

Laravel

Routing, Controller, Request / Response

Installazione e configurazione

Installazione di Laravel 8.x

Attraverso Laravel Installer



```
composer global require laravel/installer  
  
laravel new blog
```

Attraverso Composer Create-Project



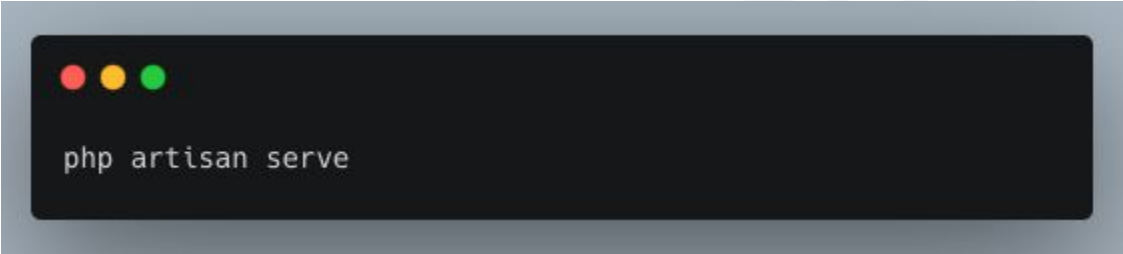
```
composer create-project --prefer-dist laravel/laravel blog
```

Configurazione - config files

- Tutti i file di configurazione di Laravel sono contenuti nella cartella config.
 - file diversi a seconda delle funzionalità
- Possono cambiare a seconda dell'ambiente, quindi?
 - Laravel utilizza la libreria Dot-Env
 - si compila un file .env contenente le variabili d'ambiente
 - le variabili d'ambiente sono leggibili tramite la funzione env()
 - il file .env non viene versionato su git, al suo posto si mantiene un file .env.example

Esecuzione di Laravel

- Può essere eseguito all'interno di qualsiasi web server che supporti l'interprete Php, essendo Laravel scritto in Php
- Tuttavia, in caso di Apache, Nginx ecc. saranno necessarie alcune configurazioni particolari, ad esempio impostare la document root sulla cartella public
- La soluzione per sviluppare in locale? Utilizzare **artisan**, il tool a linea di comando di Laravel

A terminal window with a dark background and a light gray border. In the top-left corner, there are three colored circles: red, yellow, and green. The text 'php artisan serve' is displayed in a light gray monospace font.

```
php artisan serve
```

Routing



Routing


Tutte le rotte sono definite in appositi files php contenuti nella cartella routes e vengono caricati automaticamente dal framework.

- Il file web.php definisce le rotte dell'interfaccia web, assegnate al “middleware” web che permette la gestione di sessione e protezione da attacchi CSRF.
- Le rotte contenute nel file api.php, al contrario, sono stateless e assegnate al middleware “api”.
- Per lo sviluppo di REST API si utilizzano queste ultime.

Routing

Sono definite da una url e da una funzione da eseguire:

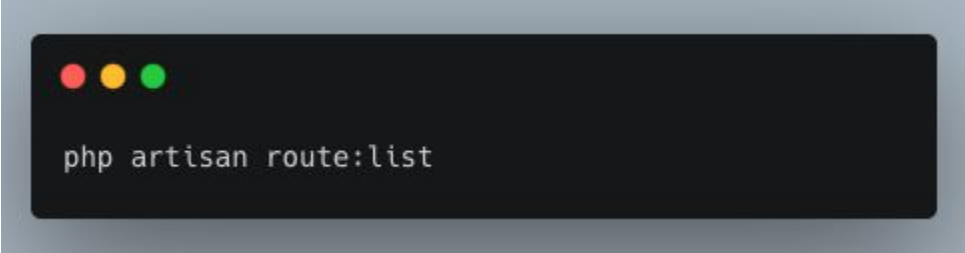
- closure alias funzione anonima, oppure
- stringa in formato controller@metodo)



```
Route::get('/test', function() {  
    //Do something  
});  
  
Route::get('/user', 'UserController@index');
```


Routing list

- Se ho necessità di visualizzare un elenco di tutte le rotte disponibili nella mia applicazione, posso richiedere ad artisan un elenco:

A terminal window with a dark background and a light gray border. In the top-left corner, there are three colored circles: red, yellow, and green. The text 'php artisan route:list' is displayed in a light gray monospace font.

```
php artisan route:list
```

I Controller

I Controller - cosa sono

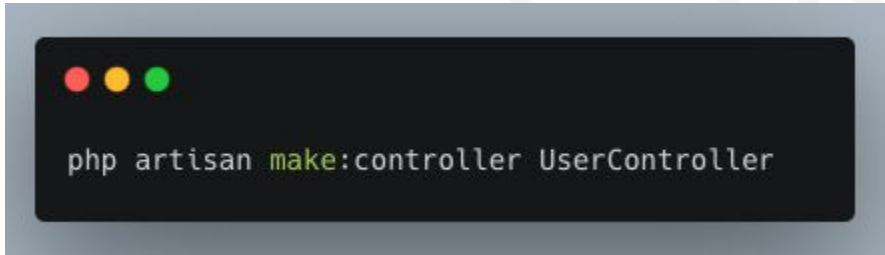
- Laravel permette di definire alcuni comportamenti direttamente nei file routes
- Questo approccio però non è scalabile e genera file di rotte molto lunghi e complessi
- Per gestire i vari entry point di una applicazione Laravel è invece previsto l'utilizzo dei controller, come da pattern MVC
- I Controller non sono altro che particolari classi PHP posizionate nel percorso

`app/Http/Controllers`

I Controller - come crearli

- Basta eseguire il comando `artisan make:controller`, seguito dal nome del controller che si intende creare.

Ad esempio:

A terminal window with a dark background and a light gray border. At the top left, there are three colored circles: red, yellow, and green. Below them, the command `php artisan make:controller UserController` is displayed in a light gray monospace font. The word `make` is highlighted in green.

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

I Controller - cosa fanno

- All'interno di un Controller devono essere implementate le funzioni corrispondenti alle entry point della nostra applicazione, mappate nelle rotte
- Laravel ragiona a richieste e risposte; le funzioni dei controller quindi hanno in input una richiesta e sono obbligati a restituire una risposta tramite il return

I Controller - esempio

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\User;
use Illuminate\Http\Request;

class UserController extends Controller
{
    /**
     * Show the profile for the given user.
     */
    public function show(Request $request)
    {
        return view('user.profile', ['user' => User::findOrFail($request->input('id'))]);
    }
}
```

Request e Response



L'oggetto request

Una delle responsabilità dei controller è quella di agire sull'oggetto **request**. La miglior strategia per ottenere l'oggetto in questione è quella utilizzare la *dependency injection* di Laravel definendo un parametro tipizzato tra i parametri del metodo.

```
<?php


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

class UserController extends Controller
{
    public function store(Request $request)
    {
        $name = $request->input('name');

        //
    }
}
```


L'oggetto Request

Laravel permette di accedere ai parametri della richiesta (parametri GET o POST) attraverso l'oggetto Request, anziché attraverso le variabili superglobals `$_GET`, `$_POST`, `$_REQUEST` ecc.




```
$name = $request->input('name'); // -> $_REQUEST['name']
```

Per approfondire: <https://laravel.com/docs/8.x/requests>

Accesso a valori JSON in input

Nel caso in cui la richiesta che arriva alla nostra applicazione contenga un body in formato JSON, è possibile accedere a questi valori attraverso il metodo input visto in precedenza.

- La richiesta dovrà però avere il Content-Type header correttamente settato su application/json (Postman lo imposta automaticamente nel caso il body sia JSON).
- Laravel consente l'utilizzo della sintassi “dot” per accedere alle proprietà di un determinato input



```
$name = $request->input('user.name');
```

Response

Da un controller possiamo ritornare una risposta attraverso la return. Solitamente la risposta è una vista elaborata (stringa HTML).



```
return response()  
    ->view('hello', $data, 200)  
    ->header('Content-Type', $type);
```

Nelle REST API, invece, le risposte devono essere prodotte in formato JSON

Per approfondire: <https://laravel.com/docs/8.x/responses>

JSON Response

Il metodo `json` imposta automaticamente l'header `Content-Type` a `application/json`, e converte la struttura dati passata in JSON (nell'esempio, un array):



```
return response()->json([  
  'name' => 'Abigail',  
  'state' => 'CA'  
], 200);
```

Altri tipi di response

- redirect (url o named route)
- file, file download





Q & A