your app is alive at `http://127.0.0.1:8000`.

Code examples

Example 1 — Plain PHP: the language under Laravel

```
<?php
$name = "Ali Raza";
$marks = 88;

echo "Student: " . $name . "<br>";
echo "Marks: " . $marks . "<br>";

if ($marks >= 40) {
    echo "Result: PASS";
}
```

What it does: This is raw PHP, no Laravel. Variables start with `$`, text joins with a dot `.`, and `echo` sends output to the browser. The `if` checks the marks and prints the result. Laravel is written in this same language — the framework just organises it so big apps stay clean.

What you will see: three lines in the browser:

```
Student: Ali Raza
Marks: 88
Result: PASS
```

Example 2 — Birth of the series project: laravel new + artisan serve

```
laravel new school-app
cd school-app
php artisan serve
```

What it does: The Laravel installer (a small Composer tool) creates a complete app — hundreds of files, perfectly organised — inside a new `school-app` folder. Then Artisan starts Laravel's own development server. The important folders on the tour: `routes/` (your URLs), `app/Models/` (your data), `resources/views/` (your pages) and `.env` (your secret settings — never shared).

What you will see: the terminal answers, and the app stays alive until you press Ctrl+C:

```
   INFO  Server running on [http://127.0.0.1:8000].

  Press Ctrl+C to stop the server
```

Open that address in the browser and Laravel's welcome page greets you — a real, running app, built in one minute.

Example 3 — The MVC map in three tiny files

```
// routes/web.php — the address book
Route::get('/welcome', [WelcomeController::class, 'show']);

// app/Http/Controllers/WelcomeController.php — the manager
public function show()
{
    $student = Student::find(1);            // Model: fetch the data
    return view('welcome', ['student' => $student]);  // View: show it
}
```

```
{{-- resources/views/welcome.blade.php — the face --}}
<h1>Welcome, {{ $student->name }}!</h1>
```

What it does: One request, three workers. The **route** hears the address `/welcome` and calls the **controller**. The controller asks the **Model** for student number 1 and hands it to the **view**. The Blade view prints the name inside the page. Every lesson in this series is this same dance, step by step.

What you will see: visiting `http://127.0.0.1:8000/welcome` shows a page whose heading reads `Welcome, Ali Raza!` — the name travelled database → Model → Controller → View → browser.

Quick recap

- PHP is the server-side language; Laravel is its famous framework — a skeleton plus a toolbox.
- **MVC** is the map: Model (data), View (pages), Controller (the manager between them).
- Composer delivers packages; Artisan is the command-line helper; `php artisan serve` makes the app alive.
- Setup happens once: PHP + Composer → `laravel new school-app` → serve.

Common mistakes

1. **Opening PHP files directly in the browser** — double-clicking a `.php` file shows code or downloads it, because no server ran it. Fix: always visit the app through `php artisan serve` and its `127.0.0.1:8000` address.
2. **Skipping Composer** — trying to download Laravel as a zip and copy files by hand ends in missing pieces. Fix: install Composer first, then let `laravel new` (or `composer create-project`) build the project properly.
3. **Editing the welcome page to "start"** — beginners hack the demo page and learn nothing about structure. Fix: leave the welcome page; from LV.2 you will add your own routes, controllers and views the

real way.

4. **Sharing the .env file** — it holds passwords and secret keys. Fix: `.env` stays on your machine only; it is already listed in `.gitignore` — keep it that way.

Practice

1. Install PHP and Composer on your machine, create the `school-app` project, run `php artisan serve` and open the welcome page. Write down the five folder names from the tour and one line each: what lives there?
2. Write the tiny plain-PHP file from Example 1 with your own name and marks, run it (a `test.php` inside `public/` works for a quick taste — delete it after), and change the pass mark to 50. What prints now?

Practice quiz

1. What is MVC, in one line per letter?

- Model = pages, View = data, Controller = database
- Model = your data and its rules, View = what the user sees (Blade pages), Controller = the manager in the middle that takes the request, asks the Model, and hands data to the View
- Model = routes, View = controllers, Controller = views
- MVC is a database only

Answer: Model = your data and its rules, View = what the user sees (Blade pages), Controller = the manager in the middle that takes the request, asks the Model, and hands data to the View

2. Why can the browser never see your PHP code?

- Because PHP files are hidden
- Because PHP runs on the server: the browser sends a request, PHP thinks, and answers with plain HTML
- Because browsers cannot read .php files at all
- Because Laravel encrypts every file

Answer: Because PHP runs on the server: the browser sends a request, PHP thinks, and answers with plain HTML

3. How do you start a Laravel app after the one-time setup?

- By double-clicking a .php file in the browser
- With php artisan serve, and the app is alive at http://127.0.0.1:8000
- With composer serve
- By opening the project folder

Answer: With php artisan serve, and the app is alive at http://127.0.0.1:8000

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-intro · More free tools and notes at https://scopetreasure.com