

# Laravel - Views

Generar vistas empleando Blade

# Views

Las vistas proporcionan una forma conveniente de **colocar el código HTML en archivos separados**.

Las vistas **separan la lógica del controlador** o de la aplicación **de la lógica de presentación** y se almacenan en el directorio **resources/views**.

Al usar Laravel, las plantillas de vista suelen estar escritas con el lenguaje de plantillas **Blade**.



# Views - Ejemplo sencillo

*<!-- View almacenada en resources/views/**greeting.blade.php** -->*

*<html>*

*<body>*

*<h1>Hola, {{ \$name }}</h1>*

*</body>*

*</html>*



# Views - Ejemplo sencillo

Para cargar la vista anterior se puede configurar, en web.php, la ruta siguiente:

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




# Crear una vista

Puedes crear una vista **colocando un archivo con la extensión .blade.php en el directorio resources/views** de tu aplicación o utilizando el **comando Artisan make:view**:

***php artisan make:view greeting***

La extensión **.blade.php** indica al framework que el archivo **contiene una plantilla Blade**. Las plantillas Blade incluyen HTML y directivas de Blade, que permiten mostrar valores fácilmente, crear estructuras condicionales "if", iterar sobre datos y mucho más.



# Vistas en subcarpetas

Las vistas también pueden estar anidadas dentro de subdirectorios, dentro del directorio resources/views. Se puede utilizar la **notación de "punto" ( . ) para hacer referencia** a la ruta de las vistas.

Por ejemplo, si tu vista se encuentra en resources/views/admin/profile.blade.php, puedes traerla desde una de las rutas (en web.php) o controladores de tu aplicación mediante:

```
return view('admin.profile', $data);
```

Nota: estos directorios **NO deben contener .** en su nombre.



# Pasar datos a las vistas

Como se ha visto en el ejemplo previo, se puede pasar datos mediante un array.

```
return view('greetings', ['name' => 'Juan Luis', 'occupation' => 'informático']);
```

Al pasar información de esta manera, los datos deben ser un **array con pares clave => valor**.

Después de proporcionar datos a una vista, se puede **acceder a cada valor dentro de la vista usando las claves del array**, por ejemplo:

```
<?php echo $name; ?>
```



# Pasar datos a las vistas - Método WITH

Como alternativa a pasar un array completo de datos a la función view, se puede utilizar el **método 'with' para agregar datos individuales a la vista.**

with() devuelve una instancia del objeto de vista, lo que **permite encadenar** métodos antes de retornarla:

```
return view('greeting')  
    ->with('name', Juan Luis)  
    ->with('occupation', 'informático');
```





# Trabajar con datos pasados

*<!-- View almacenada en resources/views/greeting.blade.php -->*

*<!-- Laravel reemplazará {{ \$name }} y {{ \$occupation }}  
con los valores proporcionados en el array. -->*

*<html>*

*<body>*

*<h1>Hola, {{ \$name }}</h1>*

*<p>Tu ocupación es: {{ \$occupation }}</p>*

*</body>*

*</html>*



# Optimizar las vistas

Con Blade, **las vistas se compilan bajo demanda.**

Cuando se ejecuta una solicitud que renderiza (carga) una vista, **Laravel verifica si existe una versión compilada** de la vista.

Si el archivo existe, Laravel comprobará si la vista ha sido modificada recientemente, en comparación con la versión compilada.

**Si la vista compilada no existe o la versión sin compilar ha cambiado, Laravel recompilará la vista.**



# Optimizar las vistas

La compilación de vistas durante la solicitud puede tener un pequeño impacto en el rendimiento.

Por ello, Laravel proporciona el **comando Artisan view:cache** para **pre-compilar todas las vistas** utilizadas por la aplicación. Para mejorar el rendimiento, se puede ejecutar este comando como parte del proceso de despliegue:

***php artisan view:cache***

Por el contrario, para limpiar la cache de vistas, se puede usar el siguiente comando:

***php artisan view:clear***

