

Laravel Controllers

Lógica de gestión de solicitudes

Controladores

Organiza la **lógica de gestión** (o control) **de solicitudes** utilizando **clases "controller"**.

Por ejemplo, una clase 'UserController' podría gestionar todas las solicitudes entrantes relacionadas con los usuarios, incluyendo mostrar, crear, actualizar y eliminar usuarios (CRUD).

Por defecto, en Laravel los controladores se almacenan en el directorio **'app/Http/Controllers'**.



Crear un controlador

Para generar rápidamente un nuevo controlador, puedes ejecutar el comando Artisan **make:controller**.

Ejemplo para crear un controlador para gestionar solicitudes sobre “usuarios”:

```
php artisan make:controller UserController
```

También puedes crear un controlador manualmente en la carpeta `app/Http/Controllers`, con formato “[nombre]**Controller.php**”.

Un controlador puede tener cualquier número de métodos públicos para responder a solicitudes HTTP entrantes.




Ejemplo de controlador - UserController.php

```
<?php
Namespace App\Http\Controllers;
use App\Models\User;
use Illuminate\View\View;

class UserController extends Controller{ // Recomendable extender Controllers, no obligatorio

    /** Mostrar perfil de un user. */
    public function show(string $id): View {
        return view('user.profile', [
            'user' => User::findOrFail($id)
        ]);
    }
}
```




Ejemplo de Controlador - invocar UserController.php

Una vez **escrito una clase de controlador** y un método, se puede **definir una ruta** (en web.php) hacia dicho método de la siguiente manera:

```
use App\Http\Controllers\UserController;
```

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

Cuando una solicitud entrante coincida con la URI de la ruta especificada, se invocará el método 'show' de la clase 'App\Http\Controllers\UserController' y los parámetros de la ruta se pasarán al método.



Controladores de una sola acción - __invoke()

Si solo se requiere controlar una única acción puede ser conveniente declarar una clase de controlador con un único método dedicado a dicha acción. Para lograr esto, se puede definir un único método **__invoke()** dentro del controlador:

```
public function __invoke() {  
    // lógica ...  
}
```

Cuando se declara un controlador de una única acción, no es necesario indicar el método en la ruta de llamada. Si UserController solo tuviera __invoke():

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



Controladores - métodos propios

Ejemplo de Controlador *UserController.php* con método propio:

```
namespace App\Http\Controllers;
class userController{
    public function index() {
        $mensajes = [ ['titulo' => 'primer mensaje'], ['titulo' => 'segundo mensaje'] ];
        return view('chat', ['mensajes' => $mensajes ]);
    }
}
```

En *web.php*:

```
Route::get('chat', [ UserController::class, 'index' ]->name('chat');
```

Los métodos necesarios se pasan con la clase como parámetros en array.



Controladores - crear con Artisan

Ejemplo de generación de controlador *MensajeController.php* por comandos empleando artisan (ubicados en la carpeta de proyecto):

php artisan make:controller UserController

- Añadir **-i** o **--invokable** (invoke) para que sea invocable (añade método `__invoke()`)
- Añadir **-r** o **--resource** (resource) para que incluya los métodos `index()`, `create()`, `store()`, `show()`, `edit()`, `update()`, `destroy()`
- Opción **--api** para que incluya los métodos `index()`, `store()`, `show()`, `update()`, `destroy()`

Controladores con Middleware

Al invocar controladores definiendo la ruta es posible añadir Middleware:


```
Route::get('/profile', [UserController::class, 'show']->middleware('auth'));
```



Controladores con Middleware

También se puede **registrar un Middleware desde la clase del controlador, con un método Middleware estático**:

```
use Illuminate\Routing\Controllers\HasMiddleware;  
use Illuminate\Routing\Controllers\Middleware;  
class UserController extends Controller implements HasMiddleware {  
  
    // Registrar Middleware que se debería ejecutar cuando lleguen peticiones al Controlador  
    public static function middleware(): array {  
        return [  
            'auth',  
            new Middleware('log', only: ['index']),  
            new Middleware('subscribed', except: ['store']),  
        ];  
    }  
}
```



Controladores de recursos - Resource

Si se considera cada modelo Eloquent (**tablas** identificadas por el ORM) **de una aplicación como un "recurso", es habitual realizar las mismas acciones CRUD** sobre cada recurso.

Debido a este caso de uso común, Laravel **asigna las rutas típicas de crear, leer, actualizar y eliminar ("CRUD") a un controlador con una sola línea de código.**

Como hemos visto, estos controladores se pueden crear empleando el tag de "-r" o --resources con el comando de artisan create:controller:

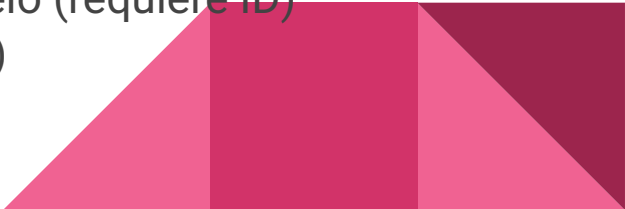
```
php artisan make:controller PhotoController --resource
```



Controladores de recursos - Resource

php artisan make:controller PhotoController --resource

Este comando generará un controlador en **app/Http/Controllers** llamado *PhotoController.php*, que contendrá un método para cada operación del recurso:

- `index()`: para mostrar listado de registros
 - `show()`: para mostrar un registro (requiere ID)
 - `create()`: para mostrar un formulario de creación de registro
 - `store()`: para guardar registro nuevo en el modelo
 - `edit()`: para mostrar formulario de edición de un registro existente (requiere ID)
 - `update()`: para actualizar un registro existente en el modelo (requiere ID)
 - `destroy()`: para eliminar un registro existente (requiere ID)
- 

Controladores de recursos - Resource - Ruta

Para definir rutas a los métodos de los controladores de registro, **basta con registrar una única ruta “Route::resource”** que apunte al controlador:

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

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

Esta **única declaración de ruta crea múltiples rutas** para manejar las diversas acciones a realizar sobre el recurso.

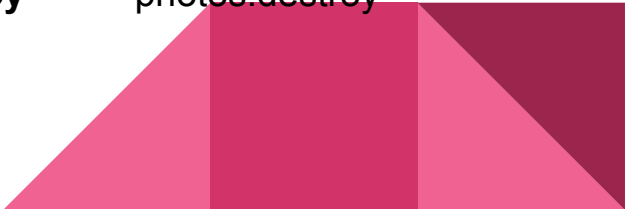
Además, siempre se puede obtener una visión general rápida de las rutas de la aplicación ejecutando el comando

```
php artisan route:list
```



Controladores de recursos - Resource - Ruta

Verb	URI	Action	Route Name
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy



Controladores de recursos - Recurso no localizado

Por lo general, **se genera una respuesta HTTP 404 si un recurso** del modelo vinculado **no se encuentra**.

Sin embargo, **se puede personalizar este comportamiento llamando al método `missing()` al definir la ruta al controlador** de recurso.

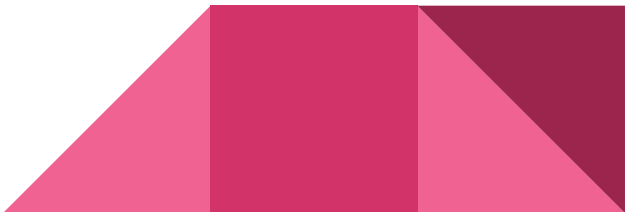
Este método acepta una closure que se ejecutará si no se puede encontrar recurso para alguna de las rutas del recurso (es decir, ID de registro no localizado).



Controladores de recursos - Recurso no localizado

Ejemplo de control personalizado para un recurso no localizado:

```
use App\Http\Controllers\PhotoController;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Redirect;  
  
Route::resource('photos', PhotoController::class)  
->missing(function (Request $request) {  
    // Redirige al método index() si no se encuentra el registro/recurso  
    return Redirect::route('photos.index');  
});
```

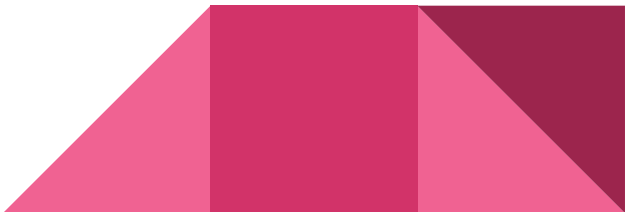


Controladores de recursos - dar nombre

Por defecto, **todas las acciones de los controladores de tipo resource tienen un nombre de ruta**; no obstante, se **pueden sobrescribir estos nombres** pasando un array al **método names()** con los nombres de ruta deseados. Por ejemplo:

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

```
Route::resource('photos', PhotoController::class)->names([  
    'create' => 'photos.build'  
]);
```



Controladores de recursos y el orden de las rutas

Recordatorio: **El orden con el que se declaran las rutas importa.**

Las rutas de controladores de recursos no son excepción; **si tenemos métodos propios en dichos controladores, conviene definirlos antes que** la definición de ruta de **los métodos de recursos** “Route::resource()”.

Por ejemplo:

```
use App\Http\Controller\PhotoController;
```

```
Route::get('/photos/popular', [PhotoController::class, 'popular']);
```

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

Con esta declaración no existe riesgo de que la ruta photos/popular se confunda con un photos/{photos}.




Controladores - Inyeccion de métodos externos

Los controladores permite incluir llamadas a métodos de otros controladores o clases. Esto puede servir, por ejemplo, para procesar datos y posteriormente redirigir a otra ruta.

Si tenemos la siguiente Ruta para el controlador UserController:

```
use App\Http\Controllers\UserController;  
Route::put('/user/{id}', [UserController::class, 'update']);
```

Interesa que, tras realizar el update, se redirija al usuario devuelta a la primera página “/users”.



Controladores - Inyeccion de métodos externos

```
namespace App\Http\Controllers;  
use Illuminate\Http\RedirectResponse;  
use Illuminate\Http\Request;
```

```
class UserController extends Controller {
```

```
    // Método Update the un user dado.
```

```
    public function update(Request $request, string $id): RedirectResponse { // Clase Request ANTES de los parámetros pasados
```

```
        // Buscar y actualizar al usuario
```

```
        $user = User::findOrFail($id); // Lanza un error si no encuentra el usuario
```

```
        $user->name = $request->name;
```

```
        // repetir para el resto de datos de User
```

```
        $user->save(); // Guarda los cambios en la base de datos
```

```
        // Tras finalizar la actualización, redirige al listado de usuarios - indicando estado y mensaje
```

```
        return redirect('/users')->with('success', 'Usuario actualizado correctamente.');
```

```
    }
```

```
}
```

