Simple MVC

Rappel de l'architecture des dossiers :

- src/Controller : Contient tous les controllers.
- src/Model : Contient tous les managers.
- src/View : Contient tous les templates twig.

Autres rappels :

- Le dossier racine de twig est le dossier src/View
- Les fichiers s'appellent du même nom que la classe qu'ils contiennent
- Les controllers héritent de la classe *AbstractController*
- Les managers héritent de la classe *AbstractManager*

Etapes d'ajout d'un formulaire en partant de la création du formulaire jusqu'à l'insertion dans la base de données :

- Initialisation : Création du controller qui contrôlera la route
- HTML : Création du fichier de template twig
- Control : Vérification des champs du formulaire dans le controller sur la request post
- Manager : Création du manager pour insérer dans la base
- Liaison du controller et du manager

Astuces : Les étapes sont dans l'ordre.

Initialisation de la route

Pour afficher quelque chose dans le navigateur, il faut initialiser une url qu'on appellera également **route** à l'aide d'un controller.

Une route se compose de : */nomducontroller/nomdelaméthode*

Il faut donc définir quelle est l'adresse que l'on souhaite initialiser.

Ex : Prenons le cas d'un formulaire d'inscription. La route sera : `/user/register`

On doit créer dans le dossier **src/Controller** un fichier **UserController.php** qui contiendra la classe **UserController** avec la méthode **register()**

La méthode du controller devra appeler le template twig que vous souhaitez afficher.

```
// UserController.php
namespace App\Controller;

class UserController extends AbstractController {       // Pas besoin
d'utiliser de use car AbstractController

// est dans le même namespace que UserController
```

```
        public funtion register() {
        return $this->twig->render('User/register.html.twig'); // Ce
template n'existe pas encore, nous allons le créer à l'étape suivante
        }
}
```

HTML : Création du template

Il faut maintenant créer le template twig dans le dossier **src/View**. Dans l'exemple précédent le fichier du template s'appelle **register.html.twig** et se trouve dans le dossier **src/View/User**. Si le dossier *User* n'existe pas, à vous de le créer. Par convention, on range les templates qui concernent la même partie fonctionnelle dans le même dossier.

```
{% extends 'layout.html.twig' %}        // Héritage du layout principal qui
contient les entêtes html,
                                                                        //
l'inclusion de bootstrap et notre style.css

{% block content %}
    <div class="container">
        <!-- Default form login -->
        <form class="text-center border border-light p-5" method="post">

            <p class="h4 mb-4">Sign up</p>

                        <!-- Firstname -->
            <input type="text" name="firstname" class="form-control mb-4"
placeholder="Firstname">

            <!-- Lastname -->
            <input type="text" name="lastname" class="form-control mb-4"
placeholder="Lastname">

            <!-- Email -->
            <input type="email" name="email" id="defaultLoginFormEmail"
class="form-control mb-4" placeholder="E-mail">

            <!-- Password -->
            <input type="password" name="password"
id="defaultLoginFormPassword" class="form-control mb-4"
placeholder="Password">

            <!-- Sign in button -->
            <button class="btn btn-info btn-block my-4" type="submit">Sign
up</button>

            <!-- Register -->
            <p>back
                <a href="/user/signin">Sign in</a>
            </p>
```

```
        </form>
        <!-- Default form login -->
    </div>
{% endblock %}
```

Ne pas oublier :

- Mettre la méthode POST sur le formulaire
- Mettre les attributs 'name' sur les champs
- Mettre un bouton submit

Lors du clic sur submit, si le champ d'action du formulaire n'est pas défini, la page est rechargée en POST, la même route est donc appelée :

```
POST http://localhost:8000/user/register
```

Control : Vérification du formulaire

Même adresse donc même controller et même méthode mais en POST cette fois-ci.

Nous vérifions le formulaire comme nous l'avons appris en vérifiant bien au préalable que nous sommes dans la méthode POST.

```
// UserController.php
namespace App\Controller;

class UserController extends AbstractController {

        public funtion register() {
                if ($_SERVER['REQUEST_METHOD'] === 'POST') {
                        // Création d'un tableau associatif local pour plus
                        // de simplicité d'utilisation
            $user = [
                'firstname' => $_POST['firstname'],
                'lastname'  => $_POST['lastname'],
                'email'     => $_POST['email'],
                'password'  => $_POST['password']
            ];

            // Vérification des champs rentrés dans le formulaire
            // checkUser est une méthode privée définie plus bas
            $errors = $this->checkUser($user);
            if (empty($errors)) { // S'il n'y a pas d'erreurs
                // Pas d'erreurs, on insère l'utilisateur
                // dans la base de données en utilisant le manager que nous
                // allons créer à l'étape suivante

                // La liaison de la DERNIERE ETAPE se fera ici
            }

        }
```

```
        return $this->twig->render('User/register.html.twig');
        }

        /**
     * Return an errors array. Check each fields.
     * @param  array $user informations d'un utilisateur
     * @return array       tableau associatif des erreurs
     */
    private function checkUser($user) {
        $errors = [];

        if (!isset($user['firstname']) || empty($user['firstname'])) {
            $errors['firstname'] = true; // Vous pouvez gérer les erreurs
avec des chaînes de caractères si vous préférez
        }

        // Hahaha vous ne croyez quand même pas que j'allais vous donner
        // cette méthode en entier

        return $errors;
    }
}
```

Manager : Insertion dans la base

On crée un fichier manager dans le dossier **src/Model**. Il hérite de **AbstractManager** On crée la méthode d'intéraction avec la base de données qui nous intéresse. Dans notre exemple c'est la méthode **insert** pour insérer un utilisateur dans la base.

```
// src/Model/UserManager.php
namespace App\Model;

/**
 * Manage interaction with user table in the database
 */
class UserManager extends AbstractManager
{
    /**
     *
     */
    const TABLE = 'user'; // NOM DE LA TABLE A COMPLETER !!!!!!!!!

    /**
     *  Initializes this class.
     */
    public function __construct()
    {
        parent::__construct(self::TABLE);
    }

    /**
```

```
     * Insère un utilisateur par rapport au tableau associatif $user
     * passé en paramètre.
     * @param array $user       tableau associatif des champs d'un
utilisateur
     * @return int                     renvoie l'identifiant de
l'utilisateur créé
     */
    public function insert(array $user): int
    {
        // prepared request
        // pdo est hérité de la classe parente, allez y jeter un oeil ;)
        $statement = $this->pdo->prepare("INSERT INTO $this->table
(firstname, lastname, email, password) VALUES (:firstname, :lastname,
:email, :password)");

        $statement->bindValue('firstname', $item['firstname'],
\PDO::PARAM_STR);
        $statement->bindValue('lastname', $item['lastname'],
\PDO::PARAM_STR);
        $statement->bindValue('email', $item['email'], \PDO::PARAM_STR);
        $statement->bindValue('password', $item['password'],
\PDO::PARAM_STR);

        if ($statement->execute()) {
            return (int)$this->pdo->lastInsertId();
        }
    }

}
```

Liaison du controller et du manager

Dernière étape, appeler la méthode **insert($user)** du **UserManager** dans le **UserController**

Rappel :

- Le UserManager est une classe

Il faut instancier le manager pour récupérer un objet et ensuite appeler la méthode voulue sur cet objet.

```
use App\Model\UserManager; // Les use se font TOUJOURS en début de fichier

$userManager = new UserManager();
$userManager->insert($user);
```

Ensuite à vous de choisir si vous voulez effectuer une redirection `header('Location: /user/index');exit;` c'est à dire afficher une autre page ou bien réafficher le même template twig.

Pour cette dernière étape, à vous de jouer. Vous devez l'insérer au bon endroit du **UserController**