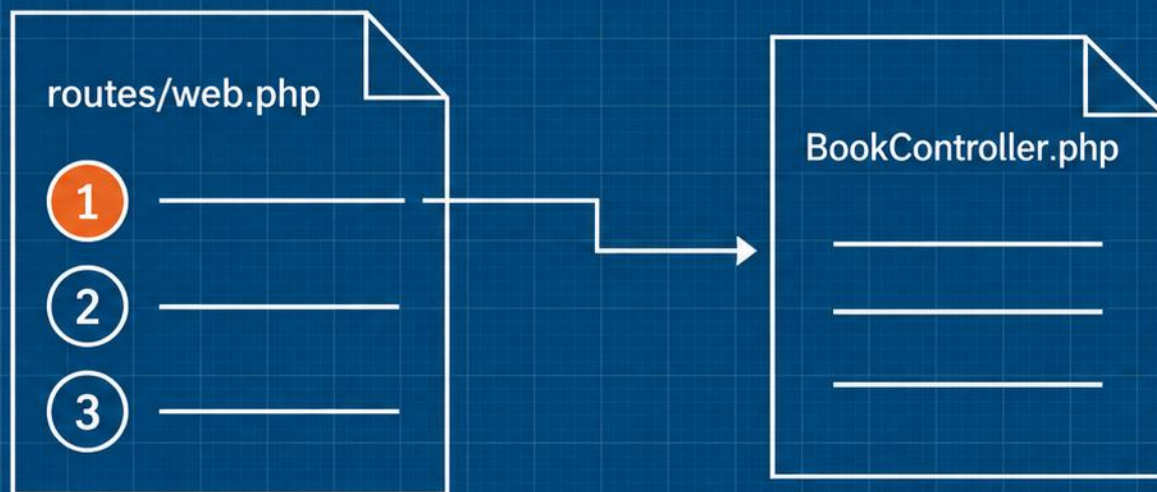


Bloc Laravel · chapitre 8

Controllers

Le code des routes passe dans une classe. Dans routes/web.php, chaque route tient sur une ligne.



Des fonctions anonymes au contrôleur

Un contrôleur est une classe dont les méthodes répondent aux requêtes.

```
1 $ php artisan route:list --path=livres
2 GET|HEAD livres ..... books.index > routes/web.php:16
3 POST livres ..... routes/web.php:20
4 GET|HEAD livres/{id} ..... books.show > routes/web.php:24
```



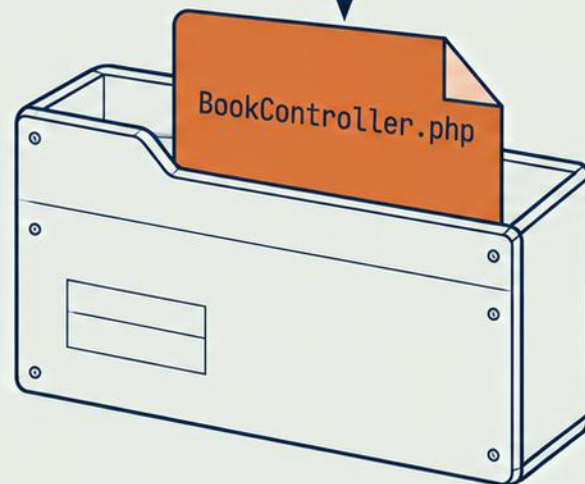
Avec une fonction de dix lignes par route, le code des pages finit par cacher les adresses.

Artisan crée le contrôleur

```
1 $ php artisan make:controller BookController
2 INFO Controller [app/Http/Controllers/BookController.php] created successfully.
```

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class BookController extends Controller
8 {
9     //
10 }
```

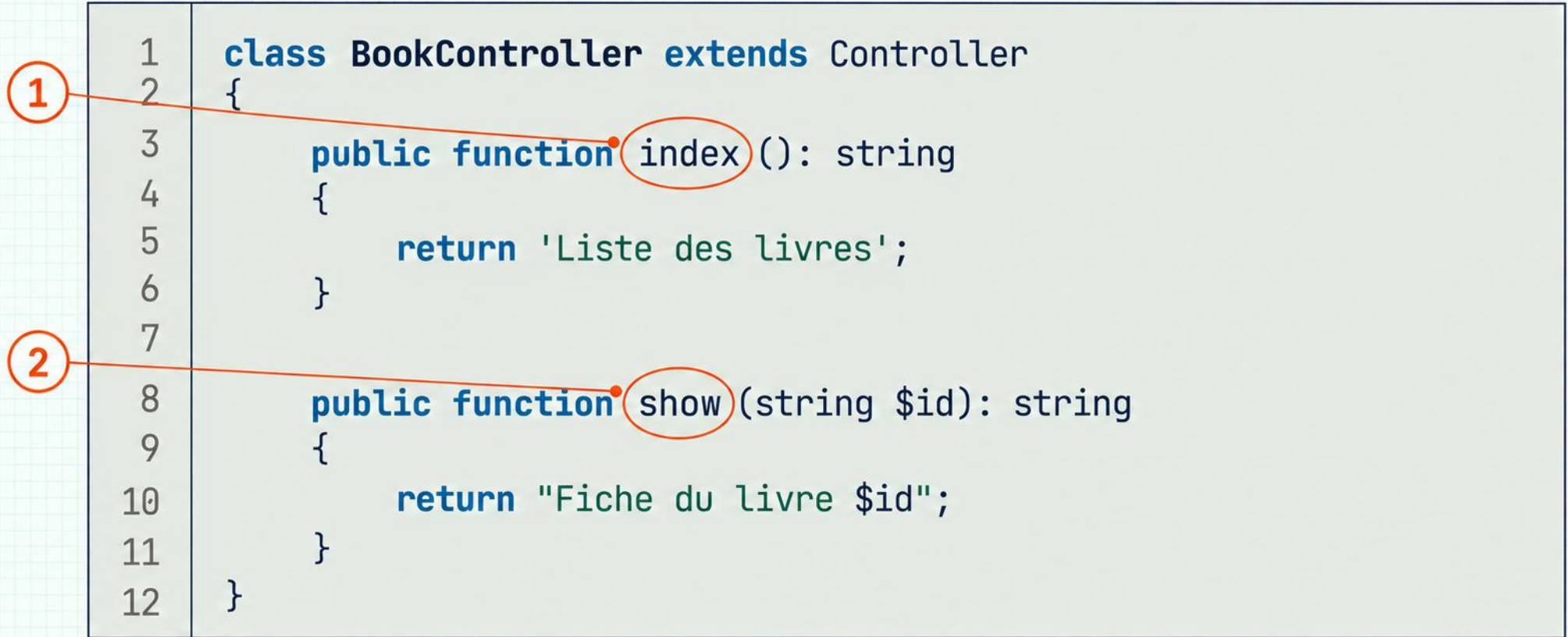
app/Http/Controllers/



Le namespace suit le dossier, règle PSR-4. Controller est une classe abstraite, et elle est vide.

Le code des routes devient des méthodes

```
1  class BookController extends Controller
2  {
3      public function index(): string
4      {
5          return 'Liste des livres';
6      }
7
8      public function show(string $id): string
9      {
10         return "Fiche du livre $id";
11     }
12 }
```



- Les deux méthodes renvoient les mêmes phrases que les fonctions anonymes du chapitre Routing.

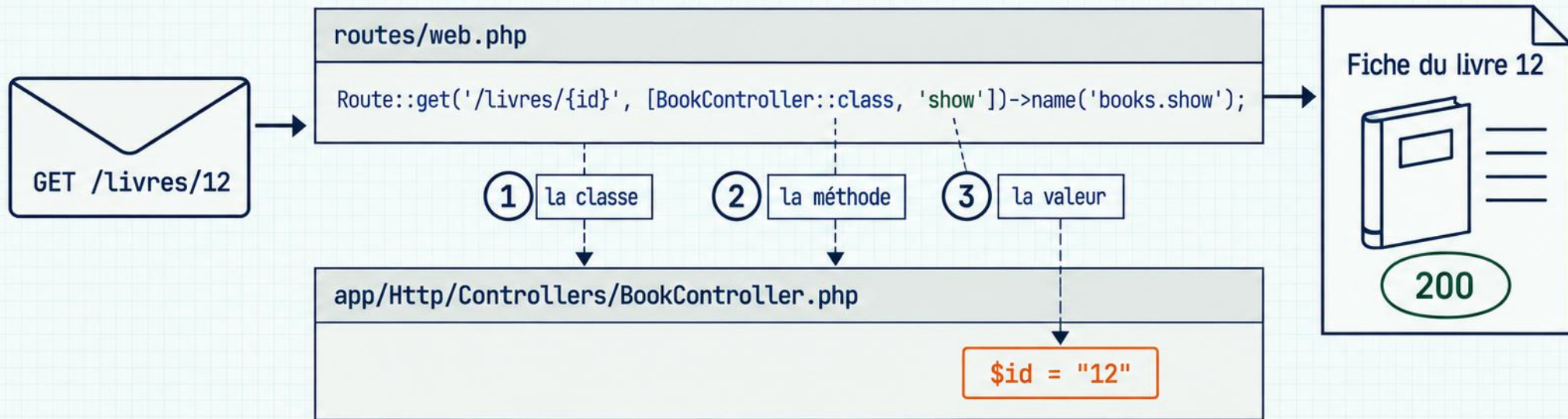
Importez la classe, puis désignez la méthode avec un tableau

```
1 use App\Http\Controllers\BookController;  
2 use App\Http\Controllers\PostController;  
3 use Illuminate\Support\Facades\Route;  
4  
5 // ...  
6 Route::get('/livres', [BookController::class, 'index'])->name('books.index');  
7 ...  
8 Route::get('/livres/{id}', [BookController::class, 'show'])->name('books.show');
```

The diagram consists of two orange lines. The first line starts from the `BookController` in the first line of code, goes down, then left, then down again to a small circle positioned above the `class` keyword in the `[BookController::class, 'index']` array on line 6. The second line starts from the `BookController` in the first line of code, goes down, then left, then down again to a small circle positioned above the `show` string in the `[BookController::class, 'show']` array on line 8.

Les routes gardent leur nom, `books.index` et `books.show`.

Laravel crée un objet BookController et appelle show()



`route:list` indique maintenant la classe et la méthode appelées.

```
1 $ php artisan route:list --path=livres
2 GET|HEAD  livres ..... books.index > BookController@index
3 POST      livres ..... routes/web.php:19
4 GET|HEAD  livres/{id} ..... books.show > BookController@show
```

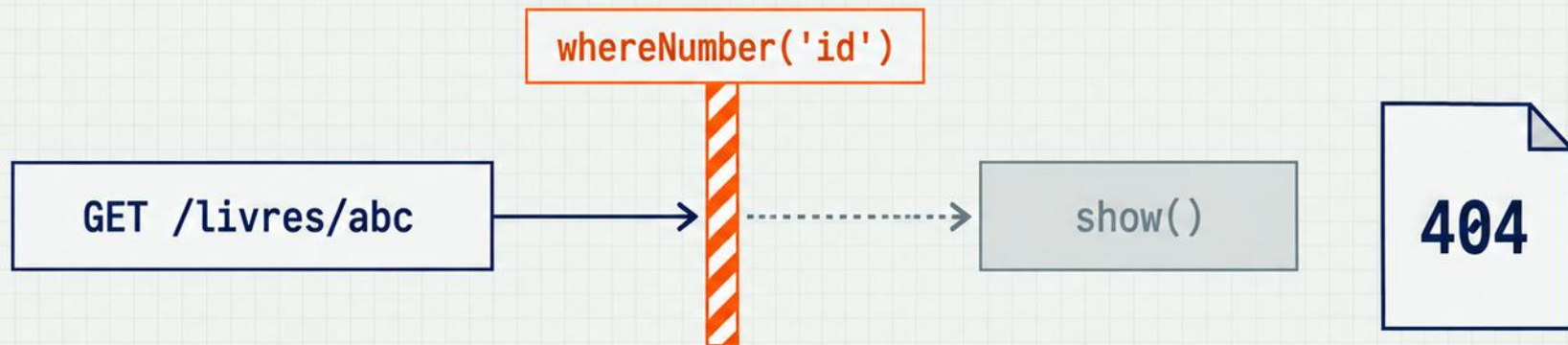
Quand une méthode déclare un paramètre int, ajoutez whereNumber à sa route

Avec `show(int $id)`, l'adresse `/livres/abc` répond 500 :

```
1 App\Http\Controllers\BookController::show(): Argument #1 ($id) must be of type int, string given
```

```
1 Route::get('/livres/{id}', [BookController::class, 'show'])->whereNumber('id')->name('books.show');
```

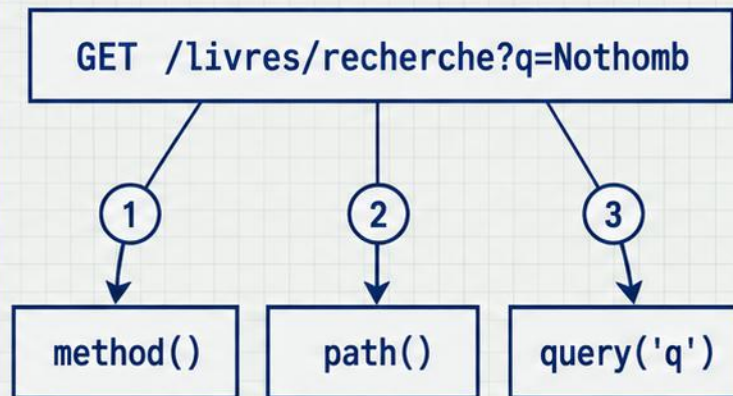
Avec la contrainte, `abc` ne correspond plus à la route : Laravel répond 404 sans appeler `show()`.



Request \$request donne la requête à la méthode

Quand un paramètre a le type Request, Laravel lui donne comme valeur l'objet qui représente la requête en cours.

```
1 public function search(Request $request): string
2 {
3     $word = $request->query('q', 'rien');
4
5     return "Recherche de « $word », requête {$request->method()}
6         sur {$request->path()}";
7 }
```



```
1 $ curl -s "http://premier-test-laravel.test/livres/recherche?q=Nothomb"
2 Recherche de « Nothomb », requête GET sur livres/recherche
3 $ curl -s http://premier-test-laravel.test/livres/recherche
4 Recherche de « rien », requête GET sur livres/recherche
```

Laravel fabrique la réponse à partir de ce que renvoie la méthode

La méthode renvoie	Type de retour	Réponse
"Fiche du livre \$id"	string	200, text/html, la phrase
view('books.links')	View	200, text/html, le HTML du template
redirect()->route('books.show', 12)	RedirectResponse	302, Location: .../livres/12
response()->json([...])	JsonResponse	200, application/json
['id' => 12, 'title' => 'Livre 12']	array	la même réponse

View, RedirectResponse et JsonResponse sont des classes de Laravel : importez-les en haut du contrôleur.

Un tableau PHP devient une réponse JSON

```
1 public function data(int $id): array
2 {
3     return ['id' => $id, 'title' => "Livre $id", 'available' => true];
4 }
```



200 · application/json

{"id":12,"title":"Livre 12","available":true}

```
1 $ curl -s -i http://premier-test-laravel.test/livres/12/donnees
2 HTTP/1.1 200 OK
3 ...
4 Content-Type: application/json
5 ...
6 {"id":12,"title":"Livre 12","available":true}
```

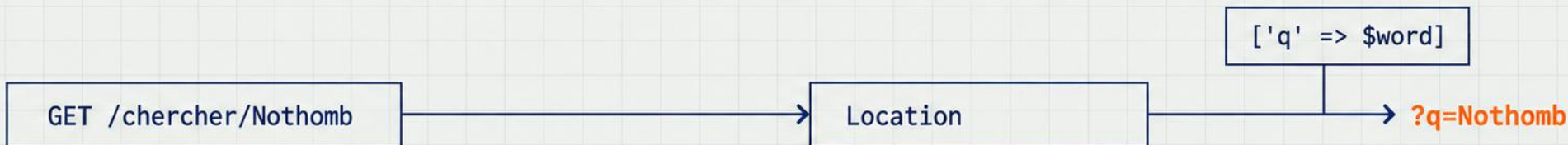
Le JSON est un texte structuré qu'un autre programme sait lire.

Rediriger vers une route, avec un paramètre d'URL

```
1 public function legacySearch(string $word): RedirectResponse
2 {
3     return redirect()->route('books.search', ['q' => $word]);
4 }
```

```
1 $ curl -s -I http://premier-test-laravel.test/chercher/Nothomb
2 HTTP/1.1 302 Found
3 ...
4 Location: http://premier-test-laravel.test/livres/recherche?q=Nothomb
5 ...
```

Quand vous donnez à `route()` une valeur dont le nom n'apparaît pas dans le chemin, Laravel l'ajoute à la fin de l'adresse comme paramètre d'URL



Écrivez le type de retour de chaque méthode

Sans return, index() déclarée : string répond 500 :

```
1 App\Http\Controllers\BookController::index(): Return value must be of type string, none returned
```

Sans : string non plus, Laravel envoie une page vide avec le code 200, et aucun message.

: string

500

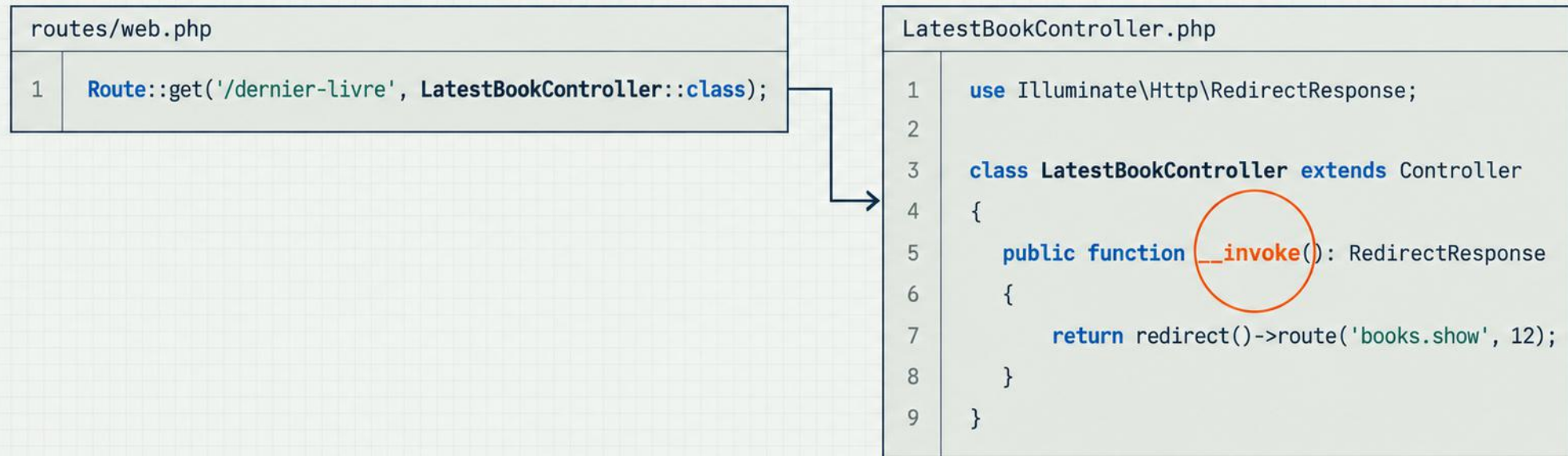
```
App\Http\Controllers\BookController::index():  
Return value must be of type string, none returned
```

200

NON CONFORME

__invoke : une classe, une seule action

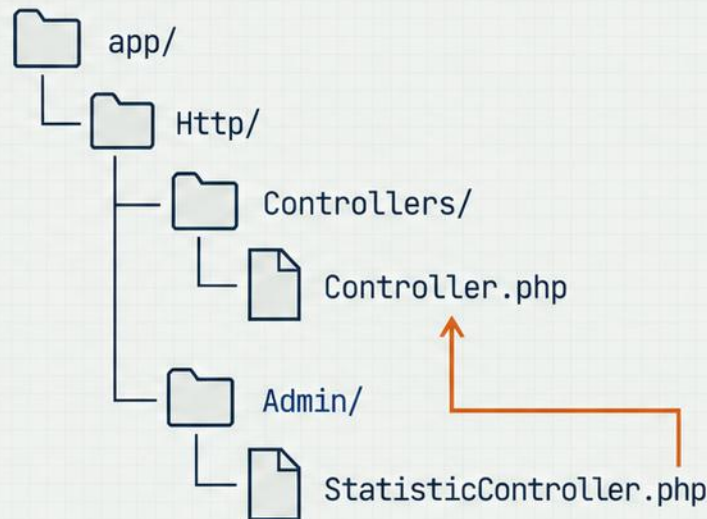
Une classe qui ne contient qu'une méthode, __invoke, s'appelle un contrôleur à action unique. Sa route donne la classe seule.



Ranger un contrôleur dans un sous-dossier

```
1 $ php artisan make:controller Admin/StatisticController
2 INFO Controller [app/Http/Controllers/Admin/StatisticController.php] created successfully.
```

```
1 namespace App\Http\Controllers\Admin;
2
3 use App\Http\Controllers\Controller;
4
5 use Illuminate\Http\Request;
6
7 class StatisticController extends Controller
8 {
9     //
10 }
```



Le namespace se termine par Admin, et Artisan importe Controller.
Le sous-dossier range le code, il ne protège pas la page.

Quand ça ne marche pas

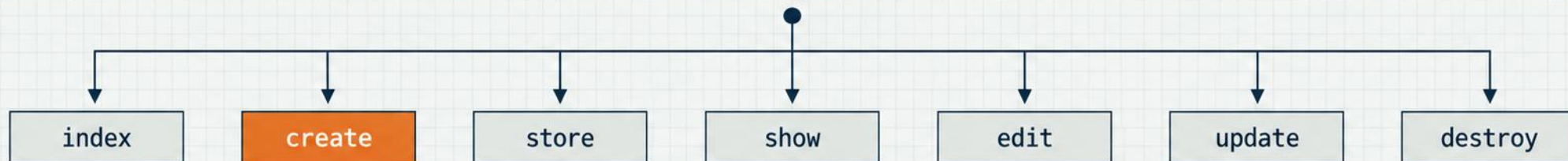


Message	Cause
Target class [BookController] does not exist.	import oublié dans routes/web.php
Call to undefined method App\Http\Controllers\BookController::list()	méthode absente du contrôleur
ERROR Controller already exists.	contrôleur déjà créé, fichier intact

`route:list` signale l'import oublié, mais pas la méthode absente.

Une ligne, sept routes, sept méthodes

```
1 Route::resource('recettes', RecipeController::class);
```



```
1 $ php artisan route:list --path=recettes
2 GET|HEAD      recettes ..... recettes.index > RecipeController@index
3 ...
4 GET|HEAD      recettes/create ..... recettes.create > RecipeController@create
5 GET|HEAD      recettes/{recette} ..... recettes.show > RecipeController@show
6 ...
```

Laravel déclare `recettes/create` avant `recettes/{recette}`. Les méthodes restent vides jusqu'au chapitre Blog.

À vous : trois contrôleurs et une lecture

70 minutes, en séance ou chez vous.

- Ex. 1 : le cinéma.
- Ex. 2 : le parking, `__invoke`.
- Ex. 3 : la météo, JSON et redirection.
- Ex. 4, sur papier : routes et erreurs.

