Documentation technique

Security Authentification

1. About this documentation

This is a documentation technique of Authentication from Open Classrooms project 8 for Company TODo&Co

Author – Chun Cheung Duret

2. Folders of Authentication

type	Folder	description
Configration	Config/packages/security.taml	Configuration of authentification
Entity	src/Entity/User.php	User entity
Controller	Src/Controller/SecurityController.php	Controller of login login_check and logout
Authentication	Src/Security/LoginFormAuthenticator.php	Methods of authentifiction
Loginpage	Template/security/login.html.twig	Login form

3. Security installation

3.1. Install Security bundle , run command:

```
composer require security
```

after run this command, will install security.yaml in the application root path config\packages\security.yaml

3.2.1 The user (providers)

The user provider loads users from the database based on a “ UserIdentifier ”example: email or username.

```
provider
  providers:
    app_user_provider:
      entity:
        class: App\Entity\User
        property: username
```

3.2.2 The Fiewall & Authenticating User (Firewalls)

The firewall is the core of securing the application, the firewall is checked ervery request if it needs an authenticated user. The firewall also take care of authticating this user example(login form);

```
  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_check
        always_use_default_target_path: true
        default_target_path: /
```

3.2.3 Access Control(Authorization)(access_control)

Using access control and the authorization checker, you control the required permissions to perform a specific action or visit a specific URL

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

4. The User

4.1 Create a User class for authentication

if you already have “symfony/maker-bundl “run the command:

```
Symfony console make:user
```

This command will ask you name of user, typically, this will be User, of course you can give a name whatever you want to store in the database

```
PS C:\MAMP\htdocs\security> symfony console make:user

The name of the security user class (e.g. User) [User]:

 >User

Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:

 > yes

 Enter a property name that will be the unique "display" name for the user (e.g. email, username, uuid) [email]:

 > username

 Will this app need to hash/check user passwords? Choose No if passwords are not needed or will be checked/hashed by some other system (e.g. a single sign-on server).

 Does this app need to hash/check user passwords? (yes/no) [yes]:

 > yes
```

4.2 Entiy User

The new User entity created in src/Entity/User.php

```php
#[ORM\Table('user')]
#[ORM\Entity(repositoryClass: UserRepository::class)]
#[UniqueEntity('email')]
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    #[ORM\Column()]
    #[ORM\Id]
    #[ORM\GeneratedValue(strategy: 'AUTO')]
    private ?int $id = null;

    #[ORM\Column(length: 25, unique: true)]
    private ?string $username = null;

    #[ORM\Column(length: 64)]
    private ?string $password = null;

    #[ORM\Column(length: 60, unique: true)]
    private ?string $email = null;
```

4.3 Entity method

4.3.1 getUserIdentifier():

this User class implements UserInterface, and the getUserIdentifer() method will return username, that's because we set username be the unique "display" name for the user when we generate the User class

```
public function getUserIdentifier(): string
    {
        return $this->username;
    }
```

You are free to change the getUserIdenttifier() method return value e.g :

1. return $this->email;
2. config\package\security.yaml to change the user provider property: username to email

```
# https://symfony.com/doc/current/security.html#loading-the-user-the-user-
provider
  providers:
    app_user_provider:
      entity:
        class: App\Entity\User
        property: username
```

4.3.2 getRoles():

This method deals with permissions

```
public function getRoles(): array
    {
        $roles = $this->roles;
        // guarantee every user at least has ROLE_USER
        $roles[] = 'ROLE_USER';

        return array_unique($roles);
    }
```

5. Database setting

5.1 configuration connection of database

In root folder .env (.env.local) to set your owner database connection.

```
DATABASE_URL="mysql://{username}:{Password}@127.0.0.1:3306/{database}?serverVersion=8"
```

5.2 Create database

Following the command to create you database:

```
symfony console doctrine:database:create
```

5.3 Migrations

Following the command to make our migration

```
symfony console make:migration
```

5.4 Migrate

Following the command to make migrate to create user table in the database

```
symfony console doctrine:migratios:migrate
```

6. Registering the User

6.1 Hashing Passwords

Our application require a user to log in with a password.

6.1.1 PasswordAuthenticatedUserInterface

First make sure our Entity\User class implements the PasswordAuthenticatedUserInterface

```php
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    public function getPassword(): ?string
    {
        return $this->password;
    }
    public function setPassword(string $password): void
    {
        $this->password = $password;
    }
}
```

6.1.2 Configuration security.yaml

Second check our security.yaml password_hashers configuration, normally when we make:user should have done this.

```
 password_hashers:
    Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
'auto'
```

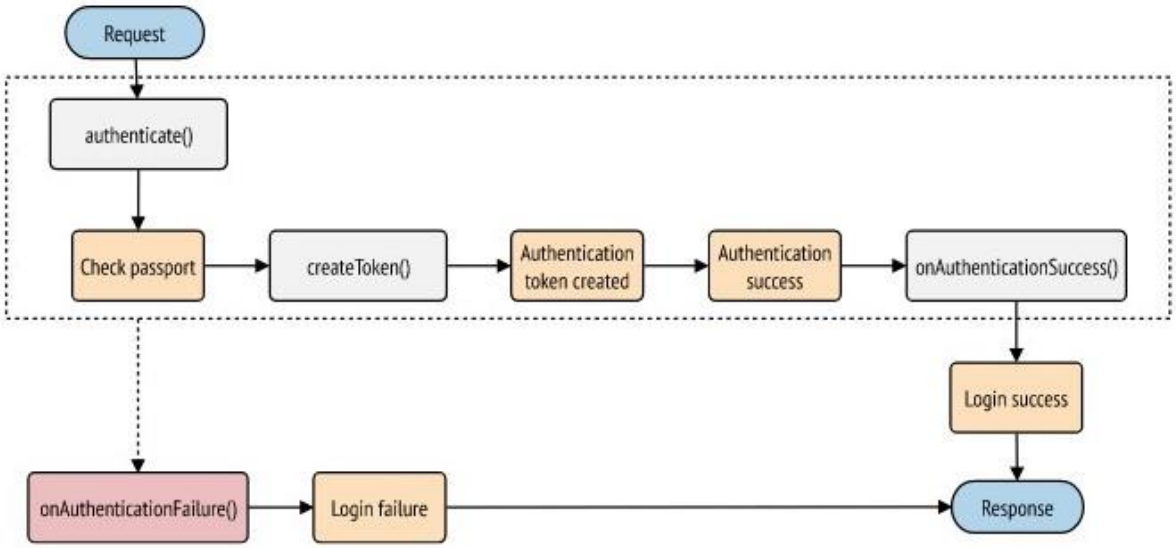
6.1.3 Implementation:

Now we can implement UserPasswordHasherInterface to hash the password before insert in to the database.

```
//src/Service/UserService

class UserService
{
    public function __construct(
        private readonly EntityManagerInterface $em,
        private readonly UserRepository $userRepository,
        private readonly UserPasswordHasherInterface $passwordHasher
    ) {
    }
    public function creatOneUserService(User $user)
    {
        $password = $this->passwordHasher->hashPassword($user, $user->getPassword());
        $user->setPassword($password);
        $this->em->persist($user);
        $this->em->flush();
     }
}
```

7. Authenticating Users



7.1 Authenticaor

7.1.1 Generate authenticator

Use following command to create one authenticator:

```
symfony console make:auth

 What style of authentication do you want? [Empty authenticator]:

  [0] Empty authenticator

  [1] Login form authenticator

> 1

The class name of the authenticator to create (e.g. AppCustomAuthenticator):

> SecurityAuthenticator

Choose a name for the controller class (e.g. SecurityController) [SecurityController]:

>

Do you want to generate a '/logout' URL? (yes/no) [yes]:

>

created: src/Security/SecurityAuthenticator.php

updated: config/packages/security.yaml

updated: src/Controller/SecurityController.php

created: templates/security/login.html.twig
```

this command will create:

In Folder	New files	
src/controller/	SecurityController.php	
src/	Security/SecurityAuthenticaor.php	
template/security/	index.html.twig	
config/packages/	security.yaml	

7.2 SecurityController

By default path " /login" method and render called "index", it's free to change the name

```
// src\Controller\SecurityController

class SecurityController extends AbstractController
{
    #[Route(path: '/login', name: 'login')]
    public function login(AuthenticationUtils $authenticationUtils): Response
    {
        $error = $authenticationUtils->getLastAuthenticationError();
        $lastUsername = $authenticationUtils->getLastUsername();

        return $this->render('security/login.html.twig', ['last_username' => $lastUsername,
'error'         => $error]);
     }
}
```

7.2.2 configuration firwalls

now we have to firwalls in security.yaml file, we need define one firewall for our application , here I used 'custom_authenticator ' to replaced ' form_login ' firewall

```
  custom_authenticator: App\Security\SecurityAuthenticator
      logout:
        path: logout
```

7.3 What Authenticator do?

When user submit ' login ' button , then the authenticator will take charge rest work.

- Method authenticate -→ return Passport
- Method onAuthenticationSucces → return RedirectResponse
- Method getLoginUrl → return login path

7.4 Metthod authenticate

7.4.1 authenticate() :

1. who trying to login (username -> identifier)
2. Login proof (password)
3. Store identifier to session
4. Create passport(username, password)

```
8     public function authenticate(Request $request): Passport
9     {
10        $username = $request->request->get('username', '');
11        $request->getSession()->set(Security::LAST_USERNAME, $username);
12
13        return new Passport(
14            new UserBadge($username),
15            new PasswordCredentials($request->request->get('password', '')),
16            [
17                new CsrfTokenBadge('authenticate', $request->request->get('_csrf_token')),
18            ]
19        );
20    }
```

7.4.2 onAuthenticationSuccesse:

When the user pass the authenticate method if success, then onAuthenticationSuccess method will redirect user to ‘ homepage ’ or you can set the page you want the user to go.

```
public function onAuthenticationSuccess(Request $request, TokenInterface $token, string
$firewallName): ?Response
    {
        if ($targetPath = $this->getTargetPath($request->getSession(), $firewallName)) {
            return new RedirectResponse($targetPath);
        }

        // For example:
        return new RedirectResponse($this->urlGenerator->generate('homepage'));
        //throw new \Exception('TODO: provide a valid redirect inside '.__FILE__);
    }
```

8. Resume

8.1 Intall symfony Security

8.3 Database Configuration

8.4 Migrations

8.5 Password Hashing

8.6 Authenticating User