Développer en back-end

Gestion du routage

Formatrice : Elidrissi Asmae

OFPPT
La Voie de l'Avenir

Plan du cour

Créer Des Routes

Créer Des Routes

Définir une route dans Laravel

- Laravel offre un système de routage simple pour lier une URI à un code à exécuter.

- Les routes sont définies dans le fichier routes/web.php (fichier principal pour les routes du site).

Laravel propose plusieurs fichiers de routage par défaut :

- **api.php :** Routes pour l'API
- **channels.php :** Gestion des canaux de diffusion
- **console.php :** Routes de la console
- **web.php :** Routes normales du site web

Créer Des Routes

Exemple de route :

Fichier routes/web.php

```
use Illuminate\Support\Facades\Route;
Route::get('/greeting', function () {
return 'Bienvenue sur le site !';
});
```

Créer Des Routes

Les méthodes HTTP

Définition :

Le HTTP (Hypertext Transfer Protocol) est un protocole de communication entre un client (navigateur) et un serveur.

- Requête : Le client demande une ressource au serveur.
- Réponse : Le serveur envoie une réponse (souvent une page HTML).

Méthodes principales utilisées avec Laravel :

GET : Récupérer une ressource immuable.

– Exemple : Route::get($uri, function () { /* ... */ });

POST : Ajouter ou modifier une ressource (souvent pour les formulaires).

– Exemple : Route::post($uri, function () { /* ... */ });

PUT : Ajouter ou remplacer une ressource.

– Exemple : Route::put($uri, function () { /* ... */ });

DELETE : Supprimer une ressource.

– Exemple : Route::delete($uri, function () { /* ... */ });

Créer Des Routes

Particularités des méthodes HTTP

Parfois, vous devrez peut-être enregistrer une route qui répond à plusieurs verbes HTTP. Vous pouvez le faire en utilisant la méthode **match**. Ou, vous pouvez même enregistrer une route qui répond à tous les verbes HTTP en utilisant la méthode **any**

- Répondre à plusieurs méthodes HTTP :

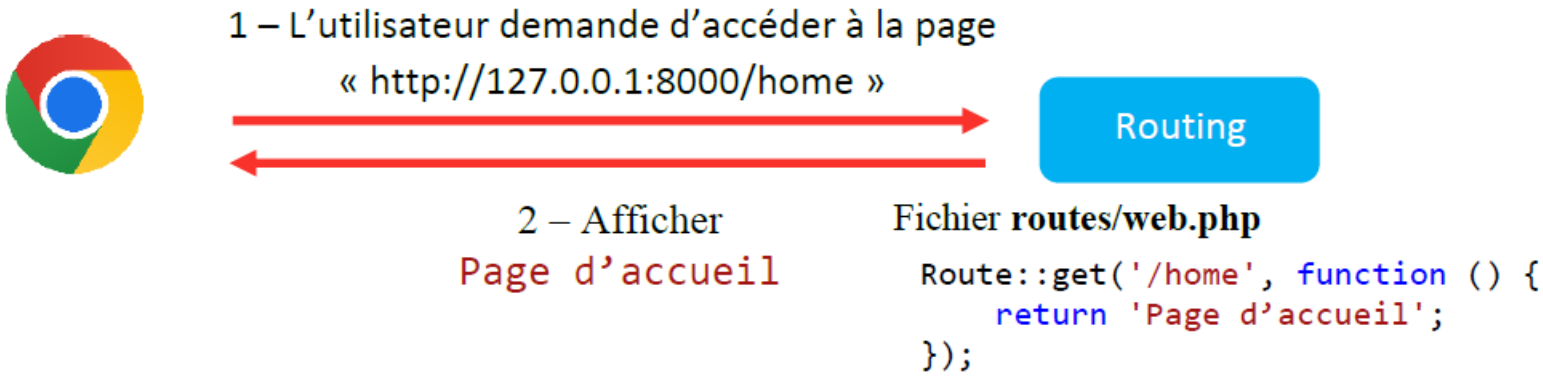
Utilisation de ‘match’ :

```
Route::match(['get', 'post'], '/', function () { /* ... */ });
```

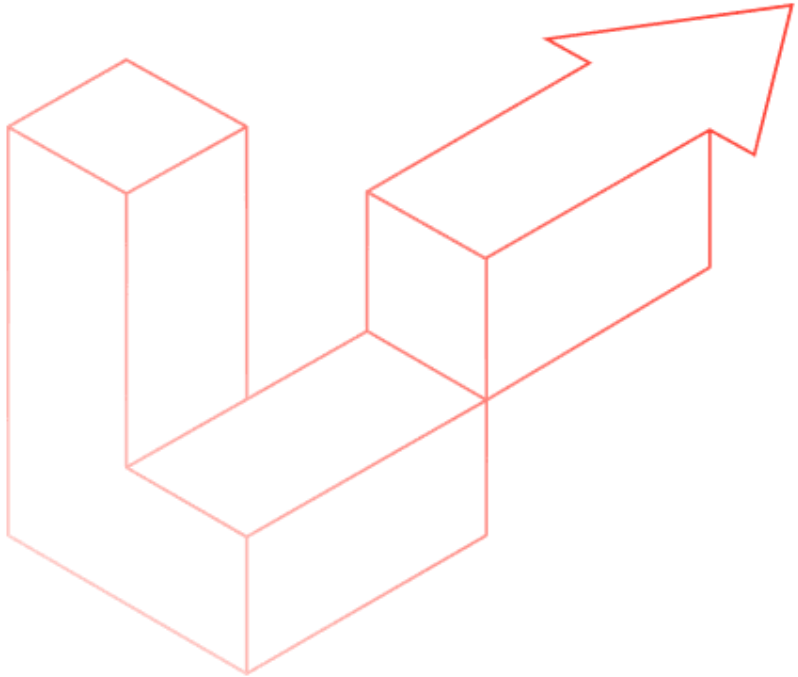
Répondre à tous les verbes HTTP avec ‘any’ :

```
Route::any('/', '/', function () { /* ... */ });
```

Créer Des Routes



Paramètres des routes



Paramètres des routes

❖ Déclarer des paramètres

A l'installation, Laravel a une seule route qui correspond à l'url de base. Voyons maintenant comment créer d'autres routes. Imaginons que nous ayons 3 pages qui doivent être affichées avec ces urls :

- ✓ http://127.0.0.1:8000/1
- ✓ http://127.0.0.1:8000/2
- ✓ http://127.0.0.1:8000/3

J'ai fait apparaître en gras la partie spécifique de l'url pour chaque page. Il est facile de réaliser cela avec ce code :

- ✓ `Route::get('1', function() { return 'Je suis la page 1 !'; });`
- ✓ `Route::get('2', function() { return 'Je suis la page 2 !'; });`
- ✓ `Route::get('3', function() { return 'Je suis la page 3 !'; });`

Paramètres des routes

- Utiliser les paramètres capturés

On peut utiliser un paramètre pour une route qui accepte des éléments variables en utilisant des accolades. Regardez ce code :

```
Route::get('{n}', function($n) {
return 'Je suis la page ' . $n . ' !';
});
```

Les paramètres des routes sont toujours entre crochets {} et doivent être composés de caractères alphabétiques et des tirets bas « _ ».

Paramètres des routes

- Utiliser plusieurs paramètres

Il est possible de créer des routes avec termes séparés par des barres obliques « / » et avec plusieurs paramètres.

```
Route::get('posts/{postId}/comments/{commentId}', function ($postId,
$commentId) {
/* ... */
});
```

Paramètres des routes

- Paramètres optionnels

Le point d'interrogation « ? » permet de rendre le paramètre optionnel.
Chaque variable qui est passé à la fonction de rappel de la route doit avoir une valeur par défaut. (Cas de paramètre optionnel)

```
Route::get('page/{n?}', function($n = null) {
if ($n === null)
return 'Je suis une page sans numéro !';
else
return 'Je suis la page ' . $n . ' !';
});
```

Paramètres des routes

- Routes nommées

Comme son nom l'indique, nous pouvons nommer les routes, ce qui permet de générer des URL ou des redirections pour des routes spécifiques.

Une façon simple de créer une route nommée est fournie par la méthode name enchaînée sur la façade Route. Le nom de chaque route doit être unique :

```
Route::get('/', function () { /*...*/ })->name("homepage");
```

Pour générer une URL en utilisant le nom de la route

```
$url = route('home'); ou <a href="{{ route('homepage') }}">
```

Réponses

Réponses

Toutes les routes et tous les contrôleurs doivent renvoyer une réponse à renvoyer au navigateur de l'utilisateur. Laravel propose plusieurs façons différentes de renvoyer des réponses.
La réponse la plus basique consiste à renvoyer une chaîne à partir d'une route ou d'un contrôleur. Le framework convertira automatiquement la chaîne en une réponse HTTP complète :

```
Route::get('/', function () {
return 'Bienvenue sur le site !';
});
```

En plus de renvoyer des chaînes à partir de vos routes et de vos contrôleurs, vous pouvez également renvoyer des tableaux. Le framework convertira automatiquement le tableau en une réponse JSON :

```
Route::get('/', function () {
return [1, 2, 3];
});
```

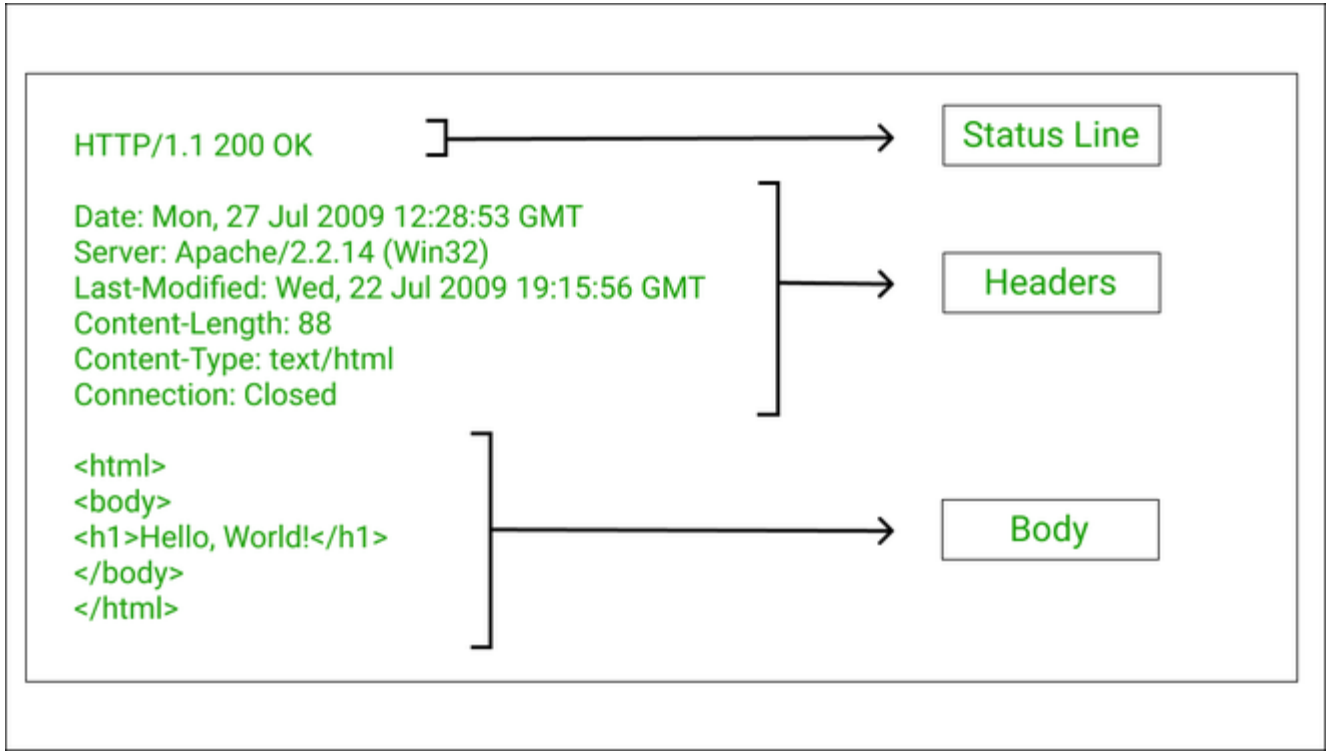
Réponses

- Les réponses textuelles

En règle générale, vous ne renverrez pas simplement de simples chaînes ou des tableaux à partir de vos actions de routage. Au lieu de cela, vous renverrez des instances **Illuminate\Http\Response** ou des vues complètes.

Le renvoi d'une instance **Response** complète vous permet de personnaliser le code d'état HTTP et les en-têtes de la réponse. Une instance **Response** hérite de la classe **Symfony\Component\HttpFoundation\Response**, qui fournit diverses méthodes pour créer des réponses HTTP :

```
Route::get('/', function () {
return response('Bienvenue sur le site !', 200)
->header('Content-Type', 'text/plain');
});
```


HTTP/1.1 200 OK
Status Line
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
Headers
<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
Body

HTTP STATUS CODES

2xx Success

- 200 Success / OK

3xx Redirection

- 301 Permanent Redirect
- 302 Temporary Redirect
- 304 Not Modified

4xx Client Error

- 401 Unauthorized Error
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed

5xx Server Error

- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout

Réponses

- Les réponses JSON

La méthode **json** définira automatiquement l'en-tête Content-Type sur **application/json**, ainsi que convertira le tableau donné en JSON à l'aide de la fonction **json_encode** PHP :

```
Route::get('/', function () {
return response()->json([1, 2, 3]);
});
```

Réponses

- Les redirections

La fonction **redirect()** peut recevoir un paramètre pour lui indiquer à quel endroit elle doit effectuer la redirection. Pour effectuer une simple redirection :

```
Route::get('ancienne', function () {
return redirect('nouvelle');
});
```

ou

```
Route::redirect('ancienne', 'nouvelle’);
```

Par défaut, **Route::redirect** renvoie un **302** code d'état. Vous pouvez personnaliser le code d'état à l'aide du troisième paramètre facultatif :

```
Route::redirect('ancienne', 'nouvelle', 301);
```

Réponses

- Les redirections

Redirection vers des actions du contrôleur

Vous pouvez également générer des redirections vers les actions du contrôleur. Pour ce faire, passez le contrôleur et le nom de l'action à la méthode action :

```
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\IndexController;

Route::get('/', [IndexController::class, 'index']);
```

Groupes de routes

Groupes de routes

❖ **Préfixes de routes**

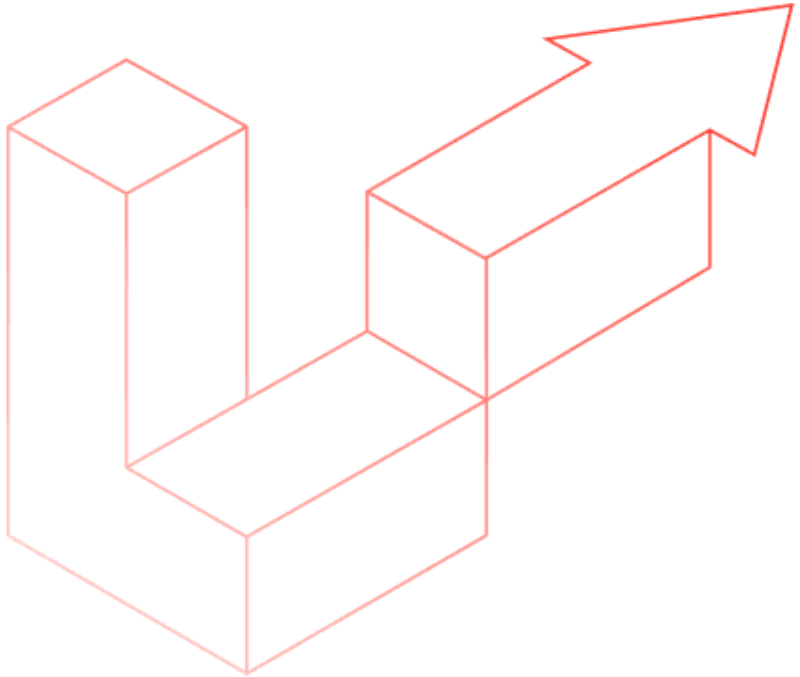
La méthode **prefix** peut être utilisée pour préfixer chaque route du groupe avec un URI donné. Par exemple, vous pouvez préfixer tous les URI suivants :

```
Route::get('/admin/stats', function () { /* ... */ });
Route::get('/admin/users', function () { /* ... */ });
Route::get('/admin/logs', function () { /* ... */ });
```

Au sein du groupe avec **admin** :

```
Route::prefix('admin')->group(function () {
Route::get('/stats', function () {
// Matches The "/admin/stats" URL
});
Route::get('/users', function () {
// Matches The "/admin/users" URL
});
Route::get('/logs', function () {
// Matches The "/admin/logs" URL
});
});
```

Routes et vues



Routes et vues

- Les vues sont les fichiers .blade.php utilisés pour rendre le frontend de Laravel.
- Elles utilisent le moteur de templating Blade.
- C'est la manière par défaut de construire une application full-stack avec Laravel.

```
Route::view('/', 'home');
ou
Route::get('/', function () {
return view('home');
});
```

Routes et vues

- Passer des paramètres à la vue

Pour passer des paramètres à la vue, vous pouvez utiliser un tableau.

```
Route::view('/', 'home', ['name' => "ALLAOUI"]);
ou
Route::get('/', function () {
return view('home', ['name' => "ALLAOUI"]);
});
```

Le paramètre name sera accessible dans la vue avec {{ $name }} ou {!! $name !!}.

Note importante : Si le paramètre est requis dans la vue et qu'il est manquant, la requête échouera et une erreur sera lancée.

Commandes utiles

Laravel Artisan

- Les commandes utiles de Laravel Artisan:

1. Afficher toutes les routes définies par l'application

```
php artisan route:list
```

2. Afficher le middleware de chaque route

```
php artisan route:list -v
```

3. Filtrer les routes par URI spécifique (par exemple, API)

```
php artisan route:list --path=api
```

4. Masquer les routes des packages tiers

```
php artisan route:list --except-vendor
```

5. Générer un cache de routes pour améliorer les performances

```
php artisan route:cache
```

6. Après ajout de nouvelles routes, régénérez le cache

```
php artisan route:cache
```

Vider le cache des routes

```
php artisan route:clear
```


(count($workerpro
$test_files = array_
$sequentialTests = 1;
$files = [];
$maxBatchSize = $valgrind ? 1 :
$averageFilesPerWorker = max(1,
$batchSize = min($maxBatchSize,
while (count($files) <= $batchSize
foreach ($fileConflictsWith
if (isset($activeConflicts
$waitingTests[$conflictKey]
continue 2;
$files[] = $file;
as $file) {

TP

À la prochaine !