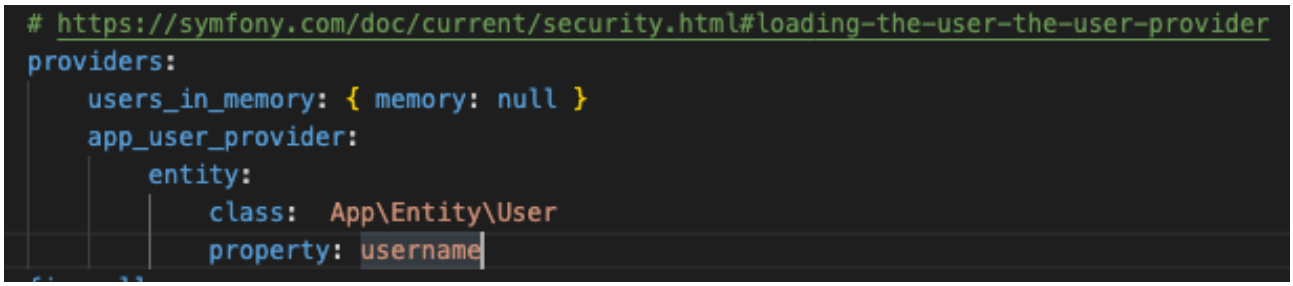
L’authentification sur l’application TODOLiST

1. Security.yaml

Un fichier important qui aide à la gestion de l’authentification dans Symfony c’est le fichier security.yaml. Vous trouverez le fichier en suivant le chemin suivant :

"config\packages\security.yaml"

Arretons nous en premier sur la parti providers. Symfony utilise ici Doctrine pour charger l’entité Utilisateur que nous lui avons désigné avec App\Entity\User et va utiliser la propriété username comme la capture d’écran ci-dessous nous le montre sous "property: username".
Si besoin ces paramètres sont modifiables, n’hésitez pas à vous référer à la documentation officielle de symfony qui est commentée dans la capture d’écran.



Veuillez aussi notez que dans password_hashers il est indiquez quelle interface de hachage vous utilisez dans l’entité de connection. Ici on utilise PasswordAuthentificatedUserInterface, si vous êtes amené à changer d’interface dans votre entité, n’oubliez pas de le modifier ici également!

Le pare-feu ou firewalls en anglais. le pare-feu définit quelles parties de votre application sont sécurisées et comment vos utilisateurs pourront s’authentifier.
En mode "dev" il s’agit juste d’un faux pare-feu qui s’assure que ne bloquez pas accidentellement les outils de développement.

Le pare-feu gère toute les URL et s’assure de l’authentification.Il permet d’utiliser plusieurs modes d’authentification!
Lorsqu’il est activé, il redirige les utilisateurs non identifiés vers login_path. Nous avons choisi "login" qui est le nom de la route que nous avons donné dans la class SecurityController. Elle se retrouve dans l’annotation de la fonction login:
  #[Route('/login', name:"login")] .Elle est aussi modifiable.
L’objectif de form_login est de redirigé vers le contrôleur définit dans "check_path" tant que les identifiants ne seront pas valide.

Logout comme sont nom l’indique désigne la route pour la déconnection.

```yaml
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        lazy: true
        provider: app_user_provider
        form_login:
            login_path: login
            check_path: login
            always_use_default_target_path:  true
            default_target_path:  /
        logout:
            path: logout
```

Un utilisateur peut donc maintenant ce connecter mais vous ne voulez pas pour autant qu'il est accès à toute les pages de l'application.
Access_control vous permet de définir suivant le rôle d'un utilisateur ses accès .
Ici on a défini que le chemin login est accessible à tous. Que les pages sous le chemin users n'est accessible qu'aux administrateurs et que le reste du site est accessible aux utilisateurs connectés.

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

Enfin dernière section importante de security.yaml c'est role_hierarchy :
Ici on indique que tout les administrateurs possèdent également le rôle utilisateur. Il est possible d'ajouter d'autres rôle comme commenté.

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

2. L'entité

Vous retrouverez l'entité pour la connexion avec le chemin suivant :

App\Entity\User

Vous constaterez la présence des interfaces: UserInterface et PasswordAuthenticatedUserInterface ainsi que la contrainte d'unicité sur l'email.

```php
#[ORM\Entity(repositoryClass: UserRepository::class)]
#[UniqueEntity("email", message: "Cet email est déjà utilisé")]
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
```

Les fonctions "getRoles" et "setRoles" sont utilent pour renvoyer ou modifier le rôle d'un User

```php
public function setRoles(array $roles): void
{
    $this->roles = $roles;
}

/**
 * @return array<string>
 */
public function getRoles(): array
{
    $roles = $this->roles;

    $roles[] = 'ROLE_USER';

    return array_unique($roles);
}
```

3. Le contrôleur

Vous trouverez le contrôleur pour la sécurité sous le chemin suivant :

"App\Controller\SecurityController"

La fonction login dont nous avons déjà parlé contient la route et utilise "AuthenticationUtils".

Vous devrez certainement faire des modifications dans ce contrôleur si vous souhaitez changer de la méthode de hachage ou pour d'autre modification.