Introduction to Laravel

The PHP Framework for Web Artisans

Kazi Rifat Morshed

Student Intern
ICT Cell
Khulna University

July 7, 2026

Overview

Introduction to Laravel: Prerequisite Knowledge & Skills

- Basic understanding of **web development** (HTML, CSS, JavaScript)
- Familiarity with **object-oriented programming (OOP)**
- Knowledge of **database** concepts (MySQL/MariaDB)
- Basic understanding of **PHP**
- **Architecture patterns** (MVC) and **design principles**
- Some prior **struggling** experience with PHP and DBMS web projects

Introduction to Laravel: Prerequisite Knowledge & Skills

- Basic understanding of **web development** (HTML, CSS, JavaScript)
- Familiarity with **object-oriented programming (OOP)**
- Knowledge of **database** concepts (MySQL/MariaDB)
- Basic understanding of **PHP**
- **Architecture patterns** (MVC) and **design principles**
- Some prior **struggling** experience with PHP and DBMS web projects

Introduction to Laravel: Prerequisite Knowledge & Skills

- Basic understanding of **web development** (HTML, CSS, JavaScript)
- Familiarity with **object-oriented programming (OOP)**
- Knowledge of **database** concepts (MySQL/MariaDB)
- Basic understanding of **PHP**
- **Architecture patterns** (MVC) and **design principles**
- Some prior **struggling** experience with PHP and DBMS web projects

Introduction to Laravel: Prerequisite Knowledge & Skills

- Basic understanding of **web development** (HTML, CSS, JavaScript)
- Familiarity with **object-oriented programming (OOP)**
- Knowledge of **database** concepts (MySQL/MariaDB)
- Basic understanding of **PHP**
- **Architecture patterns** (MVC) and **design principles**
- Some prior **struggling** experience with PHP and DBMS web projects

Introduction to Laravel: Prerequisite Knowledge & Skills

- Basic understanding of **web development** (HTML, CSS, JavaScript)
- Familiarity with **object-oriented programming (OOP)**
- Knowledge of **database** concepts (MySQL/MariaDB)
- Basic understanding of **PHP**
- **Architecture patterns** (MVC) and **design principles**
- Some prior **struggling** experience with PHP and DBMS web projects

Introduction to Laravel: Prerequisite Knowledge & Skills

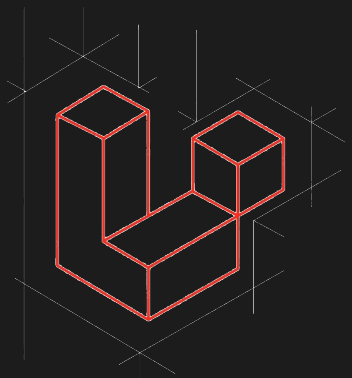
- Basic understanding of **web development** (HTML, CSS, JavaScript)
- Familiarity with **object-oriented programming (OOP)**
- Knowledge of **database** concepts (MySQL/MariaDB)
- Basic understanding of **PHP**
- **Architecture patterns** (MVC) and **design principles**
- Some prior **struggling** experience with PHP and DBMS web projects

Introduction to Laravel: PHP, dead?

The Future Programmer
@TheProgrammerMe

1995: PHP is dead, learn ColdFusion
2002: PHP is dead, learn ASP .net
2003: PHP is dead, learn Django
2004: PHP is dead, learn Ruby on Rails
2010: PHP is dead, learn Flask
2011: PHP is dead, learn AngularJS
2016: PHP is dead, learn Next.js
2022: okay this is awkward

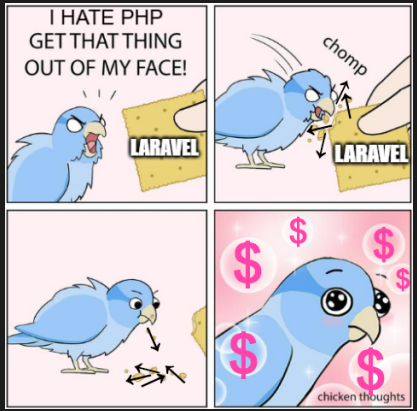
Introduction to Laravel: PHP, not dead yet



Laravel

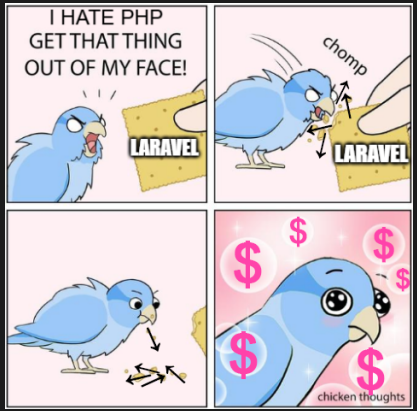
Introduction to Laravel: What is Laravel?

- A free, open-source PHP web framework
- Comes with a rich set of libraries, features and tools for web development
- Makes life easier for developers by providing elegant syntax and conventions
- Follows the MVC (Model-View-Controller) architectural pattern
- Designed for building modern web applications



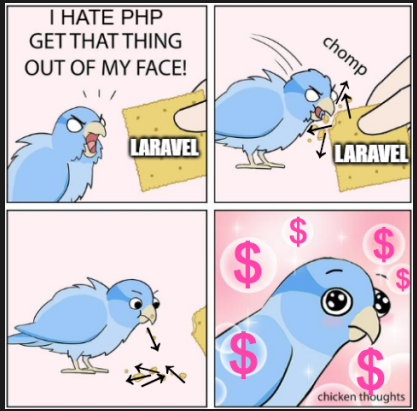
Introduction to Laravel: What is Laravel?

- A free, open-source PHP web framework
- Comes with a rich set of libraries, features and tools for web development
- Makes life easier for developers by providing elegant syntax and conventions
- Follows the MVC (Model-View-Controller) architectural pattern
- Designed for building modern web applications



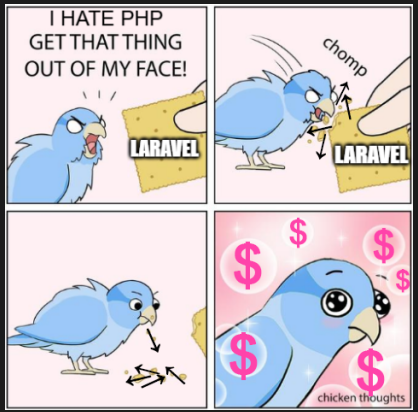
Introduction to Laravel: What is Laravel?

- A free, open-source PHP web framework
- Comes with a rich set of libraries, features and tools for web development
- Makes life easier for developers by providing elegant syntax and conventions
- Follows the MVC (Model-View-Controller) architectural pattern
- Designed for building modern web applications



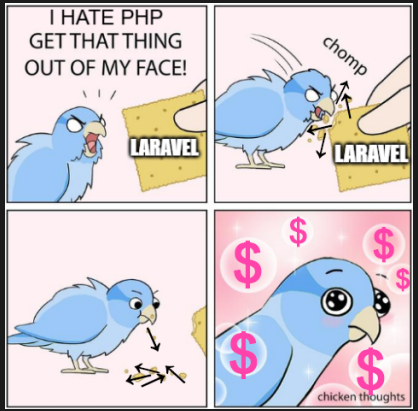
Introduction to Laravel: What is Laravel?

- A free, open-source PHP web framework
- Comes with a rich set of libraries, features and tools for web development
- Makes life easier for developers by providing elegant syntax and conventions
- Follows the MVC (Model-View-Controller) architectural pattern
- Designed for building modern web applications



Introduction to Laravel: What is Laravel?

- A free, open-source PHP web framework
- Comes with a rich set of libraries, features and tools for web development
- Makes life easier for developers by providing elegant syntax and conventions
- Follows the MVC (Model-View-Controller) architectural pattern
- Designed for building modern web applications



Introduction to Laravel: Laravel vs PHP

Aspect	Laravel	PHP
Type	Full-stack web framework	Server-side scripting language
Structure	Conventions for MVC, config, tests and queues	You design folder layout and patterns yourself
Routing	Declarative routes with middleware pipeline	`$_GET`/.htaccess or custom router
Database	Eloquent ORM, migrations, factories	PDO or raw SQL
CLI & tooling	Artisan commands, migrations, queues	Composer scripts or bash
Extensibility	90 000+ Laravel-specific packages plus generic PHP libs	Any PHP lib via Composer

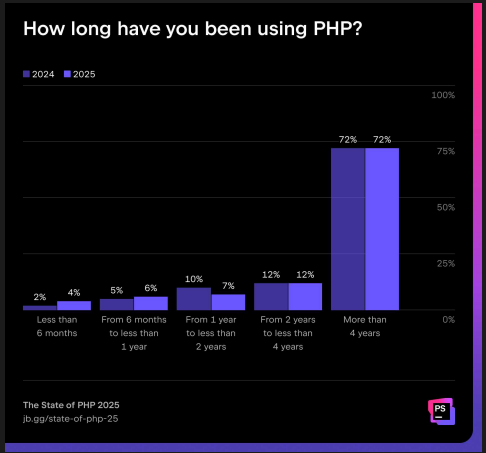
NOTE: Though, it is illogical to compare between a programming language and a framework built on same language.

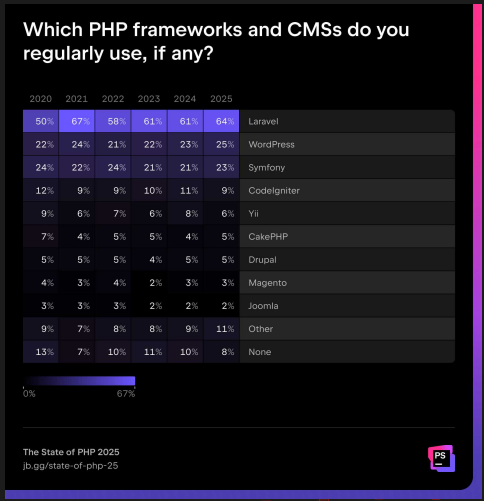
Introduction to Laravel: Laravel vs PHP

- **Performance** reality check
  - **I/O still dominates:** Database calls and network latency, not PHP byte-code, are the bottleneck in most production workloads; **Laravel's caching**, eager-loading and Octane workers help mitigate that.
- **Where raw PHP can still win**: Ultra-lean APIs, edge-deployed scripts or micro-services with stringent memory budgets can shave a few milliseconds by skipping framework bootstrapping.
- **When Laravel outperforms:** Long-running apps using Laravel Octane (Swoole/RoadRunner) stay active between requests, handling more requests faster than traditional setups.

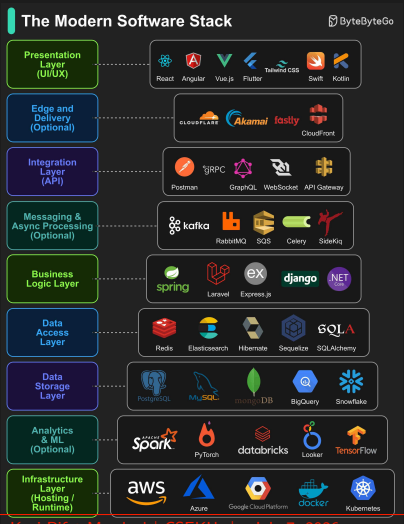
Source: https://tailkits.com/blog/laravel-vs-php/

Introduction to Laravel: Laravel vs PHP





Introduction to Laravel: Exactly, where are we?



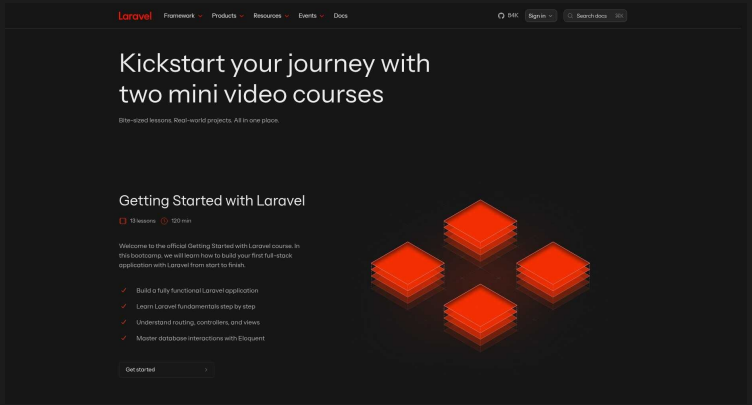
Introduction to Laravel: Why Laravel?

- Modular **packaging system** with a dedicated dependency manager
- Multiple ways for accessing relational databases
- Ease database migration, seeding, and version control
- Built-in support for unit testing and test-driven development
- Robust security features for authentication, authorization, and data protection
- Task scheduling and queuing for background jobs
- Utilities that aid in application deployment and maintenance
- Orientation toward syntactic sugar for elegant code

Introduction to Laravel: Laravel Release History

Version	Release date	Bug Fixes Until	Security Fixes Until	PHP version
1.0	June 2011	–	–	–
7	March 3, 2020	October 6, 2020	March 3, 2021	7.2 – 8.0
8	September 8, 2020	July 26, 2022	January 24, 2023	7.3 – 8.1
9	February 8, 2022	August 8, 2023	February 6, 2024	8.0 – 8.2
10	February 14, 2023	August 6, 2024	February 4, 2025	8.1 – 8.3
11 (Unsupported)	March 12, 2024	September 3, 2025	March 12, 2026	8.2 – 8.4
12 (Supported)	February 24, 2025	August 13, 2026	February 24, 2027	$\geq$ 8.2
13 (Latest)	March 17, 2026	Q3 2027	March 17, 2028	$\geq$ 8.3
14 (Planned)	Q1 2027	–	–	–

Learning Path: Best Learning Resource



Laravel Learn: Getting Started with Laravel (click here)

NOTE: This slide is based on this official learning resources by Laravel.

Learning Path: Getting Started with Laravel **Fundamentals**

1. **What are we building?** - Meet Chirper, a Twitter-like microblogging app
2. **Setting up your Laravel project** - Install Laravel, PHP, and create your first project
3. **Your first route** - Create routes and Blade templates for your homepage
4. **Deploying your app** - Deploy to Laravel Cloud with Git basics
5. **What is MVC?** - Understand Model-View-Controller pattern and create your first controller
6. **Working with the database** - Migrations, seeding, and database tools
7. **Our first model** - Build Eloquent models to interact with databases

Learning Path: Getting Started with Laravel **Fundamentals**

1. **What are we building?** - Meet Chirper, a Twitter-like microblogging app
2. **Setting up your Laravel project** - Install Laravel, PHP, and create your first project
3. **Your first route** - Create routes and Blade templates for your homepage
4. **Deploying your app** - Deploy to Laravel Cloud with Git basics
5. **What is MVC?** - Understand Model-View-Controller pattern and create your first controller
6. **Working with the database** - Migrations, seeding, and database tools
7. **Our first model** - Build Eloquent models to interact with databases

Learning Path: Getting Started with Laravel **Fundamentals**

1. **What are we building?** - Meet Chirper, a Twitter-like microblogging app
2. **Setting up your Laravel project** - Install Laravel, PHP, and create your first project
3. **Your first route** - Create routes and Blade templates for your homepage
4. **Deploying your app** - Deploy to Laravel Cloud with Git basics
5. **What is MVC?** - Understand Model-View-Controller pattern and create your first controller
6. **Working with the database** - Migrations, seeding, and database tools
7. **Our first model** - Build Eloquent models to interact with databases

Learning Path: Getting Started with Laravel **Fundamentals**

1. **What are we building?** - Meet Chirper, a Twitter-like microblogging app
2. **Setting up your Laravel project** - Install Laravel, PHP, and create your first project
3. **Your first route** - Create routes and Blade templates for your homepage
4. **Deploying your app** - Deploy to Laravel Cloud with Git basics
5. **What is MVC?** - Understand Model-View-Controller pattern and create your first controller
6. **Working with the database** - Migrations, seeding, and database tools
7. **Our first model** - Build Eloquent models to interact with databases

Learning Path: Getting Started with Laravel **Fundamentals**

1. **What are we building?** - Meet Chirper, a Twitter-like microblogging app
2. **Setting up your Laravel project** - Install Laravel, PHP, and create your first project
3. **Your first route** - Create routes and Blade templates for your homepage
4. **Deploying your app** - Deploy to Laravel Cloud with Git basics
5. **What is MVC?** - Understand Model-View-Controller pattern and create your first controller
6. **Working with the database** - Migrations, seeding, and database tools
7. **Our first model** - Build Eloquent models to interact with databases

Learning Path: Getting Started with Laravel **Fundamentals**

1. **What are we building?** - Meet Chirper, a Twitter-like microblogging app
2. **Setting up your Laravel project** - Install Laravel, PHP, and create your first project
3. **Your first route** - Create routes and Blade templates for your homepage
4. **Deploying your app** - Deploy to Laravel Cloud with Git basics
5. **What is MVC?** - Understand Model-View-Controller pattern and create your first controller
6. **Working with the database** - Migrations, seeding, and database tools
7. **Our first model** - Build Eloquent models to interact with databases

Learning Path: Getting Started with Laravel **Fundamentals**

1. **What are we building?** - Meet Chirper, a Twitter-like microblogging app
2. **Setting up your Laravel project** - Install Laravel, PHP, and create your first project
3. **Your first route** - Create routes and Blade templates for your homepage
4. **Deploying your app** - Deploy to Laravel Cloud with Git basics
5. **What is MVC?** - Understand Model-View-Controller pattern and create your first controller
6. **Working with the database** - Migrations, seeding, and database tools
7. **Our first model** - Build Eloquent models to interact with databases

Learning Path: Getting Started with Laravel **Fundamentals** (continued)

8. **Showing the feed** - Blade components and polished UI design
9. **Creating and storing Chirps** - Forms, validation, and mass assignment
10. **Edit and delete Chirps** - Complete CRUD operations with RESTful routing
11. **Basic authentication: Registration** - Build user registration and password hashing
12. **Basic authentication: Login/Logout** - Sessions, middleware, and route protection
13. **What's Next?** - Extend Chirper with new features and continue learning

Learning Path: Getting Started with Laravel **Fundamentals** (continued)

8. **Showing the feed** - Blade components and polished UI design
9. **Creating and storing Chirps** - Forms, validation, and mass assignment
10. **Edit and delete Chirps** - Complete CRUD operations with RESTful routing
11. **Basic authentication: Registration** - Build user registration and password hashing
12. **Basic authentication: Login/Logout** - Sessions, middleware, and route protection
13. **What's Next?** - Extend Chirper with new features and continue learning

Learning Path: Getting Started with Laravel **Fundamentals** (continued)

8. **Showing the feed** - Blade components and polished UI design
9. **Creating and storing Chirps** - Forms, validation, and mass assignment
10. **Edit and delete Chirps** - Complete CRUD operations with RESTful routing
11. **Basic authentication: Registration** - Build user registration and password hashing
12. **Basic authentication: Login/Logout** - Sessions, middleware, and route protection
13. **What's Next?** - Extend Chirper with new features and continue learning

Learning Path: Getting Started with Laravel **Fundamentals** (continued)

8. **Showing the feed** - Blade components and polished UI design
9. **Creating and storing Chirps** - Forms, validation, and mass assignment
10. **Edit and delete Chirps** - Complete CRUD operations with RESTful routing
11. **Basic authentication: Registration** - Build user registration and password hashing
12. **Basic authentication: Login/Logout** - Sessions, middleware, and route protection
13. **What's Next?** - Extend Chirper with new features and continue learning

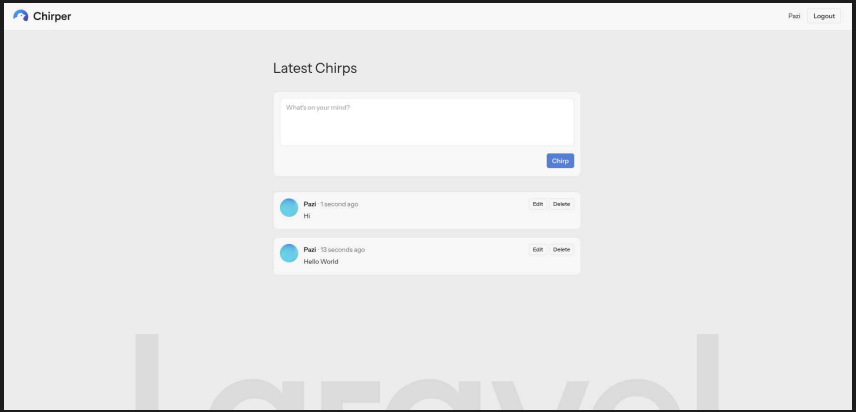
Learning Path: Getting Started with Laravel **Fundamentals** (continued)

8. **Showing the feed** - Blade components and polished UI design
9. **Creating and storing Chirps** - Forms, validation, and mass assignment
10. **Edit and delete Chirps** - Complete CRUD operations with RESTful routing
11. **Basic authentication: Registration** - Build user registration and password hashing
12. **Basic authentication: Login/Logout** - Sessions, middleware, and route protection
13. **What's Next?** - Extend Chirper with new features and continue learning

Learning Path: Getting Started with Laravel **Fundamentals** (continued)

8. **Showing the feed** - Blade components and polished UI design
9. **Creating and storing Chirps** - Forms, validation, and mass assignment
10. **Edit and delete Chirps** - Complete CRUD operations with RESTful routing
11. **Basic authentication: Registration** - Build user registration and password hashing
12. **Basic authentication: Login/Logout** - Sessions, middleware, and route protection
13. **What's Next?** - Extend Chirper with new features and continue learning

Learning Path: What are we building? I



Learning Path: What are we building? II

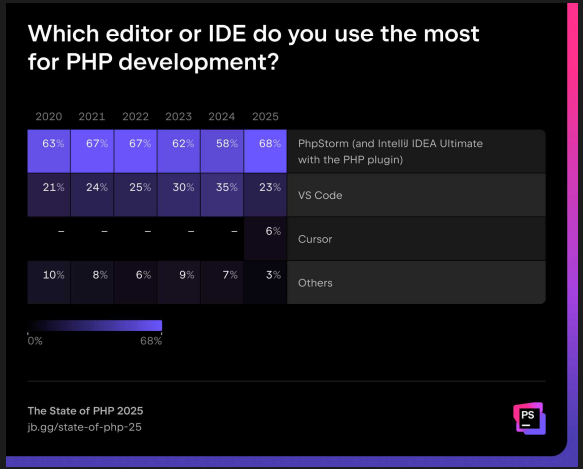
See the live demo on Laravel Cloud: http://chirper.laravel.cloud/.

- Register an account
- Post chirps
- See a live feed
- Edit or delete your own chirps
- Only modify your own chirps.

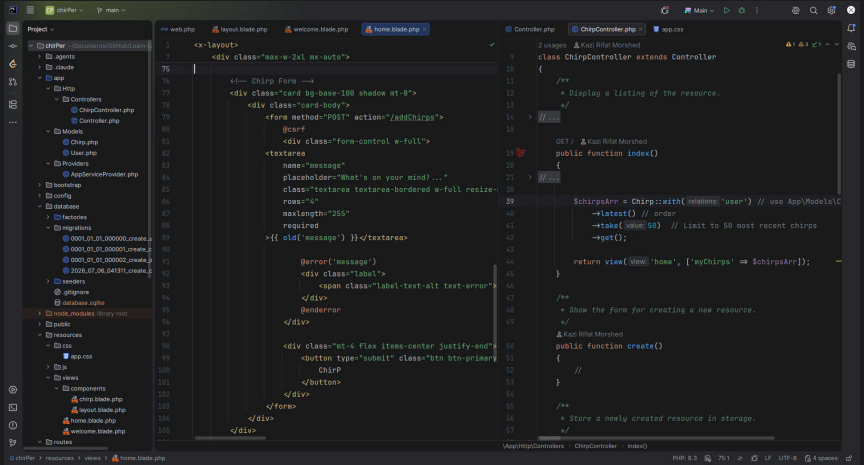
Learning Path: Install Laravel I

- Install PHP and Composer:
  - Debian/Ubuntu: `sudo apt update && sudo apt install -y php`
  - Arch Linux: `sudo pacman -S php`
- Install Laravel: https://laravel.com/docs/12.x#installing-php
  - **Linux:** `/bin/bash -c "$(curl -fsSL https://php.new/install/linux/8.4)"`
  - **Windows:** (Follow official instructions)
  - **Composer:** `composer global require laravel/installer`
- Check Installation: `php -v` and `laravel -v`
- Create a new Laravel project: `laravel new chirper && cd chirper`
- Run the development server: `composer run dev`
- Access the application in a web browser: `http://localhost:8000`

Learning Path: IDE for Laravel



Learning Path: IDE for Laravel



Learning Path: Explore the Project Structure

```
AGENTS.md    composer.json      package.json       routes/
app/         composer.lock      package-lock.json  storage/
artisan*     config/            phpunit.xml        tests/
boost.json   database/          public/            vendor/
bootstrap/   README.md          vite.config.js
CLAUDE.md    node_modules/      resources/
```

- **routes/** — Define application URLs and behavior
- **app/** — **Models** and **controllers** live here
- **resources/views/** — Blade templates **View** in HTML
- **database/** — Migrations and SQLite database file

You'll primarily work in just a couple of these directories!

Learning Path: Your First Route

In Laravel, web routes are defined in the `routes/web.php` file:

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

```

Listing 1: Defining a basic route in routes/web.php

Let's follow that welcome view to `resources/views/welcome.blade.php`.

Learning Path: Your First Route

Let's play with this code!

```
Route::get('/', function () {
    // return view('welcome');
    return view('home'); // home not found or not created yet
});

```

- Refreshing now gives an error—Laravel can't find the home view yet.
- Create a new file named `home.blade.php` in `resources/views`.
- The error is gone, but the page is still blank, so let's build something.
- If you're following along on `laravel.com/learn`, you'll find the full code below the lesson.

Learning Path: Copy-Pasteable Code Snippets

To copy the full source files directly into your code editor during the presentation, click the links below to open/extract the files:

- **Home Blade File**: **[Click to Open/Copy]**

Notice that $*slot* in the main section? That's where the content from our individual pages will be injected. So basically, any other stuff that we add to a file using this layout will be put right there.

Note: File attachments are supported in standard desktop PDF viewers (like Adobe Acrobat Reader, Okular, and Evince). If viewing in a browser PDF reader, check the sidebar/attachments panel.

Also, look at how we're handling the title:
`{{ isset($title) ? $title . ' - Chirper' : 'Chirper' }}`.
We're using some PHP here to say: if this title variable is set, let's use it and append " - Chirper". If not, let's just revert to "Chirper". This gives us consistent branding across all pages!

With Tailwind CSS v4, we can customize our design system directly in app.css.
Update resources/css/app.css: **Tailwind CSS File**:
[Click to Open/Copy]

Learning Path: Applying the Layout Component

Let's update our `home.blade.php` to use this layout.

```
<x-layout>
    <x-slot:title> Welcome </x-slot:title>
    <div class="max-w-2xl mx-auto">
        <div class="card bg-base-100 shadow mt-8">
            <div class="card-body">
                <div>
                    <h1 class="text-3xl font-bold">Welcome to Chirper!</h1>
                    <p class="mt-4 text-base-content/60">This is your brand new Laravel application. Time to make it sing (or chirp)!</p>
    </div> </div> </div> </div>
</x-layout>

```

Learning Path: Applying the Layout Component: What's Happening

```
...
<main class="flex-1 container mx-auto px-4 py-8">
    {{ $slot }}
</main>
...
```

Listing 2: layout.blade.php

```
<x-layout>
    <x-slot:title> Welcome </x-slot:title>
    ...
</x-layout>

```

Listing 3: home.blade.php

Reminder: `Route::get('/', function () return view('home'););`

Learning Path: Understanding MVC in Laravel

Model

- Manage your data and your business logic
- Fetch chirps(data) from the database

View

- Display information to your users
- Show the chirps nicely in `home.blade.php`

Controller

- Handle requests and coordinate responses
- Decide which chirps to show and how to present them

Learning Path: MVC Architecture: A Restaurant Analogy

- **The Waiter (Controller)**: This is what you would think of as the waiter in a restaurant. It takes your order, it coordinates everything, and then it brings you the food.
- **The Kitchen (Model)**: Well, the kitchen would then be the model. This is what prepares the food, it manages the ingredients—your data, I guess, in an application.
- **The Presentation (View)**: And then there's the presentation—how the food looks on the plate when it arrives. It's your view.

Learning Path: The MVC Architecture Pattern implemented in Laravel

- Model-View-Controller pattern that keeps Laravel code organized in a structure.
- Each piece has its job in the MVC structure: Model, View, Controller.
- This keeps the code organized, and future you will thank current you.
- How to create MVC:
  - **Model**: `php artisan make:model Chirp`
  - **View**: Create a Blade template in `resources/views` directory.
  - **Controller**: `php artisan make:controller ChirpController` (Let's do it now!)
    Creates a new file in `app/Http/Controllers/ChirpController.php`
  - **Route**: `routes/web.php`

Learning Path: The MVC Flow

- **Route**: This slash parameter, this root of your application, receives the request
- **Route calls Controller**: Route calls the ChirpController, specifically the index method of this ChirpController class
- **Controller prepares data**: Controller prepares the data—it's waiting at our table to decide how are we going to present this data (our sample chirps)
- **Controller passes to View**: Controller passes all this information to the view—in this case, that home.blade.php
- **View renders**: View then renders that HTML with the data
- **User sees page**: User sees this page!

Learning Path: Add an Index Method

After `php artisan make:controller ChirpController`, we're going to add an index method to handle our homepage.

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class ChirpController extends Controller {
    public function index() {
        return view('home');
    }
}

```

Listing 4: app/Http/Controllers/ChirpController.php

The index method is a **convention for displaying a list of things** (or in our case, the main page).

Learning Path: Update the Route

So in our web routes, how do we change this to instead point to our index method in our ChirpController?

```php
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ChirpController;

Route::get('/', [ChirpController::class, 'index']);

```

Listing 5: routes/web.php

Learning Path: Passing Data to View

Controllers really shine when we need to prepare data and then serve that data, like a waiter of a restaurant.

File `app/Http/Controllers/ChirpController.php`: **[Click to Open/Copy]**

The task of View is showing the data. It will get data from Controller. Normally, the Controller will get data from Model which is responsiable for communicating with the database.

File `resources/views/home.blade.php`: **[Click to Open/Copy]**

Learning Path: Method Descriptions I

Controller Creation Command with Resourceful Methods

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

- `index()`: This method is responsible for displaying the list of chirps(socail media posts). It **retrieves the chirps(posts) from the database** and **passes them to the view**.
  Displaying a listing of all resources. It's listing all of the chirps, this home page.
- `create()`: This method is responsible for showing the form to create a new chirp. It returns a view with the form.
  Might be showing a form to create a new chirp.

Learning Path: Method Descriptions II

- `store(Request $request)`: This method is responsible for storing a new chirp in the database. It validates the request data and creates a new chirp record.
  Where we're creating and saving a new chirp.
- `edit(Chirp $chirp)`: This method is responsible for showing the form to edit an existing chirp. It retrieves the chirp from the database and passes it to the view.
  Displaying a form for editing a single chirp
- `update(Request $request, Chirp $chirp)`: This method is responsible for updating an existing chirp in the database. It validates the request data and updates the chirp record.
  Would be to update that single chirp.
- `destroy(Chirp $chirp)`: This method is responsible for deleting an existing chirp from the database. It deletes the chirp record.
  Deleting a chirp.

Let's Continue from Laravel Learn

Link (Click Here)

Thank You!

Any Questions?

Appendix: Disclaimer

- All image credit goes to the respective owners.
- All images were used for educational purposes only.
- We apologize for not mentioning all sources or attributions due to limited time for slide preparation.