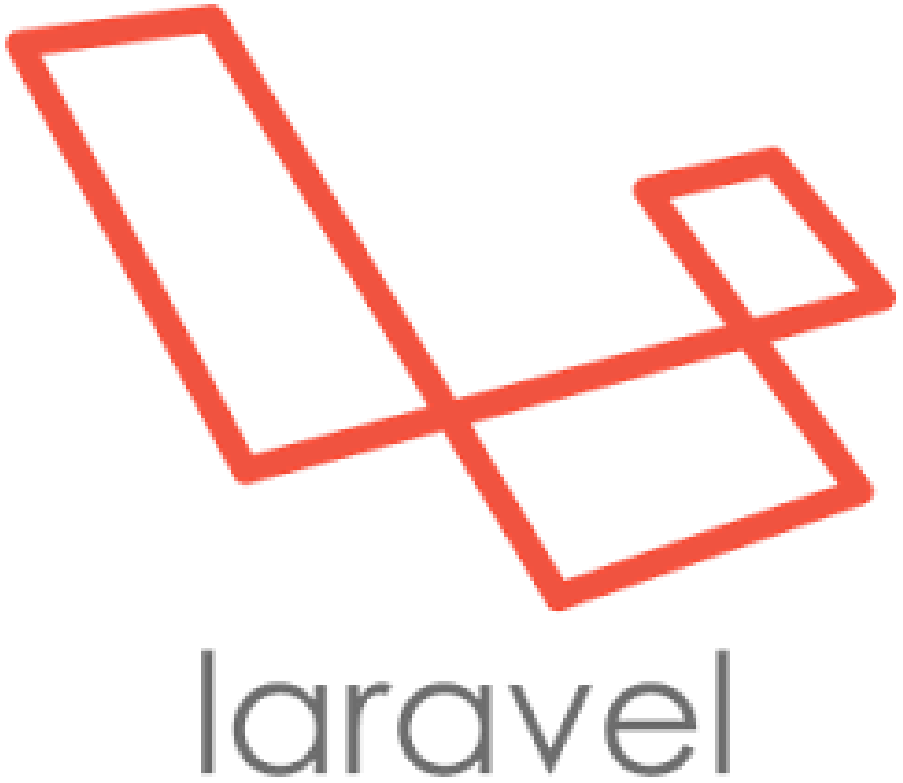
Curso de Laravel



Capítulo 3: Layout

Laravel: Plantillas

Blade

- Laravel utiliza Blade para la definición de plantillas y vistas
- Blade es una librería para generar vistas
- Permite generar plantillas
- Permite generar layouts
- Permite sustituir secciones con datos recibidos por el controlador y cargarlas en la vista de forma dinámica
- Tiene la capacidad de indexar otras plantillas o generar herencias

Laravel: Plantillas

Archivos Blade

- Todas las plantillas Blade deben tener la extención .blade.php pero no es necesario en una llamada

- Ejemplo
  Si utilizamos una llamada a la vista view(‘home’)

  Puede tomar la plantilla home.blade.php

  También las vistas pueden ser llamadas a subdirectorios con un punto, ejemplo
  view(‘user.listar’) la vista recogerá un archivo ubicado en resources/views/user/listar.blade.php

Laravel: Plantillas

Mostrar Datos

- Blade usa como método básico para mostrar variables las llaves dobles {{ }}, dentro de ellas se insertan las variables o funciones del contenido a mostrar

```
Hola {{ $name }}

La hora actual es {{ time() }}
```

Laravel: Plantillas

Mostrar Datos

- Blade como método de seguridad escapa automáticamente caracteres de tipo inyección de código o ataques de tipos XSS, si no deseamos que eso ocurra, se puede definir la salida de la siguiente forma:

```
Hola {{!! $name !!}}
```

Laravel: Plantillas

Mostrar una variable sólo si existe

- Para mostrar una variable solo si existe se realiza de la siguiente forma:

```
{{ isset($name) ? $name: 'No existe' }}
```

- O un atajo de Blade

```
{{ $name or 'No existe' }}
```

Laravel: Plantillas

Comentarios

- Para añadir comentarios los símbolos son los siguientes
  {{-- --}}

```
{{-- Esto es un comentario, no se mostrará en la vista --}}
```

Laravel: Plantillas

Estructuras de control

- Las estructuras de control if se definen de la siguiente forma

```
@if( count($clientes) ===0 )
   No hay clientes registrados
@elseif (count($clientes)>1)
   Hay {{ count($clientes) }} Clientes
@else
   Hay un solo cliente
@endif
```

Laravel: Plantillas

Iteraciones

- El siguiente ejemplo muestra iteraciones for - while

```
@for ($i=0; $i<10; $i++)
   El valor actual es: {{ $i }}
@endfor

@while (true)
   <p>Esta iteración no terminará </p>
@endwhile

@foreach($clientes as $cliente)
   <p>Cliente: {{ $cliente->nombre }}</p>
@endforeach
```

Laravel: Plantillas

Incluir una plantilla dentro de otra plantilla

- La función para incluir plantillas es @include

```
@include('nombre_vista')
```

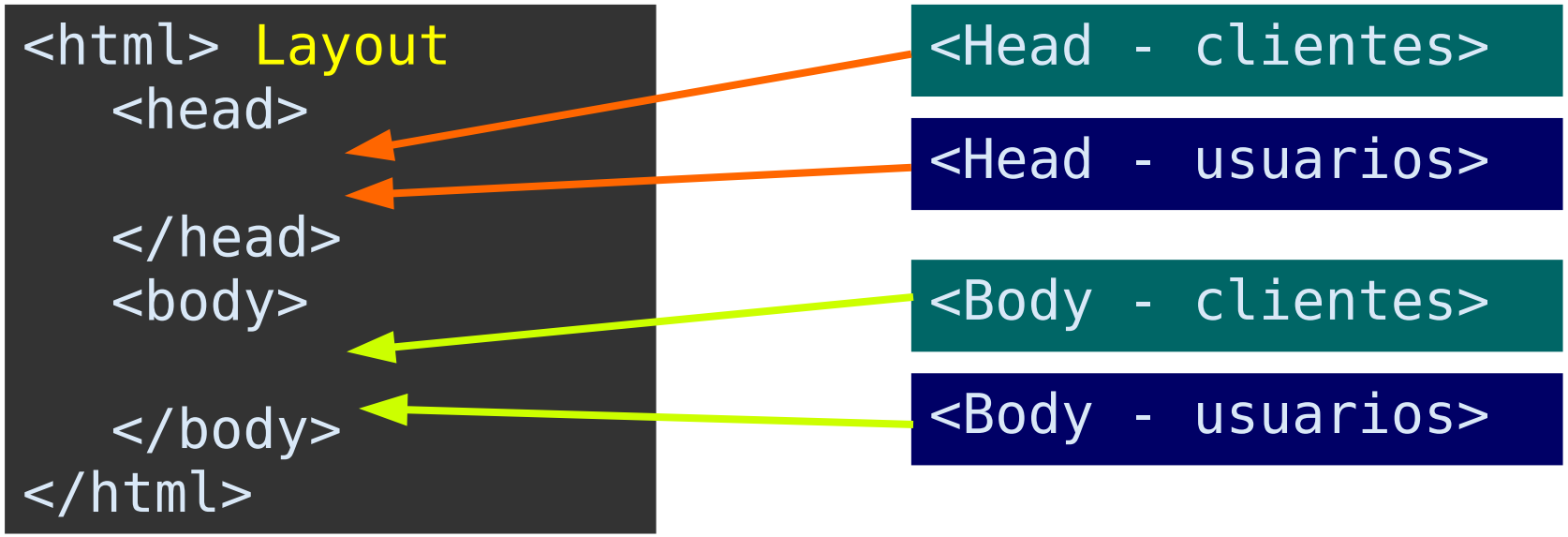
- Podemos enviar datos a la vista de la siguiente forma:

```
@include('nombre_vista',array('clientes'=>'data')
```

Laravel: Plantillas

Layouts

- Blade puede utilizar capas o layouts para crear una estructura HTML separada por componentes



Laravel: Plantillas

Layout

- Ejemplo: se crea un directorio layout con una plantilla maestra “master”

```
resources/views/layouts/master.blade.php
```

Laravel: Plantillas

Código de master.blade.php

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>
      @section('title')
        Título de la web
      @show
    </title>
  </head>
  <body>
      @yield('content')
  </body>
</html>
```

Laravel: Plantillas

Plantilla/Vista extendida

- Se crea una nueva plantilla extendida que extiende las secciones del archivo master

```
@extends('layouts.master')
@section('title')
   @parent
  <p>Título de la vista</p>
@endsection
@section('content')
      Contenido de la vista
@endsection
```

Laravel: Plantillas

Resumen Layouts de vistas en Blade

- @extends: usa como herencia una plantilla “padre” con contenido general donde se modificarán sus secciones.
- @section: Permite añadir contenido en las plantillas “hijas” en la sección definida por la plantilla “padre”
- @parent: Carga en la posición indicada el contenido definido por el padre para esa sección
- @yield: permite que las plantillas “hijas” añadan contenido en esa sección. También se puede establecer contenido por defecto:

```
@yield('section','contenido por defecto')
```

Ejercicio

Realizar lo siguiente

- Descargar e instalar Bootstrap en el proyecto desde http://getbootstrap.com
- Descomprimir e instalar la librería en public/assets/bootstrap (crear carpeta asset/bootstrap)
- Generar una plantilla base de nombre master.blade.php al interior de un directorio “/resources/views/layouts” copiando el código del siguiente sitio: https://getbootstrap.com/docs/3.3/examples/starter-template/
- Cambiar las rutas por las que corresponden en el proyecto (css y js)

```
{{ url('/assets/bootstrap/css/bootstrap.min.css')   }}
{{ url('/assets/bootstrap/js/bootstrap.min.js')  }}
```

Ejercicio

Realizar lo siguiente

- Reemplazar el contenido al interior de la etiqueta por lo siguiente

```
@include('sections.navbar')
```

- Al interior del div de clase “container”, reemplazar el contenido interior por lo siguiente

```
@yield('content')
```

Ejercicio

Realizar lo siguiente

- Crear una vista en el archivo “sections/navbar” con nombre navbar.blade.php y generar un menú.
- Crear un archivo home.blade.php y extendiéndose de la plantilla master
- Vincular el archivo de ruta para la petición “/” a la vista home.blade.php