PHP & Laravel Backend Engineering

Week 6 – Introduction to Laravel Framework

Program: PHP & Laravel Backend Engineering
Week: 6
Level: Beginner
Duration: 3 Days

□ Learning Objectives

By the end of this week, you will be able to:

- □ Set up a Laravel development environment
- □ Understand Laravel's MVC architecture
- □ Use Artisan command line tool
- □ Create basic routes and views
- □ Work with Blade templating
- □ Understand Laravel's directory structure
- □ Configure environment settings

Day 1 – Laravel Fundamentals

□ What is Laravel?

Laravel is a powerful PHP framework that makes web development a breeze. Think of it as a Swiss Army knife for PHP developers - it comes with all the tools you need to build modern web applications quickly and securely.

Real-world analogy:
Imagine you're building a house. Instead of making every nail, hammer, and tool yourself, you go to a hardware store that has everything ready to use. Laravel is like that hardware store for web development!

Why Laravel?

- □ **MVC Architecture** - Keeps your code clean and organized
- □ **Built-in Features** - Authentication, routing, sessions out of the box
- □ **Composer Integration** - Easy package management
- □ **Artisan CLI** - Your command-line best friend
- □ **Blade Templating** - Clean, elegant, and powerful

What makes Laravel special:

Feature	What it does	Why it matters
Eloquent ORM	Makes database easy	Write less SQL, more PHP
Routing	Handles URLs	Clean, readable URLs
Blade Templates	HTML with PHP power	Reusable, secure templates
Authentication	User login system	Built-in, secure login
Artisan	Command tools	Automate repetitive tasks

□ Setting Up Laravel

Prerequisites:

- PHP >= 8.0
- Composer (PHP package manager)
- MySQL or MariaDB
- Node.js & NPM (for frontend assets)

Step 1: Check Your Environment Before installing Laravel, make sure you have everything:

```
# Check PHP version
php --version
# Should show PHP 8.0 or higher

# Check Composer
composer --version
# Should show Composer version

# Check Node.js
node --version
# Should show Node.js version
```

Step 2: Create a New Laravel Project

```
# Create a new Laravel project
composer create-project laravel/laravel my-laravel-app

# Navigate to your project directory
cd my-laravel-app
```

What just happened?

- `composer create-project` downloads Laravel and sets up a complete project
- It installs all required packages and dependencies
- Creates the entire folder structure for you
- Takes about 1-2 minutes depending on your internet speed

Step 3: Configure Your Environment

```
# Copy the environment file
cp .env.example .env

# Generate an application key
php artisan key:generate
```

Why these steps matter:

- `.env` file stores your database credentials and app settings
- Application key encrypts your sessions and cookies
- These are security essentials!

Step 4: Start the Development Server

```
# Start the built-in PHP server
php artisan serve
```

Visit your application: Open your browser and go to: http://127.0.0.1:8000

What you should see:

- Laravel welcome page with logo
- Links to documentation and resources
- Confirmation that everything is working!

Common Troubleshooting:

Problem	Solution
Port already in use	Use `php artisan serve --port=8001`
Permission errors	Run `chmod -R 755 storage bootstrap/cache`
Database connection failed	Check `.env` file credentials

Laravel Directory Structure

Understanding the Laravel Folder Structure

Think of Laravel's directory structure like a well-organized office building - each folder has a specific purpose!

```
my-laravel-app/
├── app/                # 🏢 Your main office - Core application code
│   ├── Http/          # 📮 Mail room - Controllers, middleware
│   │   ├── Controllers/  # 🎮 Control center - Handle requests
│   │   └── Middleware/   # 🛡️ Security guards - Check permissions
│   ├── Models/        # 📊 Data department - Database models
│   └── Providers/     # 🔧 Service providers - Register services
├── config/            # ⚙️ Settings room - Configuration files
├── database/          # 🗄️ Database room - Migrations and seeders
│   ├── migrations/    # 📋 Database blueprints
│   ├── seeders/       # 🌱 Sample data
│   └── factories/     # 🏭 Data factories
├── public/            # 🚪 Front door - Web entry point
│   ├── index.php      # 🎯 Main entrance file
│   └── assets/        # 🎨 Public files (CSS, JS, images)
├── resources/         # 📚 Library - Views and assets
│   ├── js/           # ⚡ JavaScript files
│   ├── css/          # 🎨 CSS files
│   └── views/        # 👁️ Blade templates (HTML)
├── routes/           # 🚦 Traffic control - Application routes
│   ├── web.php       # 🌐 Web routes
│   └── api.php       # 🔌 API routes
├── storage/          # 📦 Warehouse - Logs, compiled views, caches
│   ├── logs/         # 📝 Error logs
│   ├── framework/    # 🔧 Framework cache
│   └── app/          # 💾 Application cache
├── tests/            # 🧪 Lab room - Test files
├── vendor/           # 📦 Package room - Composer packages
├── .env              # 🔐 Secret file - Environment variables
├── artisan           # 🛠️ Tool belt - Command line tool
└── composer.json     # 📋 Shopping list - Dependencies
```

Key Folders You'll Use Most:

Folder	What you'll do there	Example files
`app/Http/Controllers`	Write page logic	`UserController.php`, `PostController.php`
`app/Models`	Create database models	`User.php`, `Post.php`
`resources/views`	Build HTML pages	`welcome.blade.php`, `dashboard.blade.php`
`routes/web.php`	Define URLs	`Route::get('/', 'HomeController@index')`
`database/migrations`	Create database tables	`create_users_table.php`

Pro Tip:

- **90% of your work** will be in `app/`, `resources/views/`, and `routes/`
- **Never edit files in `vendor/`** - they're managed by Composer
- **Keep `storage/` writable** - Laravel needs to write cache and logs here

🎯 Creating Your First Route

What are Routes? Routes are like street signs for your application - they tell Laravel where to send visitors when they type a URL.

File to edit: `routes/web.php`

```php
<?php

// Import the Route class - this gives us access to routing methods
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

// Example 1: Basic Route - Returns simple text
Route::get('/', function () {
    return 'Welcome to Laravel!';
});

// Example 2: Route with Parameters - Dynamic content
Route::get('/hello/{name}', function ($name) {
    return "Hello, $name! Welcome to our website!";
});

// Example 3: Route with Optional Parameters
Route::get('/greet/{name?}', function ($name = 'Guest') {
    return "Hello, $name!";
});

// Example 4: Route with Multiple Parameters
Route::get('/user/{id}/profile', function ($id) {
    return "Showing profile for user ID: $id";
});

// Example 5: Route that returns a view (we'll create this next)
Route::get('/welcome', function () {
    return view('welcome');
});

// Example 6: Route with name (for easy linking later)
Route::get('/about', function () {
    return 'About Us Page';
})->name('about');
```

Understanding Route Methods:

Method	When to use	Example
`Route::get()`	Display pages	`Route::get('/home', ...)`
`Route::post()`	Submit forms	`Route::post('/login', ...)`
`Route::put()`	Update data	`Route::put('/user/1', ...)`
`Route::delete()`	Delete data	`Route::delete('/user/1', ...)`
`Route::any()`	Any method	`Route::any('/webhook', ...)`

Testing Your Routes: After adding these routes, visit these URLs in your browser:

- http://127.0.0.1:8000/ → "Welcome to Laravel!"
- http://127.0.0.1:8000/hello/John → "Hello, John! Welcome to our website!"
- http://127.0.0.1:8000/greet → "Hello, Guest!"
- http://127.0.0.1:8000/greet/Sarah → "Hello, Sarah!"

- http://127.0.0.1:8000/user/5/profile → "Showing profile for user ID: 5"

🎨 Creating Your First View

What are Views? Views are the HTML pages your users see. Laravel uses Blade templating to mix PHP and HTML safely.

Step 1: Create a View File Create this file: `resources/views/welcome.blade.php`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ config('app.name') }}</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            max-width: 800px;
            margin: 50px auto;
            padding: 20px;
            background: #f5f5f5;
        }
        .container {
            background: white;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }
        h1 {
            color: #333;
            text-align: center;
        }
        .info {
            background: #e3f2fd;
            padding: 15px;
            border-radius: 5px;
            margin: 20px 0;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Welcome to {{ config('app.name') }}!</h1>

        <div class="info">
            <h3>🎉 Your First Laravel View!</h3>
            <p><strong>Current Time:</strong> {{ now()->format('g:i A, l, F j, Y') }}</p>
            <p><strong>Laravel Version:</strong> {{ app()->version() }}</p>
            <p><strong>Environment:</strong> {{ config('app.env') }}</p>
        </div>

        <h3>📚 What you've learned so far:</h3>
        <ul>
            <li>✅ Installed Laravel successfully</li>
            <li>✅ Created your first routes</li>
            <li>✅ Built your first Blade view</li>
            <li>✅ Used Blade syntax ({{ }})</li>
        </ul>

        <p><em>This view file is located at: <code>resources/views/welcome.blade.php</code></em></p>
    </div>
</body>
</html>
```

Understanding Blade Syntax:

Blade Syntax	What it does	Example
`{{ $variable }}`	Echoes variable safely	`{{ $name }}`
`{!! $html !!}`	Echoes raw HTML (dangerous)	`{!! $content !!}`
`{{ config('key') }}`	Gets config value	`{{ config('app.name') }}`
`{{ now() }}`	Current time	`{{ now()->format('Y-m-d') }}`

Step 2: Update Your Route to Use the View

Add this route to `routes/web.php`:

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

What happens when you visit /welcome:

1. Laravel finds the route `/welcome`
2. It calls `view('welcome')`
3. Laravel looks for `resources/views/welcome.blade.php`
4. It processes the Blade syntax
5. It sends the final HTML to your browser

Step 3: Create a Dynamic View with Data

Let's create a more advanced view that accepts data:

Route:

```
Route::get('/user/{name}', function ($name) {
    return view('user', [
        'name' => $name,
        'age' => 25,
        'hobbies' => ['Reading', 'Coding', 'Gaming']
    ]);
});
```

View file: `resources/views/user.blade.php`

```
<!DOCTYPE html>
<html>
<head>
    <title>User Profile: {{ $name }}</title>
</head>
<body>
    <h1>User Profile</h1>
    <p><strong>Name:</strong> {{ $name }}</p>
    <p><strong>Age:</strong> {{ $age }}</p>

    <h3>Hobbies:</h3>
    <ul>
        @foreach($hobbies as $hobby)
            <li>{{ $hobby }}</li>
        @endforeach
    </ul>
</body>
</html>
```

Visit: http://127.0.0.1:8000/user/Alice to see your dynamic view in action!

Day 2 – Working with Controllers & Blade

✅ Creating a Controller

Using Artisan:

```
php artisan make:controller PageController
```

app/Http/Controllers/PageController.php:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PageController extends Controller
{
    public function home()
    {
        return view('pages.home', [
            'title' => 'Home Page',
            'users' => ['John', 'Jane', 'Mike']
        ]);
    }

    public function about()
    {
        return view('pages.about', [
            'title' => 'About Us'
        ]);
    }
}
```

✅ Updating Routes

routes/web.php:

```
use App\Http\Controllers\PageController;

Route::get('/', [PageController::class, 'home'])->name('home');
Route::get('/about', [PageController::class, 'about'])->name('about');
```

✅ Blade Templating Basics

resources/views/layouts/app.blade.php:

```
<!DOCTYPE html>
<html>
<head>
    <title>@yield('title', config('app.name'))</title>
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
</head>
<body>
    <nav>
        <a href="{{ route('home') }}">Home</a>
        <a href="{{ route('about') }}">About</a>
    </nav>

    <div class="container">
        @yield('content')
    </div>

    <script src="{{ asset('js/app.js') }}"></script>
</body>
</html>
```

resources/views/pages/home.blade.php:

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

@section('title', 'Home')

@section('content')
    <h1>Welcome to {{ $title }}</h1>

    <h2>Users List</h2>
    <ul>
        @foreach($users as $user)
            <li>{{ $user }}</li>
        @endforeach
    </ul>
@endsection
```

Day 3 – Environment & Configuration

Environment Configuration

.env file:

```
APP_NAME=Laravel
APP_ENV=local
APP_DEBUG=true
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```

Using Environment Variables

config/app.php:

```
'name' => env('APP_NAME', 'Laravel'),
'env' => env('APP_ENV', 'production'),
'debug' => (bool) env('APP_DEBUG', false),
```

Practice Exercises

1. **Create a Contact Page**
    - Create a new route and controller
    - Add a contact form with name, email, and message
    - Display the submitted data
2. **Dynamic Navigation**
    - Create a navigation partial
    - Highlight the current page
    - Add authentication links
3. **Configuration**
    - Add a custom configuration value
    - Use it in your views
    - Try different environments (.env vs production)

Resources

- Laravel Documentation
- Laracasts

- Laravel News
- Laravel Daily

Tips for Success

- **Practice daily** - Build small projects
- **Learn Blade** - Master the templating engine
- **Use Artisan** - It's your best friend
- **Read the docs** - Laravel has excellent documentation
- **Debug with `dd()`** - Dump and die is your friend
- **Version Control** - Use Git from day one
- **Test your code** - Learn PHPUnit basics

Week 6 Summary

What you learned:

- Setting up a Laravel project
- Understanding MVC architecture
- Using Artisan commands
- Creating routes and controllers
- Working with Blade templates
- Configuring environments

Next week: Building a complete Laravel application with databases!