Chapitre 4
Routage et contrôleur

1. Fonctionnement du routage dans Symfony

1.1 Définition

Comme nous l'avions vu dans le chapitre sur l'architecture du framework, Symfony n'utilise pas la mise en relation entre l'URL et le système de fichiers pour servir les pages web.

Les URL sont gérées par le routage (composant **Router** du framework). Le rôle de ce composant est de trouver l'action à exécuter pour une requête donnée et pour cela, il s'appuie sur un ensemble de règles de routage définies par le développeur.

Remarque

*Si vous êtes familier des systèmes de réseaux, vous avez probablement déjà entendu parler de ce terme, voire du **routeur**. Dans un contexte applicatif, cela correspond à l'**action** sélectionnée pour une **requête** donnée ; les actions sont contenues dans des contrôleurs sous forme de méthodes.*

1.2 Le répertoire public et le contrôleur frontal

À la racine de votre projet, vous avez un répertoire nommé **public**. Il contient tous vos fichiers publics. Ce sont typiquement des images, des feuilles de style CSS, des fichiers de scripts JavaScript, et, plus largement, tous les fichiers destinés à être servis directement par le serveur web.

Pour l'URL **http://monjournal.local/robots.txt** (ou bien **http://localhost:8000/robots.txt** si vous utilisez le serveur web intégré de PHP), le fichier **robots.txt** du répertoire **public** sera servi.

Tandis que pour **http://monjournal.local/hello/world**, comme le dossier **public** ne contient pas de sous-dossier **hello** avec, à l'intérieur un fichier **world**, le contrôleur frontal est invoqué et c'est votre application qui, après avoir analysé la requête HTTP, peut décider de retourner une réponse, dont le corps serait « Hello world! ».

Pour ce faire, le Kernel fait appel au composant Router (cf. Architecture du framework - Architecture de Symfony pour un schéma explicatif).

1.3 Une requête, une action

Le rôle du routage est donc de sélectionner l'action à invoquer selon un ensemble de règles. Ces règles sont en rapport avec la requête HTTP envoyée par le client, requête qui est la seule entité permettant au routage de pouvoir faire son choix.

Les principales règles de routage sont :

– la méthode de la requête (GET, POST, etc.)

– le *path* de l'URL (http://www.website.com**/hello/world**)

– l'hôte de l'URL (http://**www.website.com**/hello/world)

Pour le *path* et l'hôte, un système similaire aux REGEX (expressions régulières) est disponible et permet de définir des URL dynamiques.

Nous allons maintenant aborder les différentes règles de routage au travers d'exemples.

2. Définition des routes

Une route est une règle de routage. Chaque route est constituée de différentes règles, et pointe vers une action donnée.

Par défaut, le fichier de routage de l'application est **config/routes.yaml**, celui-ci étant complété par les fichiers se trouvant dans **config/routes/** ; dans ce dernier, on trouve notamment le fichier **annotations.yaml** permettant d'activer la configuration du routage via des annotations :

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

kernel:
   resource: ../../src/Kernel.php
   type: annotation
```

2.1 Les différents formats de définition

Comme évoqué dans le chapitre Mise en place d'un projet Symfony, section Structure de l'application, il existe différentes manières de définir la configuration d'une application Symfony : via des **annotations** ou les **attributs PHP** mais également par des **fichiers de configuration**. Plusieurs formats sont disponibles pour les fichiers de configuration : **YAML** (*YAML Ain't Markup Language*), **XML** (*eXtensible Markup Language*) ou PHP.

En ce qui concerne le routage, ce sont principalement les **annotations** ou les **attributs PHP** (**si l'application fonctionne sur PHP 8**) et les fichiers au format **YAML** qui sont utilisés, les formats **XML** et **PHP** étant, de manière générale dans Symfony, de plus en plus délaissés.

Les informations obligatoires pour la définition des routes sont :

– le path : il correspond à l'URI de l'application sollicitée ;
– l'action : c'est la méthode d'un contrôleur à laquelle sera associé le path ; l'exécution sera donc déclenchée par la réception d'une requête à ce path.

Pour la configuration avec des annotations, on utilise **@Route** sur les actions à exécuter ; la valeur définie dans cette annotation est le path :

```
/**
 * @Route("/")
 */
public function index(): Response
{
    ...
}
```

Avec les attributs de PHP 8, la définition ressemble énormément à la précédente :

```
#[Route("/")]
public function index(): Response
{
    ...
}
```

Lors d'une configuration en YAML, il est nécessaire de donner un nom à la route (avec les annotations, un nom par défaut est attribué) et le nom de l'action du contrôleur, en plus du path :

```
index:
    path: /
    controller: App\Controller\DefaultController::index
```

Ici, la clé de premier niveau **index** représente le nom de la route ; le path et la spécification de l'action doivent être écrits avec un décalage de plusieurs espaces (quatre en général) par rapport au nom de la route.

En plus de ces informations obligatoires, d'autres, optionnelles, peuvent être ajoutées.

2.2 Les options sur la définition des routes

Selon le format de configuration des routes, les options complémentaires de configuration peuvent revêtir la forme de clés supplémentaires dans le fichier YAML ou bien d'attributs nommés dans l'annotation **@Route** ou l'attribut **#[Route]**. Par exemple, il est possible d'indiquer le verbe HTTP (GET, POST…) par lequel l'action sera sollicitée. Voici la définition en annotation pour limiter à une invocation en HTTP GET et POST :

```
/**
 * @Route("/", methods={"GET","POST"})
 */
public function index(): Response
{
    ...
}
```

Avec les attributs PHP :

```
#[Route("/", methods: ["GET","POST"])]
public function index(): Response
{
    ...
}
```

Et la même chose via un fichier YAML :

```
index:
    path: /
    controller: App\Controller\DefaultController::index
    methods: GET|POST
```

Les options de configuration du routage seront présentées dans la suite de ce chapitre.

3. Configurer le path

3.1 Illustration par l'exemple : /hello/world

La règle de routage la plus importante est le **path** ; elle correspond à la variable superglobale **$_SERVER['PATH_INFO']**.

Configurons une route pour un path donné : **/hello/world**.

Annotations ou attributs PHP

Selon les définitions du chapitre Architecture du framework - Le modèle de conception MVC, les actions sont des méthodes de classe appelées « contrôleur » et c'est donc au-dessus de ces actions que nous devons placer notre règle de routage sous la forme d'annotation ou d'attribut :

```
namespace App\Controller;

use
Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class WelcomeController extends AbstractController
{
   /**
    * @Route("/hello/world")
    */
   public function hello()
   {
       return new Response('Hello world!');
   }
}
```

Avec les attributs, nous utiliserions :

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;

class WelcomeController extends AbstractController
```

```
{
    #[Route("/hello/world")]
    public function hello()
    {
        return new Response('Hello world!');
    }
}
```

À noter qu'avec les attributs, il n'y a pas besoin d'utiliser une quelconque directive **use** pour rendre visible une classe !

Ici, l'action **hello** sera exécutée pour toute requête dont le path est **/hello/world**, cette règle de routage étant définie grâce à l'annotation **@Route** ou l'attribut PHP.

Pour que le routage du contrôleur soit effectif, il faut qu'il soit activé dans le fichier **config/routes/annotations.yaml**. Pour nous faciliter la tâche, Symfony autorise la configuration de tous les contrôleurs d'une application en référençant un dossier en tant que ressource ; c'est d'ailleurs la configuration par défaut présente dans ce fichier :

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

Une fois les règles de routage configurées, le fait de demander l'URL http://monjournal.local/hello/world au travers de votre navigateur affiche une page dont le contenu est « Hello world! ».

Le nom des routes avec les annotations et les attributs

Nous allons voir ci-dessous avec les autres formats qu'un nom doit être assigné aux routes. Avec les annotations ou les attributs, ce n'est pas obligatoire car un nom est généré automatiquement.

Ce nom suit le format : **bundle_controleur_action**. Dans l'exemple précédent, le nom de la route est donc : **app_welcome_hello**.