CONTROLADORES EN LARAVEL

Son clases intermediarias entre las clases que gestionan las bases de datos (modelo) y las que gestionan la vista.

Recordad, Laravel es un modelo MVC donde todo pasa a través del controlador.

Creando controladores

Controlador principal

En Laravel hay un controlador principal, la clase Controller, que está definida en app/http/Controllers.

Su código es el siguiente::

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Routing\Controller as BaseController;

class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
}
```

namespace asigna un espacio de nombres, por si hay duplicados.
use permite incluir una clase necesaria.
La clase Controller se define en este php, es el controlador principal de la aplicación.

Creación de controladores en Laravel:

Para crear nuevos controladores en Laravel, podemos hacerlo de dos formas:

1. Con **artisan**: Se genera de forma automática
2. A mano: Creando un archivo vacío en la carpeta Controllers

No vamos a utilizar la opción manual, que consistiría en crear un nuevo fichero dentro de la carpeta de controladores

Creación de un controlador con artisan:

Abrimos un terminal, y escribimos:

```
$ php artisan make:controller NombreControlador
```

Ejercicio 1:
Vamos a crear un controlador llamado PostController.

```
$ php artisan make:controller PostController
```

Al ejecutarlo, Laravel genera un nuevo fichero PostController.php en la carpeta Controllers, con el siguiente contenido:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostController extends Controller
{
    //
}
```

- El espacio de nombres será el mismo para todos los controladores.
- Utiliza la clase Request.
- La clase extenderá el controlador principal, pero estará vacía para que yo ponga todo el código.

Ejercicio 2:
Otra opción es ejecutar el mismo comando, con la opción --resource, para que me añada los métodos automáticamente:

```
$ php artisan make:controller --resource NombreControlador
```

En este caso, me creará el fichero en Controllers, con el siguiente código

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
```

```
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        //
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        //
```

```
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        //
    }
}
```

Como puede verse:

- Los métodos vienen con la documentación previamente escrita.
- Los métodos están vacíos, se han añadido para facilitar el desarrollo

Enrutando Controladores

En el fichero web.php, donde definimos las rutas, podemos utilizar el método index de un controlador, en vez de una función closure definida por nosotros. Ese método index hará las veces de función closure.

EJEMPLO:
Vamos a definir una ruta “/post”. Luego utilizaremos como segundo parámetro, en vez de una “function”, el nombre de un controlador definido por mí, por ejemplo, PostController.

Como el segundo parámetro es una función closure, tendré que tendré que indicar un método de ese controlador, ya que se espera una función. En este caso, suele utilizarse el index:

Web.php

```
Route::get('/post', '\App\Http\Controllers\PostController@index');
```

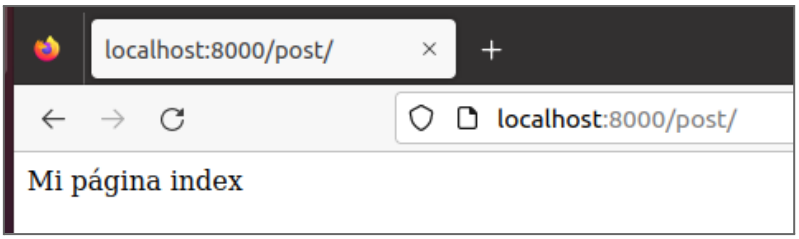
Nota: En versiones anteriores de Laravel no era necesario añadir la ruta completa del controlador (espacio de nombres). En las actuales sí es necesario.

Para probarlo:

1. Añado la ruta anterior en web.php
2. Modifico mi PostController.php, para que el método index retorne algo:

```
PostController.php
Controllers > PostController.php > PostController > index
public function index()
{
    //
    return "Mi página index";
}
```

Al poner la URL /post en el localhost obtendré:



EJERCICIO

¿ Se te ocurre cómo pasar parámetros a los métodos del controlador ? Intenta averiguarlo sin mirar la sección siguiente.

Pasando datos a los controladores

Podemos pasar un parámetro al index de un controlador mediante una variable de ruta.

Ejemplo:

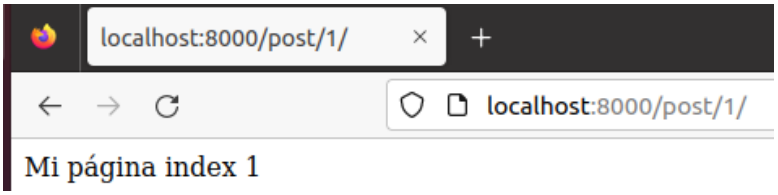
a) Defino la ruta con una variable, que será mi parámetro

```
Route::get('/post/{param}', '\App\Http\Controllers\PostController@index');
```

b) Añado el parámetro al método index de mi controlador

```
public function index($miParam)
{
    //
    return "Mi página index $miParam";
}
```

c) Al poner la ruta como URL, la variable “param” será pasada como parámetro al index



Efectivamente, el índex se está utilizando como closure function, por lo que toda variable de la ruta será pasada como parámetro, al igual que cuando utilizábamos funciones closure.

EJERCICIO

Prepara una ruta pasando dos parámetros a tu controlador, y utilizándolos en el response.

Recursos y controladores

Otra opción para definir rutas en web.php es mediante el método:

```
Route::resource
```

Este método nos permitirá crear rutas especiales.

EJEMPLO:
Si añado la siguiente ruta:

web.php

```
Route::resource('posts', '\App\Http\Controllers\PostController');
```

En este caso no estamos enviando una request al método index del controlador. Lo que hace Laravel es generar varias rutas.

Vamos a listar las rutas con **artisan**, y veré que se han añadido unas cuantas…

```
$ php artisan route:list
```

```
alumno@alumno-VirtualBox:~/proyectos/aplicacion1$ php artisan route:list

  GET|HEAD        / ..........................................................................
  POST            _ignition/execute-solution ignition.executeSolution › Spatie\LaravelIgnition › ExecuteSolu…
  GET|HEAD        _ignition/health-check ignition.healthCheck › Spatie\LaravelIgnition › HealthCheckControll…
  POST            _ignition/update-config ignition.updateConfig › Spatie\LaravelIgnition › UpdateConfigContr…
  GET|HEAD        api/user ...................................................................
  GET|HEAD        posts ..................................................... posts.index › PostController@index
  POST            posts ..................................................... posts.store › PostController@store
  GET|HEAD        posts/create ............................................ posts.create › PostController@create
  GET|HEAD        posts/{post} ............................................. posts.show › PostController@show
  PUT|PATCH       posts/{post} ............................................ posts.update › PostController@update
  DELETE          posts/{post} ............................................ posts.destroy › PostController@destroy
  GET|HEAD        posts/{post}/edit ....................................... posts.edit › PostController@edit
  GET|HEAD        sanctum/csrf-cookie ..... sanctum.csrf-cookie › Laravel\Sanctum › CsrfCookieController@show

                                                                                Showing [13] routes
```

Podemos ver que para la ruta “posts”, se han creado:

1. Varias rutas y subrutas, cada una estará asociada a un método diferente de nuestro controlador.
2. La variable {post} en algunas rutas
3. Un alias (name) para cada ruta. El alias consiste en la palabra “posts”, un punto, y el nombre del método que hará las veces de función callable. Podríamos utilizar esos “names” (posts.index, etc) en los links de nuestra aplicación.

Las rutas con POST, nos permiten realizar accesos a nuestra BD, de forma muy segura.

Lo que ha hecho Laravel, es crear esos “recursos”, esas rutas, automáticamente para cada uno de los métodos de la clase. Dependiendo del método request (get, post, …), cambia la función.

Probando las rutas generadas:

1. Añadimos código a los métodos. Si la ruta tiene una variable, añadiremos un parámetro.
2. Vamos al navegador y probamos.

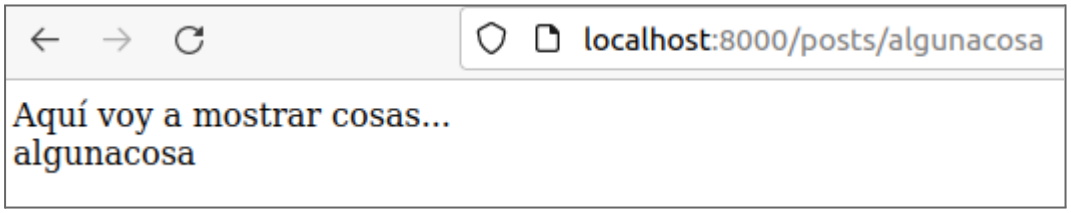
EJEMPLO:

En nuestro ejemplo, el show es llamado cuando se hace un request get a la ruta “`posts/{variable}`”.

Vamos a modificar el show para que retorne algo en el response.

```
public function show($variable)
    {
        //
        return "Aquí voy a mostrar cosas...<br>$variable";
    }
```

Cuando escribo la ruta que va a llamar a este método:



EJERCICIO

Vas a generar un controlador con recursos:

1. Crea una ruta con resource que llame a dicho controlador.
2. Comprueba con artisan qué rutas se han creado.
3. Prueba todas las rutas con request GET, añadiendo el código necesario en el controlador.