Controller e Middleware

LaraMind

Sidrit Trandafili

@sidis405

@strandafili

- Sviluppatore PHP 15+ anni
- Laravel da 4 anni
- Laravel Certified
- Back-end
  - php
  - java
  - node
  - python
  - dart
- Front-end
  - VueJS
  - Vanilla JS
  - TailwindCSS
  - Sass
  - dart
- Devops + Sysops da 14 anni

Controllers

Controllers

In Laravel 5.6, tutti i controller vengono creati nella cartella Controllers, che si trova in App/Http/Controllers.

Tutti i controller hanno bisogno di un namespace App\Http\Controllers e ereditano dalla classe Controller.

Creare un semplice controller

Richiamiamo l'esempio della tabella multipli dalla sezione Routing.

Creeremo un controller, con un singolo metodo, che stamperà a schermo la nostra tabellina.
Per definire un controller bisogna dargli un nome. Sceglieremo TabellaController come nome della classe per il nostro controller.
Tutti i controller in Laravel devono contenere la parola Controller alla fine.

Creare un semplice controller

Per questa operazione è possibile proseguire in due modi.

- Creiamo il file manualmente
- Usiamo *Artisan* per creare il controller.

La seconda e l'opzione più facile ed è quella che sceglieremo.

```
php artisan make:controller TabellaController
```

Creare un semplice controller

Questo comando creerà la classe TabellaController nella cartella Controllers, nel file TabellaController.php.

```
<?php


namespace App\Http\Controllers;


use Illuminate\Http\Request;


class TabellaController extends Controller
{
        //
}
```

Creare un semplice controller

Trasferiamo adesso la logica per creare la tabella, dal file delle routes al nostro controller.

Creiamo un metodo create dentro il nostro TabellaController.

```
public function create($numero = 2){

                for(range(1, 10) as $x){

                        echo "$x * $numero = ". $x * $numero ."<br>";

                }

}
```

Creare un semplice controller

In questo modo possiamo alleggerire il nostro web.php, sostituendo la nostra route iniziale con la seguente

```
Route::get('/tabella-multipli/{numero?}', 'TabellaController@create')
->where('numero',  '[0-9]+');
```

In questo modo, ogni volta che alla nostra applicazione viene richiesto l'url sopracitato, il framework eseguirà la funzione create che si trova in TabellaController

Creare un Resource controller

I Resource Controllers vengono usati per creare funzionalità CRUD (Create, Read, Update, Delete).

Supponiamo di avere un modello Libri, per il quale vogliamo creare un controller che svolge operazioni CRUD nel controller.

Con un singolo comando Artisan è possibile creare il controller assieme a tutti i metodi necessari, includendo la possibilità di specificare anche il modello per il quale stiamo creando questa struttura.

```
php artisan make:controller LibroController --resource -m Libro
```

Una volta eseguito il comando, vi verrà chiesto se creare il modello Libro.

Rispondete y e premete invio.

In questo modo verrà creato a seguito anche il controller LibroController.

Creare un Resource controller

Nel file del controller troveremo i seguenti metodi:

- index() - per listare tutti i libri
- create() - per vedere il form della creazione di un libro
- store() - per salvare il libro appena creato
- show() - per mostrare un libro
- edit() - per vedere il form della modifica di un libro
- update() - per salvare il libro appena modificato
- destroy() - per cancellare un libro

Creare un Resource controller

Creare un Resource controller

I Resource Controllers vengono usati per creare funzionalità CRUD (Create, Read, Update, Delete).

Supponiamo di avere un modello Libri, per il quale vogliamo creare un controller che svolge operazioni CRUD nel controller.

Con un singolo comando Artisan è possibile creare il controller assieme a tutti i metodi necessari, includendo la possibilità di specificare anche il modello per il quale stiamo creando questa struttura.

```
php artisan make:controller LibroController --resource -m Libro
```

Una volta eseguito il comando, vi verrà chiesto se creare il modello Libro.

Rispondete y e premete invio.

In questo modo verrà creato a seguito anche il controller LibroController.

Una funzionalità molto utile di Laravel è che non è necessario creare una route separata nel file routes.

Basta aggiungere la seguente riga in **web.php** e il framework si occuperà di mappare tutte le URL.

```
Route::resource('libro', 'LibroController');
```

Creare un Resource controller

Url, verbi HTTP, metodi e nomi delle routes vengono mappati come segue:

- /libro - GET - index - libro.index
- /libro/create - GET - create - libro.create
- /libro - POST - store - libro.store
- /libro/{libro} - GET - show - libro.show
- /libro/{libro}/edit - GET - edit - libro.edit
- /libro/{libro} - (PUT oppure PATCH) - update - libro.update
- /libro/{libro} - DELETE - destroy - libro.destroy

Creare un Resource controller

È possibile specificare una Route Resource per gestire solo certe routes, o per escludere certe routes.

Ecco la sintassi per ciascuno di questi scenari:

```
Route::resource('libro', 'LibroController')->only('index', 'show');
```

```
Route::resource('libro', 'LibroController')
->except('create', 'store', 'update', 'destroy');
```

Cosa sono i Middleware e come crearli?

I Middleware sono il modo più facile per verificare o filtrare una richiesta HTTP prima di passarla al Controller.

Si trovano nella cartella Middleware dentro app/Http.

Creazione di un semplice Middleware

Procederemo a creare un Middleware per il nostro LibroController. Questo verificherà che se un libro è di un anno in particolare specificato da noi, non verrà salvato nel database.

Chiameremo questo Middleware VerificaAnno e lo creeremo con un comando Artisan.

```
php artisan make:middleware VerificaAnno
```

Una volta eseguito, questo Middleware avrà il suo metodo handle(), nel quale aggiungeremo la nostra logica.

Creazione di un semplice Middleware

```php
<?php

namespace App\Http\Middleware;

use Closure;

class VerificaAnno
{

    public function handle($request, Closure $next)
    {

        $anni = ['1928','1939'];

        if (in_array($anni, $request->anno)) {
            return redirect('/');
        }

        return $next($request);
    }
}
```

Creazione di un semplice Middleware

- Abbiamo definito un array $anni per il quali rifiuteremo il salvataggio dati
- Controlliamo con la funzione php in_array se l'anno specificato per il nostro libro è contenuto nell'array definito
- Se il controllo risulta positivo, reindirizziamo l'utente alla pagina home.
- Altrimenti alla richiesta viene permesso di completare il suo ciclo.

Creazione di un semplice Middleware

Per poter connettere un middleware a un controller, basta definirlo come segue nel metodo __constructor() del controller stesso:

```php
<?php
namespace App\Http\Controllers;
use App\Libro;
use Illuminate\Http\Request;
use App\Http\Middleware\VerificaAnno;

class LibroController extends Controller

        public function __construct()
        {
                $this->middleware(VerificaAnno::class);
        }
```

Creazione di un semplice Middleware

I Middleware possono essere registrati in app/Http/Kernel.php, nell'array $routemiddleware con un alias e il namespace completo della classe.

```
protected $routeMiddleware = [
        'auth' => \Illuminate\Auth\Middleware\Authenticate::class,
        ...
        ...
        'anno' => \App\Http\Middleware\VerificaAnno::class
];
```

Creazione di un semplice Middleware

In questo modo non abbiamo bisogno di richiamare la classe del middleware nel nostro controller ma solo il suo alias.

```php
<?php
namespace App\Http\Controllers;
use App\Libro;
use Illuminate\Http\Request;

class LibroController extends Controller

        public function __construct()
        {
                $this->middleware('anno');
        }
```