

PREMIÈRE PAGE : CONTRÔLEUR, ROUTE & VUE :

Cours Symfony 4

Mohamed CHEMINGUI

Institut Supérieur des Arts Multimédia de la Manouba

3.1 CONTRÔLEUR

Un élément indispensable de l'architecture MVC

le contrôleur = classe en PHP...

Contenant une méthode / fonction PHP

Qui va permettre de lire une requête et renvoyer une réponse,

→ recevoir une requête et retourner une réponse.

Il « utilise » tous les autres composants (base de données, formulaires, templates, etc.) pour générer la réponse suite à notre requête ;

2 modes de création:

1. Création manuelle
2. Création avec la commande

« php bin/console make:controller CurrentHourController »

3.1.1 CRÉATION MANUELLE D'UN CONTRÔLEUR

Créer une page /today / date – qu'affiche la date d'aujourd'hui.

Pour ce faire, créez ;

- une classe PHP «TodayDateController» sous le dossier src\Controller
- et une fonction «date» à l'intérieur

NB: par convention le nom de contrôleur doit suivre les règles en ci-dessous:

- Chaque première lettre des mots de nom est en majuscule
- Le nom est suffixé par « Controller »

3.1.1 CRÉATION MANUELLE D'UN CONTRÔLEUR

```
1  <?php
2
3  // src/Controller/TodayDateController.php
4
5  namespace App\Controller; //Tous les controleurs sont définis dans un namespace App\Controller
6
7  use Symfony\Component\HttpFoundation\Response; // « importer la classe Response, que le contrôleur doit renvoyer.
8
9
10 class TodayDateController
11 {
12
13
14     public function date()
15     {
16         $date = date("d/m/Y");
17
18         return new Response(
19             '<html><body>Today is: '.$date.'</body></html>'
20         ); //Le contrôleur crée et renvoie un objet Response.
21     }
22 }
23
```

3.1.1 CRÉATION MANUELLE D'UN CONTRÔLEUR

Explication du contenu de Contrôleur :

Ligne 5 : Symfony tire parti de la fonctionnalité d'espace de noms de PHP pour créer un espace de noms dans toute la classe de contrôleur.

Ligne 7 : « use » importe la classe Response, que le contrôleur doit renvoyer.

Ligne 10 : La classe peut techniquement être appelée n'importe quoi, mais elle est suffixée par Controller par convention (naming rules).

Ligne 18 : Le contrôleur crée et renvoie un objet Response.

3.1.1 CRÉATION MANUELLE D'UN CONTRÔLEUR

La méthode «date» de contrôleur:

- Renvoie la date d'aujourd'hui,
- Doit être liée à une URL publique (par exemple /today/date)
- Elle est appelée lorsqu'un utilisateur y accède,

→ une route dans le fichier « config / routes.yaml » dans l'étape suivante

3.1.2 CRÉATION / GÉNÉRATION D'UN CONTRÔLEUR

« php bin/console make:controller CurrentHourController »

- php bin/console : Une commande représente en quelque sorte la ligne de commande de Symfony. Elle va nous permettre de créer nos contrôleurs, créer des entités,...
- make:controller: pour créer un contrôleur nommé « CurrentHourController »

À vos claviers « php bin/console make:controller CurrentHourController »

3.1.2 CRÉATION / GÉNÉRATION D'UN CONTRÔLEUR

2 fichiers ont été créés :

- 1-« CurrentHourController.php » un contrôleur généré dans src\Controller.
- 2- « index.html.twig » une vue générée dans templates/current_hour

```
created: src/Controller/CurrentHourController.php  
created: templates/current_hour/index.html.twig
```

Success!

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

3.1.2 CRÉATION / GÉNÉRATION D'UN CONTRÔLEUR

Voilà le contenu de fichier src/Controller/CurrentHourController.php

```
<?php

namespace App\Controller;

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

class CurrentHourController extends AbstractController
{
    /**
     * @Route("/current/hour/", name="current-hour")
     */

    public function index()
    {
        |

        return $this->render('current_hour/indexn.html.twig', [
```

3.1.2 CRÉATION / GÉNÉRATION D'UN CONTRÔLEUR

Une route a été définie en utilisant les annotations

« Contrôleur » extends un `AbstractController` pour utiliser des methodes tels que `(render (), redirectToRoute ())` et `createNotFoundException ()` .

La méthode `render ()` rend un `template` et place ce contenu dans un objet `Response`,

Les templates dans `Symfony` sont créés avec `Twig`.

3.2 ROUTE

Lien entre une URL et un contrôleur

Les routes peuvent être configurés:

- YAML,
- XML,
- PHP,
- ou à l'aide d'annotations. recevoir une requête et retourner une réponse.

nous allons voir les 2 méthodes :

1. Avec YAML,
2. Avec les Annotations,

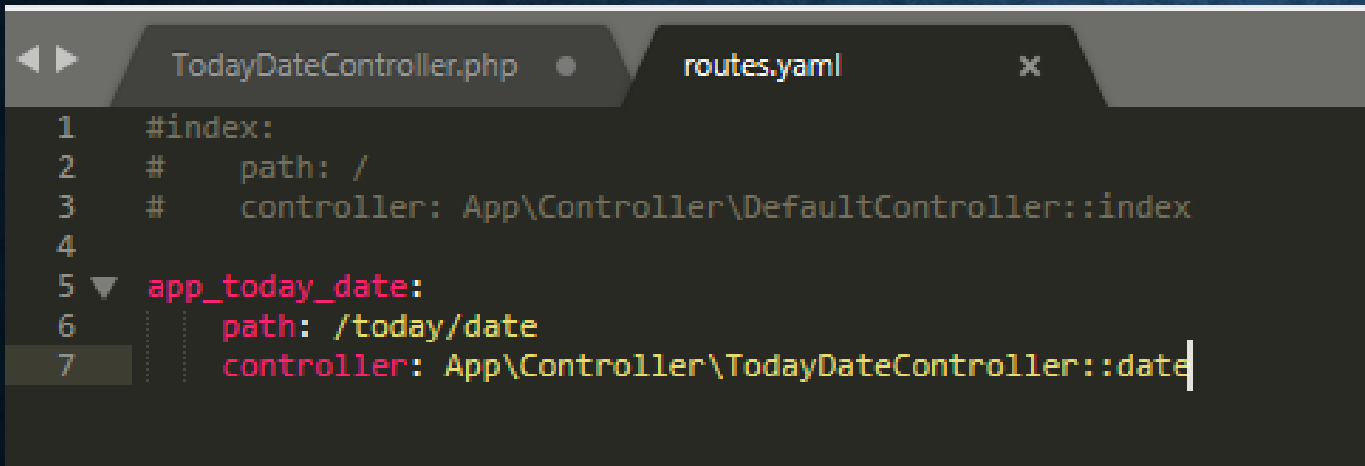
3.2 ROUTE

YAML VS ANNOTATIONS

Yaml	Annotations
Plus difficile à lire	Simple à lire et configurer
Chaque mise à jour, de la route ou les paramètres, vous devez ouvrir le contrôleur correct et fichier yaml.	Facile à mettre à jour
Plus organisée et des concepts séparés.	Route et contrôleur très proches l'un de l'autre

3.2.1 CRÉATION ROUTE AVEC YAML

1. Ouvriez le fichier « fichier config/routes.yaml »
2. Ajoutez les lignes en ci-dessous »



```
1 #index:
2 #   path: /
3 #   controller: App\Controller\DefaultController::index
4
5 ▼ app_today_date:
6     path: /today/date
7     controller: App\Controller\TodayDateController::date
```

Ligne 5 : son nom comme une variable en fait, pas d'espace, séparé par des (_)

Ligne 6 : son chemin, est ce qui va s'afficher dans le navigateur du client, (/tutorial, /a-propos, /accueil, ...) la aussi éviter les espaces, utiliser plutôt le tiret à la place (-)

Ligne 7 : nom du contrôleur puis la méthode PHP qui va traiter la requête de l'utilisateur et lui renvoyer une réponse,

Pour tester aller <http://localhost:8000/today/date>

3.2.2 CRÉATION ROUTE AVEC LES ANNOTATIONS

1. Exécutez la commande suivante pour que notre application supporte les annotations « `composer require doctrine/annotations` »
2. Commentez Le code de `routes.yaml`

```
1  #index:
2  #   path: /
3  #   controller: App\Controller\DefaultController::index
4
5  ▼ #app_today_date:
6  #   path: /today/date
7  #   controller: App\Controller\TodayDateController::date
```


3.2.2 CRÉATION ROUTE AVEC LES ANNOTATIONS

3. Ajoutez au niveau du Contrôleur `TodayDateController` :

- « `use Symfony\Component\Routing\Annotation\Route;` » pour importer les annotations
- la route de notre page comme suit :

```
11  class TodayDateController
12  {
13      /**
14       * @Route("/today/date", name="today_date")
15       */
16  }
```

L'annotation `@Route` permet d'associer un nom à la route pour qu'on puisse l'appeler

Pour tester aller <http://localhost:8000/today/date>

3.2.3 MULTI ROUTE

Associer plusieurs routes a notre méthode (* @Route("/"))

```
/**
 * @Route("/")
 * @Route("/current/hour/", name="current-hour")
 */
```

Pour tester aller <http://localhost:8000/current/hour>
<http://localhost:8000/>

3.2.3 MULTI ROUTE

Associer plusieurs routes principale à un méthode controlleur



```
1 <?php
2
3 namespace App\Controller;
4
5 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController; //pour heriter AbstractController
6 use Symfony\Component\Routing\Annotation\Route;
7
8 /**
9  * @Route("/controller")
10 */
11
12
13 class CurrentHourController extends AbstractController
14 {
15     /**
16      * @Route("/current/hour")
17      */
18     public function index()
19     {
20         return $this->render('current_hour/index.html.twig', [
21             'controller_name' => 'CurrentHourController',
22         ]); // retourner la nouvelle template + le nom du controlleur
23     }
24 }
```

<http://localhost:8000/controller/current/hour>

3.2.3 MULTI ROUTE

Définir un paramètre dans la route et l'ajouter dans l'action

```
8
9
10 class CurrentHourController extends AbstractController
11 {
12     /**
13      * @Route("/current/hour/{nom}", name="current/hour")
14      */
15     public function index(String $nom)
16     {
17         return $this->render('current_hour/index.html.twig', [
18             'controller_name' => $nom,
19         ]); // retourner la nouvelle template + le nom du controlleur
20     }
21 }
22
```

<http://localhost:8000/controller/current/hour/isamm> par exemple¹⁸

3.2.3 MULTI ROUTE

Récupérer un paramètre en utilisant l'objet requête

Ajoutez « use Symfony\Component\HttpFoundation\Request; »

```
9  class CurrentHourController extends AbstractController
10 {
11
12     /**
13      * @Route("/current/hour/{nom}", name="current-hour")
14      */
15
16     public function index(Request $request)
17     {
18         $nom = $request->get('nom');
19
20         return $this->render('current_hour/index.html.twig', [
21             'controller_name' => $nom,
22         ]); // retourner la nouvelle template + le nom du controlleur
23     }
24 }
```

<http://localhost:8000/controller/current/hour/isamm> par exemple

3.3 TWIG

Moteur de Template pour PHP.

Les vues sont gérées par le moteur de Template « Twig ».

Pourquoi TWIG ?

- Permet de séparer le code PHP du code html (lisibilité)
- Offre la possibilité de modifier un fichier sans influencer le deuxième
- Il y a quelques fonctionnalités en plus, comme l'héritage de templates (nous le verrons)
- Facilite le travail d'équipe

3.3 TWIG

Trois types de balises pour l'affichage:

{% ... %} : pour executer une action

{# ... #} : pour definir un commentaire

{{ ... }} : pour afficher

A voir dans le contenu:

« templates\current_hour\index.html.twig »

3.3 TWIG

APPELER UNE VUE DEPUIS LE CONTROLLEUR

- Au sein de la méthode que nous allons appeler le template.
- Cela se fait très simplement avec la méthode `$this->render()`.
- Cette méthode prend en paramètre le nom du template et retourne un objet de type `Response` pour afficher le contenu de notre template.

3.3 TWIG

Ouvrez le fichier `index.html.twig` qui se trouve dans `templates/current_hour`, nous allons remplacer son contenu par ce qui suit:

```
1  {% extends 'base.html.twig' %}
2
3  {% block title %}Hello CurrentHourController!{% endblock %}
4
5  {% block body %}
6  <style>
7      .example-wrapper { margin: 1em auto; max-width: 800px; width: 95%; font: 18px/1.5 sans-serif; }
8      .example-wrapper code { background: #F5F5F5; padding: 2px 6px; }
9  </style>
10
11  <div class="example-wrapper">
12      <h1>The current time is {{ time }}! ✓</h1></div>
13  {% endblock %}
14
```

3.3 TWIG

Ouvrez le fichier « CurrentHourController.php » qui se trouve dans src\Controller, nous allons remplacer son contenu par ce qui suit:

```
class CurrentHourController extends AbstractController
{
    /**
     * @Route("/current/hour/", name="current-hour")
     */

    public function index()
    {
        $hour = date("h:i:sa");

        return $this->render('current_hour/indexn.html.twig', [
            'time' => $hour,
        ]); // retourner la nouvelle template + le nom du controleur
    }
}
```

Pour Tester: <http://localhost:8000/current/hour/>

3.3 TWIG

Faire appel à une route depuis une page TWIG:

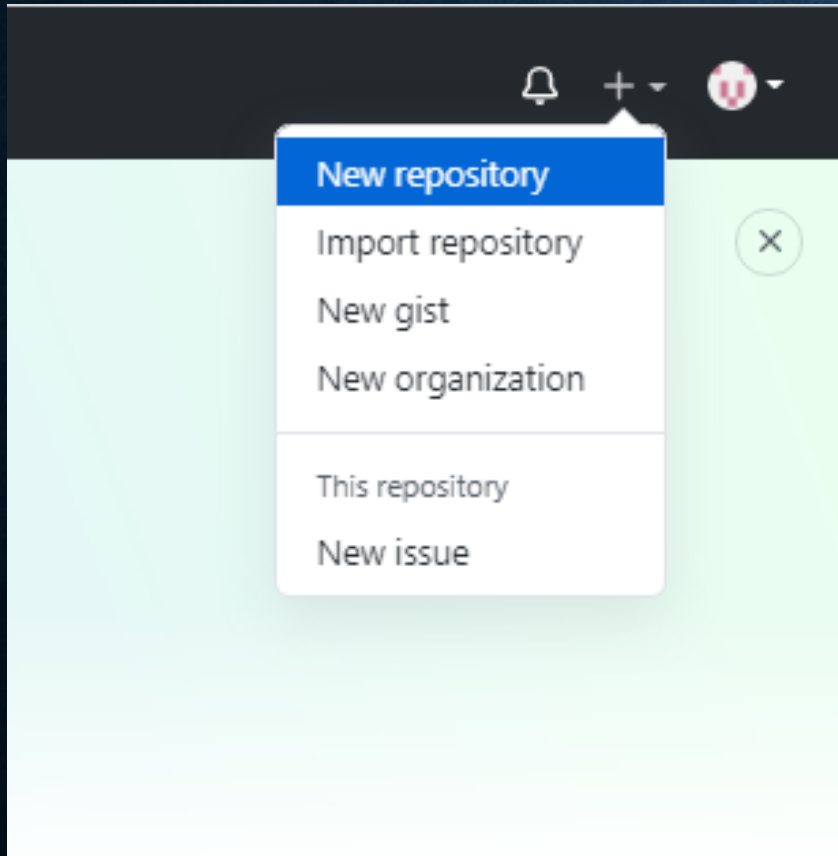
- C'est la fonction « path » qu'il faut utiliser depuis un template

Twig :

```
<a href="{{ path('modifiervoiture', {mat: voiture.id})
}}" target="_blank">Modifier</a>
```


3.4 GITHUB



1. Connectez vous sur sur Git Hub
2. Créez un nouveau projet



Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * Repository name *

 medchemingui / nom-du-projet 

Great repository names are short and memorable. Need inspiration? How about super-duper-octo-telegram?

Description (optional)

☐  Public
Anyone on the internet can see this repository. You choose who can commit.

☒  Private
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

[Create repository](#)

3.4 GITHUB

3. Lancez l'invite de commande dans le dossier du projet
4. Exécutez les commande suivantes:
 - `git init`
 - `git config --global user.email mohamedchemingui@gmail.com`
 - `git config --global user.name medchemingui`
5. Récupérez le lien du projet à partir du GITHUB

Quick setup — if you've done this kind of thing before



Set up in Desktop

or

HTTPS

SSH

<https://github.com/medchemingui/nom-du-projet.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a README file.

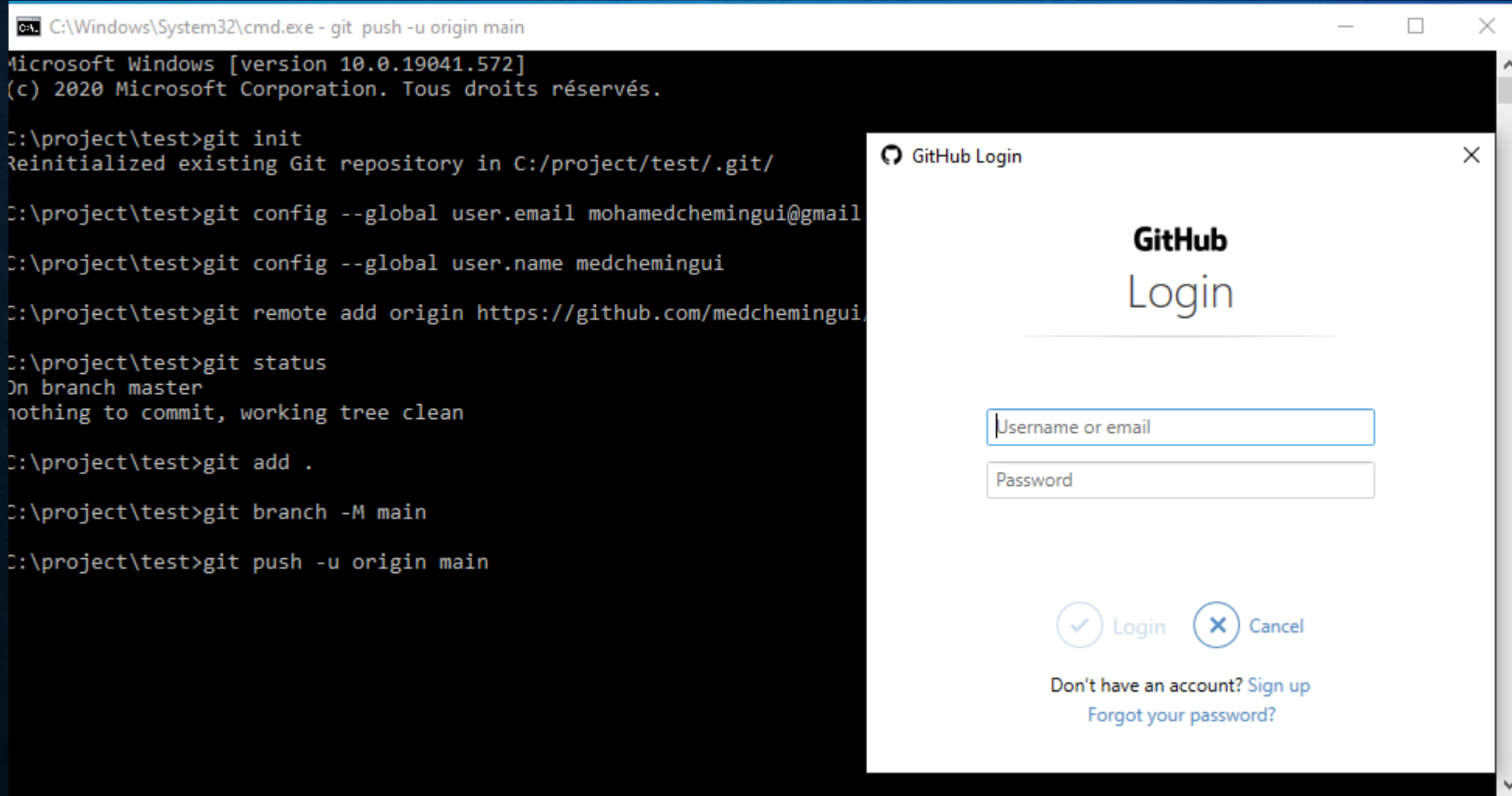
3.4 GITHUB

6. Exécutez les commande suivantes:

- `git remote add origin « +lien copié à l'étape 5 »`
- `git branch -M main`
 - si il y a un erreur de création de branch:
 - « exécuter `git commit -m "v1"`
- `git push -u origin main`

3.4 GITHUB

Après l'exécution de la commande « `git push -u origin main` »
vous allez avoir cette interface → Connectez vous



3.4 GITHUB

Actualisez votre projet sur Git Hub

medchemingui / nom-du-projet Private

<> Code ⓘ Issues 🔗 Pull requests ⚙️ Actions 📁 Projects 🛡️ Security 📈 Insights ⚙️ Settings

main 1 branch 0 tags Go to file Add file Code

unknown Add initial set of files ce6f635 21 days ago 1 commits

bin	Add initial set of files	21 days ago
config	Add initial set of files	21 days ago
migrations	Add initial set of files	21 days ago
public	Add initial set of files	21 days ago
src	Add initial set of files	21 days ago
templates	Add initial set of files	21 days ago
tests	Add initial set of files	21 days ago
translations	Add initial set of files	21 days ago
.env	Add initial set of files	21 days ago
.env.test	Add initial set of files	21 days ago
.gitignore	Add initial set of files	21 days ago
composer.json	Add initial set of files	21 days ago
composer.lock	Add initial set of files	21 days ago
phpunit.xml.dist	Add initial set of files	21 days ago
symfony.lock	Add initial set of files	21 days ago

3.4 GITHUB

1. git status pour vérifier s'il y a des fichiers non synchronisés

```
C:\project\test>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Nous allons créer un contrôleur par exemple

```
C:\project\test>php bin/console make:controller CurrentHourController

created: src/Controller/CurrentHourController.php
created: templates/current_hour/index.html.twig

Success!

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

3.4 GITHUB

2. git status pour vérifier s'il y a des fichiers non synchronisés

```
C:\project\test>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    src/Controller/CurrentHourController.php
    templates/current_hour/

nothing added to commit but untracked files present (use "git add" to track)
```

Pour synchroniser lancer les commandes suivantes:

- git add .
- git status
- git commit -m "v2"
- git push

3.5 RÉSUMÉ

Le rôle du routeur est de déterminer quel route utiliser pour la requête courante.

Le rôle d'une route est d'associer une URL à une action du contrôleur.

Le rôle du contrôleur est de retourner au noyau un objet Response, qui contient la réponse HTTP à envoyer à l'internaute (page HTML ou redirection).

Le rôle des vues est de mettre en forme les données que le contrôleur lui donne, afin de former une page HTML, un e-mail, etc.