Laravel - Views

Topics : Laravel
Written on December 22, 2023

In Laravel, views are used to render HTML content and display it to the user. Views provide a way to separate the application's logic from its presentation layer. Here's a brief overview of working with views in Laravel:

Creating Views:

Views in Laravel are typically stored in the `resources/views` directory. You can create a new view file with the `.blade.php` extension. For example, you might have a file named `welcome.blade.php`:

```
<!-- resources/views/welcome.blade.php -->
<!DOCTYPE html>
<html>
<head>
    <title>Welcome</title>
</head>
<body>
    <h1>Hello, {{ $name }}</h1>
</body>
</html>
```

Passing Data to Views:

You can pass data to views using the `view` function or the `with` method:

```
// Using the view function
return view('welcome', ['name' => 'John']);

// Using the with method
return view('welcome')->with('name', 'John');
```

In the above examples, the `$name` variable is passed to the `welcome.blade.php` view.

Blade Templating:

Laravel uses the Blade templating engine to simplify writing views. Blade allows you to use template inheritance, control structures, and more:

```
<!-- resources/views/layouts/app.blade.php -->
```

```
<!DOCTYPE html>
<html>
<head>
    <title>@yield('title', 'Default Title')</title>
</head>
<body>
    <div class="container">
        @yield('content')
    </div>
</body>
</html>

<!-- resources/views/welcome.blade.php -->
@extends('layouts.app')

@section('title', 'Welcome')

@section('content')
    <h1>Hello, {{ $name }}</h1>
@endsection
```

In the above example, the `welcome.blade.php` view extends the `app.blade.php` layout and provides content for the `title` and `content` sections.

Including Sub-Views:

You can include other views within a view using the `@include` directive:

```
<!-- resources/views/welcome.blade.php -->
@include('partials.header')

<h1>Hello, {{ $name }}</h1>

@include('partials.footer')
```

View Composer:

View composers allow you to bind data to a view each time it is rendered. This is useful for sharing data across multiple views:

```
// In a service provider or a dedicated service provider class
View::composer('welcome', function ($view) {
    $view->with('name', 'John');
});
```

View Caching:

To improve performance, you can cache views using the `view:cache` Artisan command:

```
php artisan view:cache
```