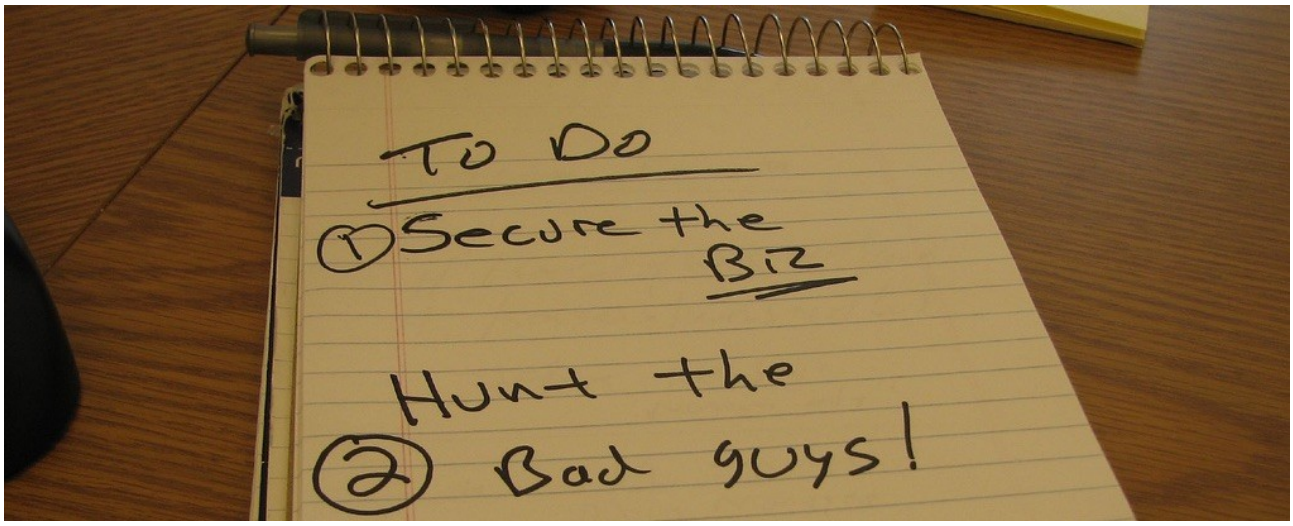
ToDo & Co

Documentation technique



Technologies et installation

Le projet utilise la version 6.4 de Symfony ainsi que la version 8.1 de PHP.

Pour installer le projet sur votre machine, clonez le repository grâce à la commande suivante :

→ **$ git clone https://github.com/HyacineAlnuma/PHP-P8.git**

Pour ce qui est des librairies utilisées, tout est listé dans le fichier composer.json. Afin d'installer celles-ci sur votre machine, il vous faudra dans un premier temps installer composer :

→ https://getcomposer.org/download/

Enfin, il vous suffira de lancer la commande suivante :

→ **$ composer install**

Pour mettre en place la base de données en local, modifier la variable DATABASE_URL dans le fichier .env en mettant votre propre configuration.

Ensuite lancer les commandes suivantes afin de créer les bases de données (dev/prod et test) et charger les fixtures :

→ **$ php bin/console doctrine:database:create**

→ **$ php bin/console doctrine:schema:update --force**

→ **$ php bin/console doctrine:database:create --env=test**

→ **$ php bin/console doctrine:schema:update –force --env=test**

→ **$ php bin/console doctrine:fixtures:load**

→ **$ php bin/console doctrine:fixtures:load --env=test**

Pour ce qui est des extensions de PHP, il vous faudra installer Xdebug afin de pouvoir générer le rapport de couverture de code :

→ https://xdebug.org/docs/install

Vous pouvez désormais lancer le serveur de Symfony grâce à la commande suivante :

→ **$ symfony server:start**

Le site sera à ce moment consultable à l'adresse suivante :

→ https://localhost:8000

Afin de tester le code et générer le rapport de couverture, lancez la commande suivante :

→ **vendor/bin/php --coverage-html public/test-coverage**

Authentification

Fichiers

Type	Fichier
Entité	src/Entity/User.php
Controller	src/Controller/SecurityController.php
Template	templates/security/login.html.twig
Authentification	src/Security/AppCustomAuthenticator.php
Configuration	config/packages/security.yaml

Fonctionnement

Tout d'abord le controller va envoyer des données à la vue, notamment le dernier nom d'utilisateur utilisé ou en cas d'erreur.

La vue va être affichée grâce à la méthode render qui va appeler le template login.html.twig. C'est dans ce template que se trouve le formulaire d'authentification que l'utilisateur va remplir avec ses informations de connexion.

Ces credentials sont ensuite récupérer par la classe AppCustomAuthenticator. Dans celle-ci un « passeport » va être créé avec les crendentials de l'utilisateur et le token csrf récupéré dans le formulaire. Une fois l'authentification validée, l'utilisateur est redirigé vers la route task_list grâce à la RedirectResponse.

Vous pouvez retrouver ici toute la documentation concernant la gestion de l'authentification dans une application Symfony :

→ https://symfony.com/doc/current/security.html

Configuration

Toute les configurations qui concernent l'authentification se trouvent dans le fichier security.yaml cité plus haut. Dans ce fichier plusieurs choses sont possibles :

- Modifier le fichier en charge de la gestion de l'authentification par l'application :

```yaml
        main:
            lazy: true
            provider: app_user_provider
            custom_authenticator: App\Security\AppCustomAuthenticator
            logout:
                path: app_logout
                # where to redirect after logout
                # target: app_any_route
```

- Modifier le contrôle d'accès des différentes page de l'application :

```yaml
    access_control:
        - { path: ^/users/list, roles: ROLE_ADMIN }
        - { path: ^/users/edit, roles: ROLE_ADMIN }
        - { path: ^/tasks, roles: ROLE_USER }
        - { path: ^/login, roles: PUBLIC_ACCESS }
        - { path: ^/users/create, roles: PUBLIC_ACCESS }
        - { path: ^/, roles: ROLE_USER }
```

- Modifier le système d'encryptage du mot de passe des utilisateurs :

```yaml
    security:
        password_hashers:
            # By default, password hashers are resource intensive and take time. This is
            # important to generate secure password hashes. In tests however, secure hashes
            # are not important, waste resources and increase test times. The following
            # reduces the work factor to the lowest possible values.
            Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
                algorithm: auto
                cost: 4 # Lowest possible value for bcrypt
                time_cost: 3 # Lowest possible value for argon
                memory_cost: 10 # Lowest possible value for argon
```

Entités et Base de données

Voici quelques commandes utiles qui vous permettrons de gérer les entités ainsi que la mise à jour des tables dans la base de données.

Afin de créer ou de mettre à jour une entité, lancez la commande suivante :

→ **php bin/console make:entity**

Une fois que vous avez créé ou modifié une entité, afin de mettre à jour la base de données vous devez dans un premier temps créer une migration grâce à la commande suivante :

→ **php bin/console make:migrations**

Pour exécuter la migration lancez la commande suivante :

→ **php bin/console doctrine:migrations:migrate**

Votre base de données est désormais à jour !