Symfony 5 : contrôleur

Nizar Rouatbi

Technologue en informatique à Iset Sousse

nizar.rouatbi@gmail.com

Plan

Symfony

Rôle

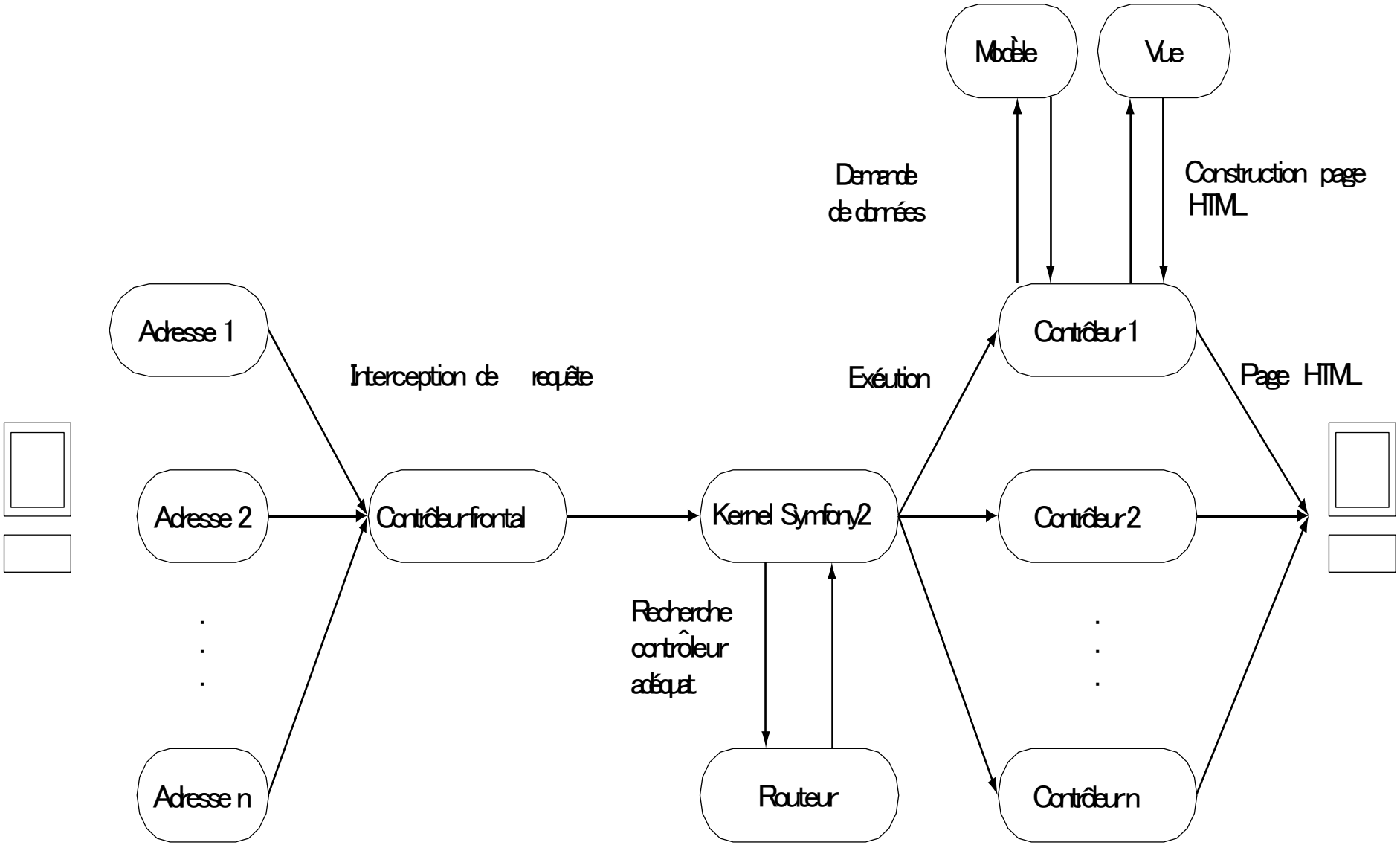
- Un élément indispensable de l'architecture MVC
- Il reçoit une requête et il interagit avec les différents composants d'une application **Symfony** :
  - les vues
  - les services
  - les modèles
  - les constructeurs de formulaires
  - ...
- pour retourner une réponse

Symfony

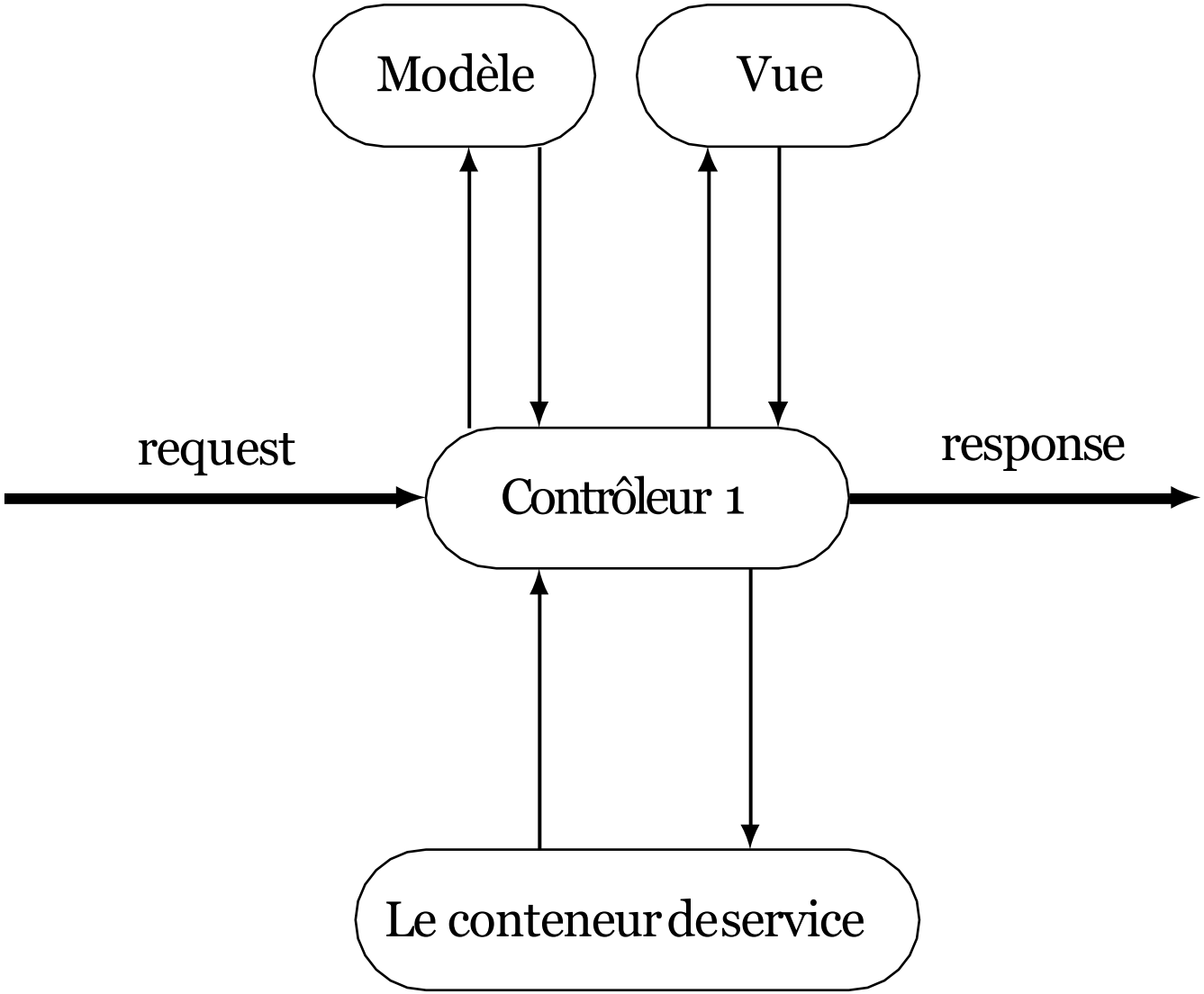
Techniquement

- Un contrôleur est une classe **PHP** qui hérite d'`AbstractControl`
- Chaque méthode (action) de contrôleur est associée à une route
- Dans un contrôleur, il n'y a que du code **PHP** (pas de **HTML** ni **CSS** ni **JS**)

Symfony


Modèle
Vue
Demande de données
Construction page HTML
Adresse 1
Interception de requête
Exécution
Contrôleur 1
Page HTML
Adresse 2
Contrôleur frontal
Kernel Symfony2
Contrôleur 2
Recherche contrôleur adéquat
Adresse n
Routeur
Contrôleur n

Symfony


Modèle
Vue
request
Contrôleur 1
response
Le conteneur de service

Symfony

Explication

- `request` et `response` sont deux objets
- `request` contient les données concernant la requête utilisateur
- `response` correspond à la réponse préparée puis retourner par le contrôleur
- Les services, les modèles... vont nous permettre de réaliser tout le travail nécessaire pour préparer le contenu de la réponse.

Symfony

Pour générer un contrôleur nommé `HomeController`

```
php bin/console make:controller HomeController
```

Le résultat est

```
created: src/Controller/HomeController.php
created: templates/home/index.html.twig
```

Constats

- `HomeController.php` : un contrôleur généré dans `src/controller`
- `index.html.twig`: une vue générée dans `templates/home`
- `home` : un répertoire créé pour le contrôleur `HomeController`qui contiendra toutes ses vues. Par défaut, **Symfony** cherchera les vues dans ce répertoire.

Symfony

Pour générer un contrôleur sanstemplate

```
php bin/console make:controller HomeController --no-
   template
```

Si on oublie de spécifier le nom, Symfony nous le rappellera

```
Choose a name for your controller class (e.g.
   GrumpyPizzaController):
```

Symfony

Plusieurs modes de routage avec **Symfony**

- par annotation (par défaut en **Symfony**)
- dans un fichier **YAML**
- dans un fichier **XML**
- dans un fichier **PHP**

Pour le vérifier, allez dans `config/routes/annotations.yaml`

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

Symfony

Code généré pour `HomeController`

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home", name="home_route")
     */
    public function index()
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}
```

Symfony

Explication

- Tous les contrôleurs sont définis dans un namespace `App\Controller`
- La méthode `index` retourne la vue `home/index.html.twig` et lui envoie un paramètre `controller_name` avec comme valeur le nom du contrôleur `HomeController`

  La méthode `index` est annotée par `@Route` qui définit le chemin qui permettra d'exécuter cette méthode
- L'annotation `@Route` permet d'associer un nom à la route pour qu'on puisse l'appeler

Pour tester, allez sur `localhost:8000/home`

Symfony

Routage dans un fichier **YAML** : démarche

- supprimer l'annotation de la méthode `index` du contrôleur `HomeController`
- vérifier que la route `/home` n'est plus accessible depuis le navigateur
- commenter la partie concernant l'annotation des contrôleurs dans `annotations.yaml`
- définir les routes dans `routes.yaml`

Symfony

Nouveau contenu de `HomeController`

```
namespace App\Controller;

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

class HomeController extends AbstractController
{

    public function index()
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}
```

Symfony

Nouveau contenu de `annotations.yaml`

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

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

Symfony

Définissons les routes dans `routes.yaml`

```
home:
    path: /home
    controller: App\Controller\HomeController::index
```

Pour tester, allez sur `localhost:8000/home`

Symfony

Routage dans un fichier **XML** : démarche

- commenter code de `routes.yaml`
- vérifier que la route `/home` n'est plus accessible depuis le navigateur
- créer un fichier `routes.xml` dans `config`
- définir les routes dans `routes.xml`

Symfony

Définissons les routes dans `routes.xml`

```
<?xml version="1.0" encoding="UTF-8" ?>
<routes xmlns="http://symfony.com/schema/routing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://symfony.com/schema/routing
        https://symfony.com/schema/routing/routing-1.0.xsd">

    <route id="home" path="/home"
           controller="App\Controller\HomeController::index"/>

</routes>
```

Pour tester, allez sur `localhost:8000/home`

Symfony

Routage dans un fichier **PHP** : démarche

- supprimer le fichier `routes.xml`
- vérifier que la route `/home` n'est plus accessible depuis le navigateur
- créer un fichier `routes.php` dans `config`
- définir les routes dans `routes.php`

Symfony

Définissons les routes dans `routes.php`

```
<?php
use App\Controller\HomeController;
use Symfony\Component\Routing\Loader\Configurator\
   RoutingConfigurator;

return function (RoutingConfigurator $routes) {
    $routes->add('home', '/home')
        ->controller([HomeController::class, 'index']);

    };
?>
```

Pour tester, allez sur `localhost:8000/home`

Symfony

Pour la suite

- Nous n'utiliserons que le routage par annotations.
- Pensez à supprimer tous les fichiers `routes.*`

Pensez à réactiver le routage par annotation dans `annotations.yaml`

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

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

Il est aussi possible d'associer plusieurs routes à notre méthode

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/")
     * @Route("/home", name="home_route")
     */
    public function index()
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}
```

Il est aussi possible d'associer une route à un contrôleur, la méthode `index` sera exécutée en allant sur la route `/controller/home`

```php
namespace App\Controller;

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

/**
 * @Route("/controller")
 */
class HomeController extends AbstractController
{
    /**
     * @Route("/home")
     */
    public function index()
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}
```

On peut utiliser les expressions régulières pour définir des contraintes sur les paramètres

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age}", name="home_route", requirements={"age"="\d
        {2,3}"})
     */
    public function index(int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Pour tester, allez sur `localhost:8000/home/50`

La contrainte peut être collée au paramètre (sans l'attribut `requirements`)

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age<\d+>}", name="home_route")
     */
    public function index(int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Pour tester, allez sur `localhost:8000/home/5`

On peut aussi rendre ce paramètre optionnel en lui attribuant une valeur par défaut

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age<\d+>}", name="home_route")
     */
    public function index(int $age = 7)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Pour tester, allez sur `localhost:8000/home`

Pour accepter la valeur `null`

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age?}", name="home_route")
     */
    public function index(?int $age)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $age,
        ]);
    }
}
```

Pour tester, allez sur `localhost:8000/home`

On peut aussi définir des constantes qu'on récupère comme des paramètres de substitution mais ils ne font pas partis de la route

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{age}", name="home_route", defaults={"nom": "wick
        ", "prenom": "john"})
     */
    public function index(int $age, string $nom, string $prenom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => "$age $nom $prenom",
        ]);
    }
}
```

Pour tester, allez sur `localhost:8000/home/45`

Symfony

L'objet `request` permet de

- récupérer les paramètres de substitution
- récupérer les variables hors routes (les paramètres libres)
- récupérer la méthode de la requête HTTP
- récupérer le nom de la route
- ...

Pour récupérer un paramètre de substitution en utilisant l'objet `request`

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{nom}", name="home_route")
     */
    public function index(Request $request)
    {
        $nom = $request->attributes->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour tester, allez sur `localhost:8000/home/wick`

On peut aussi utiliser le raccourci `request->get('nom parametre')`

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{nom}", name="home_route")
     */
    public function index(Request $request)
    {
        $nom = $request->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour tester, allez sur `localhost:8000/home/wick`

Pour les variables hors routes, pas besoin de les déclarer dans la route

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home", name="home_route")
     */
    public function index(Request $request)
    {
        $nom = $request->query->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour tester, allez sur `localhost:8000/home?nom=wick`

On peut aussi utiliser le raccourci `request->get('nom parametre')`

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home", name="home_route")
     */
    public function index(Request $request)
    {
        $nom = $request->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour tester, allez sur `localhost:8000/home?nom=wick`

Symfony

Autres informations contenues dans `$request`

- `$request->server` : les variables de serveur
- `$request->cookies` : les variables de cookie
- `$request->getMethod()` : le type de la méthode de la requête **HTTP**
- `$request->attributes->get('_route')` ou son raccourci `$request->get('_route')` : le nom de la route
- `$request->request->get('var')` : une variable envoyée par le biais de la méthode `POST`
- ...

Symfony

Remarque

Par défaut, chaque action d'un contrôleur peut être exécutée quel que soit le type de la méthode **HTTP**. Cependant, il est possible de spécifier pour chaque action les méthodes **HTTP** autorisées.

Exemple

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home", name="home_route", methods={"GET","POST"})
     */
    public function index(Request $request)
    {
        $nom = $request->query->get('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

L'action `index` sera exécutée pour les deux méthodes **HTTP** `GET` et `POST`.

Symfony

Commençons par générer un deuxième contrô leur nommé `VehiculeController`

```
php bin/console make:controller VehiculeController
```

Le résultat est

```
created: src/Controller/VehiculeController.php
created: templates/vehicule/index.html.twig
```

Symfony

Considérons le contenu suivant pour `HomeController`

```
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{nom}",name="home_route")
     */
    public function index(string $nom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Objectif

- Dans `VehiculeController` : générer une route pour le contrôleur `HomeControlleur`
- Ensuite, utiliser cette route pour rediriger vers `HomeControlleur`

Dans `VehiculeController`, on génère une URL puis on redirige

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    /**
     * @Route("/vehicule", name="vehicule_route")
     */
    public function index()
    {
        $url = $this->generateUrl('home_route', array(
            'nom' => 'abruzzi',
        ));
        return new RedirectResponse($url);

    }
}
```

Pour tester, allez sur `localhost:8000/vehicule`

On peut aussi utiliser la méthode `redirect`

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    /**
     * @Route("/vehicule", name="vehicule_route")
     */
    public function index()
    {
        $url = $this->generateUrl('home', array(
            'nom' => 'abruzzi',
        ));
        return $this->redirect($url);

    }
}
```

Pour tester, allez sur `localhost:8000/vehicule`

On peut aussi utiliser le raccourci `redirectToRoute`

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    /**
     * @Route("/vehicule", name="vehicule_route")
     */
    public function index()
    {
        return $this->redirectToRoute('home_route', ['nom' => 'abruzzi'
           ]);
    }
}
```

Pour tester, allez sur `localhost:8000/vehicule`

Symfony

La méthode `redirect` permet aussi de rediriger vers une URL externe

```
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    /**
     * @Route("/vehicule", name="vehicule_route")
     */
    public function index()
    {
        return $this->redirect('http://symfony.com/doc');
    }
}
```

Pour tester, allez sur `localhost:8000/vehicule`

Pour renvoyer une page d'erreur, on peut utiliser `HTTPException`

```php
namespace App\Controller;

// les use précédents
use Symfony\Component\HttpKernel\Exception\HttpException;

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{nom?}",name="home_route")
     */
    public function index(?string $nom)
    {
        if (!isset($nom)) {
            throw new HttpException(
                404,
                'On ne peut vous afficher la page de cette personne'
            );
        }
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Symfony

Pour tester

- allez sur une URL avec paramètre `localhost:8000/home/wick`
- et une deuxième sans `localhost:8000/home`

On peut aussi utiliser la méthode `createNotFoundException`

```
namespace App\Controller;

// les use précédents
use Symfony\Component\HttpKernel\Exception\HttpException;

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{nom?}",name="home_route")
     */
    public function index(?string $nom)
    {
        if (!isset($nom)) {
            throw $this->createNotFoundException('On ne peut vous
               afficher la page de cette personne');
        }
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Symfony

Deux utilisations possibles pour l'objet `response`

- explicite : en construisant la réponse
- implicite : on n'utilise pas l'objet `response` pour retourner la réponse mais il sera utilisé en coulisses, nous n'avons pas à le manipuler directement.

Utilisation explicite

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{nom?}",name="home_route")
     */
    public function index(?string $nom)
    {
        $response = new Response(
            "<p>Bonjour $nom</p>",
            Response::HTTP_OK,
            ['content-type' => 'text/html']
        );
        return $response;
    }
}
```

Pour tester, allez sur `localhost:8000/home/wick`

Symfony

Utilisation implicite

```
namespace App\Controller;

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

class HomeController extends AbstractController
{
    /**
     * @Route("/home/{nom?}",name="home_route")
     */
    public function index(?string $nom)
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```