



# Routing de Symfony

Comment créer des pages dans Symfony ?



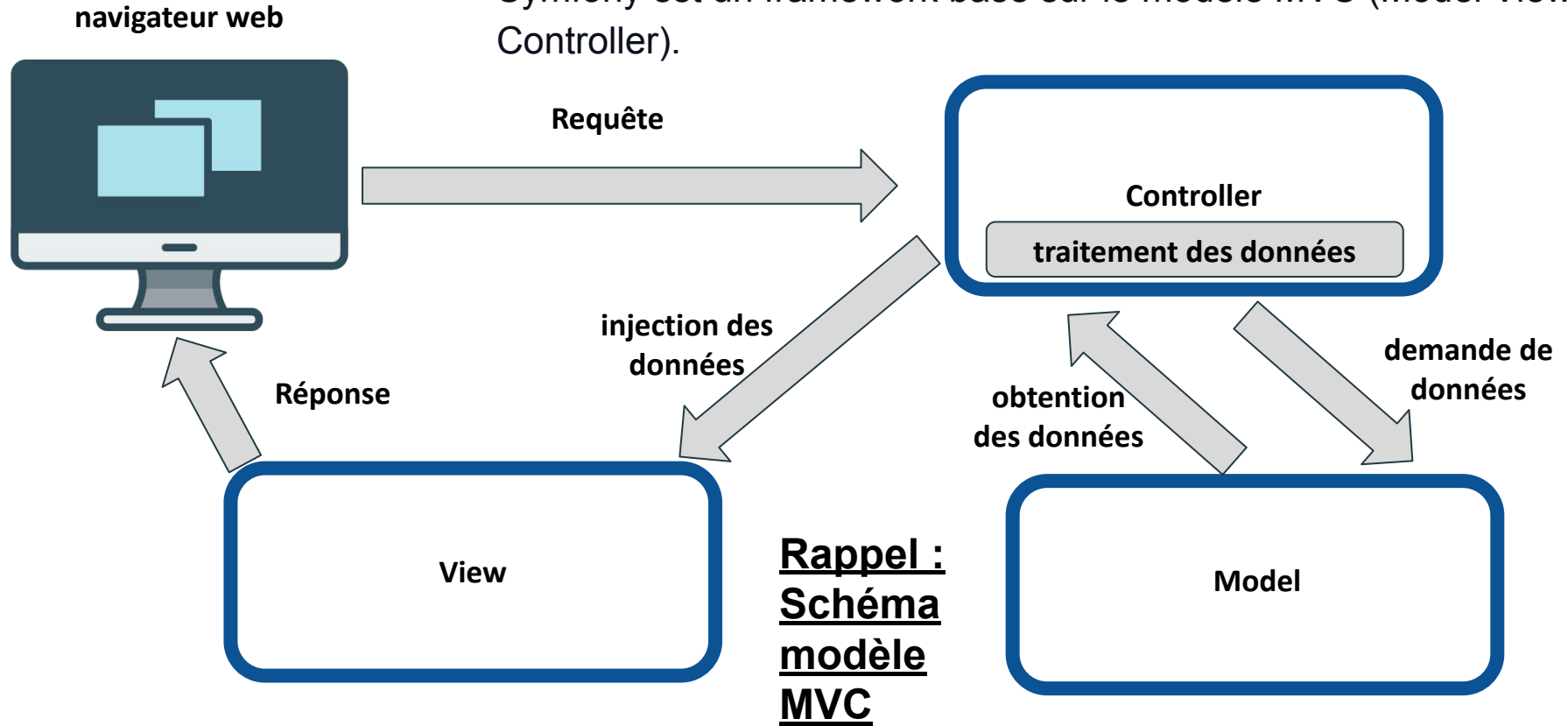
# Routing

- Après avoir créer le projet, on va créer les premières pages. Grâce à Symfony, on est plus obligé de créer un fichier PHP par page.
- Avec Symfony le mécanisme de routes permet de lier une URL à une portion de code (une méthode dans une classe) à exécuter quand cette URL est demandée
- Les routes seront créer dans les controllers
- Les controllers reçoivent la requête HTTP et réaliser les actions que demande la requête en utilisant les vues et les modèles.



# Controller

- Symfony est un framework basé sur le modèle MVC (Model View Controller).





# Controller

- Les controllers sont créés dans le dossier Controller du dossier src du projet Symfony.
- Le nom du controller commencera toujours par une majuscule et se finira toujours par Controller (ex: MainController)
- Pour créer un controller on utilise la commande
  - `php bin/console make:controller`

```
C:\xampp\htdocs\projet_symfony
λ php bin/console make:controller

Choose a name for your controller class (e.g. AgreeableChefController):
> MainController

created: src/Controller/MainController.php
created: templates/main/index.html.twig
```

Success!

Next: Open your new controller class and add some pages!



# Controller

- Le fichier MainController.php ressemble à ceci

```
MainController.php X
src > Controller > MainController.php > ...
1  <?php
2
3  namespace App\Controller;
4
5  use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
6  use Symfony\Component\HttpFoundation\Response;
7  use Symfony\Component\Routing\Annotation\Route;
8
9  class MainController extends AbstractController
10 {
11     #[Route('/main', name: 'main')]
12     public function index(): Response
13     {
14         return $this->render('main/index.html.twig', [
15             'controller_name' => 'MainController',
16         ]);
17     }
18 }
19
```

On remarquera :



- le namespace qui correspond à l'emplacement du fichier dans le projet (src étant remplacé par App)
- des importations de classe avec des use
- la classe MainController qui hérite de l'AbstractController et qui est importé ligne 5



# Route

- Les routes seront créées dans les fichiers Controller
- Elles seront immédiatement suivie d'une méthode
- Les routes permettent de réaliser la méthode qui est associée à la route
- Les routes seront écrites suivant les syntaxes suivantes :

Route, terme que l'on retrouve dans l'url



```
/**
 * @Route("/home", name="home")
 */
public function home()
{
}
```

Syntaxe avant Symfony 6

Nom de la route

```
#[Route('/home', name: 'home')]
public function home()
{
}
```

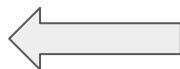
Syntaxe depuis Symfony 6



# Route

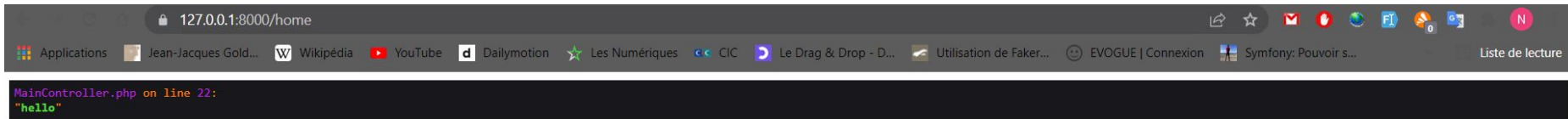
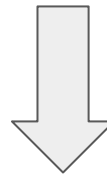
- Les méthodes des routes demandent toujours un réponse
- Pour contrer cela on peut utiliser la fonction dd() qui signifie dump and die.

```
#[Route('/home', name: 'home')]
public function home()
{
    dd('hello');
}
```



Code

Résultat





# Route

- Comme les fonctions demandent toujours un réponse on peut utiliser la classe Response qui a été importé lors de la création du Controller

```
#[Route('/home', name: 'home')]
public function home()
{
    return new Response('hello');
}
```



Code

Résultat



hello





# Route

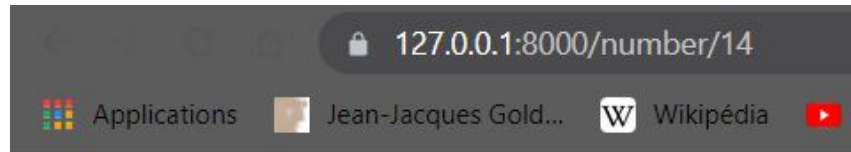
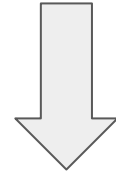
- De la même manière qu'avec PHP, on peut mettre des paramètres dans l'URL (par exemple l'id)
- Ce paramètre s'appelle une wildcard et s'écrit entre accolade dans la route
- Il devra ensuite être mis en paramètre de la fonction sous forme de variable

```
#[Route('/number/{id}', name: 'number')]  
public function number($id)  
{  
    return new Response($id);  
}
```



Code

Résultat



14



1



# Route

- Bravo moussaillon, tu peux maintenant créer des routes sur ton projet.

