
Laravel

SEARCH

Documentation

Laracasts

Prologue

Release Notes

Upgrade Guide

Contribution Guide

API Documentation

Getting Started

Installation

Configuration

Directory Structure

Request Lifecycle

Dev Environments

Homestead

Valet

Core Concepts

Service Container

Service Providers

Facades

Contracts

The HTTP Layer

Routing

Middleware

CSRF Protection

Controllers

Requests

Responses

Views

Session

Validation

Frontend

Blade Templates

Localization

Frontend Scaffolding

Compiling Assets

Security

Routing

- # **Basic Routing**
- # **Route Parameters**
 - # Required Parameters
 - # Optional Parameters
 - # Regular Expression Con
- # **Named Routes**
- # **Route Groups**
 - # Middleware
 - # Namespaces
 - # Sub-Domain Routing
 - # Route Prefixes
- # **Route Model Binding**
 - # Implicit Binding
 - # Explicit Binding
- # **Form Method Spoofing**
- # **Accessing The Current Rou**

Basic Routing

The most basic Laravel routes sin
expressive method of defining rou

```
Route::get('foo', function ()  
    return 'Hello World';  
});
```

The Default Route Files

All Laravel routes are defined in y
files are automatically loaded by
your web interface. These routes

LARAVEL: RUTAS, VISTAS Y PLANTILLAS BLADE

DISEÑO DE SISTEMAS SOFTWARE

Contenido

1. Rutas
2. Vistas
3. Plantillas Blade

Laravel: rutas y vistas

RUTAS

Rutas

- Las rutas de la aplicación se indican en los ficheros:

`routes/`

`api.php`

→ Rutas para API

`channels.php`

→ Suscripciones a eventos

`console.php`

→ Rutas para consola

`web.php`

→ Rutas para acceso Web

- Cualquier ruta no definida → Error 404

Rutas

- Para cada ruta se define:
 - Método HTTP de la petición: GET, POST, PUT, DELETE
 - Ruta URL (sin el dominio, el cual está ya definido en la configuración: `config/app.php`)
 - Respuesta de una petición a esa ruta
- Podremos devolver tres tipos de respuestas:
 - Un valor (por ejemplo una cadena)
 - Una vista
 - Enlazar con un método de un controlador

Rutas

- Ejemplo de ruta para peticiones tipo GET a la URL raíz de nuestra aplicación web:

```
Route::get('/', function() { // URL raíz de la app
    return '¡Hola mundo!';
});
```

- Ejemplo de respuesta a peticiones tipo POST para la ruta “foo/bar”:

```
Route::post('foo/bar', function() {
    return '¡Hola mundo!';
});
```

- De la misma forma podemos definir rutas para PUT o DELETE

Rutas

- También podemos definir una ruta que responda a varios tipos de peticiones:

```
Route::match(['GET', 'POST'], '/', function() {  
    return '¡Hola mundo!';  
});
```

- O que responda a todas:

```
Route::any('foo', function() {  
    return '¡Hola mundo!';  
});
```

Rutas con parámetros

- Para añadir parámetros a las rutas se indican entre llaves {}:

```
Route::get('user/{id}', function($id) {  
    return 'User ' . $id;  
});
```

- En este caso el parámetro sería obligatorio
- Si queremos indicar que un parámetro es opcional tenemos que añadir el símbolo ?:

```
Route::get('user/{name?}', function($name = null) {  
    return $name;  
});
```

- Para generar un enlace a una ruta usamos el método:

```
$url = url('foo');
```


Rutas con nombre

- Podemos asignar nombre a las rutas:

```
Route::get('user/{id}', function($id) {  
    //  
})->name('user.show');
```

- Se puede generar un enlace a una ruta usando su nombre

```
$url = route('user.show', ['id' => 1]);
```

Laravel: rutas y vistas

VISTAS

Vistas

- Presentan el resultado de forma visual al usuario, el cual podrá interactuar con él y volver a realizar una petición
- Permiten separar toda la parte de presentación de resultados de la lógica (controladores) y de la base de datos (modelos)
- No realizan ningún tipo de consulta ni procesamiento de datos, simplemente reciben datos y los prepararán para mostrarlos
- Se almacenan en la carpeta `resources/views` como ficheros PHP

Vistas

- Podrán contener:
 - Código HTML
 - Assets (CSS, imágenes, Javascript, etc. que estarán en la carpeta `public`)
 - Algo de código PHP (o plantillas con Blade) para presentar los datos de entrada como un resultado HTML

Vistas

- Ejemplo de vista guardada en `resources/views/home.php`:

```
<html>
  <head>
    <title>Mi Web</title>
  </head>
  <body>
    <h1>¡Hola <?php echo $name; ?>!</h1>
  </body>
</html>
```

- Para asociarla con una ruta, en el fichero `routes/web.php` añadimos:

```
Route::get('/', function() {
    return view('home', ['name' => 'John']);
});
```

Vistas

- Al construir una vista podemos pasarle parámetros de varias formas:

```
view('home', ['name' => 'Pedro', 'age' => 18]);  
  
view('home')->with('name', 'Javi')->with('age', 18);
```

- Para hacer referencia a una vista que está en una subcarpeta:

```
"resources/views/user/profile" → "user.profile"
```

- Por ejemplo:

```
Route::get('user/profile/{id}', function($id) {  
    $user = // Cargar usuario a partir de $id  
    return view('user.profile', ['user' => $user]);  
});
```

HTML, CSS y plantillas Blade

PLANTILLAS BLADE

Plantillas Blade

- Laravel utiliza Blade para la definición de plantillas en las vistas
- Permite realizar todo tipo de operaciones con los datos: sustitución de variables o de secciones, herencia entre plantillas, definición de *layouts*, etc.
- Los ficheros de Blade tienen que tener la extensión `.blade.php`
- Para hacer referencia a una vista que utiliza Blade no tenemos que indicar la extensión, por ejemplo:

`"home.blade.php"` → `"view('home');"`

Comentarios / Mostrar valores

- **Comentarios con** `{{-- comentario --}}`:

```
{{-- Este comentario no se mostrará en HTML --}}
```

- **Para mostrar datos usamos** `{{ var / función() }}`:

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

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

```
{{-- Si no queremos escapar los datos (CUIDADO): --}}
```

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

- **Operador ternario:**

```
{{ isset($name) ? $name : 'Default' }}
```

```
{{ $name ?? 'Default' }}
```

Estructuras de control

- Estructuras de control `if`:

```
@if( count($users) === 1 )  
    Solo hay un usuario!  
@elseif (count($users) > 1)  
    Hay muchos usuarios!  
@else  
    No hay ningún usuario :(  
@endif
```

- Estructuras de control `unless`:

```
@unless (Auth::check())  
    Usuario no identificado  
@endunless
```

Estructuras de control

- Bucles:

```
@for ($i = 0; $i < 10; $i++)  
    El valor actual es {{ $i }}  
@endfor  
  
@while (true)  
    <p>Soy un bucle while infinito!</p>  
@endwhile
```

- Dentro de un bucle se puede usar @continue y @break

Estructuras de control

- Más bucles:

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

@forelse ($users as $user)
    <li>{{ $user->name }}</li>
@empty
    <p>No hay usuarios</p>
@endforelse
```

Sub-vistas

- Incluir una plantilla dentro de otra plantilla:

```
@include('view_name')
```

```
{{-- También podemos pasarle un array de datos  
    como segundo parámetro: --}}
```

```
@include('view_name', array('some'=>'data'))
```

Componentes

- Los componentes son pequeñas sub-vistas parametrizables
- Deben estar en la carpeta `/resources/views/components`

```
<!-- /resources/views/components/alert.blade.php -->  
<div class="alert alert-danger">  
    {{ $slot }}  
</div>
```

- Al incluir un componente podemos incluir contenido que se mostrará en el lugar de la variable `$slot`

```
<x-alert>  
    <strong>Whoops!</strong> Something went wrong!  
</x-alert>
```

Componentes

- Se pueden definir slots adicionales

```
<!-- /resources/views/components/alert.blade.php -->  
<div class="alert alert-danger">  
    <div class="alert-title">{{ $title }}</div>  
    {{ $slot }}  
</div>
```

- Para asignarles valor se usa `<x-slot name='nombre'>`

```
<x-alert>  
    <x-slot name='title'>  
        Forbidden  
    </x-slot>  
    You are not allowed to access this resource!  
</x-alert>
```

Layouts

- Definición de un Layout

```
<!-- /resources/views/layouts/master.blade.php -->
<html>
  <head>
    <title>@yield('title')</title>
  </head>
  <body>
    @section('menu')
      Contenido del menu
    @show
    <div class="container">
      @yield('content')
    </div>
  </body>
</html>
```


Layouts

- Para extender una plantilla

```
@extends('layouts.master')
@section('title', 'Título de la página')
@section('menu')
    @parent
    <p>Este contenido es añadido al menú principal.</p>
@endsection
@section('content')
    <p>Este apartado aparecerá en la sección
        "content".</p>
@endsection
```