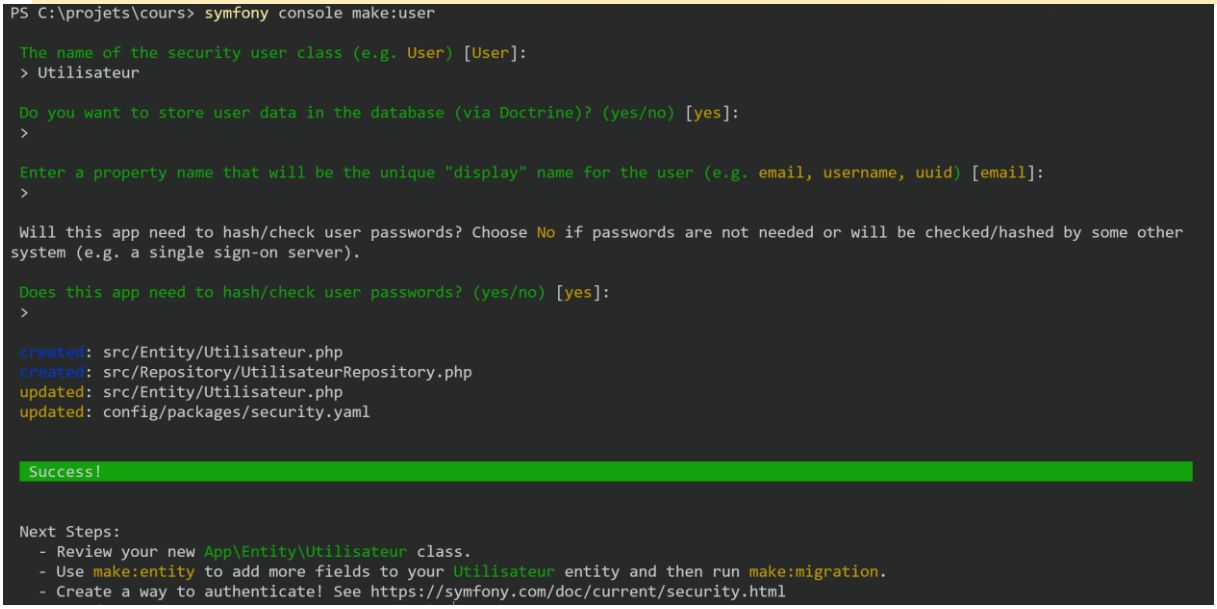
AUTHENTIFICATION DANS SYMFONY

1 CREATION DE L'ENTITE UTILISATEUR

Utilisez la console Symfony pour créer automatiquement la classe utilisateur :

```
symfony console make:user
```



Vérifiez que votre classe utilisateur a bien été créée puis ajouter les attributs que vous désirez :

2 FORMULAIRE D'INSCRIPTION

Générez un formulaire lié à la création de l'utilisateur.

```
symfony console make:form
```

Modifez le formulaire comme suit :

```
$builder
            ->add('nom', TextType::class, [
                'label' => "Votre nom",
                'attr' => [
                    'placeholder' => "Entrez votre nom"
                ]
            ])
            ->add('prenom', TextType::class, [
                'label' => "Votre prénom",
                'attr' => [
                    'placeholder' => "Entrez votre prénom"
                ]
            ])
            ->add('email', EmailType::class, [
                'label' => "Votre email",
                'attr' => [
                    'placeholder' => "Entrez votre email"
                ]
            ])
            ->add('password', RepeatedType::class, [
                'type' => PasswordType::class,
                'invalid_message' => "Le mot de passe et sa confirmation doivent être identiques",
                'label' => "Votre mot de passe",
                'required' => true,
                'first_options' => [
                    'label' => "Mot de passe",
                ],
                'second_options' => [
                    'label' => "Confirmez votre mot de passe",
                ],
            ])
            ->add('submit', SubmitType::class, [
                'label' => "S'inscrire",
            ])

        ;
```

3 GESTION DES MOTS DE PASSE

Vérifiez le fichier security.yaml

Il a été modifié automatiquement lorsque nous avons utilisé la console avec make:user

On voit apparaitre une catégorie encoders ce qui permet à Symfony d'encoder les mots de passes. L'indication auto permet d'indiquer à Symfony d'utiliser la meilleure méthode pour encoder les mots de passe

Dans le contrôleur il faut ajouter le code pour encoder les mots de passe :

On injecte tout d'abord dans la fonction gérant le formulaire l'interface UserPasswordEncoderInterface

```php
/**
     * @Route("/inscription", name="inscription")
     */
    public function inscription(Request $request, UserPasswordEncoderInterface $encoder): Response
    {
        $utilisateur = new Utilisateur;
        $form = $this->createForm(UtilisateurType::class, $utilisateur);
        $form->add("submit", SubmitType::class, array('label' => 'Créer'));

        $form->handleRequest($request);
        if($form->isSubmitted() && $form->isValid()){
            $password = $encoder->encodePassword($utilisateur,$utilisateur->getPassword());
            $utilisateur->setPassword($password);
            $this->em->persist($utilisateur);
            $this->em->flush();
            return $this->redirectToRoute("accueil",[]);
        }

        return $this->render('utilisateur/inscription.html.twig', [
            "form" => $form->createView()
        ]);
    }
```

4 FORMULAIRE DE CONNEXION

La commande symfony make:auth permet de créer un système d'authentification.

Les différentes questions nous permettent de réaliser étape par étape le nécessaire pour l'authentification.

```
symfony make:auth
```

```
PS C:\projets\cours> 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):
 > LoginFormAuthenticator

 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/LoginFormAuthenticator.php
 updated: config/packages/security.yaml
 created: src/Controller/SecurityController.php
 created: templates/security/login.html.twig

  Success!

 Next:
 - Customize your new authenticator.
 - Finish the redirect "TODO" in the App\Security\LoginFormAuthenticator::onAuthenticationSuccess() method.
 - Review & adapt the login template: templates/security/login.html.twig.
```

Modifiez le template login.html.twig pour correspondre à vos besoins.

Allez dans src/security/ et modifiez la fonction onAuthenticationSuccess pour une fois que l'utilisateur est connecté le renvoyer vers une apge de votre choix.

Ici vers la liste des produits :

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

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

Dans le fichier security.yaml modifiez la catégorie access_control pour contrôler certaines pages et ne les rendre accessible qu'aux utilisateurs.

```yaml
# Easy way to control access for large sections of your site
# Note: Only the *first* access control that matches will be used
access_control:
    # - { path: ^/admin, roles: ROLE_ADMIN }
    - { path: ^/produits, roles: ROLE_USER }
```

Assurez-vous d'être déconnecté et testez l'accès à la page de la liste des produits.

Connectez-vous et faites de même.