Symfony : contrôleur

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com

Plan

Plan

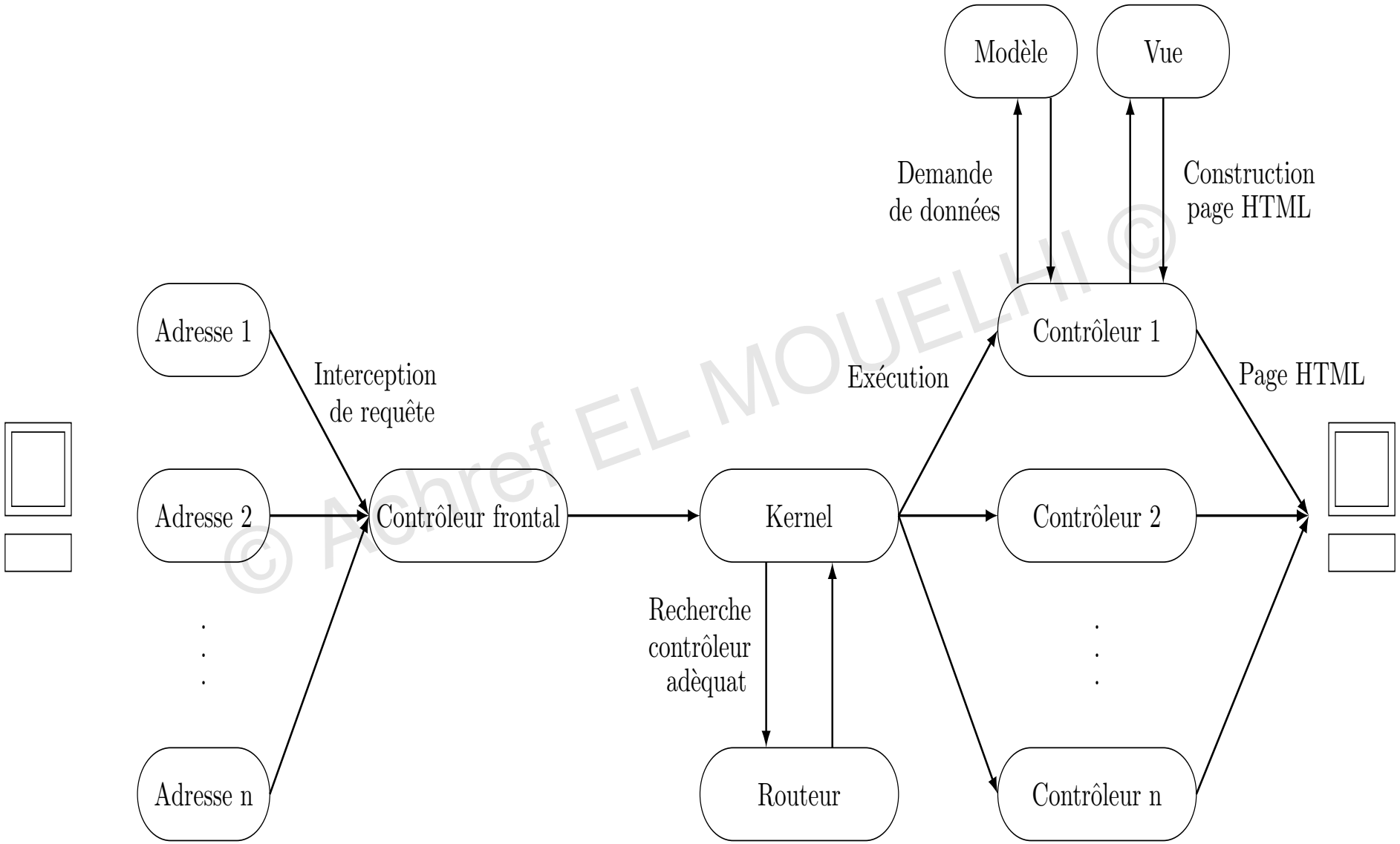
Plan

Symfony

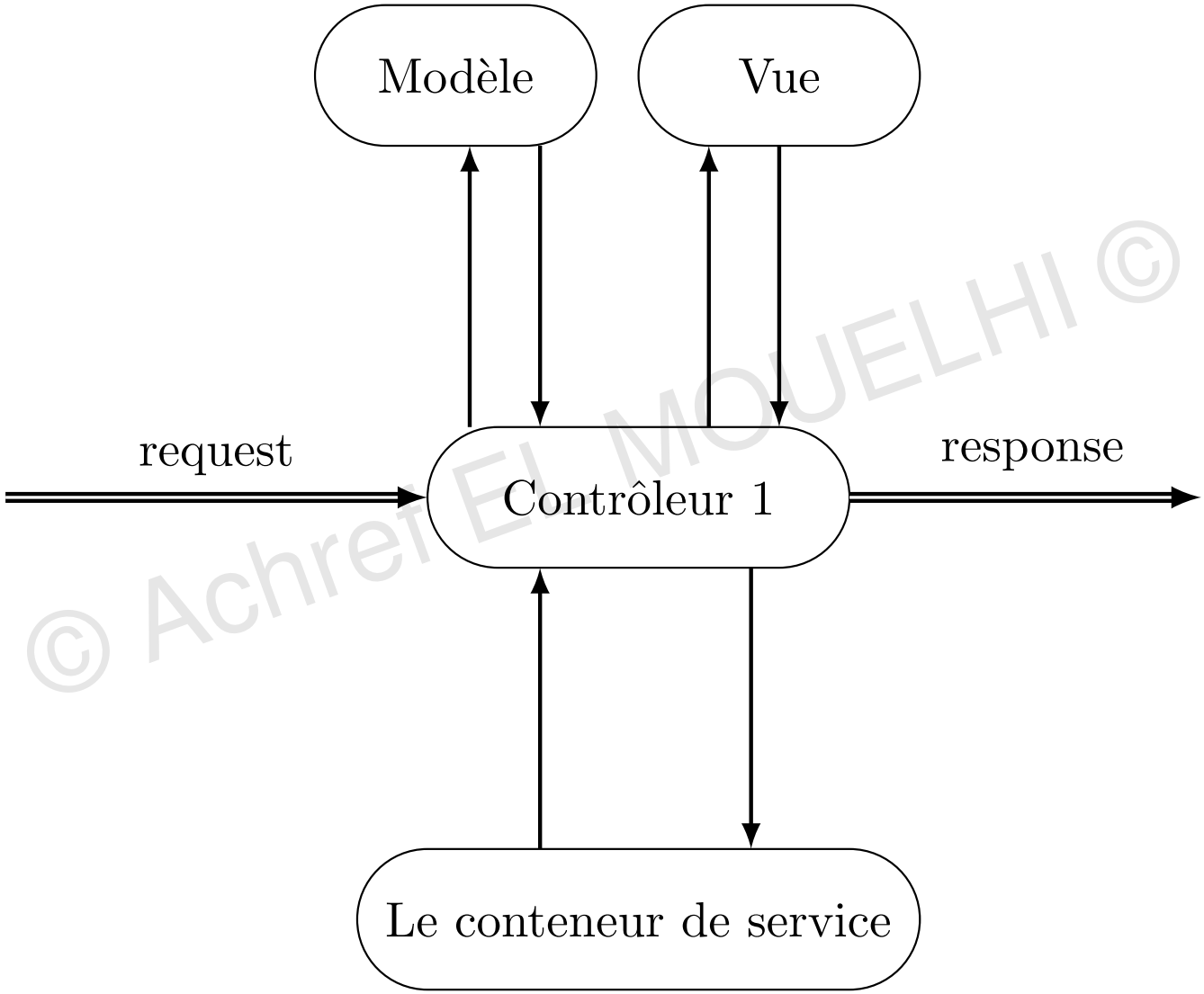
Rôle

- 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



Symfony



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

Techniquement

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

Symfony

Quelques méthodes de la classe `AbstractController`

- `render()` : permet de rendre une vue **Twig** avec des paramètres.
- `redirectToRoute()` : redirige vers une autre route en utilisant son nom.
- `redirect()` : redirige vers une **URL** spécifique.
- `json()` : retourne une réponse **JSON**.
- `createNotFoundException()` : génère une exception de type 404.
- `isGranted()` : vérifie si un utilisateur a une permission spécifique.
- `getUser()` : retourne l'utilisateur actuellement authentifié.
- ...

Symfony

Question

La classe `AbstractController` est-elle obligatoire pour les contrôleurs **Symfony** ?

Symfony

Question

La classe `AbstractController` est-elle obligatoire pour les contrôleurs **Symfony** ?

Réponse

Non, `AbstractController` n'est pas obligatoire. Les contrôleurs peuvent :

- Hériter de `AbstractController` pour bénéficier des méthodes utilitaires.
- Être des classes simples avec uniquement les méthodes nécessaires.
- Utiliser l'autowiring pour obtenir les dépendances.

Symfony

Exemple sans hériter d'`AbstractController`

```
use Symfony\Component\HttpFoundation\Response;

class HomeController
{
    public function index(): Response
    {
        return new Response('Hello, Symfony!');
    }
}
```

Symfony

Installer le `MakerBundle` avant de générer le contrôleur (Si ce n'est pas le cas)

```
composer require symfony/maker-bundle --dev
```

Symfony

Pour générer un contrôleur nommé `HomeController`

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

Symfony

Pour générer un contrôleur nommé `HomeController`

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

Ou

```
symfony console make:controller HomeController
```

Symfony

Résultat

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

Symfony

Résultat

```
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 sans template

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

Symfony

Pour générer un contrôleur sans template

```
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 attribut : recommandé depuis la version 5.2.
- par annotation : par défaut avant la version 5.2, déprécié depuis mais encore fonctionnel.
- dans un fichier **YAML**
- dans un fichier **XML**
- dans un fichier **PHP**

Symfony

Remarques

- Depuis **Symfony** 5.2, le routage par attributs est non seulement recommandé, mais c'est aussi le mode activé par défaut dans les nouveaux projets **Symfony**.
- Les modes de routage basés sur **YAML**, **XML** et **PHP** ne sont plus activés par défaut depuis les versions récentes (**Symfony** 5.x et ultérieures).
- Ils sont encore disponibles, mais nécessitent une configuration manuelle : l'ajout de fichiers de configuration appropriés ou la modification des paramètres dans le fichier `config/routes.yaml`.

Symfony

Pour le vérifier, allez dans `config/routes.yaml` **(**`annotations.yaml` **dans les versions précédentes)**

```
controllers:
    resource:
        path: ../src/Controller/
        namespace: App\Controller
    type: attribute
```

Symfony

Code généré pour `HomeController`

```php
namespace App\Controller;

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

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

Symfony

Explication

- Tous les contrôleurs sont définis dans un namespace `App\Controller`.
- Une méthode définie dans un contrôleur est appelée action.
- L'action `index` retourne la vue `home/index.html.twig` et lui envoie un paramètre `controller_name` avec comme valeur le nom du contrôleur `HomeController`.
- L'action `index` est précédée par `#[Route]` qui définit le chemin qui permet de l'exécuter.
- L'attribut `#[Route]` permet d'associer un nom à la route pour qu'on puisse l'appeler.

Symfony

Explication

- Tous les contrôleurs sont définis dans un namespace `App\Controller`.
- Une méthode définie dans un contrôleur est appelée action.
- L'action `index` retourne la vue `home/index.html.twig` et lui envoie un paramètre `controller_name` avec comme valeur le nom du contrôleur `HomeController`.
- L'action `index` est précédée par `#[Route]` qui définit le chemin qui permet de l'exécuter.
- L'attribut `#[Route]` permet d'associer un nom à la route pour qu'on puisse l'appeler.

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

Symfony

Il est possible d'associer une route à un contrôleur et une deuxième à la méthode `index`. Cette dernière sera exécutée en allant à `localhost:8000/home/index`

```php
namespace App\Controller;

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

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

Symfony

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

```php
namespace App\Controller;

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

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

Symfony

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

```php
namespace App\Controller;

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

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

Pour tester, allez à `localhost:8000/home` ou `localhost:8000`

Symfony

Deux types de paramètre

- Paramètre de substitution : `/home/wick/john` (à définir dans la route avec {nomParametre})
- Paramètre hors route : `/home?nom=wick&prenom=john` (à ne pas inclure dans la route)

Symfony

Pour récupérer un paramètre de substitution, il faut le définir dans la route et l'ajouter comme paramètre de l'action

```
namespace App\Controller;

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

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

Symfony

Pour récupérer un paramètre de substitution, il faut le définir dans la route et l'ajouter comme paramètre de l'action

```php
namespace App\Controller;

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

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

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

Symfony

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

```
namespace App\Controller;

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

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

Symfony

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

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

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

Symfony

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

```
namespace App\Controller;

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

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

Symfony

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

```php
namespace App\Controller;

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

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

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

Symfony

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

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

Symfony

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

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

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

Symfony

Pour accepter la valeur `null`

```php
namespace App\Controller;

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

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

Symfony

Pour accepter la valeur `null`

```php
namespace App\Controller;

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

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

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

Symfony

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

```
namespace App\Controller;

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

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

Symfony

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

```
namespace App\Controller;

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

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

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

Symfony

Considérons le contrôleur suivant

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

    #[Route('/home/index', name: 'app_home2')]
    public function index2()
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => "HomeController",
        ]);
    }
}
```

Symfony

Constats

- En allant à `localhost:8000/home/wick` ⇒ `Hello wick!` s'affiche.
- En allant à `localhost:8000/home/index` ⇒ `Hello index!` s'affiche.

Symfony

Constats

- En allant à `localhost:8000/home/wick` ⇒ `Hello wick!` s'affiche.
- En allant à `localhost:8000/home/index` ⇒ `Hello index!` s'affiche.

Conclusion

L'action `index2` n'est jamais exécutée car la route demandée match avec la route de l'action `index`.

Symfony

Solution

- Depuis **Symfony 5.1**, il est possible d'associer une valeur de priorité à chaque route.
- Une route sans la propriété `priority` a par défaut la priorité 0.
- Si la route demandée match avec plusieurs routes définies, alors celle avec la priorité max sera exécutée.

Symfony

Ajoutons une priorité à la deuxième route

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

    #[Route('/home/index', name: 'app_home2', priority: 2)]
    public function index2()
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => "HomeController",
        ]);
    }
}
```

Symfony

Constats

- En allant à `localhost:8000/home/wick` ⇒ `Hello wick!` s'affiche.
- En allant à `localhost:8000/home/index` ⇒ `Hello HomeController!` s'affiche.

Symfony

Constats

- En allant à `localhost:8000/home/wick` ⇒ `Hello wick!` s'affiche.
- En allant à `localhost:8000/home/index` ⇒ `Hello HomeController!` s'affiche.

Question

Comment on faisait avant **Symfony 5.1** ?

Symfony

Constats

- En allant à `localhost:8000/home/wick` ⇒ `Hello wick!` s'affiche.
- En allant à `localhost:8000/home/index` ⇒ `Hello HomeController!` s'affiche.

Question

Comment on faisait avant **Symfony 5.1** ?

Réponse

L'exécution est séquentielle, on jouait sur l'ordre de définition de nos actions (routes).

Symfony

Pour consulter les routes disponibles

```
php bin/console debug:router
```

Symfony

Pour consulter les routes disponibles

```
php bin/console debug:router
```

Le résultat

```
------------------------ -------- -------- ------ ----------------------------------
 Name                     Method   Scheme   Host   Path
------------------------ -------- -------- ------ ----------------------------------
 app_home2                ANY      ANY      ANY    /home/index
 _preview_error           ANY      ANY      ANY    /_error/{code}.{_format}
 _wdt                     ANY      ANY      ANY    /_wdt/{token}
 _profiler_home           ANY      ANY      ANY    /_profiler/
 _profiler_search         ANY      ANY      ANY    /_profiler/search
 _profiler_search_bar     ANY      ANY      ANY    /_profiler/search_bar
 _profiler_phpinfo        ANY      ANY      ANY    /_profiler/phpinfo
 _profiler_search_results ANY      ANY      ANY    /_profiler/{token}/search/results
 _profiler_open_file      ANY      ANY      ANY    /_profiler/open
 _profiler                ANY      ANY      ANY    /_profiler/{token}
 _profiler_router         ANY      ANY      ANY    /_profiler/{token}/router
 _profiler_exception      ANY      ANY      ANY    /_profiler/{token}/exception
 _profiler_exception_css  ANY      ANY      ANY    /_profiler/{token}/exception.css
 app_home                 ANY      ANY      ANY    /home/{nom}
------------------------ -------- -------- ------ ----------------------------------
```

Symfony

On peut aussi spécifier le nom de la route

```
php bin/console debug:router app_home
```

Symfony

On peut aussi spécifier le nom de la route

```
php bin/console debug:router app_home
```

Le résultat

```
+--------------+-------------------------------------------------------------+
| Property     | Value                                                       |
+--------------+-------------------------------------------------------------+
| Route Name   | app_home                                                    |
| Path         | /home/{nom}                                                 |
| Path Regex   | {^/home/(?P<nom>[^/]++)$}sDu                                |
| Host         | ANY                                                         |
| Host Regex   |                                                             |
| Scheme       | ANY                                                         |
| Method       | ANY                                                         |
| Requirements | NO CUSTOM                                                   |
| Class        | Symfony\Component\Routing\Route                             |
| Defaults     | _controller: App\Controller\HomeController::index()         |
| Options      | compiler_class: Symfony\Component\Routing\RouteCompiler     |
|              | utf8: true                                                  |
+--------------+-------------------------------------------------------------+
```

Symfony

Pour tester si une route match avec une route définie (A ne pas essayer depuis une console Bash)

```
php bin/console router:match /home/john
```

Symfony

Pour tester si une route match avec une route définie (A ne pas essayer depuis une console Bash)

```
php bin/console router:match /home/john
```

Le résultat

```
[OK] Route "app_home" matches


+--------------+-------------------------------------------------------------+
| Property     | Value                                                       |
+--------------+-------------------------------------------------------------+
| Route Name   | app_home                                                    |
| Path         | /home/{nom}                                                 |
| Path Regex   | {^/home/(?P<nom>[^/]++)$}sDu                                |
| Host         | ANY                                                         |
| Host Regex   |                                                             |
| Scheme       | ANY                                                         |
| Method       | ANY                                                         |
| Requirements | NO CUSTOM                                                   |
| Class        | Symfony\Component\Routing\Route                             |
| Defaults     | _controller: App\Controller\HomeController::index()         |
| Options      | compiler_class: Symfony\Component\Routing\RouteCompiler     |
|              | utf8: true                                                  |
+--------------+-------------------------------------------------------------+
```

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
- ...

Symfony

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

```
namespace App\Controller;

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

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

Symfony

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

```php
namespace App\Controller;

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

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

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

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

```php
namespace App\Controller;

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

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

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

```php
namespace App\Controller;

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

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

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

On peut aussi utiliser le raccourci `attributes` pour récupérer tous les paramètres

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/home/{nom}/{prenom}', name: 'app_home')]
    public function index(Request $request): Response
    {
        $params = $request->attributes->get('_route_params');
        return $this->render('home/index.html.twig', [
            'controller_name' => implode(" ", $params),
        ]);
    }
}
```

On peut aussi utiliser le raccourci `attributes` pour récupérer tous les paramètres

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/home/{nom}/{prenom}', name: 'app_home')]
    public function index(Request $request): Response
    {
        $params = $request->attributes->get('_route_params');
        return $this->render('home/index.html.twig', [
            'controller_name' => implode(" ", $params),
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home/wick/john`

Symfony

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

```php
namespace App\Controller;

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

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

Symfony

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

```php
namespace App\Controller;

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

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

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

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

```php
namespace App\Controller;

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

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

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

```php
namespace App\Controller;

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

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

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

Symfony

Pour récupérer tous les paramètres hors routes

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/home', name: 'app_home')]
    public function index(Request $request): Response
    {
        $params = $request->query->all();
        return $this->render('home/index.html.twig', [
            'controller_name' => implode(" ", $params),
        ]);
    }
}
```

Symfony

Pour récupérer tous les paramètres hors routes

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/home', name: 'app_home')]
    public function index(Request $request): Response
    {
        $params = $request->query->all();
        return $this->render('home/index.html.twig', [
            'controller_name' => implode(" ", $params),
        ]);
    }
}
```

Pour tester, allez à `localhost:8000/home?nom=wick&prenom=john`

Symfony

Exercice

- Créer un contrôleur `Calcul`
- Ajouter une action `apply` accessible via la route `/calcul/{op}`
- Le paramètre `op` accepte les valeurs `plus`, `moins`, `fois` ou `div`
- En saisissant `/calcul/plus?var1=10&var2=5` ⇒ **Résultat :** `10 + 5 = 15`
- En saisissant `/calcul/fois?var1=10&var2=5` ⇒ **Résultat :** `10 * 5 = 50`
- En saisissant `/calcul/moins?var1=10&var2=5` ⇒ **Résultat :** `10 - 5 = 5`
- En saisissant `/calcul/div?var1=10&var2=5` ⇒ **Résultat :** `10 / 5 = 2`
- En saisissant `/calcul/div?var1=10&var2=0` ⇒ **Résultat :** `10 / 0 = Infinity`
- En saisissant `/calcul/x?var1=10&var2=5` ⇒ **Résultat :** `erreur`

Symfony

Pour récupérer le nom d'une route surtout en cas de multi-routing

```
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/index', name: "app_home_index")]
    #[Route('/home', name: 'app_home')]
    public function index(Request $request): Response
    {
        $route = $request->attributes->get('_route');
        return $this->render('home/index.html.twig', [
            'controller_name' => $route,
        ]);
    }
}
```

Symfony

Constats

- En allant à `localhost:8000/home` ⇒ `Hello app_home!` s'affiche.
- En allant à `localhost:8000/index` ⇒ `Hello app_home_index` s'affiche.

Symfony

Pour récupérer le nom d'une route surtout en cas de multi-routing

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/index', name: "app_home_index")]
    #[Route('/home', name: 'app_home')]
    public function index(Request $request): Response
    {
        $route = $request->getPathInfo();
        return $this->render('home/index.html.twig', [
            'controller_name' => $route,
        ]);
    }
}
```

Symfony

Constats

- En allant à `localhost:8000/home` ⇒ `Hello /home` s'affiche.
- En allant à `localhost:8000/index` ⇒ `Hello /index` s'affiche.

Symfony

Autres informations contenues dans `$request`

- `$request->server` : les variables de serveur
- `$request->cookies` : les variables de cookie
- `$request->getMethod()` : le nom de la méthode **HTTP**
- `$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 quelle que soit la méthode **HTTP**.
- Cependant, il est possible de spécifier pour chaque action les méthodes **HTTP** autorisées.

Symfony

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

```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: 'app_home', methods: ["GET", "POST"])]
    public function index(Request $request)
    {
        $route = $request->getMethod();
        return $this->render('home/index.html.twig', [
            'controller_name' => $route,
        ]);
    }
}
```

Symfony

Constat

En allant à `localhost:8000/home` ⇒ `Hello GET!` s'affiche.

Symfony

Expression Language

- Composant **Symfony** introduit dans la version 2.4
- Inspiré par **SpEL** (**Sp**ring **E**xpression **L**anguage)
- Basé sur la syntaxe de **Twig**
- Permettant de simplifier les contrôles

Symfony

Exemple

```
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: 'app_home', methods: ["GET", "POST"], condition: "context.getMethod
       () in ['GET', 'POST']")]
    public function index(Request $request)
    {
        $methode = $request->getMethod();
        return $this->render('home/index.html.twig', [
            'controller_name' => $methode,
        ]);
    }
}
```

Symfony

Explication

- L'action `index` sera exécutée pour les deux méthodes **HTTP** `GET` et `POST`.
- `context` : instance de `Symfony\Component\Routing\RequestContext`.

Symfony

Explication

- L'action `index` sera exécutée pour les deux méthodes **HTTP** `GET` et `POST`.
- `context` : instance de `Symfony\Component\Routing\RequestContext`.

Autres variables disponibles dans **EL**

- `request` : instance de `Symfony\Component\HttpFoundation\Request`.
- `this` : pour accéder aux attributs de l'objet courant.
- ...

Symfony

Opérateurs arithmétiques supportés

- +
- -
- *
- /
-
- ** (puissance)

Symfony

Opérateurs de comparaison

- `==`
- `===`
- `!=`
- `!==`
- `<`
- `>`
- `<=`
- `>=`
- `matches`

Symfony

Opérateurs logiques

- `not` ou `!`
- `and` ou `&&`
- `or` ou `||`

Symfony

Opérateurs logiques

- `not` ou `!`
- `and` ou `&&`
- `or` ou `||`

Opérateurs de concaténation (string)

~

Symfony

Opérateurs pour les tableaux

- `in`
- `not in`
- `..`

Symfony

Opérateurs pour les tableaux

- `in`
- `not in`
- `..`

Opérateurs ternaires

- `condition ? vrai : faux`
- `condition ?: faux`
- `condition ? vrai`

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: 'app_home')]
    public function index(?string $nom): Response
    {
        $response = new Response(
            "<p>Bonjour $nom</p>",
            Response::HTTP_OK,
            ['content-type' => 'text/html']
        );
        return $response;
    }
}
```

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: 'app_home')]
    public function index(?string $nom): Response
    {
        $response = new Response(
            "<p>Bonjour $nom</p>",
            Response::HTTP_OK,
            ['content-type' => 'text/html']
        );
        return $response;
    }
}
```

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

Symfony

Utilisation implicite

```
namespace App\Controller;

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

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

Symfony

Commençons par générer un deuxième contrôleur nommé `VehiculeController`

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

Symfony

Commençons par générer un deuxième contrôleur nommé `VehiculeController`

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

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

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

Symfony

Objectif

Dans `VehiculeController` : générer une route et rediriger vers l'action `index` de `HomeControlleur`

Symfony

Une première solution consiste à utiliser la méthode `forward`

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'app_vehicule')]
    public function index(): Response
    {
        return $this->forward('App\Controller\HomeController::index', [
            'nom' => 'abruzzi',
        ]);
    }
}
```

Symfony

Une première solution consiste à utiliser la méthode `forward`

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'app_vehicule')]
    public function index(): Response
    {
        return $this->forward('App\Controller\HomeController::index', [
            'nom' => 'abruzzi',
        ]);
    }
}
```

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

Symfony

Constat

La route affichée dans la barre d'adresse (`/vehicule`) ne correspond pas au contenu affiché (appartenant au contrôleur `HomeController`).

Symfony

Constat

La route affichée dans la barre d'adresse (`/vehicule`) ne correspond pas au contenu affiché (appartenant au contrôleur `HomeController`).

Remarque

L'ordre des paramètres de l'action (`index`) n'a pas d'importance : la correspondance se fait par nom.

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

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'app_vehicule')]
    public function index(): Response
    {
        $url = $this->generateUrl('app_home', array(
            'nom' => 'abruzzi',
        ));
        return new RedirectResponse($url);
    }
}
```

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

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'app_vehicule')]
    public function index(): Response
    {
        $url = $this->generateUrl('app_home', array(
            'nom' => 'abruzzi',
        ));
        return new RedirectResponse($url);
    }
}
```

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

On peut aussi utiliser la méthode `redirect` de `AbstractController`

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'app_vehicule')]
    public function index(): Response
    {
        $url = $this->generateUrl('app_home', array(
            'nom' => 'abruzzi',
        ));
        return $this->redirect($url);
    }
}
```

On peut aussi utiliser la méthode `redirect` de `AbstractController`

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'app_vehicule')]
    public function index(): Response
    {
        $url = $this->generateUrl('app_home', array(
            'nom' => 'abruzzi',
        ));
        return $this->redirect($url);
    }
}
```

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

Symfony

On peut aussi utiliser le raccourci `redirectToRoute`

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'app_vehicule')]
    public function index(): Response
    {
        return $this->redirectToRoute('app_home', [
            'nom' => 'abruzzi'
        ]);
    }
}
```

Symfony

On peut aussi utiliser le raccourci `redirectToRoute`

```php
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'app_vehicule')]
    public function index(): Response
    {
        return $this->redirectToRoute('app_home', [
            'nom' => 'abruzzi'
        ]);
    }
}
```

Pour tester, allez à `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\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'app_vehicule')]
    public function index(): Response
    {
        return $this->redirect('http://symfony.com/doc');
    }
}
```

Symfony

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

```
namespace App\Controller;

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

class VehiculeController extends AbstractController
{
    #[Route('/vehicule', name: 'app_vehicule')]
    public function index(): Response
    {
        return $this->redirect('http://symfony.com/doc');
    }
}
```

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

Symfony

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

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: 'app_home')]
    public function index(?string $nom): Response
    {
        if (!isset($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 à une URL avec paramètre `localhost:8000/home/wick`
- et une deuxième sans `localhost:8000/home`

Symfony

On peut aussi utiliser la méthode `createNotFoundException`

```
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: 'app_home')]
    public function index(?string $nom): Response
    {
        if (!isset($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

Commençons par créer un contrôleur `ErrorController`

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

Symfony

Commençons par créer un contrôleur `ErrorController`

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

Le résultat est

```
created: src/Controller/ErrorController.php
```

Symfony

Préparons le contenu de ce contrôleur

```php
<?php

namespace App\Controller;

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

class ErrorController extends AbstractController
{
    public function index(Request $request): Response
    {
        $route = $request->getPathInfo();
        return new Response("La route '$route' n'est pas définie", 404);
    }
}
```

Symfony

Dans `config/routes.yaml`, redirigeons toutes les autres requêtes vers ce nouveau contrôleur

```
controllers:
    resource:
        path: ../src/Controller/
        namespace: App\Controller
    type: attribute


fallback_route:
    path: /{any}
    controller: 'App\Controller\ErrorController::index'
    requirements:
        any: '.*'
```

Symfony

Deux types de paramètre d'application

- prédéfinis : quelle que soit l'application **Symfony**
- personnalisés : à définir par le développeur

Pour récupérer le chemin absolu vers notre projet

```
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: 'app_home')]
    public function index(?string $nom): Response
    {
        $path = $this->getParameter('kernel.project_dir');
        return $this->render('home/index.html.twig', [
            'controller_name' => $path,
        ]);
    }
}
```

Pour récupérer le chemin absolu vers notre projet

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/home/{nom?}', name: 'app_home')]
    public function index(?string $nom): Response
    {
        $path = $this->getParameter('kernel.project_dir');
        return $this->render('home/index.html.twig', [
            'controller_name' => $path,
        ]);
    }
}
```

En allant `localhost:8000/home/wick` ⇒ `Hello C:\Users\elmou\Desktop\cours_symfony!` est affiché

Symfony

Autres paramètres définis

- `kernel.charset` pour l'encodage.
- `kernel.cache_dir` pour le chemin vers le dossier du cache (`C:\Users\user\Desktop\cours_symfony\var\cache\dev`).
- `kernel.logs_dir` pour le chemin vers le dossier de journalisation (`C:\Users\user\Desktop\cours_symfony\var\log`).
- ...

Symfony

Paramètres personnalisés

- Pour définir nos paramètres, il faut aller dans `config/services.yaml` et ajouter le nouveau paramètre dans la section `parameters` sous format : `clé: valeur`.
- Ensuite, on utilise la méthode `$this->getParameter()` pour récupérer le paramètre.

Symfony

Commençons par définir le paramètre suivant dans la section `parameters` **de** `config/services.yaml`

```
# This file is the entry point to configure your own
    services.
# Files in the packages/ subdirectory configure your
    dependencies.

# Put parameters here that don't need to change on
   each machine where the app is deployed
# https://symfony.com/doc/current/best_practices/
   configuration.html#application-related-
   configuration
parameters:
    nom: 'wick'
```

Pour récupérer le paramètre personnalisé

```
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/home', name: 'app_home')]
    public function index(): Response
    {
        $nom = $this->getParameter('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

Pour récupérer le paramètre personnalisé

```php
namespace App\Controller;

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

class HomeController extends AbstractController
{
    #[Route('/home', name: 'app_home')]
    public function index(): Response
    {
        $nom = $this->getParameter('nom');
        return $this->render('home/index.html.twig', [
            'controller_name' => $nom,
        ]);
    }
}
```

En allant `localhost:8000/home` ⇒ `Hello wick!` est affiché.

Symfony

Bonnes pratiques

- Faire hériter son contrôleur de `AbstractController`.
- Depuis **Symfony** 6, l'utilisation des attributs (pour le routage, la sécurité...) est privilégiée aux annotations, qui étaient auparavant utilisées avec **Symfony** 5.
- Favoriser l'injection de dépendance pour l'utilisation des services.
- Ne plus organiser l'application par **Bundle** : Les bundles ont été utilisés dans les premières versions de **Symfony** pour créer des fragments réutilisables.
- Organiser par namespaces pour faciliter la navigation dans le projet.