Práctica UD-5 Sesión-2

Objetivos	• Aprender conceptos de **enrutamiento** y **controladores** para el desarrollo de aplicaciones webs con Laravel.
Instrucciones de entrega	• No hay que entregar nada.

Abre una línea de comandos y ve a la ruta donde tiene el directorio ***htdocs*** de Xampp (c:\xampp\htdocs) después crea un nuevo proyecto con nombre ***PHPPracticaUD5Sesion2***:

- A través de composer

```
C:\xampp\htdocs>composer create-project laravel/laravel
PHPPracticaUD5Sesion2
```

- O con el comando de Laravel directamente:

```
C:\xampp\htdocs>laravel new PHPPracticaUD5Sesion2
```

- Después entre dentro de la carpeta ***PHPPracticaUD5Sesion2*** recién creada e instale los paquetes ***npm*** que trae Laravel por defecto para evitar el aviso en *NetBeans*:

```
C:\xampp\htdocs\PHPPracticaUD5Sesion2>npm install
```

- Comprobamos que todo ha ido arrancando el servidor embebido:

```
C:\xampp\htdocs\PHPPracticaUD5Sesion2>php artisan serve
```

- Después navegamos a la URL:
  - http://127.0.0.1:8000

- Una vez hecho eso importa el proyecto en NetBeans o en tu IDE o editor favorito.

Enrutamiento

- Documentación oficial:
  - https://laravel.com/docs/8.x/routing

- Para definir el enrutamiento en Laravel consistirá en la mayoría de los casos en editar el fichero ***routes\web.php*** donde se definirá las rutas del siguiente modo:

```
Route::get('/saludar', function () {
    return '¡Hola Mundo!';
});
```

- El enrutado permite registrar las rutas que responden a cualquier verbo HTTP y se utiliza la clase ***Route***:
  - *Route::get($uri, $callback);*
  - *Route: :post($uri, $callback);*
  - *Route: :put($uri, $callback);*
  - *Route: :patch($uri, $callback);*
  - *Route::delete($uri, $callback);*
  - *Route::options($uri, $callback);*

- También es posible registrar una ruta que responda a varios verbos HTTP, esto se hace utilizando el método *match*.

```
Route::match(['get', 'post'], '/', function () {
    //
});
```

- O bien, registrar una ruta que responda a todos los verbos HTTP utilizando el método *any*:

```
Route::any('/', function () {
    //
});
```

- Podemos ver qué rutas están definidas en nuestra aplicación a través del comando de artisan siguiente:

```
C:\xampp\htdocs\PHPPracticaUD5Sesion2>php artisan route:list
```

Enrutamiento básico

- Edita el archivo *web.php* y añade la siguiente ruta:

```
Route::get('/saludar', function () {
    return '¡Hola Mundo, definir una ruta básica!';
});
```

- Por último pruebe el resultado navegando a la URL:
  - http://127.0.0.1:8000/saludar

Redireccionamiento

- Para definir una ruta que redirige a otra URI, se puede utilizar el método *Route::redirect*.

- Edita de nuevo el archivo *web.php* y añade la siguiente ruta:

```
Route::redirect('/saludarRedirect', '/saludar');
```

- Prueba el resultado navegando a la URL:
  - http://127.0.0.1:8000/saludarRedirect

Enrutamiento a vistas

- Si tenemos una ruta que sólo necesita devolver una vista se puede usar el método *Route::view*.

- Edita de nuevo el archivo *web.php* y añade la siguiente ruta:

```
Route::view('/inicio', 'welcome');
```

- Prueba el resultado navegando a la URL:
  - http://127.0.0.1:8000/inicio

- Comprueba las rutas hasta ahora definidas:

```
C:\xampp\htdocs\PHPPracticaUD5Sesion2>php artisan route:list
```

Enrutamiento con parámetros

Parámetros obligatorios

- Si necesita capturar segmentos dentro de la ruta, por ejemplo el id de un usuario.

- Los parámetros de ruta siempre se incluyen entre llaves {} y deben constar de caracteres alfabéticos y guiones bajos (_). Los parámetros de ruta se inyectan en las funciones de llamada o los controladores de ruta según su orden.

- Edita el archivo *web.php* y añade lo siguiente:

```
Route::get('/usuario/{id}', function ($id) {
    return 'Usuario con Id:'.$id;
});
```

- Prueba el resultado navegando a la URL:
  - http://127.0.0.1:8000/usuario/123
  - http://127.0.0.1:8000/usuario/
  - http://127.0.0.1:8000/usuario

- Edita el archivo *web.php* y añade lo siguiente:

```
Route::get('/posts/{post}/comments/{comment}', function ($postId,
$commentId) {
     return 'postId:'.$postId.', commentId: '.$commentId;
});
```

- Prueba el resultado navegando a la URL:
  - http://127.0.0.1:8000/posts/123/comments/22
  - http://127.0.0.1:8000/posts/comments/22
  - http://127.0.0.1:8000/posts/1/comments/

Parámetros opcionales

- Para definir parámetros opciones se hace con el símbolo interrogacion (?).

- Edita el archivo *web.php* y añade lo siguiente:

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

- Prueba el resultado navegando a la URL:
  - http://127.0.0.1:8000/user/Pepe
  - http://127.0.0.1:8000/user/

Parámetros con expresiones regulares

- Se puede restringir el formato de los parámetros de su ruta utilizando el método ***where*** en una instancia de ruta. El método ***where*** acepta el nombre del parámetro y una expresión regular que define cómo se debe restringir el parámetro.

- Edita el archivo *web.php* y añade lo siguiente:

```
Route::get('/userRegularExpresion/{name}', function ($name) {
    return 'User name: '.$name;
})->where('name', '[A-Za-z]+');

Route::get('/userRegularExpresion/{id}', function ($id) {
    return 'User id: '.$id;
})->where('id', '[0-9]+');

Route::get('/userRegularExpresion/{id}/{name}', function ($id, $name) {
    return 'User name: '.$name.', id: '.$id;
})->where(['id' => '[0-9]+', 'name' => '[A-Za-z]+']);

Route::get('/userRegularExpresionHelper/{id}/{name}', function ($id,
$name) {
    return 'User name: '.$name.', id: '.$id;
})->whereNumber('id')->whereAlpha('name');

Route::get('/userRegularExpresionHelper/{name}', function ($name) {
    return 'User name: '.$name;
})->whereAlphaNumeric('name');

Route::get('/userRegularExpresionHelper/{id}', function ($id) {
    return 'User id: '.$id;
})->whereUuid('id');
```

- Prueba el resultado navegando a la URL:
  - http://127.0.0.1:8000/userRegularExpresion/Juan
  - http://127.0.0.1:8000/userRegularExpresion/1234
  - http://127.0.0.1:8000/userRegularExpresion/1234/juan
  - http://127.0.0.1:8000/userRegularExpresionHelper/Lucia
  - http://127.0.0.1:8000/userRegularExpresionHelper/999
  - http://127.0.0.1:8000/userRegularExpresionHelper/999/lucia

Rutas con nombre

- Las rutas con nombre permiten la generación conveniente de URL o redireccionamientos para otras rutas específicas.

- Edita el archivo *web.php* y añade lo siguiente:

```
Route::get('/user/profile', function () {
      return 'ver el perfil';
})->name('profile');
```

- Prueba el resultado navegando a la URL:
    - http://127.0.0.1:8000/profile/
    - http://127.0.0.1:8000/user/profile/

Controladores

Básicos

- Documentación oficial:
  - https://laravel.com/docs/8.x/controllers

- Crea un nuevo controller con nombre ***HomeController***, puedes hacerlo desde línea de comandos:

```
C:\xampp\htdocs\PHPPracticaUD5Sesion2>php artisan make:controller
HomeController
Controller created successfully.
```

- Comprueba en *NetBeans* o en tu IDE o editor favorito que se ha creado un archivo con el nombre ***HomeController*** que extiende de Controller.

- Edite el contenido del controlador para añadir lo siguiente:

```
public function index(){
        return '¡Hola desde el método/acción index del controlador';
    }
```

- Edita el archivo *web.php* y añade lo siguiente:

```
use App\Http\Controllers\HomeController;

Route::get('/home', [HomeController::class, 'index']);

Route::get('/home2', 'App\Http\Controllers\HomeController@index');
```

- Prueba el resultado navegando a la URL:
  - http://127.0.0.1:8000/home
  - http://127.0.0.1:8000/home2

Con una sola acción

- Cuando un controlador sólo define una acción podemos definirlo de una forma especial; podemos crearlo con Artisan con la opción ***--resource*** del comando *make: controller*, del siguiente modo para ***SingleActionController***:

```
C:\xampp\htdocs\PHPPracticaUD5Sesion2>php artisan make:controller
SingleActionController --invokable
Controller created successfully.
```

- Edite el contenido del controlador para añadir lo siguiente:

```
    public function __invoke(Request $request)
    {
        return 'Controlador con una sóla acción';
    }
```

- Edita el archivo *web.php* y añade lo siguiente:

```
use App\Http\Controllers\SingleActionController;

Route::get('/singleaction', SingleActionController::class);
Route::get('/singleaction2',
'App\Http\Controllers\SingleActionController');
```

- Prueba el resultado navegando a la URL:
  - http://127.0.0.1:8000/singleaction
  - http://127.0.0.1:8000/singleaction2

Controladores de recursos

- El enrutamiento de recursos de Laravel asigna las rutas típicas de creación, lectura, actualización y eliminación ("CRUD") a un controlador con una sola línea de código.

- Para ello podemos usar la línea de comandos y Artisan con la opción ***--resource*** del comando *make: controller* para crear rápidamente un controlador para manejar estas acciones:

```
C:\xampp\htdocs\PHPPracticaUD5Sesion2>php artisan make:controller
PhotoController --resource
Controller created successfully.
```

- Comprueba el controlador recién creado.

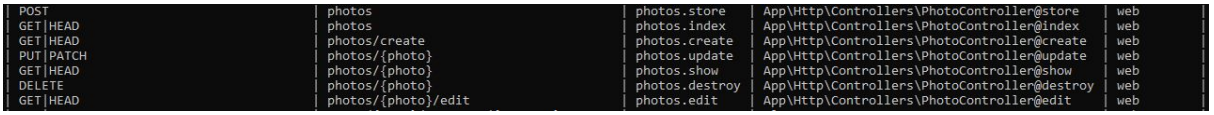
- Añade ahora la ruta especial de la siguiente forma:

```
use App\Http\Controllers\PhotoController;

Route::resource('photos', PhotoController::class);
```

- Comprueba las rutas hasta ahora definidas y fíjate en las que corresponden a photos:

```
C:\xampp\htdocs\PHPPracticaUD5Sesion2>php artisan route:list
```



Especificar el modelo de recursos

- También se pueden enlazar un modelo a los métodos del controlador de recursos, para ello se usa la opción ***--model*** al generar el controlador:

```
php artisan make:controller UserController --resource --model=User
```

- Comprueba el controlador que se acaba de crear.

Inyección de dependencias

Inyección vía constructor

- Laravel utiliza el ***service container*** para inyectar en el constructor del controlador cualquier dependencia que se pueda necesitar.

```php
<?php

namespace App\Http\Controllers;

use App\Repositories\UserRepository;

class UserController extends Controller
{
    /**
     * The user repository instance.
     */
    protected $users;

    /**
     * Create a new controller instance.
     *
     * @param  \App\Repositories\UserRepository  $users
     * @return void
     */
```

```
public function __construct(UserRepository $users)
    {
        $this->users = $users;
    }
}
```

Inyección vía métodos

- Además de la inyección del constructor, también se pueden escribir dependencias en los métodos del controlador. Un caso de uso común para esto es la inyección de la instancia *Iluminate\Http\Request* en los métodos del controlador:

```
public function store(Request $request)
    {
        $name = $request->name;

        //
    }
```

- Si el método del controlador también espera la entrada de un parámetro de la ruta, por ejemplo {id}, hay que enumerar los argumentos de ruta después de las otras dependencias. Por ejemplo, si su ruta se define así:

```
Route::put('/user/{id}', [UserController::class, 'update']);
```

- El método se definiría así:

```
public function update(Request $request, $id)
    {
        //
    }
```