Créer et utiliser un contrôleur Symfony

1. Objectifs

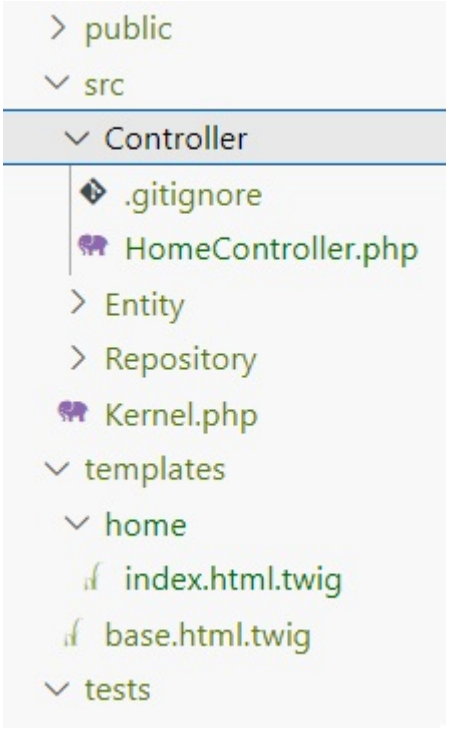
- Etre capable d'utiliser les contrôleurs Symfony

2. Qu’est-ce qu’un contrôleur ?

- Dans le modèle MVC (Modèle-Vue-Contrôleur), le contrôleur contient la logique concernant les actions effectuées par l’utilisateur. En pratique, l'utilisation de contrôleurs dans Symfony permet de séparer la logique de traitement des routes, offrant ainsi une meilleure organisation et maintenabilité du code.
- Un **contrôleur** est une classe qui va contenir différentes méthodes. Chaque méthode correspond généralement à une action (URL) de votre application Symfony.
- Concrètement, un contrôleur est matérialisé par une classe, où chaque méthode représente une action spécifique. Ainsi, chaque action est associée à une route.
- Les contrôleurs sont une composante fondamentale du framework Symfony. Plutôt que de définir la logique de traitement des demandes directement dans les fichiers de routage, Symfony préconise l'organisation de cette logique à l'aide des classes de contrôleur.
- Au lieu de définir la logique de réponse aux différents requêtes dans les fichiers du **Routing** comme on a vu dans les exemples précédents, il est recommandé d’organiser ces comportements dans des **Controllers**.
- Les **Controllers** permettent de regrouper les logiques de réponse aux **requêtes http** reliées dans la même classe.

apcpedagogie.com

3. Rôle d'un contrôleur



- Le rôle d'un contrôleur est de recevoir une requête (qui a déjà été sélectionnée par une route) et de définir la réponse appropriée. En d'autres termes, il s'agit de gérer la logique de traitement des demandes HTTP.
- Les contrôleurs sont destinés à regrouper la logique de traitement des demandes associée dans une seule classe.
- Dans votre projet Symfony, les contrôleurs sont généralement stockés dans le répertoire `src/Controller`.
- Le modèle MVC (Modèle-Vue-Contrôleur) est un design pattern qui vise à diriger le flux de données entre le modèle (les données), la vue (l'interface utilisateur) et le contrôleur (la logique de traitement). Dans ce modèle, le contrôleur est responsable de la coordination entre le modèle et la vue.
- ***La tâche d'un contrôleur est de recevoir une requête (qui a déjà été sélectionnée par une route) et de définir la réponse appropriée pour la fournir au client.***

apcpedagogie.com

4. ## Créer un contrôleur sous Symfony

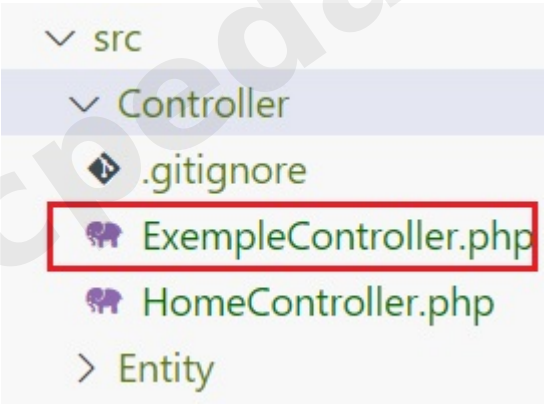
   - Pour créer un contrôleur à partir de la ligne de commande, ouvrez votre terminal et naviguez jusqu'au répertoire racine de votre projet Symfony.
   - Symfony CLI, l'outil en ligne de commande fourni avec Symfony, vous permet de créer rapidement un fichier contenant la structure de base d'un contrôleur.

   - **Créer un simple contrôleur vierge**

     - Pour créer un contrôleur, vous pouvez directement exécuter la commande suivante :
     - **`symfony console make:controller NomDeMonController`**



     - Si tout se passe bien, vous trouverez le contrôleur créé dans le répertoire spécifié par Symfony.



> *Par convention, les contrôleurs sont en notation **Camel Case** et terminent tous avec le mot **Controller**. Cela permet de les différencier des services et des modèles auxquels ils font référence.*

5. Dans l'exemple suivant, nous nommerons notre contrôleur **ApprenantController**.
6. Retournez dans votre dossier **src/Controller**, vous y trouverez votre nouveau contrôleur. Nous allons ensuite créer une nouvelle méthode dans notre classe `ApprenantController` que nous

nommerons `apprenants`.

```
<?php
namespace App\Controller;

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

class ApprenantController extends AbstractController
{
    /**
     * @Route("/apprenants", name="apprenants")
     */
    public function apprenants(): Response
    {
        return $this->render('apprenants/index.html.twig');
    }
}
```

7. Maintenant, nous devons faire le lien entre notre fichier **routes.yaml** et notre contrôleur. Nous devons donc définir notre route pour utiliser notre contrôleur.

```
app_apprenants:
    path: /apprenants
    controller: 'App\Controller\ApprenantController::apprenants'
```

8. Par défaut, Symfony recherche les contrôleurs dans le namespace `App\Controller`. Nous spécifions également après le `::` que la méthode à appeler est **apprenants()**.

9. **Créer un contrôleur avec les actions CRUD**

    - Vous pouvez générer un contrôleur avec les actions CRUD en utilisant la commande suivante :

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

10. Cela générera un contrôleur avec les méthodes index, new, show, edit et delete pour gérer les opérations CRUD.

apcpedagogie.com

11. ## Structure d'un contrôleur Symfony

    - Un contrôleur est une classe qui étend généralement la classe de base AbstractController et dont chaque méthode publique représente une action associée à une route.
    - **Fichier src/Controller/ApprenantController.php**

```
<?php

namespace App\Controller;

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

class ApprenantController extends AbstractController
{
    /**
     * @Route("/apprenants", name="apprenants")
     */
    public function apprenants(): Response
    {
        return $this->render('apprenants/index.html.twig');
    }

    /**
     * @Route("/apprenants/{id}", name="apprenant_show")
     */
    public function show($id): Response
    {
        // Code pour récupérer et afficher un apprenant avec l'id $id
    }
}
```

12. Une méthode d'un contrôleur Symfony retourne généralement une instance de Symfony\Component\HttpFoundation\Response, mais elle peut également retourner d'autres types de réponses comme des chaînes de caractères ou des objets JSON.
13. Dans l'exemple précédent, la méthode `apprenants()` retourne une réponse sous la forme d'un rendu de vue, et la méthode `show($id)` pourrait être utilisée pour afficher les détails d'un apprenant.

apcpedagogie.com

14. ## Liaison avec les routes

    - **Liaison avec les routes**

        - La figure suivante illustre une visualisation de la liaison entre la route et le contrôleur :

        - On voit qu'au niveau de la route, il suffit de désigner le nom du contrôleur et le nom de la méthode séparés par `::`.
        - La syntaxe à utiliser est **`NomDuController::méthode`**.

```
use App\Controller\ApprenantController;

Route::get('/apprenants', [ApprenantController::class, 'apprenants']);
Route::get('/apprenants/{id}', [ApprenantController::class, 'show']);
```

15. **Route nommée**

    - De la même manière que nous pouvons nommer une route classique, nous pouvons aussi donner un nom à une route qui pointe une méthode de contrôleur :

```
use App\Controller\WelcomeController;

Route::get('/', [WelcomeController::class, 'index'])->name('home');
```

16. Pour lister les routes avec la commande Symfony CLI :

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

17. On voit bien que l'action est réalisée par le contrôleur avec précision de la méthode à utiliser. On trouve aussi le nom de la route.

18. ## Déplacer le code des fonctions de rappel des routes

    - Les routes sont souvent chargées de code pour gérer les actions dans leurs fonctions de rappel.
    - L'objectif est de déplacer ce code des fonctions de rappel vers des méthodes de contrôleur, afin d'obtenir un fichier de routes propre qui se contente de faire le lien entre

apcpedagogie.com

les URI de l'application et les actions associées à chacune de ces URI.

- Le fichier de routes **config/routes.yaml** devient ainsi une indexation des différentes URI avec leurs actions associées, tandis que le code métier est déplacé dans les méthodes de contrôleur.
- **Exemple de fichier de routes sans fonction de rappel**

```
app_videos_index:
    path: /videos
    controller: 'App\Controller\VideoController@index'
    methods: GET

app_videos_create:
    path: /videos/creer
    controller: 'App\Controller\VideoController@create'
    methods: GET

app_videos_store:
    path: /videos/creer
    controller: 'App\Controller\VideoController@store'
    methods: POST

app_videos_show:
    path: /videos/{id}
    controller: 'App\Controller\VideoController@show'
    methods: GET
```

Sommaire du cours Laravel

Source : https://apcpedagogie.com/creer-et-utiliser-un-controleur-symfony · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com