4 minutes

Introduction aux routes Symfony

Activer les annotations

Depuis `PHP 8`, une nouvelle fonctionnalité du langage, appelée `attributes` permet d'annoter les classes, les méthodes, les fonctions, les paramètres, les propriétés et les constantes de classe.

Ces attributs permettent à un programme (comme `Symfony`) de les lire et de modifier le code lors de l'exécution.

Les attributs s'écrivent avec un dièse puis des crochets :

```
<?php

#[Attribut]
class UneClasse {}
```

Copier

`Symfony` utilisait un autre système appelé "annotations" et qui obligeait à installer une librairie pour les traiter.

Depuis la version 8 de `PHP`, `Symfony` utilise directement les attributs `PHP` et il n'est plus nécessaire d'installer de librairie.

Pour activer les annotations, il suffit de commenter tout ce qui se trouve dans `dymaproject/config/routes.yaml` et de créer un fichier `dymaproject/config/routes/attribute.yaml`.

Le nom du fichier n'est pas important, seul son contenu et son emplacement importe.

Il faut qu'il contienne :

```
controllers:
  resource: ../../src/Controller/
  type: annotation
```

Copier

Créer une route avec les attributs

De retour dans `DefaultController.php` :

```
<?php

namespace App\Controller;

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

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

```
  }

  #[Route('/blog', name: 'blog')]
  public function blog()
  {
    return new Response('<h1>Blog</h1>');
  }
}
```

Copier

Pour créer une route en utilisant le composant `Routing`, créé par `Symfony`, il suffit d'ajouter un attribut qui contient **une instance de `Route()` avec en premier argument le chemin de la route et en second un argument nommé `name` qui contient le nom de la route**.

Relancez votre serveur et essayez d'accéder à `/` et `/blog` :

```
symfony serve
```

Copier

Si vous n'avez que la page par défaut, pas de panique ! Cela signifie simplement que le cache n'a pas pris en compte la nouvelle configuration des routes (nous avons vu que `Symfony` mettait toute la configuration dans en cache dans `var/cache`).

Il faut donc **invalider le cache** pour prendre en compte la nouvelle configuration :

```
symfony console cache:clear
```

Copier

Vous n'avez plus qu'à relancer le serveur :

```
symfony serve
```

Copier

La classe `Route`

Le routing permet beaucoup de choses, il n'y a qu'à se pencher sur le constructeur de `Route` :

```
<?php

public function __construct(
  $data = [],
  $path = null,
  string $name = null,
  array $requirements = [],
  array $options = [],
  array $defaults = [],
  string $host = null,
  $methods = [],
  $schemes = [],
  string $condition = null,
  int $priority = null,
  string $locale = null,
  string $format = null,
  bool $utf8 = null,
  bool $stateless = null,
  string $env = null
)
```

Copier

Notez que le premier argument est `$data` et non `$path`. Pourtant, nous pouvons créer une `Route()` en précisant le chemin en premier argument.

La réponse est juste après dans le constructeur :

```
<?php

if (\is_string($data)) {
  $data = ['path' => $data];
}
```

Copier

Si le premier argument est une chaîne de caractères, il sera automatiquement transformé en tableau associatif `['path' => $data]` et interprété comme chemin.

Nous vous recommandons d'utiliser les paramètres nommés pour plus de clarté. Utilisez donc :

```
<?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')]
  public function index(Request $request)
  {
    dd($request);
    return new Response('<h1>Hello World !</h1>');
  }

  #[Route(path: '/blog', name: 'blog')]
  public function blog()
  {
    return new Response('<h1>Blog</h1>');
  }
}
```

Copier

Dans les prochaines leçons, nous verrons toutes les possibilités permises par le composant `Routing` de `Symfony`.