**symfony new --webapp nomProject** creare un project

**CREARE LOGIN/LOGOUT AUTOMATICAMENTE DA symfony**

**symfony composer req symfony/security-bundle**  installare supporto di Security nel project

**symfony console make:user** *Utilizzatore* crea Entity ‘Utilizzatore’ e lo associa ao controllo di login

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

> email

Vuoi psw hashata ? Y

**symfony console make:entity** *Utilizzatore* per aggiungere altre properties oltre emial, psw et ROLE (tabella; ha già ‘ROLE\_USER’

Nota: scrivi il nome delle property IN MINUSCOLO

NBBBBB MAI CANCELLARE LA TABELLA ‘ROLE’ , è utilizzata dalla Security (perché cerca ROLE\_USER)

**symfony console make:auth** crea Entity et Controller x la Security

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

[0] Empty authenticator

[1] Login form authenticator

A noi serve **1 -> Login**

The class name of the authenticator to create ->*Loggazione****Authenticator***

(convenzione: nomeCheVuoi+Authenticator), NB no **LoginAuthenticator**

Per non avere dopo un controller **LoginController**

-> è il nome della Entity ‘guardiana’ che controllerà nella DB email et psw

Chiamare il Controller *LoggazioneControlle*r (*SecurityController*)

Voui un loggaut? -> Y

*LoggazioneControlle*r 1) crea e appella html di Loggin

2) ha funz se Loggin è OK

3) ha funz dove ridirigersi dopo LogOut

Ci dice cosa fare dopo:

Next:

- Customize your new authenticator.

- Finish the redirect "TODO" in the

App\Security\LoggazioneAuthenticator : : onAuthenticationSuccess() method.

- Review & adapt the login template: templates/security/login.html.twig.

**Entrare in**  **.env**  e linea 30 dire di creare DB con Sql et in local + commentare (#) linea 31:

DATABASE\_URL="mysql://db\_user:db\_password@127.0.0.1:3306/db\_name?serverVersion=5.7&charset=utf8mb4"

*Diventa*

DATABASE\_URL="mysql://**root**:@127.0.0.1:3306/*nomeDB*" ->loginNoAjax

**symfony console doctrine:database:create** -> crea DB vuota ‘loginnoajax’

**symfony console make:migration**

equivalente **symfony console m:mi**

**symfony console doctrine:migrations:migrate** ->crea tabelle (con eventuali relazioni)

equivalente **symfony console d:m:m**

nel nostro semplice caso ho solo ‘utilizzatore’



Vado a creare i miei fakers -> **fixtures** e poi se serve faker;

**symfony composer req --dev doctrine/doctrine-fixtures-bundle** ->folder’DataFixtures’

**symfony console make:fixture** *Utilizzatore****Fixtures***  -> convenzione nomEntity + **Fixtures**

<?php

namespace App\DataFixtures;

use App\Entity\Utilizzatore;

use Doctrine\Persistence\ObjectManager;

**use Doctrine\Bundle\FixturesBundle\Fixture;**

**use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;**

class UserFixtures extends Fixture

{

**private $passwordHasher;**

**public function \_\_construct(UserPasswordHasherInterface $passwordHasher)**

**{**

**$this->passwordHasher = $passwordHasher;**

**}**

public function **load**(ObjectManager $manager)

{

for ($i = 0; $i < 10 ; $i++){

$user = new User(); es:

$user->setEmail ("user".$i."@lala.com"); ->login: user2@lala.com

$user->setPassword($this->passwordHasher->hashPassword ($user,'pass'.$i)); ->psw: pass2

$manager->persist ($user);

}

$manager->flush();

}

}

private $passwordHasher;

public function \_\_construct(UserPasswordHasherInterface $passwordHasher)

{

$this->passwordHasher = $passwordHasher;

}

use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;

$user->setPassword($this->passwordHasher->hashPassword( $user,"pass". $i));

**symfony console doctrine:fixtures:load** ->lanciare fixtures in DB

equivalente **symfony console d:f:l**

Si vous avez besoin à un moment donné d'obtenir le hash d'un password depuis la console, tapez :

**symfony console security:hash-password**

et puis tapez le password. Vous pouvez par après le copier-coller dans la table (colonne password)

Controlliamo che il Login funziona:

entrare nel giusto Project -> **cd nomeProjrct**  ( al bisogno cd.. per salire di un livello)

In un nuovo terminale, da lasciare sempre aperto, x non spengere il server:

**symfony serve** -> accende il server

**localhost:8000/login** -> path in *LoggazioneControlle*r=> il Controller della Security

NBB non cambiare la path, perché altre applicazioni fanno rif a questo

…………………………………………………

ERROR! -> nella View base.html.twig ho CSS et JS che entrano con le path, che non ho

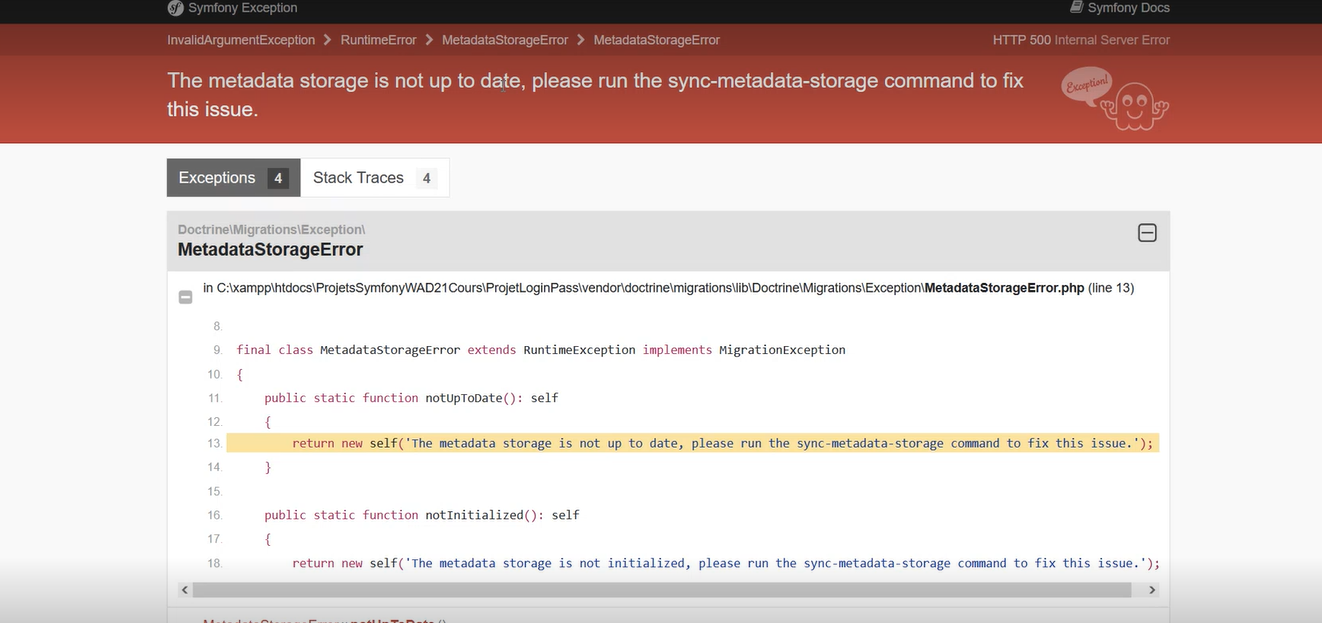
{{ encore\_entry\_script\_tags('app') }} et {{ encore\_entry\_link\_tags('app') }}

-> DEVO COMMENTARLE

**{#**{{ encore\_entry\_script\_tags('app') }}**#}** et **{#**{{ encore\_entry\_link\_tags('app') }}**#}**

……………………………………………

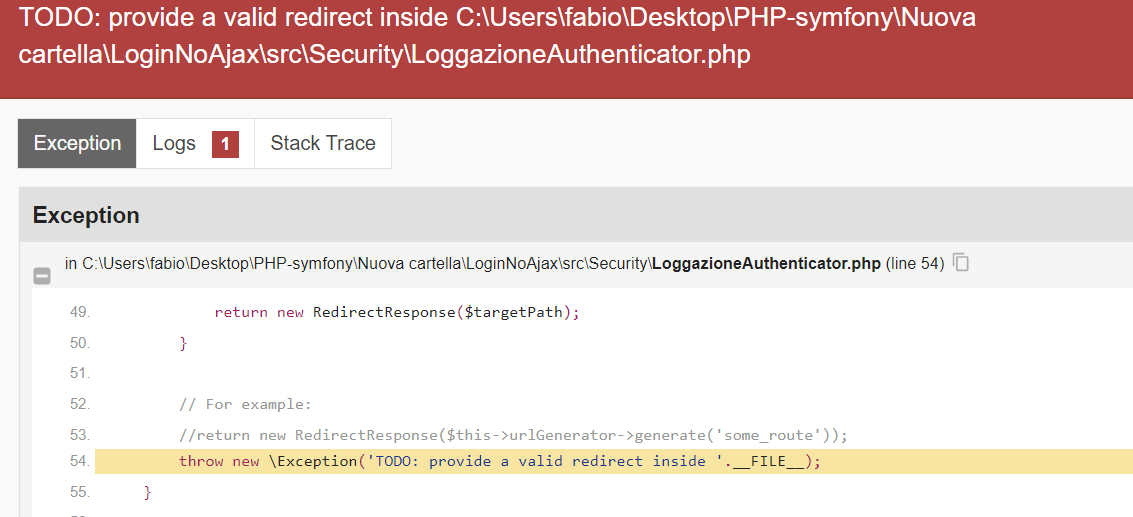
ERROR!



Basta togliere la Version il .**env** linea 30, se x caso è rimasta (serverVersion=5.7&charset=utf8mb4)

…………………………………………………….

EXCEPTION!



Non ho ancora detto dove andare se il login è a buon fine

-> Security -> *Loggazione****Authenticator.php***

Questo file.php è il Type del LoginForm (formulario di login), quindi una class, che ha:

* Method ‘authenticate’ -> controlla email (che abbiamo detto che è la UK) e la psw
* Method ‘onAuthenticationSuccess’ -> dove redirigere de loggin ok o se scrivo /login in url
* Method ‘getLoginUrl’
* Se serve creerò il method ‘onAuthenticationFailure’ per redirigere in caso loggin non ok

IN PRATICA:

1) scrivo nell’url **localhost:8000/login** ->Va nei Controller e cerca la Route ‘login’ -> è pre-fatta, andrà nel controller creato con *make:auth* : src -> Controller -> LoggazioneController

2) entr a nella function ‘login’ (auto-creata) : ‘$lastUsername’ (recupero dell’ultimo utente loggato) è ancora vuoto (e anche $error), quindi ->render la View ‘security/login.html.twig’

3) In View ‘security/login.html.twig’ non avro’ ne if $error e ne if $ lastUsername’(quindi no logOut ancora), ma affigge un semplice formulario di login con submit

4) se ok loggin -> lancia action ‘onAuthenticationSuccess’ di Entity creata con *make:auth* : src -> Security -> LoggazioneAuthenticator.php : commento throw e decommento return

return new **RedirectResponse**($this->urlGenerator->generate('some\_route'));

[identico a : return $this->**redirectToRoute** ('some\_route')]

->*'some\_route'* := **name** in Route DI UN ALTRO CONTROLLER che andrò a creare

Cambio some\_route in ‘*home’*

Se loggin non è ok -> Symfony cercherà in .. Authenticator.php l'action ‘onAuthenticationFailure’ (CHE NON ESISTE); non entrerà proprio in l'action ‘onAuthenticationSuccess’ (dove ridirigo a Homecontroller). Quindi ricarica la View ‘login.html.twig’ (perché legge sempre lo stesso url: *localhost:8000/login*), ma questa volta ho $error riempito e vedo “Credential invalid”.

5) creazione di un ALTRO controller per redirigere l’azione

**symfony console make:controller** *Home****Controller*** ->(convenzione: nomeCheVuoi+Controller)

la Route della func ‘index’ avrà anche un **name**: ‘*home’*

4’) Se loggin non è ok -> Symfony cercherà in ... Authenticator.php l'action ‘onAuthenticationFailure’, che posso creare CON QUESTO NOME (perché è un method dell’interfaccia)

**GESTIRE IL ROLE**

*Utilizzatore****Fixtures ->***

public function **load**(ObjectManager $manager)

{

for ($i = 0; $i < 10 ; $i++){

$user = new User(); es:

$user->setEmail ("user".$i."@lala.com"); ->login: user2@lala.com

$user->setPassword($this->passwordHasher->hashPassword ($user,'pass'.$i)); ->psw: pass2

$user->setRoles(['ROLE\_CLIENTE']);

$manager->**persist** ($user);

}

$user = new User(); es:

$user->setEmail ("user".10."@lala.com"); ->login: [user10@lala.com](mailto:user10@lala.com)

$user->setPassword($this->passwordHasher->hashPassword ($user,'pass10')); ->psw: pass10

$user->setRoles(['ROLE\_ADMIN’]);

$manager->**persist** ($user);

$manager->**flush**();

}

**symfony console doctrine:fixtures:load** ->per distruggere i dati in DB e reinserire i nuovi

Per donare l’accesso solo a certi ROLE ho 3 modi:

1. in security.yaml
2. in un controller
3. in una View

**a: in security.yaml**

creare il controller di ‘admin’ : *GestionController*, per esempio e poi 2 azioni:

#[Route("/gestion/action1")] ->sarà l’action x ROLE\_ADMIN -> vedi security.yaml

public function action1()

{

return $this->render('gestion/action1.html.twig');

}

#[Route("/gestion/action2")] ->sarà l’action x ROLE\_CLINTE -> vedi security.yaml

public function action2()

{

return $this->render('gestion/action2.html.twig');

}

In security.yaml : linea 37

access\_control:

# - { path: ^/admin, roles: ROLE\_ADMIN }

-> diventa -> { path: ^/ gestion/action1, roles: ROLE\_ADMIN }

# - { path: ^/profile, roles: ROLE\_USER }

-> diventa -> { path: ^/ gestion/action2, roles: ROLE\_CLIENTE }

**b: in controller**

#[Route("/autre/action1")]

public function action1()

{

**$this->denyAccessUnlessGranted("ROLE\_ADMIN");**  -> accesso solo se User ha ROLE\_ADMIN

return $this->render('autre/action1.html.twig');

}

#[Route("/autre/action2")]

public function action2()

{

$this->**denyAccessUnlessGranted**('ROLE\_CLIENTE);

// si pas d'exception...

return $this->render('autre/action2.html.twig');

}

Ovviamente se l’ user non ha quel ruolo, e cerco di entrare; si lancerà un errore (es: reparto amministrazione) -> ‘action1’ et ‘action2’ sono lanciate dal link in nav (prevedere anche link loguot)

**c: redirezione in View**

possiemo visualizzare o nascondere una <div> (o altro, es link) in base al ROLE utilizzando **is\_granted**

{% block body %}

{% if **is\_granted**('ROLE\_ADMIN') %}

<a href="{{ path ('effacer\_user') }}">effacer</a>

{% else %}

………

{% endif %}

{% endblock %}

**GESTIRE L’ errore "Access Denied"**

1) creare una class : es Security/*GestionnaireErreurAcces*.php

Questa class deve gestire 2 eventualità : o uno non è loggato o uno non ha il giusto ROLE 26.3

**CREARE UN FORMULARIO D’ISCRIZIONE AUTOMATICAMENTE DA symfony 24.2**

**GESTIRE IL LOGOUT**

In config -> packages -> **security.yaml** : linea 24

logout:

path: app\_logout

# where to redirect after logout

# target: app\_any\_route ->devo scrivere la Route di un’action

Diventa: logout:

path: app\_logout

target: dopo/logout

Nel controller, es HomeController, creo questa azione

*OPPURE* : target: '/logout' -> che già esiste in *LoggazioneController* (Controller di *make:auth*)

Nel controller LoggazioneController nell’action ‘logout’ commento Throw e

redirigo (-> return $this->redirectToRoute('/home'); -> // 'home' é il path di Route di un controller

**+** cancello void e scrivo Response

pulire la cache

symfony console cache:pool:clear cache.global\_clearer