DOCUMENTATION TECHNIQUE
ToDo & Co

Quel(s) fichier(s) modifier et pourquoi ?

Nous utilisons le framework Symfony (version 3.1.10). Cette version a été migrée vers la version 3.4

Dossiers et fichiers principaux:

- **App** ----------> (fichiers de configuration du framework et vues)
- **Bin** ----------> (console)
- **Src** ----------> (classes PHP)
- **Tests** ----------> (test unitaires et fonctionnels)
- **Var** ----------> (cache, logs, …)
- **Vendor** ----------> (librairies via composer)
- **Web** ----------->(fichiers css, js, images, etc…)

Les 2 dossiers les plus importants sont les dossiers **App** et **Src**.

1) Dossier App.

a) A la racine de ce dossier, il y a le fichier **AppKernel.php** (y ajouter un bundle en cas d'installation).

```php
if (in_array($this->getEnvironment(), ['dev', 'test'], strict: true)) {
    $bundles[] = new Symfony\Bundle\DebugBundle\DebugBundle();
    $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();
    $bundles[] = new Sensio\Bundle\DistributionBundle\SensioDistributionBundle();
    $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();
}
```

Exemple : le Bundle DoctrineFixturesBundle a été ajouté ici.

b) Le dossier **config**:

Parmi plusieurs fichiers de configuration .yml, vous serez amenés surtout à consulter voire modifier les fichiers:

- **parameters.yml** (paramètres base de données et mail).
- **security.yml** (authentification et roles).

```yaml
security:
    encoders:
        AppBundle\Entity\User: bcrypt

    providers:
        doctrine:
            entity:
                class: AppBundle:User
                property: username

    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false

        main:
            anonymous: ~
            pattern: ^/
            form_login:
                login_path: login
                check_path: login_check
                always_use_default_target_path:  true
                default_target_path:  /
            logout: ~

    role_hierarchy:
        ROLE_ADMIN:       ROLE_USER
        ROLE_SUPER_ADMIN: [ROLE_USER, ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]

    access_control:
        - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/users, roles: ROLE_ADMIN }
        - { path: ^/, roles: ROLE_USER }
```

c) Le dossier **Resources/views**:

Vous trouverez, ici, les vues (fichiers .html.twig) sur lesquelles vous serez évidemment amenés à travailler régulièrement.

2) Le dossier Src:

C'est là que se trouvent toutes nos classes.

- Controller (logique métier, annotations routes…).
- Entity (entités représentant les tables dans la base de données).
- Form (formulaires).
- Repository (requêtes DQL).

3) Le dossier Web:

Les éventuelles modifications css, js, fonts, etc, se font ici.

4) autres:

Il n'est pas nécessaire d'aller ds les dossiers **bin**, **var** et **vendor**.
Le dossier **Tests** et le fichier à la racine du projet **phpunit.xml.dist** sont en rapport avec les tests unitaires et fonctionnels. Il est fortement conseillé, à chaque modification du code, de vérifier les tests voire d'en ajouter avant de faire vos commits.
Le fichier **.travis.yml** à la racine du projet se rapporte à l'intégration continue. Des tests automatiques sont lancés avant par exemple la fusion d'une branche ou un déploiement.

Conclusion:

Vous serez donc surtout dans les dossiers **app** et **src**.

Comment s'opère l'authentification ?

1) **app\config\security.yml**
2) **src\AppBundle\Entity\User.php**
3) **src\AppBundle\Controller\SecurityController.php**

1) app\config\security.yml

- **encoders** concerne le chiffrage du mot de passe d'un utilisateur (donc relié à l'entité User). Nous utilisons l'algorithme "bcrypt".

```
security:
    encoders:
        AppBundle\Entity\User: bcrypt
```

- **providers** concerne la liste des utilisateurs. L'entité User constitue le lieu où sont stockés nos utilisateurs. La propriété username est utilisée pour procéder à l'authentification.

```
    providers:
        doctrine:
            entity:
                class: AppBundle:User
                property: username
```

- **firewalls**
  Cet élément est essentiel par rapport à la sécurité. La partie "dev" s'assure que les outils de développement (via les URLS /_profiler et /_wdt) ne soient pas bloqués par la sécurité.
  Toutes les autres URLs se trouvent dans le firewall "default". La clé "anonymous" ne nécessite aucune authentification (on peut retrouver cette session "Anonyme" dans le profiler.

Le firewall sert à configurer la façon dont les utilisateurs vont s'authentifier (formulaire de connexion? Authentification HTTP Basic? API token ?).

```yaml
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false

    main:
        anonymous: ~
        pattern: ^/
        form_login:
            login_path: login
            check_path: login_check
            always_use_default_target_path:  true
            default_target_path:  /
        logout: ~
```

- **role_hierarchy** (exemple: ROLE_ADMIN hérite de ROLE_USER).

```yaml
role_hierarchy:
    ROLE_ADMIN:       ROLE_USER
    ROLE_SUPER_ADMIN: [ROLE_USER, ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]
```

- **access_control** permet de restreindre l'accès à certaines URL de l'application selon le rôle de l'utilisateur.

```yaml
access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/users, roles: ROLE_ADMIN }
    - { path: ^/, roles: ROLE_USER }
```

2) src\AppBundle\Entity\User.php

C'est là où est stockée la liste de nos utilisateurs. L'authentification ne peut fonctionner qu'en implémentant l'interface UserInterface dont les méthodes sont:

- getRoles()
- getPassword()
- getSalt()
- getUsername()
- eraseCredentials()

Dans le fichier user.php, des contraintes peuvent être rajoutées (email unique, les champs ne peuvent être vides, vérification du format de l'adresse email, etc…).

3) src\AppBundle\Controller\SecurityController.php

3 routes sont présentes:

- /login : formulaire de connexion
- /login_check : vérification des identifiants et connexion.
- /logout : déconnexion

Gérer les accès

1) **Access_control** dans security.yml (voir page précédente).

```
    $this->addFlash( type: 'success',  message: 'La tâche a bien été supprimée.');
} elseif ($task->getUser() === null && $this->get('security.authorization_checker')->isGranted('ROLE_ADMIN')) {
    $em = $this->getDoctrine()->getManager();
    $em->remove($task);
```

2) Dans le code via le service **security.authorization_checker**.
   - Méthode denyAccessUnlessGranted()
   - Annotations grâce à SensioFrameworkBundle.

```
/**
 * @Route("/users/create", name="user_create")
 * @Security("has_role('ROLE_ADMIN')")
 */
public function createAction(Request $request)
{
    $user = new User();
```

3) Les templates Twig.

```
{% if is_granted('ROLE_ADMIN') %}
    <li><a href="{{ path('user_list') }}">Liste des utilisateurs</a></li>
    <li><a href="{{ path('user_create') }}" >Créer un utilisateur</a></li>
{% endif %}
```

[1]

[1] https://symfony.com/doc/3.1/security.html#security-securing-controller