Contents

LIVT stack (Laravel, Inertia, Vue, Tailwind).

Why this stack?

- Laravel: Handles the backend (database, routing, logic).
- Vue.js: Handles the frontend (interactivity).
- Inertia.js: The "glue." It allows you to build a Single Page Application (SPA) using classic server-side routing. You don't need a separate API; you return Vue pages directly from Laravel controllers.
- Tailwind: Utility-first CSS for styling.

Prerequisites

Before we start, ensure you have installed:

1. PHP and Composer
2. Node.js and NPM

To save hours of manual configuration, we will use Laravel Breeze. It creates the skeleton for this specific stack automatically.

Project 1: Demo Static Site

Setup

1. Create the project:

```
composer create-project laravel/laravel project1
cd project1
```

2. Install the Breeze starter kit (select Vue with Inertia when asked):

```
composer require laravel/breeze --dev
php artisan breeze:install
# Select: Vue -> Inertia -> No SSR -> PHPUnit (default)
```

3. Start the servers (keep two terminal windows open):
    - Terminal A: php artisan serve (Backend)
    - Terminal B: npm run dev (Frontend compiler)

The Shared Layout (NavBar)

In Inertia, we use Layouts to keep the navigation persistent while the content changes.

Create a file: resources/js/Layouts/MainLayout.vue

```
<script setup>
import { Link } from '@inertiajs/vue3';
</script>
```

```
<template>
  <div class="min-h-screen bg-gray-100 font-sans">
    <nav class="bg-indigo-600 text-white p-4">
      <div class="max-w-4xl mx-auto flex justify-between items-center">
        <div class="font-bold text-xl">MyProject</div>
        <div class="space-x-4">
          <Link href="/" class="hover:text-indigo-200">Home</Link>
          <Link href="/content" class="hover:text-indigo-200">Content</Link>
          <Link href="/about" class="hover:text-indigo-200">About</Link>
        </div>
      </div>
    </nav>

    <main class="max-w-4xl mx-auto mt-6 p-6 bg-white shadow rounded">
      <slot /> </main>
  </div>
</template>
```

Creating the Pages

Create three files in resources/js/Pages/.

1. Home.vue

```
<script setup>
import MainLayout from '@/Layouts/MainLayout.vue';
import { Head } from '@inertiajs/vue3';
</script>

<template>
  <Head title="Home" />
  <MainLayout>
    <h1 class="text-3xl font-bold text-gray-800">Welcome Home</h1>
    <p class="mt-4 text-gray-600">This is the landing page built with Laravel and Vue.</p>
  </MainLayout>
</template>
```

2. Content.vue

```
<script setup>
import MainLayout from '@/Layouts/MainLayout.vue';
import { Head } from '@inertiajs/vue3';
</script>

<template>
  <Head title="Content" />
  <MainLayout>
    <h1 class="text-3xl font-bold text-green-600">Our Content</h1>
```

```
    <div class="mt-4 p-4 bg-green-50 border border-green-200 rounded">
      <p>Here is some featured content using Tailwind utility classes.</p>
    </div>
  </MainLayout>
</template>
```

3. About.vue

```
<script setup>
import MainLayout from '@/Layouts/MainLayout.vue';
import { Head } from '@inertiajs/vue3';
</script>

<template>
  <Head title="About Us" />
  <MainLayout>
    <h1 class="text-3xl font-bold text-blue-600">About Us</h1>
    <p class="mt-4">We are learning the TALL stack step-by-step.</p>
  </MainLayout>
</template>
```

Define Routes

Open routes/web.php. We map URLs directly to the Vue components using Inertia::render.

```
<?php

use Illuminate\Support\Facades\Route;
use Inertia\Inertia;

// Route 1: Home
Route::get('/', function () {
  return Inertia::render('Home');
});

// Route 2: Content
Route::get('/content', function () {
  return Inertia::render('Content');
});

// Route 3: About
Route::get('/about', function () {
  return Inertia::render('About');
});
```

Test it: Go to http://127.0.0.1:8000. Click the links. Notice the page does not reload (the browser icon doesn't spin)? That is Inertia working.

Project 2: Form Processing App

Setup & Database

1. Create project2 (follow the same installation steps as Project 1).
2. Set up your .env file to connect to a database (SQLite is easiest for learning, or MySQL).
3. Create the database model and migration:

```
php artisan make:model Feedback -m
```

4. Open the migration file in database/migrations/xxxx_create_feedback_table.php:

```
public function up(): void
{
   Schema::create('feedback', function (Blueprint $table) {
      $table->id();
      $table->string('name');
      $table->string('email');
      $table->text('message');
      $table->timestamps();
   });
}
```

5. Run migration: php artisan migrate

The Controller

We need logic to handle showing the form and saving it.

```
php artisan make:controller FeedbackController
```

Open app/Http/Controllers/FeedbackController.php:

```
<?php

namespace App\Http\Controllers;

use App\Models\Feedback;
use Illuminate\Http\Request;
use Inertia\Inertia;

class FeedbackController extends Controller
{
   // Page 1: Show the form
   public function create() {
      return Inertia::render('Feedback/Create');
   }
```

```
    // Logic: Store the data
    public function store(Request $request) {
        // 1. Validate
        $validated = $request->validate([
            'name' => 'required|max:50',
            'email' => 'required|email',
            'message' => 'required|min:10'
        ]);

        // 2. Save
        Feedback::create($validated);

        // 3. Redirect
        return redirect('/success');
    }

    // Page 2: Success Message
    public function success() {
        return Inertia::render('Feedback/Success');
    }

    // Page 3: List all feedback (Admin)
    public function index() {
        return Inertia::render('Feedback/Index', [
            'feedbackList' => Feedback::latest()->get()
        ]);
    }

    // Page 4: Show single feedback detail
    public function show($id) {
        return Inertia::render('Feedback/Show', [
            'feedback' => Feedback::findOrFail($id)
        ]);
    }
}
```

Note: Allow mass assignment in app/Models/Feedback.php: protected $guarded = [];

The Routes

Open routes/web.php:

```
use App\Http\Controllers\FeedbackController;

Route::get('/', [FeedbackController::class, 'create']); // Home is the form
Route::post('/feedback', [FeedbackController::class, 'store']); // Post action
Route::get('/success', [FeedbackController::class, 'success']);
```

```
Route::get('/admin/feedback', [FeedbackController::class, 'index']);
Route::get('/admin/feedback/{id}', [FeedbackController::class, 'show']);
```

The Vue Pages (The Frontend)

Create a folder resources/js/Pages/Feedback.
1. Create.vue (The Form) This uses the useForm hook, which is crucial for handling form data and errors in Inertia.

```
<script setup>
import { useForm } from '@inertiajs/vue3';

// Initialize form data
const form = useForm({
  name: '',
  email: '',
  message: ''
});

const submit = () => {
  form.post('/feedback');
};
</script>

<template>
  <div class="max-w-md mx-auto mt-10 p-6 bg-white shadow-lg rounded-lg">
    <h1 class="text-2xl font-bold mb-6">Submit Feedback</h1>

    <form @submit.prevent="submit">
      <div class="mb-4">
        <label class="block text-gray-700">Name</label>
        <input v-model="form.name" type="text" class="w-full border p-2 rounded" />
        <div v-if="form.errors.name" class="text-red-500 text-sm">{{ form.errors.name }}</div>
      </div>

      <div class="mb-4">
        <label class="block text-gray-700">Email</label>
        <input v-model="form.email" type="email" class="w-full border p-2 rounded" />
        <div v-if="form.errors.email" class="text-red-500 text-sm">{{ form.errors.email }}</div>
      </div>

      <div class="mb-4">
        <label class="block text-gray-700">Message</label>
        <textarea v-model="form.message" class="w-full border p-2 rounded"></textarea>
        <div v-if="form.errors.message" class="text-red-500 text-sm">{{ form.errors.message }}</div>
      </div>
```

```
        <button type="submit" :disabled="form.processing" class="bg-blue-600 text-white px-4 py-2
rounded hover:bg-blue-700">
          Submit
        </button>
      </form>
    </div>
</template>
```

2. Success.vue

```
<script setup>
import { Link } from '@inertiajs/vue3';
</script>
<template>
  <div class="text-center mt-20">
    <h1 class="text-4xl text-green-600 font-bold">Thank You!</h1>
    <p class="mt-4">Your feedback has been received.</p>
    <Link href="/" class="mt-6 inline-block text-blue-500 underline">Send Another</Link>
    <br>
    <Link href="/admin/feedback" class="mt-2 inline-block text-gray-500 text-sm">View Admin
Panel</Link>
  </div>
</template>
```

3. Index.vue (Admin List) Notice how we receive feedbackList as a prop. This data came from the Laravel Controller.

```
<script setup>
import { Link } from '@inertiajs/vue3';

defineProps({
  feedbackList: Array
});
</script>

<template>
  <div class="max-w-2xl mx-auto mt-10">
    <h1 class="text-2xl font-bold mb-4">Admin: Feedback List</h1>

    <div v-for="item in feedbackList" :key="item.id" class="border-b p-4 hover:bg-gray-50">
      <div class="flex justify-between">
        <span class="font-bold">{{ item.name }}</span>
        <span class="text-gray-500 text-sm">{{ new Date(item.created_at).toLocaleDateString() }}</span>
      </div>
      <p class="truncate text-gray-600">{{ item.message }}</p>
      <Link :href="` /admin/feedback/${item.id}` " class="text-blue-500 text-sm">View Details</Link>
```

```
    </div>
  </div>
</template>
```

4. Show.vue (Detail View)

```
<script setup>
import { Link } from '@inertiajs/vue3';

defineProps({
  feedback: Object
});
</script>

<template>
  <div class="max-w-xl mx-auto mt-10 p-6 border rounded shadow">
    <Link href="/admin/feedback" class="text-gray-500 text-sm mb-4 inline-block">&larr; Back to
list</Link>

    <h1 class="text-3xl font-bold">{{ feedback.name }}</h1>
    <p class="text-blue-600 mb-6">{{ feedback.email }}</p>

    <div class="bg-gray-100 p-4 rounded">
      {{ feedback.message }}
    </div>
  </div>
</template>
```

Add Flash Message

The Backend (Laravel)

We need to tell Laravel to send a specific message along with the redirect.

Open app/Http/Controllers/FeedbackController.php. Update the store method:

```
public function store(Request $request) {
  $validated = $request->validate([
    'name' => 'required|max:50',
    'email' => 'required|email',
    'message' => 'required|min:10'
  ]);

  Feedback::create($validated);
```

```
    // CHANGE: We redirect to the form (or index) WITH a flash message
    // instead of a dedicated success page. This is a common pattern.
    return redirect('/')->with('success', 'Feedback submitted successfully!');
}
```

The Glue (Inertia Middleware)

By default, Inertia doesn't automatically send session flash data to Vue. We must explicitly "share" it.

Open app/Http/Middleware/HandleInertiaRequests.php. Look for the share method and update it:

```
public function share(Request $request): array
{
    return [
        ...parent::share($request),
        'auth' => [
            'user' => $request->user(),
        ],
        // ADD THIS SECTION
        'flash' => [
            'success' => fn () => $request->session()->get('success'),
            'error' => fn () => $request->session()->get('error'),
        ],
    ];
}
```

The Frontend (Vue)

Now, every Vue page has access to $page.props.flash.success. Let's create a global component to display it.

1. Create the Flash Component Create resources/js/Components/FlashMessage.vue:

```
<script setup>
import { usePage } from '@inertiajs/vue3';
import { computed } from 'vue';

const page = usePage();

// We make this computed so it reacts automatically when the page prop changes
const message = computed(() => page.props.flash.success);
</script>

<template>
    <div v-if="message" class="bg-green-100 border border-green-400 text-green-700 px-4 py-3 rounded
relative mb-4" role="alert">
        <strong class="font-bold">Success! </strong>
```

```
    <span class="block sm:inline">{{ message }}</span>
  </div>
</template>
```

2. Add it to your Layout Now, drop this component into the MainLayout.vue we created in Project 1 (or add it directly to Create.vue in Project 2 if you didn't use a layout there).

If you are using the layout approach:

```
<script setup>
import { Link } from '@inertiajs/vue3';
import FlashMessage from '@/Components/FlashMessage.vue'; // Import it
</script>

<template>
  <div class="min-h-screen bg-gray-100 font-sans">
    <nav class="bg-indigo-600 text-white p-4">
      </nav>

    <main class="max-w-4xl mx-auto mt-6 p-6 bg-white shadow rounded">
      <FlashMessage />

      <slot />
    </main>
  </div>
</template>
```

Verify

1. Go to your form page (/).
2. Fill it out and submit.
3. The page should reload (actually an Inertia visit), and the green "Feedback submitted successfully!" banner will appear at the top.
4. Refresh the page manually. The message will disappear (because it was "flashed" for only one request).