LARAVEL 5.8

blade

09/09/2019

INTRODUCCIÓN

Blade es el motor de plantillas simple pero potente provisto con Laravel. A diferencia de otros motores de plantillas PHP populares, **Blade** no le impide usar código PHP en sus vistas. De hecho, todas las vistas de Blade se compilan en código PHP y se almacenan en caché hasta que se modifican, lo que significa que **Blade** agrega esencialmente cero sobrecarga a su aplicación. Los archivos de vista **Blade** usan la extensión de archivo **.blade.php** y generalmente se almacenan en el directorio **resources/views**.

DEFINIENDO UN LAYOUT:

Dos de los principales beneficios de usar **Blade** son la herencia de plantillas y las secciones. Como la mayoría de las aplicaciones web mantienen el mismo diseño general en varias páginas, es conveniente definir este diseño como una sola vista **Blade**:

```
<html>
  <head>
    <title>App Name - @yield('title')</title>
  </head>
  <body>
    @section('sidebar')
      This is the master sidebar.
    @show
```

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

EXTENDER UN LAYOUT:

Al definir una vista secundaria, use la directiva de Blade **@extends** para especificar qué layout debe "heredar" la vista secundaria. Las vistas que extienden un layout **Blade** pueden inyectar contenido en las secciones del layout utilizando las directivas **@section**. el contenido de estas secciones se mostrará en el diseño usando **@yield**:

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

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

@section('sidebar')
  @parent
  <p>This is appended to the master sidebar.</p>
@endsection

@section('content')
  <p>This is my body content.</p>
@endsection
```

MOSTRAR DATOS:

Puede mostrar los datos enviados a sus vistas **Blade** envolviendo la variable entre llaves. Por ejemplo:

```
Route::get('greeting', function () {
    return view('welcome', ['name' => 'Samantha']);
});
```

Puede mostrar el contenido de la variable **name** así:

```
Hello, {{ $name }}.
```

ESTRUCTURAS DE CONTROL:

Además de la herencia de plantillas y la visualización de datos, **Blade** también proporciona accesos directos convenientes para estructuras de control PHP comunes, como declaraciones condicionales y bucles. Estos accesos directos proporcionan una forma muy limpia y concisa de trabajar con estructuras de control de **PHP**.

CONDICIONALES:

Puede construir sentencias if utilizando las directivas **@if**, **@elseif**, **@else** y **@endif**. Estas directivas funcionan de manera idéntica a sus contrapartes **PHP**:

```
@if (count($records) === 1)
    I have one record!
@elseif (count($records) > 1)
    I have multiple records!
@else
    I don't have any records!
@endif
```

Para mayor comodidad, **Blade** también proporciona una directiva **@unless**:

```
@unless (Auth::check())
    You are not signed in.
@endunless
```

Además de las directivas condicionales ya discutidas, las directivas **@isset** y **@empty** pueden usarse como atajos convenientes para sus respectivas funciones **PHP**:

```
@isset($records)
    // $records is defined and is not null...
@endisset

@empty($records)
    // $records is "empty"...
@endempty
```

DIRECTIVAS DE AUTENTICACIÓN:

Las directivas **@auth** y **@guest** pueden usarse para determinar rápidamente si el usuario actual está autenticado o es un invitado:

```
@auth
    // The user is authenticated...
@endauth

@guest
    // The user is not authenticated...
@endguest
```

SWITCH:

Las sentencias **switch** se pueden construir utilizando las directivas **@switch**, **@case**, **@break**, **@default** y **@endswitch**:

```
@switch($i)
    @case(1)
        // First case...
        @break
    @case(2)
        // Second case...
        @break
    @default
        // Default case...
@endswitch
```

CICLOS:

Además de las declaraciones condicionales, **Blade** proporciona directivas simples para trabajar con estructuras de bucle **PHP**:

```
@for ($i = 0; $i < 10; $i++)
    The current value is {{ $i }}
@endfor

@foreach ($users as $user)
    <p>This is user {{ $user->id }}</p>
@endforeach

@forelse ($users as $user)
```

```
    <li>{{ $user->name }}</li>
@empty
    <p>No users</p>
@endforelse

@while (true)
    <p>I'm looping forever.</p>
@endwhile
```

COMENTARIOS:

Blade también le permite definir comentarios en sus vistas. Sin embargo, a diferencia de los comentarios **HTML**, los comentarios **Blade** no están incluidos en el **HTML** devuelto por su aplicación:

```
{{-- This comment will not be present in the rendered HTML --}}
```

PHP:

En algunas situaciones, es útil incrustar código **PHP** en sus vistas. Puede usar la directiva Blade **@php** para ejecutar un bloque de **PHP** plano dentro de su plantilla:

```
@php
    //
@endphp
```