PHP & Laravel Backend Engineering

Week 7 - Routing, Controllers & Blade (Intermediate)

Program: PHP & Laravel Backend Engineering
Week: 7
Level: Intermediate
Duration: 3 Days

Scope and Prerequisite

This week starts **after Week 6 is completed**.

Already covered in Week 6:

- Laravel setup and first project boot
- Folder structure and environment basics
- Basic routes and simple route parameters
- First controller and route-to-controller mapping
- Blade basics (`@extends` , `@section` , `@yield`) and first views

Week 7 focuses on the next level only: cleaner architecture, advanced routing patterns, and reusable Blade systems.

Learning Objectives

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

- Design scalable route structures with groups, prefixes, and naming strategy
- Build cleaner controllers with focused responsibilities
- Use route constraints and route model binding effectively
- Build reusable Blade layouts, partials, and components
- Create a multi-page company website with maintainable template structure
- Apply CSRF protection correctly in all form submissions

Day 1 - Advanced Routing and Controller Structure

1. Route Architecture for Real Projects

Instead of adding routes one by one, organize them by feature and access level.

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\PageController;
use App\Http\Controllers\ServiceController;

Route::name('site.')->group(function () {
    Route::get('/', [PageController::class, 'home'])->name('home');
    Route::get('/about', [PageController::class, 'about'])->name('about');

    Route::prefix('services')->name('services.')->group(function () {
        Route::get('/', [ServiceController::class, 'index'])->name('index');
        Route::get('/{service:slug}', [ServiceController::class, 'show'])->name('show');
    });
});
```

Why this is better:

- Consistent URL pattern
- Predictable route names
- Easier maintenance when features grow

2. Route Constraints and Safe Parameters

```
Route::get('/team/{id}', [PageController::class, 'teamMember'])
    ->whereNumber('id')
    ->name('site.team.member');

Route::get('/pages/{section}', [PageController::class, 'section'])
    ->whereIn('section', ['company', 'mission', 'careers']);
```

Use constraints to reject invalid URLs early and reduce controller complexity.

3. Route Model Binding by Slug

```
// Route
Route::get('/services/{service:slug}', [ServiceController::class, 'show'])
    ->name('site.services.show');

// Controller
public function show(Service $service)
{
    return view('pages.services.show', compact('service'));
}
```

Result:

- Cleaner controller signatures
- Automatic 404 for missing records
- Readable URLs (`/services/web-development`)

4. Controller Organization Pattern

Keep controllers thin and feature-focused.

```
<?php

namespace App\Http\Controllers;

use Illuminate\View\View;

class PageController extends Controller
{
    public function home(): View
    {
        return view('pages.home', [
            'title' => 'Welcome',
            'hero' => 'We build reliable software products.',
        ]);
    }

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

5. Useful Artisan Commands for Routing Work

```
php artisan route:list
php artisan make:controller ServiceController
php artisan make:controller ContactController --invokable
php artisan route:clear
```

Day 1 Practical Exercise

1. Refactor existing routes into grouped, named routes.
2. Add a slug-based service details route.
3. Create `PageController` and `ServiceController` with clear method names.
4. Verify all routes with `php artisan route:list`.

Day 2 - Blade Reusability and Page Composition

1. Layout Strategy Beyond Basics

Build one master layout and compose pages from reusable parts.

```
<!-- resources/views/layouts/app.blade.php -->
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>@yield('title', config('app.name'))</title>
    <link rel="stylesheet" href="{{ asset('css/app.css') }}">
    @stack('styles')
</head>
<body>
    @include('partials.nav')

    <main class="container py-4">
        @yield('content')
    </main>

    @include('partials.footer')
    @stack('scripts')
</body>
</html>
```

2. Reusable Blade Components

Use components for repeated UI blocks.

```
<!-- resources/views/components/service-card.blade.php -->
@props(['service'])

<article class="service-card">
    <h3>{{ $service['name'] }}</h3>
    <p>{{ $service['summary'] }}</p>
    <a href="{{ route('site.services.show', $service['slug']) }}">Read more</a>
</article>
```

Usage:

```
@foreach($services as $service)
    <x-service-card :service="$service" />
@endforeach
```

3. Navigation With Active Route State

```
<!-- resources/views/partials/nav.blade.php -->
<nav>
    <a href="{{ route('site.home') }}" class="{{ request()->routeIs('site.home') ? 'active' : '' }}">Home</a>
    <a href="{{ route('site.about') }}" class="{{ request()->routeIs('site.about') ? 'active' : '' }}">About</a>
    <a href="{{ route('site.services.index') }}" class="{{ request()->routeIs('site.services.*') ? 'active' : '' }}">Services</a>
    <a href="{{ route('site.contact') }}" class="{{ request()->routeIs('site.contact*') ? 'active' : '' }}">Contact</a>
</nav>
```

4. CSRF-Ready Form Structure

```
<form method="POST" action="{{ route('site.contact.submit') }}">
    @csrf
    <input type="text" name="name" placeholder="Your name">
    <input type="email" name="email" placeholder="Your email">
    <textarea name="message" placeholder="Message"></textarea>
    <button type="submit">Send</button>
</form>
```

Day 2 Practical Exercise

1. Create a master layout with `@include` partials.
2. Build at least two Blade components (`service-card` , `page-header`).
3. Convert one long page into reusable partials and components.
4. Add active state styling to navigation links.

Day 3 - Project: Company Website (Routing + Controller + Blade)

Project Goal

Build a multi-page company website that demonstrates:

- Advanced route structure
- Clean controller design
- Reusable Blade templates and components

Important Boundary for Week 7

To avoid overlap with upcoming weeks:

- **Do not use database models/migrations here** (Week 8 topic)
- **Do not build authentication/roles here** (Week 9 topic)
- **Do not implement advanced validation/files/mail here** (Week 10 topic)

Use array/mock data in controllers for this week project.

Minimum Required Pages

1. Home (`site.home`)
2. About (`site.about`)
3. Services list (`site.services.index`)
4. Service details by slug (`site.services.show`)
5. Contact form page and submit endpoint (`site.contact` , `site.contact.submit`)

Suggested Project Structure

```
app/Http/Controllers/
  PageController.php
  ServiceController.php
  ContactController.php

resources/views/
  layouts/app.blade.php
  partials/nav.blade.php
  partials/footer.blade.php
  components/service-card.blade.php
  pages/home.blade.php
  pages/about.blade.php
  pages/services/index.blade.php
  pages/services/show.blade.php
  pages/contact.blade.php

routes/
  web.php
```

Example Contact Controller (No DB Yet)

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\View\View;

class ContactController extends Controller
{
    public function index(): View
    {
        return view('pages.contact');
    }

    public function submit(Request $request): RedirectResponse
    {
        // Week 7 focus: request flow + CSRF + routing.
        // Deeper validation/mail/database comes later.
        return back()->with('success', 'Your message was received.');
    }
}
```

Deliverables

- Working Laravel project with all required routes and pages
- Controllers with clear method names and minimal logic
- Reusable Blade layout, partials, and components
- Short README explaining route map and file structure

Presentation Guide (10-15 minutes)

1. Show route architecture (`route:list` screenshot/output)
2. Navigate all pages and dynamic service slug pages
3. Explain layout/component reuse approach
4. Explain how week 7 prepares for week 8 (database integration)

Resources

- Laravel Routing Documentation
- Laravel Controllers Documentation
- Laravel Blade Documentation

Week 7 Summary

This week extends Week 6 and avoids repeating beginner content.
You now have:

- Advanced and maintainable route architecture
- Cleaner controller design patterns
- Reusable Blade layout/component workflow
- A complete company website shell ready for Week 8 database integration