Partie 1 — cours 2:
Le Routeur de Symfony

Routeurs, Controleurs, Moteur de template Twig, et console symfony

On va afficher rapidement notre 1ère page, pour illustrer, et ensuite à partir des slides suivants on va voir en détail chaque partie (Routeur, Contrôleur, Vue Twig…)

1/ Routeur = Correspondance entre URL et des paramètres (Controleur) => config/routes.yaml (pour l'exemple)

2/ Contrôleur = Le chef d'orchestre, dans => « src/Controller » (ça fonctionne déjà si on affiche la route)

3/ Template Twig, pour la vue (installer twig)

Le Routeur de Symfony

- Le Routeur défini quelle action appeler pour chaque URL entrante.
- Il communique avec le Kernel, en lui renvoyant le Contrôleur et paramètres correspondant a l'URL.
- SEO-friendly URLs : les URLs générées et utilisées sont plus optimisées pour le référencement.
- Le Routeur peut être configuré via un fichier YAML, XML, PHP ou un système d'Annotations.

Comme nous l'avons vu dans le schéma précédent, une URL est envoyée au « Contrôleur frontal » => Le système de routing (le Routeur) est celui qui va identifier quelle action appeler en fonction de l'URL entrante

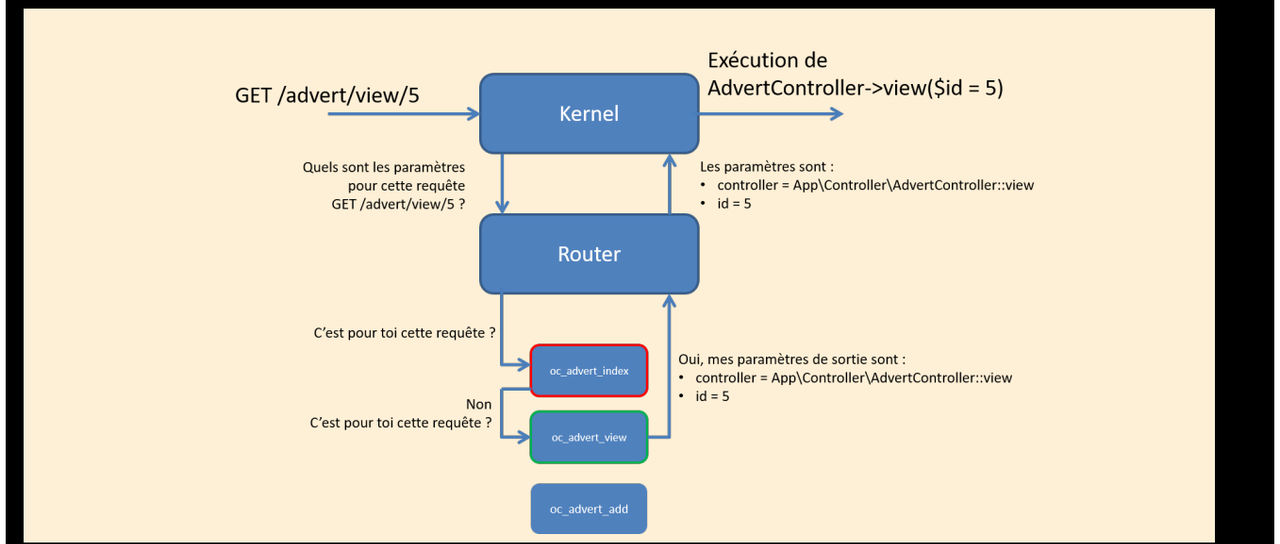
Le rôle du routeur est donc de trouver la bonne route qui correspond à l'URL appelée, et de retourner les paramètres de cette route.

Le Routeur connait donc le Contrôleur, avec les paramètres, a executer lorsque une requête HTTP a été soumise…

SEO-friendly URLs : les URLs générées et utilisées sont plus optimisées pour le référencement. => (/read/intro-to-symfony au lieu de index.php?article_id=57)

Vos routes peuvent être configurée via un fichier YAML, XML, PHP ou Annotations : même si Symfony recommande le système par annotations il n'y a aucune différence dans le fonctionnement interne.

Le Routeur



Le routeur a la liste de toutes les routes définies dans l'application.

A la demande d'une route (exemple ici : /advert/view/5), il va parcourir sa liste et renvoie la première correspondance, si elle existe, au Kernel (qui se chargera par la suite d'appeler le bon Contrôleur et faire le reste du travail).

Vous pouvez afficher la liste de vos routes avec la commande :
`php bin/console debug:router`

Les routes avec YAML

Créer des routes via la configuration YAML

```yaml
# config/routes.yaml
index:
    path: /
    controller: App\Controller\DefaultController::index

advert_add:
    path: /advert/add
    controller: App\Controller\DefaultController::add
```

- Dans notre version de symfony (4), la configuration se trouve dans : config/routes.yaml
- Attention les fichiers YAML sont stricts, veuillez a respecter les 4 espaces (et non tabulation).
- Pour chaque route est associé 1 nom unique, 1 URL (path) et 1 Contrôleur avec son action .

Ici dans notre exemple, nous avons défini 2 routes (index et advert_add):

advert_add : le nom unique et interne de la route. Il sert a identifier la route dans le code et doit être unique (c'est son id).
path : l'URL sollicité par la Request (ici : /advert/add)
controller : l'action (ici : add) contenu dans le contrôleur (ici : DefaultController) associée à la route.

Les routes avec YAML

Créer des routes via YAML avec des paramètres

```yaml
# config/routes.yaml
blog_show:
    path:       /blog/{slug}
    controller: App\Controller\BlogController::show

blog_list:
    path:       /blog/{page}
    controller: App\Controller\BlogController::list
    requirements:
        page: '\d+'
```

URL	Route	Parameters
`/blog/2`	`blog_list`	`$page = 2`
`/blog/my-first-post`	`blog_show`	`$slug = my-first-post`

- Pour ajouter des paramètres, on utilise des accolades {} dans l'URL
- Les paramètres seront passés aux actions du contrôleur
- Il est possible d'ajouter un certains nombre d'options, comme la contrainte requirements.

Dans cette exemple, on a ajouté un paramètre a notre URL, et afin de différencier les 2 routes, on indique que pour la deuxième (blog_list) le paramètre doit être un entier positif (expression régulière '\d+')

Il existe un certain nombre d'autres options, comme requirements et qui qui permettent de travailler avec les URLs, default (valeur par défaut), methods (restreindre les méthodes HTTP), condition etc.
=> https://symfony.com/doc/4.4/routing.html

Les routes via Annotations

Créer des routes via les Annotations

- composer require annotations
- Les annotations se définissent directement dans les contrôleurs - contrairement au fichier YAML

```php
class BlogController extends AbstractController
{
    /**
     * @Route("/blog", name=« blog_list")
     */
    public function list()
    {
        // …
    }
}
```

- A la demande de l'URL /blog, l'application exécutera l'action list() du contrôleur BlogController.

Installer le système d'annotations : composer require annotations.

L'installation configurera (grace à la recette Flex) le fichier config/routes/annotations.yaml

Cette configuration, dans notre exemple, défini une route nommée blog_list qui est liée à l'URL /blog.

Lorsqu'une requête sera faite pour /blog (dans le navigateur), l'application exécutera l'action list() du contrôleur BlogController.

Les routes via Annotations

Ajouter des paramètres

```
class BlogController extends AbstractController
{
   /**
    * @Route("/blog/{param}", name=« blog_list")
    */
    public function list($param)
    {
        // …
    }
}
```

- L'URL /blog/un-param, l'application exécutera l'action list($param) du contrôleur BlogController.
- Symfony exécutera l'action list() avec le paramètre $param='un-param'
- Plusieurs paramètres peuvent être définis pour votre route /blog/{id}/page/{param}/{other}

Les routes via Annotations

Ajouter des contraintes sur les paramètres

```php
class BlogController extends AbstractController
{
    /**
     * @Route("/blog/{param}", name="blog_list", requirements={"param"="\d+"})
     */
     public function list(int $param)
     {
         // …
     }
    /**
     * @Route("/blog/{slug}", name="blog_show")
     */
     public function show($slug)
     {
         // …
     }
}
```

- Une route est unique, le système Symfony doit pouvoir les différencier

Les routes blog_list d'URL /blog/{param} et blog_show d'URL /blog/{slug}, sont semblables si nous ne les différencions pas avec des contraintes sur les paramètres. Ici en indiquant que {param} doit être un entier positif (\d+), Symfony comprendra que pour l'URL /blog/10 est bien la route blog_list qui doit être appelée et non blog_show.

Dans le cas de plusieurs routes identiques, Symfony exécute la première route rencontrée.

Les routes via Annotations

Paramètres par défaut

```php
class BlogController extends AbstractController
{
    /**
     * @Route("/blog/{param?10}", name="blog_list", requirements={"param"="\d+"})
     */
     public function list(int $param)
     {
         // …
     }
    /**
     * @Route("/blog/{slug?}", name="blog_show")
     */
     public function show($slug)
     {
         // …
     }
}
```

- Il existe plusieurs façons de déclarer des valeurs par défaut a vos paramètres.

Il existe plusieurs manière de déclarer des valeurs par défaut :

Dans la méthode du contrôleur : public function list(int $param = 10)
Dans l'annotation, avec une option : defaults={"param" = 10}
Dans l'annotation : /blog/{param?10}

=> ici Met la valeur 10 par défaut au paramètre.

/blog/{param?} : affecte à null par défaut

La documentation concernant le Routing avec Symfony:
https://symfony.com/doc/4.4/routing.html

Est-ce que vous avez des questions ?