|  |
| --- |
|  |
| *Quelques notes sur l’utilisation de Symfony 6* |
|  |

|  |
| --- |
|  |

[Installation 2](#_Toc114309850)

[Prérequis d’installation 2](#_Toc114309851)

[Exécutable Symfony CLI 2](#_Toc114309852)

[Créer un environnement de test / dev 2](#_Toc114309853)

[Première page 4](#_Toc114309854)

[Création d’un contrôleur 4](#_Toc114309855)

[Ajout d’une 2ème méthode au contrôleur 5](#_Toc114309856)

# Installation

Ces notes sont liées à l’excellent cours Symfony 6 qui est [ici](https://www.youtube.com/playlist?list=PLl3CtU4THqPawV0hRF8Qqn0RVEHSjYgfy).

## Prérequis d’installation

1. PHP8 : prendre la dernière version. Fera partie d’une installation WAMP / XAMP.
2. Composer : Sur ce [site](https://getcomposer.org/download/). C’est l’équivalent de npm dans le monde PHP. Sur Windows on lance une programme d’installation.
3. Symfony CLI : Voir le paragraphe suivant.
4. Un SGBD : par exemple [MySQL](https://www.mysql.com/fr/downloads/).
5. [NodeJS](https://nodejs.org/en/download/) pour installer facilement les dépendances css et js.

1 et 2 sont nécessaires pour 3.

Pour le point 5, il s’agit essentiellement de disposer de npm (*Node Packet Manager*), par exemple pour installer les composants de Bootstrap qui va être utilisé dans la formation.

Sur un PC Windows, XAMP ou WAMP permettront de mettre rapidement en place MySQL, Apache et autres outils indispensables.

En fait avec Symfony seul MySQL sera utilisé.

## Exécutable Symfony CLI

Récupérer l’exécutable correspondant à votre O.S client [ici](https://symfony.com/download).

Selon votre O.S, choisir la bonne méthode d’installation.

Sur Windows utiliser simplement le .zip qui est [ici](https://github.com/symfony-cli/symfony-cli/releases/download/v5.4.13/symfony-cli_windows_amd64.zip).

L’extraire et s’assurer que son répertoire d’installation est dans le PATH système (ou user).

# Créer un environnement de test / dev

La documentation complète est [ici](https://symfony.com/doc/current/index.html).

Pour créer un projet Symfony, suivre les instructions de cette [page](https://symfony.com/doc/current/setup.html).

Se positionner sur le répertoire parent de la future application.

Créer une application WEB avec cette commande :

symfony new sf6Techwall --webapp

Un répertoire est créé pour le projet :

Une image contenant texte

Description générée automatiquement

* .env est un fichier contenant des variables externes aux programmes PHP. Il ne doit jamais être synchronisé sur GIT !
* composer.json décrit la liste exhaustive de tous les composants de l’application.
* bin contient des exécutables et des utilitaires, comme la console. Cette dernière permet d’utiliser symfony avec des lignes de commande ( CLI ).
* config contient les fichiers de configuration des packages et des services. Il peut aussi contenir les définitions de routes pour les requêtes mais une autre méthode est généralement préférée.
* migrations contient les programmes qui font évoluer la structure de la base de données.
* public contient index.php qui est le contrôleur frontal par lequel passe toutes les requêtes.
* src va contenir le code de l’application.
* templates qui contient des fichiers au format twig permet de partager la structure des pages entre de multiples vues de l’application.
* var contient les fichiers de cache et ceux de log.
* vendor contient toutes les bibliothèques du projet.

# Première page : Création d’un contrôleur

symfony console make :controller

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

> **first**

created: src/Controller/FirstController.php

created: templates/first/index.html.twig

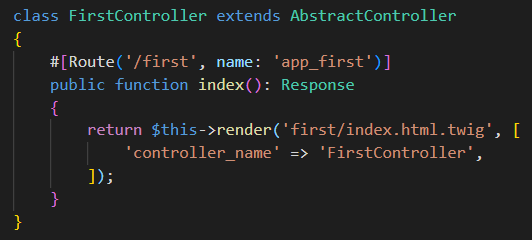
Le résultat dans les répertoires src :

Une image contenant texte

Description générée automatiquement

Symfony ajoute automatiquement la chaine « Controller » au nom du contrôleur. Il gère également la casse sur les lettres. Enfin il crée un template associé à ce contrôleur dans un dossier qui porte son nom, voir ci-dessus.

La lecture du code dans FirstController.php montre l’utilisation d’attributs PHP8 pour associer une route à une fonction dans le contrôleur. La ligne #[Route….

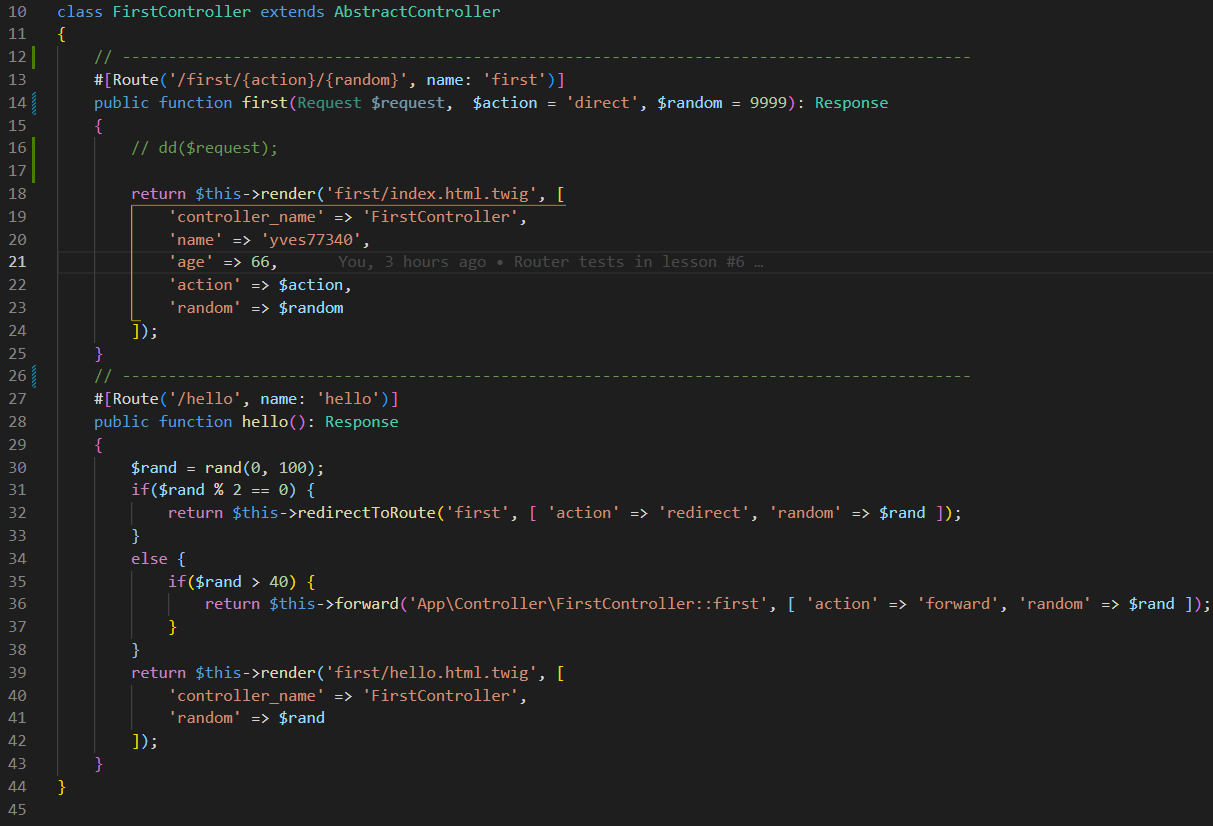


En appelant l’URL <http://127.0.0.1:8000/first> on affichera la page demandée.

# 2ème page, et ajout d’une route et d’une méthode au contrôleur

On va créer une page et sa route pour afficher un message Hello.

Puis on teste 3 méthodes de renvoi de réponse.

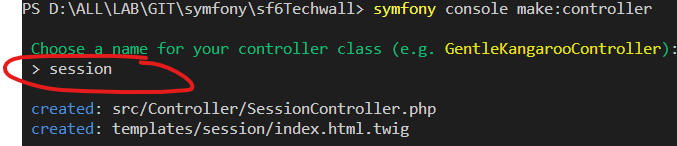


Les points à noter :

1. Ligne 13 : La route first est maintenant paramétrée.
2. Ligne 14 : La méthode associée également.
3. Sur la base d’un nombre aléatoire on utilise 3 méthodes différentes pour gérer la requête.
   1. Ligne 39 : Vu précédemment, un render().
   2. Ligne 32 : Une redirection vers first, avec des paramètres.
   3. Ligne 36 : Un Forward vers first, avec des paramètres.
4. Les méthodes render, redirectToRoute, et forward sont fournies par la classe de base AbstractController.

Dans les 3 cas, la page finale first/index.html.twig utilise bien sûr les paramètres random et action.

# Création d’un contrôleur pour la gestion de session



Les deux fichiers habituels sont créés.