2 minutes

Priorité des routes

L'ordre de définition des routes

Symfony évalue les routes dans l'ordre duquel elles sont déclarées.

Cela signifie que par défaut, lorsqu'une requêtes `HTTP` est reçu, le `Router` va regarder la première route déclarée, si elle match il exécute immédiatement le contrôleur correspondant sans regarder les autres routes.

Si la route ne correspond pas, il va regarder la seconde route. Si elle match il exécute immédiatement le contrôleur correspondant sans regarder les routes suivantes et ainsi de suite.

Définir une priorité

Ce que nous avons vu ci-dessus est le comportement par défaut. Comme nous l'avons vu dans la leçon précédente, nous pouvons modifier celui-ci en ajoutant une propriété `requirements` pour faire varier les correspondances en ajoutant des validations sur les paramètres.

Nous pouvons également ajouter une propriété `priority` qui va faire varier la priorité des routes : **`Symfony` regardera d'abord les routes par ordre de priorité et non par ordre de déclaration**.

C'est particulièrement utile lorsque nous utilisons la définition de route par attribut. En effet, il est plus difficile de savoir l'ordre de déclaration car les routes sont dans les fichiers des contrôleurs et sont donc séparées. Définir des priorités permet de remédier à ce problème.

Par défaut, toutes les routes ont une priorité de 0. Il suffit donc d'attribuer une priorité de 1 pour que la route soit prioritaire.

En reprenant notre exemple :

```
<?php

namespace App\Controller;

use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController
{
  #[Route(path: '/', name: 'index', methods: ['GET'], schemes: ['https'])]
  public function index(Request $request)
  {
    dd($request);
    return new Response('<h1>Hello World !</h1>');
  }

  #[Route(path: '/blog/{name}', name: 'blogName')]
  public function blogByName(string $name = 'all')
  {
    return new Response('<h1>Blog NAME</h1>' . $name);
  }

  #[Route(path: '/blog/{id}', name: 'blogId', priority: 1, requirements: ['id' => '\d+'])]
  public function blogById(int $id)
  {
    return new Response('<h1>Blog ID</h1>' . $id);
```

```
    }
}
```

Copier

Ici nous indiquons que la route `blogId` sera toujours prioritaire sur les autres routes de priorité 0.

Lorsqu'une requête sera reçue, le `Router` va donc vérifier si elle match en premier.

Si le paramètre ne respecte pas `requirements: ['id' => '\d+']`, c'est-à-dire si ce n'est pas un nombre entier, alors le `Router` essaiera la route suivante.

Il examinera les routes qui ont une priorité 0 dans l'ordre de déclaration.

Essayez `https://127.0.0.1:8000/blog/123` et `https://127.0.0.1:8000/blog/test`.