TD PHP MVC

Étape 18 : Login et logout

Avant de réaliser l'authentification, nous allons créer un 2ème customer lié à un app_user. Nous allons également hasher les mots de passe en base de données.

```
INSERT INTO `customer` (`id`, `firstname`, `lastname`, `address`, `city`, `phone`) VALUES
(2, 'John', 'Doe', 'somewhere', 'inTheWorld', '0987654321');

INSERT INTO `app_user` (`id`, `login`, `password`, `role`, `customer_id`) VALUES
(2, 'user@user.fr', 'MWplR1V4bWxTcUd3RVFLaA$VDxsXUhcnlLiIVHKe26oMQ', 2, 2);
```

Le mot de passe hashé ci dessus correspond en fait à « user ». J'ai utilisé un hash generator en ligne pour l'algorithme argon2id → https://argon2.online/

On fait la même chose pour hasher le mot de passe de l'utilisateur 1 « admin » qui devient par exemple en fonction des paramètres de hashage et du salt aléatoire :

$argon2id$v=19$m=1024,t=2,p=2$MHF4MTJzQndUUndwQXNFNw$XLNt/dk5T4S4F7LHBj3Kbg

On ne stocke en base que la partie après p=2$ à savoir : MHF4MTJzQndUUndwQXNFNw$XLNt/dk5T4S4F7LHBj3Kbg

La partie précédant le mot de passe hashé correspond aux paramètres de hashage, nous utiliserons toujours les mêmes dans notre application, nous allons donc stocker cette partie dans une variable static de la classe UserController.

controller > user.controller.php > UserController

```
48
49      private static $prefix = '$argon2id$v=19$m=1024,t=2,p=2$';
```

Une fois ces modifications faites en base de données, nous allons créer une route permettant de s'authentifier : /user/login

Commençons par le template

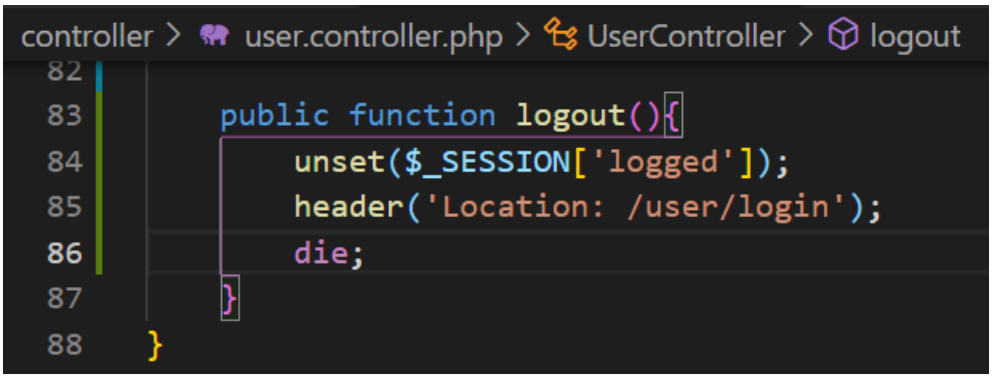
template > user > login.view.php

```
1   <h2>Authentification</h2>
2   <form action="" method="post">
3       <label for="name">Identifiant</label><br>
4       <input type="email" name="login" id="login" value="<?= $posted['login'] ?? ''; ?>" required>
5       <br><br>
6       <label for="name">Mot de passe</label><br>
7       <input type="password" name="password" id="password" value="<?= $posted['password'] ?? ''; ?>" required>
8       <br><br>
9       <div class="d-flex">
10          <input type="submit" name="validate" id="validate" value="Valider">
11          <input type="submit" name="cancel" id="cancel" value="Annuler" class="ml-2">
12      </div><br>
13      <label class="text-danger"><?= count($errors) > 0 ? 'Identifiant ou mot de passe incorrect.' : ''; ?></label>
14  </form>
```

Puis la fonction login du controller

Nous utilisons password_hash pour comparer le mot de passe en clair saisi par l'utilisateur et celui hashé en base de données.
En cas de succès nous stockons pour l'instant true dans une variable de session logged puis nous redirigeons vers la route /home, sinon nous affichons le message d'erreur dans le formulaire du template.

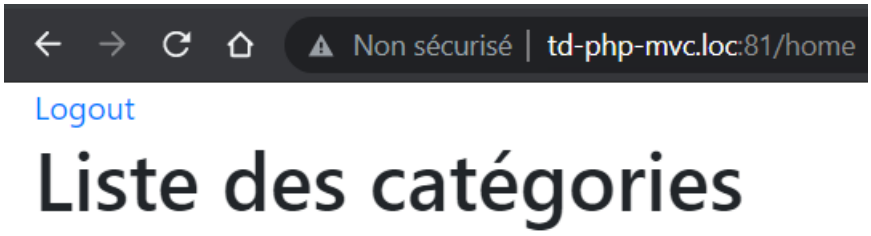
Pour le logout, nous créons pour l'instant la route /user/logout qui ira supprimer la variable de session logged. Pas besoin e template, juste la méthode logout dans le controller.



Afin de voir la différence entre utilisateur authentifié ou non-authentifié, on peut afficher dans le header de notre page index.php (point d'entrée de l'application) un lien permettant l'une ou l'autre action (login/logout) suivant l'état de l'authentification.

```php
index.php > ...
<div class="container-fluid">
    <header>
        <?= isset($_SESSION['logged']) &&  $_SESSION['logged'] == true ?
            '<a href="/user/logout">Logout</a>' :
            '<a href="/user/login">Login</a>'?>
    </header>
```

Résultat une fois authentifié :



Après un logout :