

Prise en main de Symfony 4.4

La sécurité



Objectifs :

Développer une petite application qui va gérer le déplacement de navires.

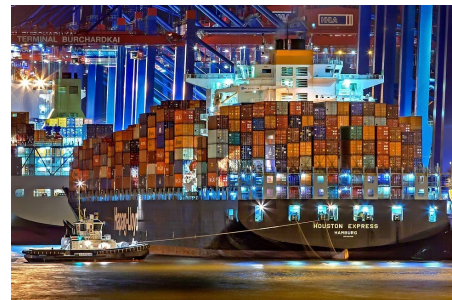
Versionner le projet

Créer des contrôleurs

Créer et valider des formulaires

Créer la base de données et les classes

Mettre en place la sécurité



Environnement :

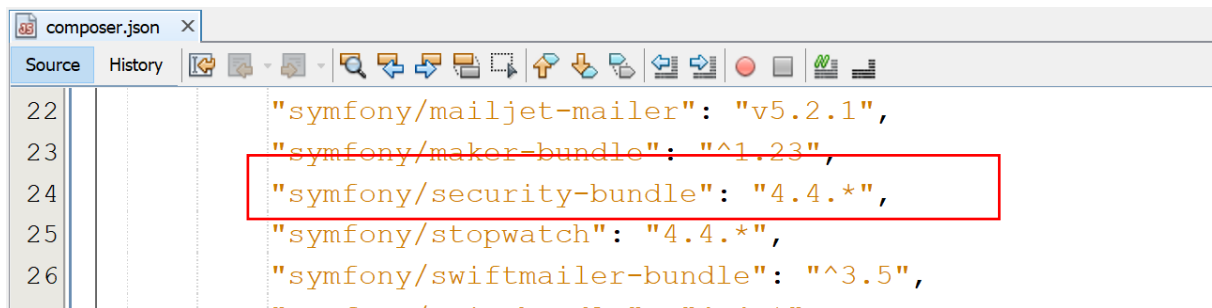
- ✓ Serveur apache : celui de wamp
- ✓ Base de données : mysql en local (wamp)
- ✓ IDE : netbeans
- ✓ virtualHost : navire.sio
- ✓ PHP 7.4.x
- ✓ Symfony 4.4
- ✓ Mysql 5.7.xx

Partie 1 : PREPARATION DU TD



Vous repartirez du TP Navire pour mettre en place la sécurité

Vérifier si le bundle security-bundle est installé (composer.json)



Vérifiez la version installée et éventuellement mettre à jour le bundle (composer.lock)

Dans un premier temps, vous allez :

- ✓ Créer la classe User et la persister en base de données :
- ✓ Mettre en place l'authentification qui se fera par l'adresse mail.

Partie 2 : LA CLASSE USER

Commande symfony : **make: user**

```
cd T:\Wampsites\CoursSymfony\DeploySymfony>php bin\console make:user
```

```
T:\Wampsites\CoursSymfony\DeploySymfony>php bin/console make:user

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

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

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

Will this app need to hash/check user passwords? Choose No if passwords are not needed or will be checked/hashed
gle sign-on server).

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

created: src/Entity/User.php
created: src/Repository/UserRepository.php
updated: src/Entity/User.php
updated: config/packages/security.yaml

Success!

Next Steps:
- Review your new App\Entity\User class.
- Use make:entity to add more fields to your User entity and then run make:migration.
- Create a way to authenticate! See https://symfony.com/doc/current/security.html

T:\Wampsites\CoursSymfony\DeploySymfony>
```



Vérifiez que le fichier security.yaml ait été modifié :

```
encoders:
    App\Entity\User:
        algorithm: auto

# https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers
providers:
    # used to reload user from session & other features (e.g. switch_user)
    app_user_provider:
        entity:
            class: App\Entity\User
            property: email

firewalls:
```

Effectuez les migrations dans la BD

```
T:\Wampsites\CoursSymfony\DeploySymfony>php bin/console make:migration

Success!

Next: Review the new migration "migrations/Version20220225152208.php"
Then: Run the migration with php bin/console doctrine:migrations:migrate
See https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html

T:\Wampsites\CoursSymfony\DeploySymfony>
```



Allez voir le fichier de migration créé :
Vous allez bien créer la table user

```
public function up(Schema $schema) : void
{
    // this up() migration is auto-generated, please modify it to your needs
    $this->addSql('CREATE TABLE user (id INT AUTO_INCREMENT NOT NULL, email VARCHAR(180) NOT NULL, roles JSON NOT NULL, password VARCHAR(255) N
}
```

Faites la migration :

```
T:\Wampsites\CoursSymfony\DeploySymfony>php bin/console doctrine:migrations:migrate

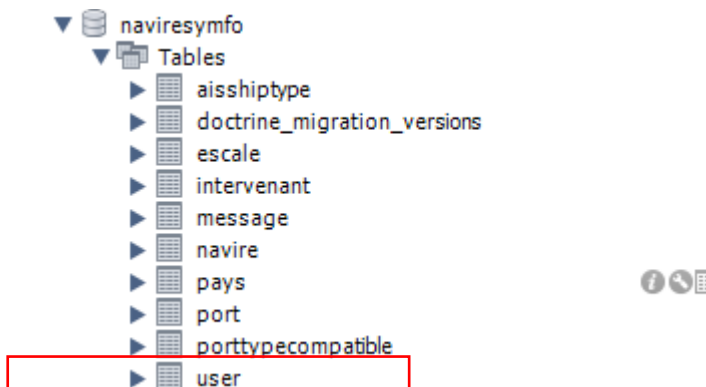
WARNING! You are about to execute a database migration that could result in schema changes and data loss. Are you sure you wish to continue? (yes/no) [yes]:
>

[notice] Migrating up to DoctrineMigrations\Version20220225152208
[notice] finished in 67.8ms, used 18M memory, 1 migrations executed, 1 sql queries

T:\Wampsites\CoursSymfony\DeploySymfony>
```



Allez voir dans la bd : la table a été créée



Créez-vous dans mysqlworkbench 2 utilisateurs :

id	email	roles	password
1	admin@navire.sio	[]	admin
2	oper@navire.sio	[]	oper
3	user@navire.sio	[]	user
NULL	NULL	NULL	NULL

Rôles : [] (2 crochets : array)



Mettre les mots de passe en clair on les chiffrera après.

Vous allez maintenant chiffrer les mots de passe : on va utiliser la commande

security:encode-password

```
T:\Wampsites\CoursSymfony\DeploySymfony>php bin/console security:encode-password

Symfony Password Encoder Utility
=====

Type in your password to be encoded:
>

-----
Key          Value
-----
Encoder used  Symfony\Component\Security\Core\Encoder\MigratingPasswordEncoder
Encoded password $argon2id$v=19$m=65536,t=4,p=1$aUVydzJT0FJZbUpRWUIwNw$dBXrCwYZRQ7RRB0jgBqeo6V8QjbRAQyoRMexn7FRSM
-----

! [NOTE] Self-salting encoder used: the encoder generated its own built-in salt.

[OK] Password encoding succeeded
```

Entrez ici le mot de passe à encoder (admin pour l'utilisateur admin@navire.sio exemple)

Mot de passe encodé à copier/coller dans la table user



Répétez l'opération pour les 2 autres utilisateurs de la base

Vous devriez obtenir ceci après les manipulations :

	id	email	roles	password
	1	admin@navire.sio		\$argi...
	2	oper@navire.sio		\$argi...
	3	user@navire.sio		\$argi...
	NULL	NULL	NULL	NULL

Partie 3 : L'AUTHENTIFICATEUR (GUARD AUTHENTICATOR)

Il s'agit maintenant de mettre en place tout le système qui va permettre de gérer l'authentification des utilisateurs.

La commande **make:auth** va créer :

- ✓ La classe d'authentification qui va se charger de gérer l'authentification. Nous l'appellerons dans notre projet ApplicationAuthenticator.
- ✓ Le contrôleur SecurityController qui va contenir la routes vers le formulaire de login
- ✓ Le template de login

```
T:\Wampsites\CoursSymfony\DeploySymfony>php bin/console make:auth
```

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

```
[0] Empty authenticator
```

```
[1] Login form authenticator
```

```
> 1
```

1 : l'authentification se fera par formulaire de login

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

```
> ApplicationAuthenticator
```

Nom de la classe qui va gérer l'authentification

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

```
>
```

Nom de la classe contrôleur qui va gérer les routes

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

```
>
```

On veut générer une route de logout

```
created: src/Security/ApplicationAuthenticator.php
```

```
updated: config/packages/security.yaml
```

```
created: src/Controller/SecurityController.php
```

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

```
Success!
```

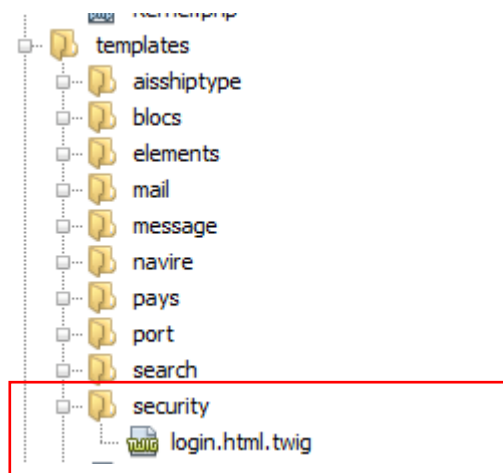
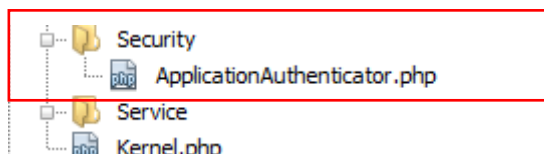
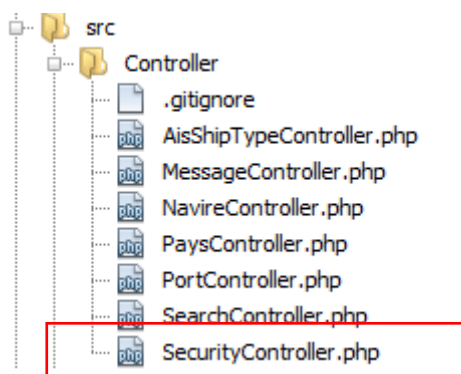
Next:

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

```
T:\Wampsites\CoursSymfony\DeploySymfony>
```



Regardons au niveau du projet ce qui a été créé :



La classe SecurityController avec les routes de login et de logout

```
class SecurityController extends AbstractController
{
    /**
     * @Route("/login", name="app_login")
     */
    public function login(AuthenticationUtils $authenticationUtils): Response
    {
        // ...12 lines ...
    }

    /**
     * @Route("/logout", name="app_logout")
     */
    public function logout()
    {
        // ...3 lines ...
    }
}
```

La classe ApplicationAuthenticator gérant l'authentification :

```
class ApplicationAuthenticator extends AbstractFormLoginAuthenticator implements PasswordAuthenticatedInterface
{
    use TargetPathTrait;

    public const LOGIN_ROUTE = 'app_login';

    private $entityManager;
    private $urlGenerator;
    private $csrfTokenManager;
    private $passwordEncoder;

    public function __construct(EntityManagerInterface $entityManager, UrlGeneratorInterface $urlGenerator, CsrfTokenManagerInterface
    {
        // ...6 lines ...
    }

    public function supports(Request $request)
    {
        // ...4 lines ...
    }

    public function getCredentials(Request $request)
    {
        // ...13 lines ...
    }

    public function getUser($credentials, UserProviderInterface $userProvider)
    {
        // ...15 lines ...
    }

    public function checkCredentials($credentials, UserInterface $user)
    {
        // ...3 lines ...
    }

    /** Used to upgrade (rehash) the user's password automatically over time ...3 lines */
    public function getPassword($credentials): ?string
    {
        // ...3 lines ...
    }

    public function onAuthenticationSuccess(Request $request, TokenInterface $token, $providerKey)
    {
        // ...8 lines ...
    }

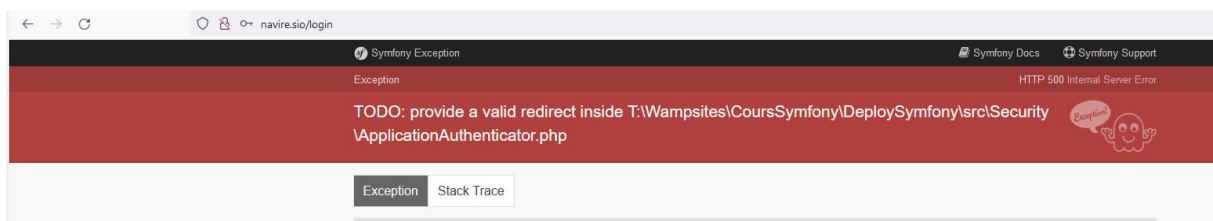
    protected function getLoginUrl()
    {
        // ...3 lines ...
    }
}
```

Vous pouvez d'ores et déjà afficher le formulaire de login grâce à la route

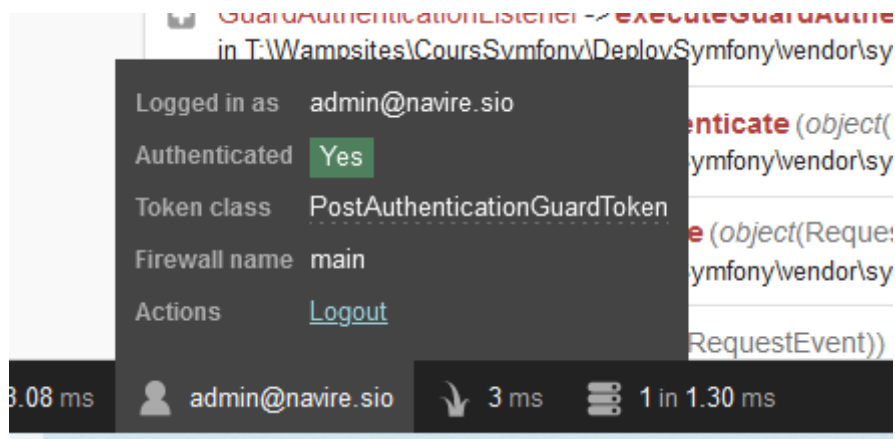
<http://navire.sio/login>



Si vous essayez de vous connecter, ça va marcher, mais la redirection ne se fera pas car vous n'avez pas précisé l'url de redirection en cas d'authentification réussie



Mais, vous êtes authentifié ... regardez la barre de debug:



C'est normal, vous n'avez pas suivi les instructions que la commande `make:auth` a affiché en fin d'exécution :

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

Vous allez donc modifier la méthode `onAuthenticationSuccess` de la classe `ApplicationAuthenticator` pour indiquer la route vers laquelle l'application se redirigera quand

l'authentification échouera. Nous choisissons de nous rediriger vers la page *home* :

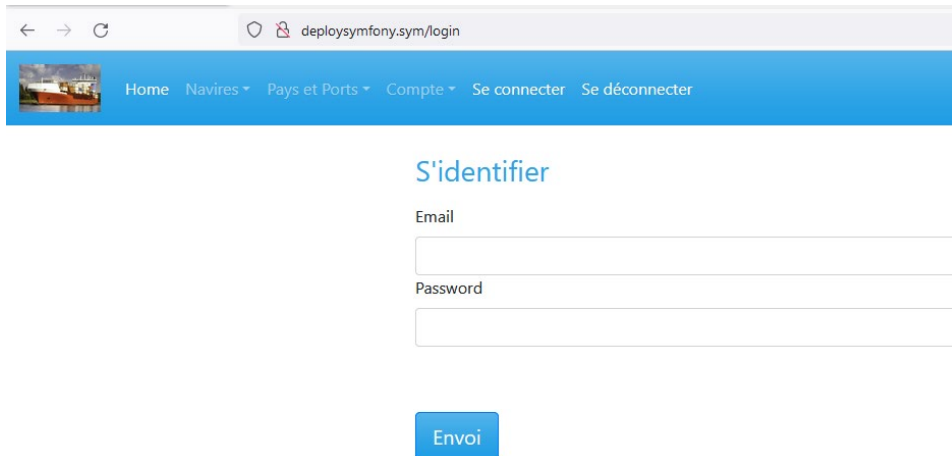
```
public function onAuthenticationSuccess(Request $request, TokenInterface $token, $providerKey)
{
    if ($targetPath = $this->getTargetPath($request->getSession(), $providerKey)) {
        return new RedirectResponse($targetPath);
    }

    // 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('home'));
}
```

Réessayez, vous devriez arriver sur la page d'accueil !

Vous pouvez aussi modifier votre page de login pour la rendre plus compatible avec votre charte graphique.

Par exemple :




... c'est bien mieux !!!



Vous remarquerez le commentaire dans votre page de login :

```
{#
    Uncomment this section and add a remember_me option below your firewall to activate remember me functionality.
    See https://symfony.com/doc/current/security/remember_me.html

    <div class="checkbox mb-3">
        <label>
            <input type="checkbox" name="_remember_me"> Remember me
        </label>
    </div>
#}
```

... on verra plus loin !!!

Il vous appartiendra de créer

- ✓ le formulaire (commande make:registration-form) d'inscription
- ✓ le formulaire de mot de passe oublié (make:reset-password)/



Seul l'administrateur créera les utilisateurs, donc il n'y a pas de raison d'envoyer un mail de confirmation.

Vous allez maintenant tester l'authentification des utilisateurs que vous avez créé en base de données : admin, oper et user.

Cliquez dans la barre de debug :

http://navire.sio/home

302 Redirect from : POST @app_login (3e1ae1)

Method: GET HTTP Status: 200 IP: ::1 Profiled on: Sat, 26 Feb 2022 04:39:05 +0000 Token: ede11d

Security Token

oper@navire.sio	✓
Username	Authenticated

Property	Value
Roles	["ROLE_USER"]
Inherited Roles	none
Token	Symfony\Component\Security\Guard\Token\PostAuthenticationGuardToken {#818 ▶}

Security Firewall

main	✓	✗	✓
Name	Security enabled	Stateless	Allows anonymous

Configuration

Key	Value
provider	security.user.provider.concrete.app_user_provider
context	main
entry_point	App\Security\ApplicationAuthenticator
user_checker	security.user_checker

Vous devriez voir ceci dans la section Security :

Vous obtenez de précieuses informations :

- ✓ Que vous êtes authentifié en tant que oper@navire.sio

Security Token

oper@navire.sio	✓
Username	Authenticated

- ✓ Que l'utilisateur authentifié (oper@navire.sic) possède le rôle ROLE_USER et que ce rôle n'hérite d'aucun autre.:

Roles	["ROLE_USER"]
Inherited Roles	none

- ✓ Que le firewall qui a intercepté l'URL de la page demandée est main et qu'il autorise les connexions anonymes :

Security Firewall			
main	✓	✗	✓
Name	Security enabled	Stateless	Allows anonymous



Voir le fichier security.yaml :

```
firewalls:
  dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security: false
  main:
    anonymous: lazy
    provider: app_user_provider
    guard:
      authenticators:
        - App\Security\ApplicationAuthenticator
```

- ✓ Que le provider est bien app_user_provider qui gère les utilisateurs en base de données grâce à l'entity APP\Entity\User et que l'identifiant est l'attribut email. Vous voyez aussi que c'est bien la classe ApplicationAuthenticator qui se charge de gérer l'authentification.

Configuration	
Key	Value
provider	security.user.provider.concrete.app_user_provider
context	main
entry_point	App\Security\ApplicationAuthenticator
user_checker	security.user_checker



Voir le fichier security.yaml :

```
providers:
    # used to reload user from session & other features
    app_user_provider:
        entity:
            class: App\Entity\User
            property: email
```



Travail à faire :

- ✓ Vous allez attribuer à l'utilisateur admin@navire.sio le rôle ROLE_ADMIN. Vous ferez ceci dans MySQLWorkbench :

	id	email	roles	password
▶	1	admin@navire.sio	["ROLE_ADMIN"]	\$argon2id\$v=19\$m=65536
	2	oper@navire.sio	[]	\$argon2id\$v=19\$m=65536
	3	user@navire.sio	[]	\$argon2id\$v=19\$m=65536

- ✓ Connectez vous en tant que  admin@navire.sio 52 m admin :

- ✓ Vérifiez le rôle :

Property	Value
Roles	["ROLE_ADMIN" "ROLE_USER"]

Les utilisateurs

Vous pourrez rajouter les noms et prénoms de l'utilisateur.

Ces champs ne peuvent pas être null.

Vous utiliserez la commande

make:entity



Vous n'oublierez pas de faire les migrations

Vous devriez obtenir ceci

Result Grid						
	id	email	roles	password	nom	prenom
	1	admin@navire.sio	["ROLE_ADMIN"]	\$argon2id\$v=19\$m=65536,...	nadmin	padmin
	2	oper@navire.sio	["ROLE_OPER"]	\$argon2id\$v=19\$m=65536,...	noper	popper
	3	user@navire.sio	["ROLE_USER"]	\$argon2id\$v=19\$m=65536,...	nuser	puser
*	NULL	NULL	NULL	NULL	NULL	NULL

Rajoutez un utilisateur avec le rôle admin :

Les rôles

Rappel : Les rôles définissent le niveau d'autorisation de chaque utilisateur.

Les rôles nécessaires à l'application :

- ✓ ROLE_ADMIN
- ✓ ROLE_OPER
- ✓ ROLE_USER

Sachant que :

- ✓ le rôle oper hérite du rôle user
- ✓ le rôle admin hérite du rôle oper

Vous attribuerez un des rôles suivants aux utilisateurs :

admin@navire.sio : ROLE_ADMIN

oper@navire.sio : ROLE_OPER

user@navire.sio : ROLE_USER

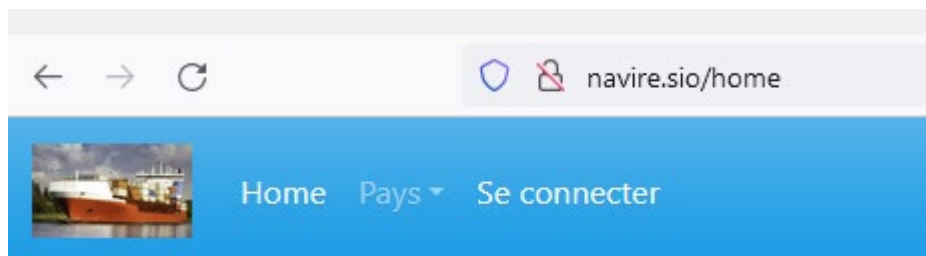
Un USER peut : visualiser l'ensemble de l'application. Il ne peut rien modifier

Une OPER peut modifier, créer, supprimer tout SAUF des utilisateurs

Un ADMIN peut créer et supprimer des utilisateurs (section Comptes de la navbar)

La nouvelle navbar en fonction de qui est connecté :

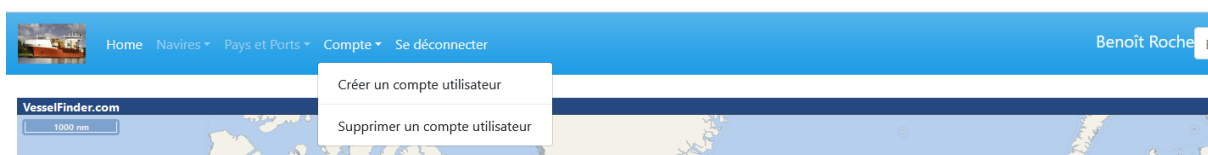
Anonyme :



Opérateur / User



Admin



Création de compte

Adresse mail

benoit.roche@gmail.com

Nom

Roche

Prenom

Benoît

Mot de passe

•••••

Répétez le mot de passe

•••••

Role

☒ ROLE_ADMIN

☐ ROLE_OPER

☐ ROLE_USER

☒ Agree terms

Créer l'utilisateur

S'identifier

Email

Password

Mot de passe oublié

Envoi

deploysymfony.sym/reset-password

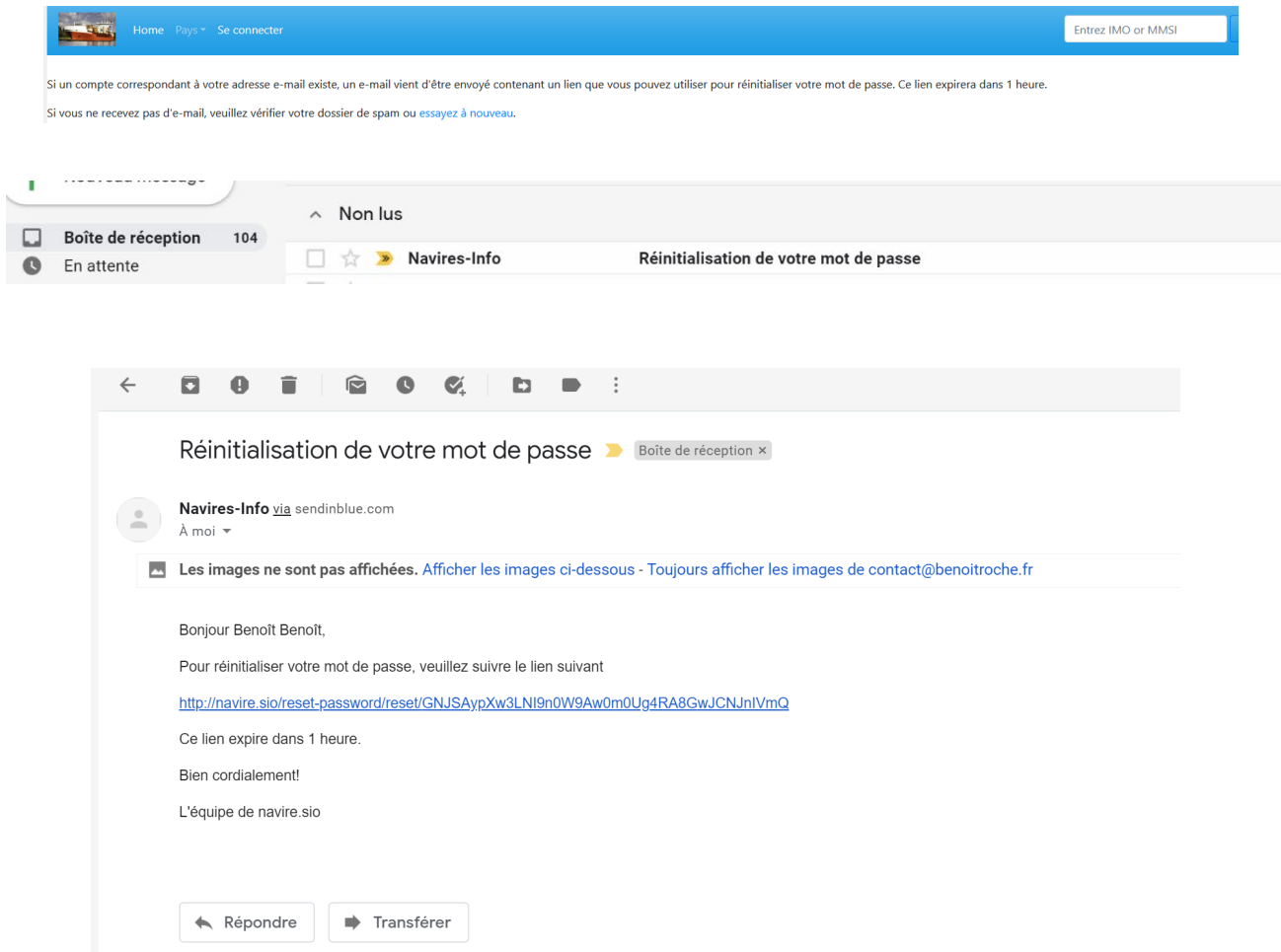
Pays et Ports • Compte • Se connecter Se déconnecter

Réinitialisation du mot de passe

Email

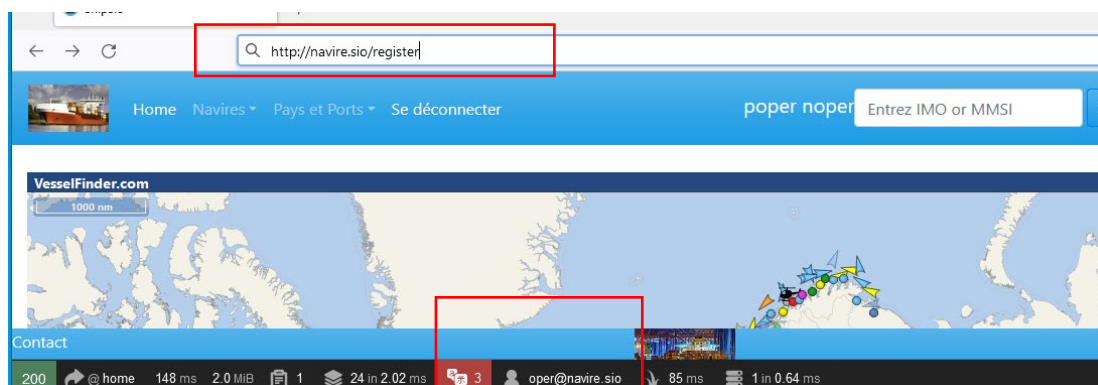
Entrez votre adresse mail et nous vous enverrons un lien pour la réinitialisation du mot de passe.

Envoyer le lien

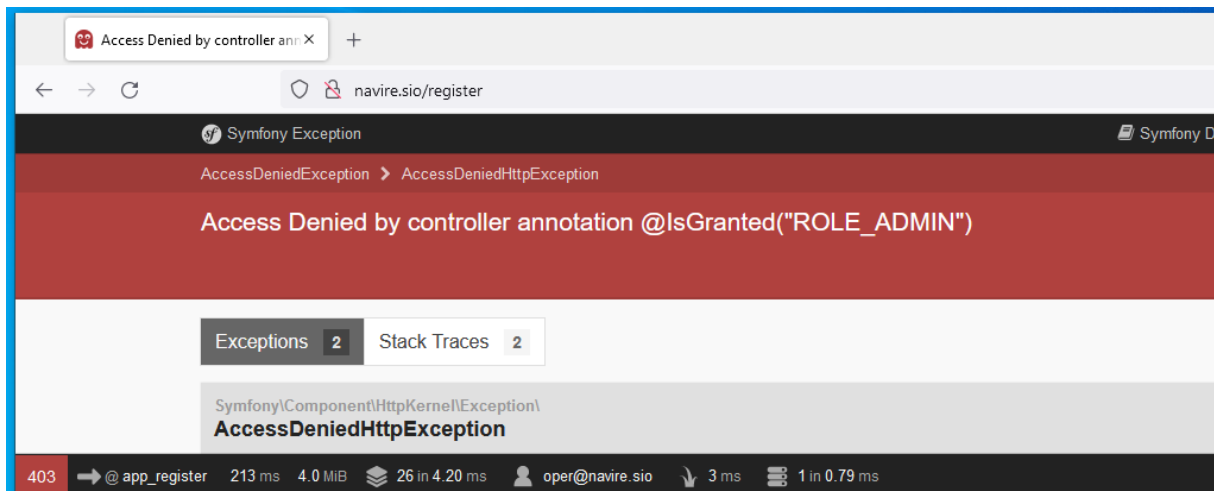


The screenshot shows the 'Réinitialisation du mot de passe' (Reset Password) form on the navire.sio website. The form has two input fields: 'Nouveau mot de passe' (New password) and 'Répétez le mot de passe' (Repeat the password). Below these fields is a blue button labeled 'Réinitialiser' (Reset).

On est connecté avec le rôle OPER, on essaie d'accéder à la page de création de comptes :



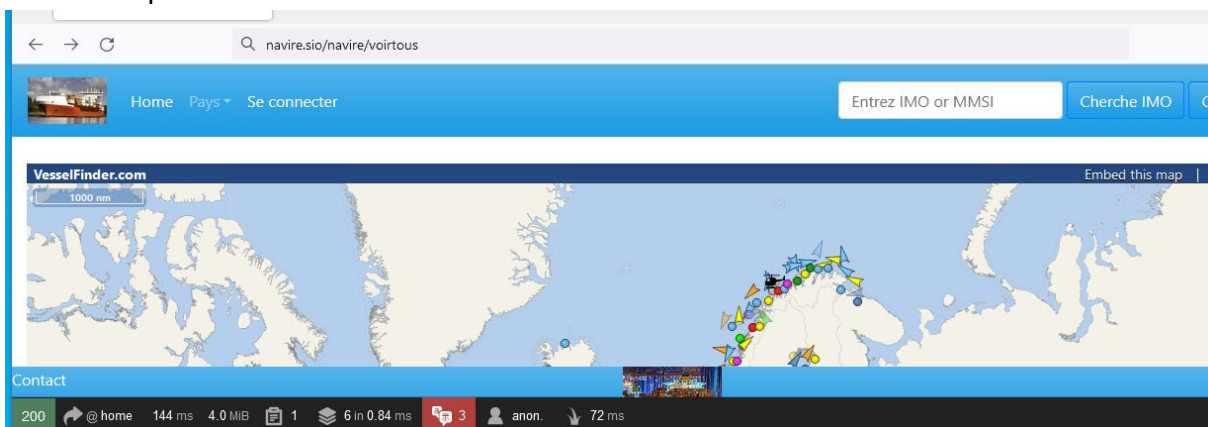
Et voilà ce qui arrive :



Pas très beau Mais

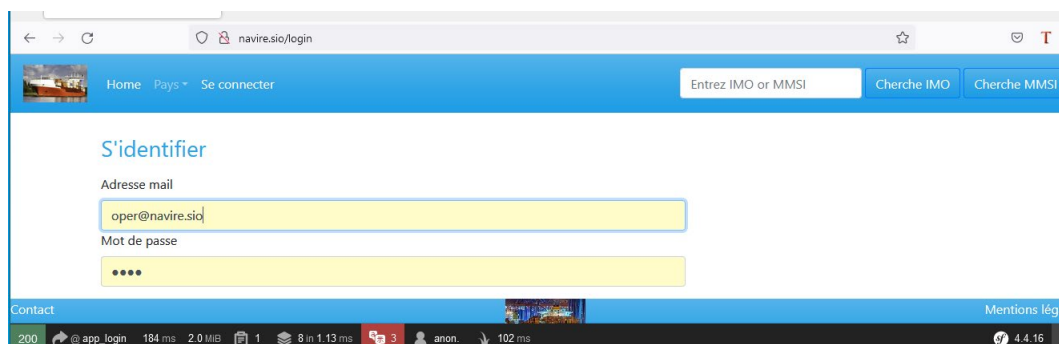
- ✓ On peut se personnalisé les pages d'erreur et donc la page 403
- ✓ On peut aussi prévenir de l'accès refusé par un message flash

On n'est pas connecté :

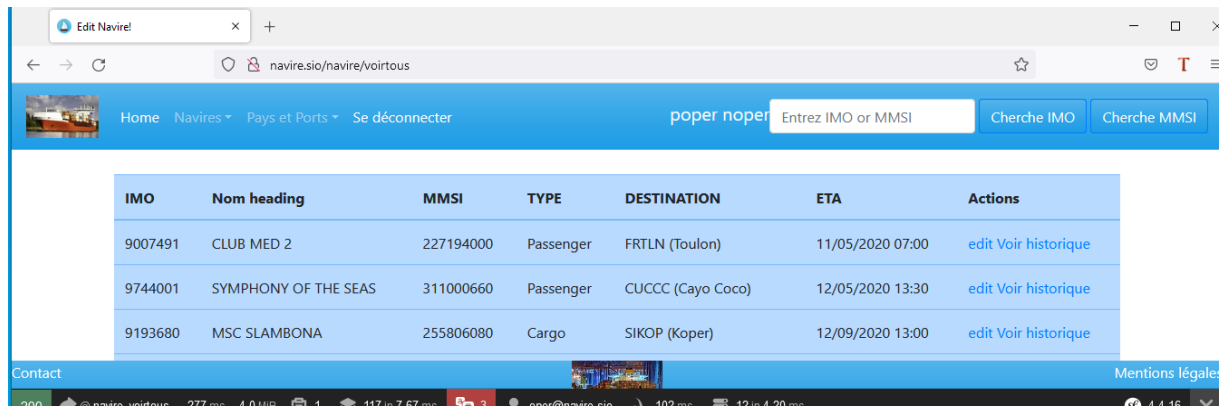


On essaie d'accéder à une page qui réclame des droits :

On est dirigé vers la page de login :



On se connecte et on est redirigé vers la ressource demandée si on en a les droits



The screenshot shows a web browser window with the URL `navire.sio/navire/voirtous`. The page has a blue header with navigation links: Home, Navires, Pays et Ports, and Se déconnecter. There is a search bar with the placeholder text "Entrez IMO or MMSI" and two buttons: "Cherche IMO" and "Cherche MMSI". Below the header is a table with the following data:

IMO	Nom heading	MMSI	TYPE	DESTINATION	ETA	Actions
9007491	CLUB MED 2	227194000	Passenger	FRTLN (Toulon)	11/05/2020 07:00	edit Voir historique
9744001	SYMPHONY OF THE SEAS	311000660	Passenger	CUCCC (Cayo Coco)	12/05/2020 13:30	edit Voir historique
9193680	MSC SLAMBONA	255806080	Cargo	SIKOP (Koper)	12/09/2020 13:00	edit Voir historique

At the bottom of the table, there is a "Contact" link and a "Mentions légales" link. The browser's status bar at the bottom shows various system icons and network information.



Il reste encore beaucoup de choses à voir, mais vous avez l'essentiel !!!

Symfony étant toujours inachevé, Il ne vous reste plus maintenant qu'à mettre tout ceci en musique !

... Dans vos prochains travaux prochain TD



Bravo pour votre travail,