Creating a layout using Laravel Blade templating engine

Creating a layout using Laravel Blade templating engine involves setting up a master layout file and then extending it in your other view files. This promotes code reusability and a consistent design across your application.

Step-by-Step Guide

1. Create the Master Layout File

First, create a master layout file that will serve as the template for your other views. This file typically includes the HTML head, common scripts, and styles.

Create a file named `app.blade.php` in the `resources/views/layouts` directory.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@yield('title', 'My Laravel App')</title>
    <link rel="stylesheet" href="{{ asset('css/app.css') }}">
    @yield('styles')
</head>
<body>
    <header>
        <!-- Include your header content here -->
    </header>

    <nav>
        <!-- Include your navigation content here -->
```

```
    </nav>

    <main>
        @yield('content')
    </main>

    <footer>
        <!-- Include your footer content here -->
    </footer>

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

2. Create Child Views

Next, create view files that extend the master layout. For example, create a `home.blade.php` file in the `resources/views` directory.

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

@section('title', 'Home Page')

@section('styles')
    <link rel="stylesheet" href="{{ asset('css/home.css') }}">
@endsection

@section('content')
    <h1>Welcome to My Laravel App</h1>
    <p>This is the home page content.</p>
@endsection

@section('scripts')
    <script src="{{ asset('js/home.js') }}"></script>
@endsection
```

3. Set Up Routes and Controllers

To display the view, set up a route and controller. In the `routes/web.php` file, add the following route:

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

Alternatively, you can create a controller and use it to return the view.

First, create a controller using Artisan command:

```
php artisan make:controller HomeController
```

Then, update the `HomeController` to return the `home` view:

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller
{
    public function index()
    {
        return view('home');
    }
}
```

Update the route in `routes/web.php`:

```
use App\Http\Controllers\HomeController;

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

4. Create Additional Views

You can create additional views following the same structure. For instance, create an `about.blade.php`:

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

@section('title', 'About Us')

@section('styles')
    <link rel="stylesheet" href="{{ asset('css/about.css') }}">
```

```
@endsection

@section('content')
    <h1>About Us</h1>
    <p>This is the about us page content.</p>
@endsection

@section('scripts')
    <script src="{{ asset('js/about.js') }}"></script>
@endsection
```

Update the routes to include the new view:

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

Or using a controller:

```
Route::get('/about', [HomeController::class, 'about']);
```

And update the `HomeController`:

```
public function about()
{
    return view('about');
}
```

5. Run and Test

Ensure you have the Laravel development server running:

```
php artisan serve
```

Visit `http://localhost:8000` to see the home page and `http://localhost:8000/about` to see the about page.

Summary

By following these steps, you create a consistent layout for your Laravel application using Blade templating. This setup allows you to extend the master layout for various views, ensuring reusability and maintaining a clean and organized codebase.

Revision #1
Created 31 July 2024 07:05:31 by AKSHAY D JOSHI
Updated 31 July 2024 07:05:51 by AKSHAY D JOSHI