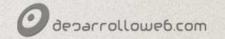




# Desarrollo de aplicaciones web con Laravel 5.1

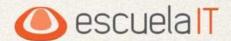
Clase 8. Plantillas con Blade y Elixir Carlos Ruiz Ruso · @micromante

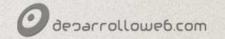




## Recordando

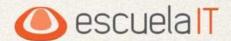
**Conceptos iniciales** 

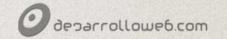




## Respuestas de tipo estándar

- Si no generamos una vista el formato sera texto plano.
- Si generamos una respuesta con Response::json la salida sera de tipo JSON
- Podemos hacer la respuesta en una funcion closure en las rutas o bien desde un controlador.
- En la clase de hoy aprenderemos a crear vistas y generar las respuestas con las variables.... vamos!





#### ¿Qué es una vista?

- Plantilla
- Extension .php o .blade.php
- Estan ubicadas en la carpeta view
- Son la parte visual de nuestra aplicación
- Podemos añadir código php en las vistas, pero usaremos blade para evitar hacer esto.
- Sintaxis limpia y facil de entender
- Mezcla php y html

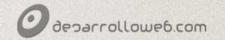




#### Caracteristicas Blade

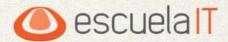
- Las variables estan rodeadas por {{...}} == <? php echo ... ?>
- No es necesario usar ';'
- Podemos usar estructuras de control tipo if, for, while...
- Podemos generar contenedores con
  - @section
  - @yield
  - @includes
- Es recomendable crear carpetas para cada modelo para organizar las vistas. Tambien creamos la carpeta layout.

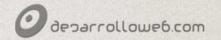




## ¿Qué es un layout?

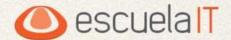
- El layout es la forma que tendríamos de agrupar todas nuestras vistas en una vista padre.
- Los layouts se pueden editar o cambiar en cualquier momento haciendo cargar un layout con unas vistas.
- Trabajamos siempre con la extensión blade para usar las funcionalidades del motor de plantillas. nombre.blade.php
- Veamos un ejemplo gráfico, siguiente!

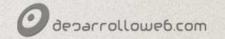




## ¿Qué es un layout?

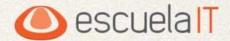
Layo	ıt Master
View	extends layout.master
Se	cions, yield, includes

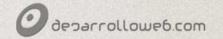




## @yield

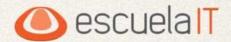
- Generamos contenedores dentro del layout donde podremos renderizar otras subvistas.
- Estas vistas heredan del layout
- Creamos un yield o sección
  - @yield('titulo')
- No podemos generar contenido por defecto.

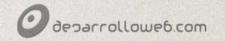




#### @section

- Generamos contenedores dentro del layout donde podremos renderizar otras subvistas.
- Estas vistas heredan del layout
- Creamos una sección
  - @section('contenido')
  - Contenido test //Solo se muestras si no es sobreescrito por otra
  - vista
  - @show

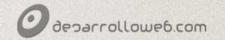




## Creamos vista y heredamos de un layout

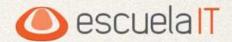
```
@extends('layout.base')
@section('titulo')
     @parent
     Contenido de ejemplo para el titulo
@stop
@section('contenido')
     @parent //Si queremos mostrar el contenido base
     Ahora ya mi contenido {{ $variable }}...
@stop
```

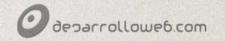




## Estructuras de control y traducción

- @if .... @elseif ... @else ... @endif
- @for(.....) .... @endfor
- @foreach(....) .... @endforeach
- @while(....) .... @endwhile
- @include('usuarios.footer', \$params) //Incluir subvistas, estas no heredan de layouts solo es php/html
- @lang('language.line') o @choice('language.line',1)





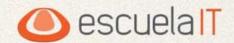
#### Devolver la vista en el controller

return view('usuarios.listado',\$data\_array); //Desde el controller

\$view = view('usuarios')->with('name', 'Victoria'); //Magic method

\$view = view('listado', \$data); //Desde el controller

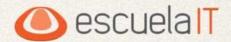
if (view()->exists('emails.customer')){ ... } //Existe la vista





## **DEMO**

Generamos nuestras vistas y probamos lo aprendido



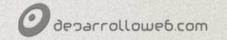


#### **Paginación**

- En nuestro controlador
  - \$users = User::paginate(10);
  - \$users = User::where(....)->paginate(10);

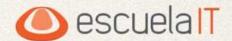
- En nuestra vista
  - {{!! \$users->render() !!}}
  - Podemos pasarle una vista para personalizar la paginación.
  - Podemos personalizar la url \$users->setPath('custom/url');
  - {!! \$users->appends(['sort' => 'votes'])->render() !!}

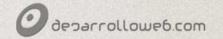




## Helpers paginación

- \$results->count()
- \$results->currentPage()
- \$results->hasMorePages()
- \$results->lastPage() (Not available when using simplePaginate)
- \$results->nextPageUrl()
- \$results->perPage()
- \$results->total() (Not available when using simplePaginate)
- \$results->url(\$page)

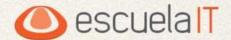


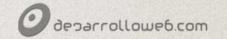


#### Ademas

Si devolvemos los resultados los convierte a JSON

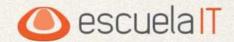
```
Route::get('users', function () {
    return App\User::paginate();
});
```





#### Más sobre Paginación

- Podriamos extender la clase de paginación y crear nuestra propia paginación avanzada. Normalmente con CSS se puede ajustar y suele sobrar.
- Documentación oficial
  - http://laravel.com/docs/5.0/pagination#usage

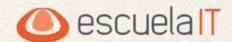




http://laravel.com/docs/5.0/elixir

npm install ← dentro del proyecto para instalar gulp watch ← ir compilando los cambios en tiempo real

Permite trabajar con scss o less



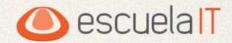


# Ejercicio para Casa

Añadimos a nuestro layout bootstrap para modelar las vistas

#### Guia ejemplo pasos

Descargar boostrap y añadirlo a public en carpeta assets Incluir en layout master los links al css y js Probar que funciona





#### **MUCHAS GRACIAS A TODOS!**

Podeis seguirme en la redes sociales como @micromante o Carlos Ruiz Ruso www.micromante.com