LE DOCUMENT TECHNIQUE

Table des matières

Informations

Nom du projet	TodoList
Type de document	Documentation technique
Langage	PHP 8.0
SGBD	Mysql
Framework	Symfony 5.4
Framework Test	PHP-UNIT
Date	04/03/2022
Auteur	Granger Kénolane

Pré-requis

OS	Windows
Env	Docker
Github	https://github.com/bangix28/P8-openclassrooms

Introduction

Contexte

Todolist est une application permettant de gérer ses tâches quotidiennes, c'est une application PHP sous le framework Symfony 5.4.

Normes de codages

Gestionnaire de librairie	Composer 2
PSR	PSR-4
Classe	PascalCase
Fonction	camelCase
Routes Symfony	`#[...] attribute native PHP`
Méthode de développement	Test driven development

Mise en place du projet

Configuration de l'environnement de dév sur PC Win :

Tous les fichiers de ce projet sont placés sur Github à l'adresse suivante : https://github.com/bangix28/P8-openclassrooms

Récupérez l'application via Git et créer une branche Locale afin de travailler sur votre propre branche.

-Ouvrez votre invite de commandes et mettez-vous à la racine du projet dans le but de construire l'image Docker via Docker build avec la commande suivante :

```
docker build -t p8 .
```

-Montez l'image via Docker-compose avec la commande suivante :

```
docker-compose up -d
```

-Installez les dépendances est le fichier .env avec composer install.

```
composer install
```

-Dans le fichier .env définissez la base de données.

```
DATABASE_URL="mysql://root:@mysql/p8"
```

-Créez la base de données.

```
php bin/console doctrine:database:create
```

- Importez les entités dans la base de données.

```
php bin/console doctrine:schema:create
```

- Installez les fixtures.

```
php bin/console doctrine:fixture:load
```

-Créez la base de données de test.

```
php bin/console doctrine:database:create --env=test
```

- Importez le schéma de la base de données de test.

```
php bin/console doctrine:schema:create --env=dev
```

Test avec PHP-UNIT

La méthode de développement est le Test Driven Development afin d'appliquer cette méthode, nous allons utiliser PHP-unit dans le but de tester le code.
L'endroit où mettre son code à tester se trouve dans le fichier à la racine du projet se nommant tests, l'arborescence est similaire au fichier Src.
Une fois votre test écrit, il faudra tester celui-ci et pour ce faire, nous utiliserons la commande suivante dans l'invite de commandes :

```
vendor/bin/phpunit tests
```

Cette commande va lancer tous les tests dans le fichier tests.
Afin de ne pas avoir à relancer cette commande à chaque changement dans le code, nous utilisons une librairie qui va compléter PHP-UNIT qui va nous permettre au moindre changement dans le fichier de tests de lancer un test, d'en refaire la commande ci-dessus.
Cette librairie se nomme Phpunit watcher et la commande à effectuer pour les tests est celle-ci :

```
vendor/bin/phpunit-watcher watch tests
```

Code coverage avec Php-Unit et Xdebug

Pour effectuer un code coverage du code, nous utilisons PHP-UNIT et le driver Xdebug afin de créer un rapport sur les classes et fonctions tester.

Nous utilisons les commandes suivantes :

```
vendor/bin/phpunit –coverage-html <nom_du_fichier _a_ créer>
```

Cette commande va générer un rapport HTML, vous pouvez l'ouvrir avec un navigateur en lançant le fichier dashboard.html du fichier que vous avez généré.

Architecture du projet

l'architecture du projet commence avec à sa racine le fichier nginx avec les fichiers de configuration du serveur nginx pour docker.
Le fichier P8 contient les fichiers du projet web.
Il y a également les fichiers vendor et var qui servent à être injectés dans le conteneur docker.

Il reste les fichiers De docker avec Dockerfile qui permet de créer l'image docker du conteneur et docker compose qui permet d'installer les images est de les paramétrer.
L'application utilise le modèle du framework Symfony 5.4 en termes d'architecture.
Pour plus d'information sur l'architecture Symfony je vous renvoie vers la documentation.

Gestionnaire de librairies

Le gestionnaire de librairie utilisée dans ce projet est Composer.
Il permet de gérer l'injection de dépendance ainsi que le rajout de nouvelle librairie tiers.
Comme nous utilisons la version Symfony flex qui permet d'avoir une configuration et une injection automatique des nouvelles librairies.

Composer.json :

minimum-stability	stable
PHP-version	8.1
license	propriétaire

Pour mettre à jour les librairies il suffit de lancer la commande suivante :
`composer update`

Pour faire une montée de version de symfony je vous renvoie vers la documentation.

Authentification

Le service d'authentification utiliser est celui de symfony 5.4 utilisant le SecurityBundle, pour plus d'information je vous renvoie vers la documentation.

Fichier Authentification

Fichier	Fonction
src/Controller/SecurityController.php	- Redirection si l'user est auth - Envoie la vue du formulaire
p8/config/packages/security.yaml	-Configuration des différents paramètres de sécurité, notamment les paramètres pour l'authentification.
src/Security/AppAuthenticator	- Créez l'objet passeport qui stock les informations utilisateur.
src/Entity/User	- Configuration de l'objet User
Templates/security/login.html.twig	- La vue du formulaire d'authentification

Fonction des fichiers

Entité	User
Champ unique	email
authenticator	App\Security\AppAuthenticator
route	app_login
password hasher algorithm	auto
Templates/security/login.html.twig	- La vue du formulaire d'authentification

L'entité utilisez pour l'authentification est User, sont champs uniques est l'email.

SecurityController : C'est le fichier contenant la logique du formulaire, ainsi que les données des erreurs.
Ce fichier doit être modifié si la logique d'affichage du formulaire doit être changée (message d'erreur, récupération de données comme l'username, etc.) ou si la redirection d'une personne connectée doit être modifiée.

Login.html.twig: Ce fichier contient le formulaire, sa structure ainsi que les noms des variables de formulaire.
Ce fichier doit être modifié si l'on veut changer le style du formulaire.

Security.yml : ce fichier contient la logique de sécurité ainsi que l'entité à utiliser pour authentifier quelqu'un.
Ce fichier doit être modifié si l'on souhaite changer l'entité reliée à l'authentification et si l'on souhaite modifier le firewalls utiliser pour l'authentification.

Moteur de template

Le moteur de template utiliser est Twig le moteur par défaut de Symfony pour plus d'information a sont propos, je vous renvoie vers la documentation.

Webpack Encore

La gestion d'asset est gérée par Webpack-encore librairie de gestion d'asset de Symfony je vous renvoie vers la documentation pour en apprendre plus.

Pour utiliser ce bundle, il faut installer le pack JavaScript avec NPM gestionnaire de paquet pour JavaScript.

Pour ce faire, vous devez installer les différents packages NPM avec la commande suivante:

```
npm install
```

Pour retrouver les différents packages qu'il va installer vous pouvez regarder le fichier package.json a la racine du projet web, il y aura les différents scripts que NPM peut exécuter.

Quand vous ajoutez où modifier un fichier d'assets, il faudra rebuild avec la commande suivante :

```
npm run dev
```