

Atelier 1 Laravel: Introduction à Laravel

Objectif

L'objectif de cette étude de cas est de vous faire manipuler Laravel en vous concentrant sur la configuration des routes, la gestion des réponses, des redirections, et l'utilisation des variables d'environnement, sans inclure l'utilisation de vues (Blade) ni de contrôleurs.

1. Créer un projet Laravel

Objectif : Créer un projet Laravel de base pour démarrer.

Étapes :

1. Ouvrez votre terminal et allez dans le répertoire où vous souhaitez créer le projet.
2. Exécutez la commande suivante pour créer un projet Laravel (remplacez NomDuProjet par le nom que vous souhaitez donner à votre projet) :

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



```
houdaifa@HoudaifaPC: ~/everything/Laravel$ composer create-project laravel/laravel atelier-1
Creating a "laravel/laravel" project at "./atelier-1"
Installing laravel/laravel (v12.11.1)
As there is no 'unzip' nor '7z' command installed zip files are being unpacked using the PHP zip extension.
This may cause invalid reports of corrupted archives. Besides, any UNIX permissions (e.g. executable) defined in the archives will be lost.
Installing 'unzip' or '7z' (21.01+) may remediate them.
- Downloading laravel/laravel (v12.11.1)
- Installing laravel/laravel (v12.11.1): Extracting archive
Created project in /home/houdaifa/everything/Laravel/atelier-1
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 111 installs, 0 updates, 0 removals
- Locking brick/math (0.14.1)
- Locking carbonphp/carbon-doctrine-types (3.2.0)
- Locking dflydev/dot-access-data (v3.0.3)
- Locking doctrine/inferno (2.1.0)
- Locking doctrine/lexer (3.0.1)
- Locking dragonmantank/cron-expression (v3.6.0)
- Locking egulias/email-validator (4.0.4)
- Locking fakerphp/faker (v1.24.1)
- Locking filp/whoops (2.18.4)
- Locking fruitcake/php-cors (v1.4.0)
- Locking graham-campbell/result-type (v1.1.4)
- Locking guzzlehttp/guzzle (7.10.0)
- Locking guzzlehttp/promises (2.3.0)
- Locking guzzlehttp/psr7 (2.8.0)
- Locking guzzlehttp/uri-template (v1.0.5)
- Locking hamcrest/hamcrest-php (v2.1.1)
```

3. Une fois le projet créé, naviguez dans le répertoire du projet avec la commande :

`cd NomDuProjet`

4. Démarrez le serveur de développement Laravel : `php`



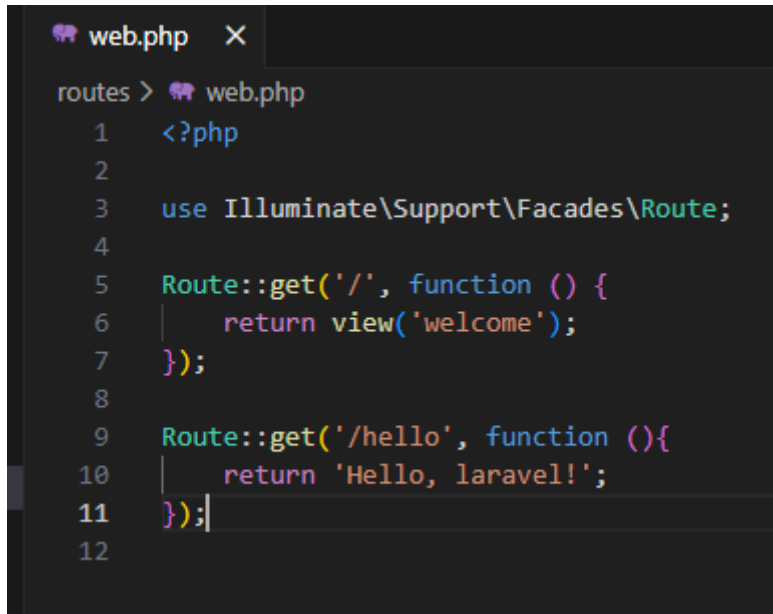
```
houdaifa@HoudaifaPC: ~/everything/Laravel/atelier-1$ php artisan serve
INFO Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
2026-01-07 10:52:18 / ..... ~ 0.44ms
2026-01-07 10:52:18 /favicon.ico ..... ~ 0.15ms
```

3. Créer et tester des routes dans Laravel

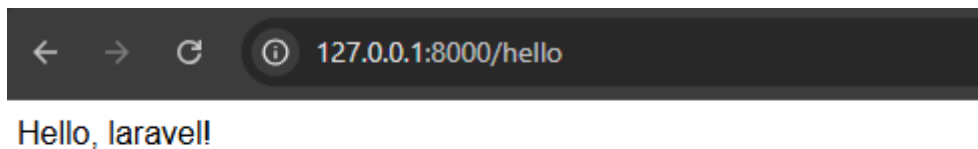
Objectif : Apprendre à définir des routes simples, dynamiques et préfixées sans utiliser de contrôleurs.

3.1 Routes de base

Dans le fichier routes/web.php, ajoutez une route qui retourne une chaîne de texte simple.



```
web.php X
routes > web.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5  Route::get('/', function () {
6      return view('welcome');
7  });
8
9  Route::get('/hello', function () {
10     return 'Hello, laravel!';
11 });
12
```

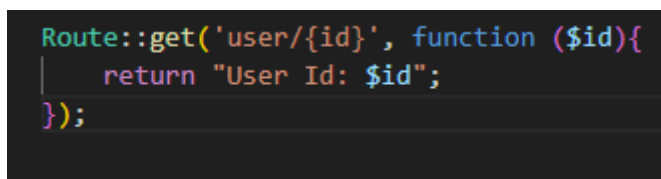


127.0.0.1:8000/hello

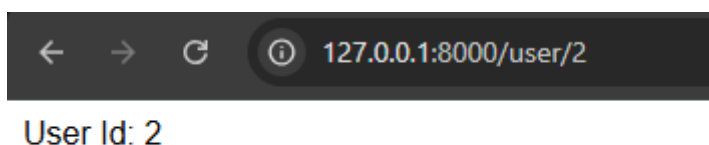
Hello, laravel!

3.2 Route dynamique avec des paramètres

Ajoutez une route qui accepte un paramètre dynamique dans l'URL.



```
Route::get('user/{id}', function ($id){
    return "User Id: $id";
});
```



127.0.0.1:8000/user/2

User Id: 2

3.3 Route avec un paramètre optionnel

Ajoutez une route qui accepte un paramètre optionnel.

```
Route::get('user/{id}/{group?}', function ($id, $group = 'default') {  
    return "User ID: {$id}, Group: {$group}";  
});
```

← → ↻ ⓘ 127.0.0.1:8000/user/2

User ID: 2, Group: default

3.4 Route préfixée

Ajoutez une route préfixée pour un groupe d'URLs sous un même préfixe (admin).

```
Route::prefix('admin')->group(function () {  
    Route::get('dashboard', function () {  
        return 'Admin Dashboard';  
    });  
  
    Route::get('settings', function () {  
        return 'Admin Settings';  
    });  
});
```

← → ↻ ⓘ 127.0.0.1:8000/admin/dashboard

Admin Dashboard

← → ↻ ⓘ 127.0.0.1:8000/admin/settings/

Admin Settings

4. Gérer les réponses et redirections

Objectif : Apprendre à retourner des réponses simples, des redirections et des réponses JSON sans vue ni contrôleur.

4.1 Retourner une réponse simple

Ajoutez une route qui retourne une réponse simple.

```
27 Route::get('/response', function () {
28     return "This is a simple response.";
29 });
30
```

← → ↻ ⓘ 127.0.0.1:8000/response

This is a simple response.

4.3 Retourner une réponse JSON

Ajoutez une route qui retourne une réponse JSON, utile pour les APIs.

```
Route::get('/json', function(){
    return response()->json(['name' => 'John', 'age' => 30]);
});
```

← → ↻ ⓘ 127.0.0.1:8000/json

Pretty-print ☐

{"name":"John","age":30}

5. Gérer les erreurs 404

Objectif : Apprendre à gérer les erreurs 404 avec la méthode fallback.

5.1 Route Fallback

Dans le fichier routes/web.php, ajoutez une route fallback pour gérer les erreurs 404.

```
Route::fallback(function () {
    return response("Page Not Found", 404);
});
```

← → ↻ ⓘ 127.0.0.1:8000/unknown

Page Not Found

6. Routes avec des conditions

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

← → ↻ ⓘ 127.0.0.1:8000/user/4

User ID: 4

← → ↻ ⓘ 127.0.0.1:8000/user/abc

Pretty-print ☐

```
{"error": "Page not found"}
```
