Initialiser un projet Symfony 4

Logiciels requis

De manière obligatoire :

- PHP >= 7.1.3
- Composer

Il est impératif que la commande php pointe vers la bonne version de php ! Pour avoir votre version de php tapez `php -v` dans le terminal.

De manière facultative :

- Node.js Uniquement si le projet utilise webpack

Webpack sert à gérer les librairies et dépendances javascript de manière plus simple. C'est utile si le projet utilise beaucoup de javascript. (Appli front par exemple, mais pas seulement. Webpack est utilisé sur MyCA)

De plus il faudra un serveur MySQL sur la machine (Laragon fera l'affaire pour windows, MAMP pour Mac, et pour linux installer un serveur mysql est assez simple)

Initialisation du projet

Symfony et les dépendances de base

Créer un nouveau dossier qui va contenir le projet, et lancer la commande suivante à l'intérieur :

```
composer create-project symfony/website-skeleton . "4.4.*"
```

Installation de Webpack

A faire uniquement si le projet va utiliser Webpack !

Lancer la commande suivante pour installer **Webpack Encore** :

```
composer require symfony/webpack-encore-bundle
```

Pour les dépendances front :

```
npm install --save-dev @symfony/webpack-encore
```

Installation de FOSUserBundle

A ne pas installer pour les projets claqués, vous gérerez les utilisateurs comme dans la fabuleuse vidéo de Lior COMPRENDRE L'AUTHENTIFICATION !

A faire si le site nécessite une gestion d'utilisateurs. (99% des cas)

Le module utilisé FOSUserBundle est un module couramment utilisé avec symfony pour cette utilisation.

Modules nécessaires

```
composer require swiftmailer-bundle
composer require symfony/translation
composer require friendsofsymfony/user-bundle "~2.0"
```

Une erreur apparaît à la fin de l'installation de ce dernier module mais c'est normal.

Création de l'entité User

src/Entity/User.php

```php
<?php
// src/Entity/User.php

namespace App\Entity;

use FOS\UserBundle\Model\User as BaseUser;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity
 * @ORM\Table(name="fos_user")
 */
class User extends BaseUser
{
```

```
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    public function __construct()
    {
        parent::__construct();
        // your own logic
    }
}
```

Le code est ici très basique. On pourra rajouter des champs notamment avec la commande `php bin/console make:entity` en précisant l'entité User.

Configuration

Modifier le fichier security.yaml :

config/packages/security.yaml

```
security:
    encoders:
        FOS\UserBundle\Model\UserInterface: bcrypt

    role_hierarchy:
        ROLE_ADMIN:       ROLE_USER
        ROLE_SUPER_ADMIN: ROLE_ADMIN

    #
https://symfony.com/doc/current/security.html#where-do-users-come-from-
user-providers
    providers:
        fos_userbundle:
            id: fos_user.user_provider.username

    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            pattern: ^/
            form_login:
                provider: fos_userbundle
                csrf_token_generator: security.csrf.token_manager
```

```
            logout:       true
            anonymous:    true

    # 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: ^/login$, role: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/register, role: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/resetting, role: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/admin/, role: ROLE_ADMIN }
```

Ensuite il faut créer le fichier de configuration de FOSUser

config/packages/fos_user.yaml

```
fos_user:
    db_driver: orm # other valid values are 'mongodb' and 'couchdb'
    firewall_name: main
    user_class: App\Entity\User
    from_email:
        address: "change@me"
        sender_name: "change@me"
```

Et enfin ajouter un paramètre au fichier framework.yaml :

config/packages/framework.yaml

```
framework:
    templating:
        engines: ['twig', 'php']
```

Création du schéma

Après avoir éventuellement ajouté des champs à l'entité User, il faut créer ou mettre à jour la base de données :

```
php bin/console make:migration
php bin/console doctrine:migrations:migrate
```

Connexion via MyCentraleAssos

Pour rajouter une connexion via MyCA, il faut avoir un jeu de clés OAUTH_ID et OAUTH_SECRET.

Il faut un module pour gérer la communication via le protocol OAuth2 :

```
composer require adoy/oauth2
```

On va modifier le .env pour rajouter les attributs nécessaires :

.env

```
OAUTH_ID=
OAUTH_SECRET=
OAUTH_BASE=https://my.centrale-assos.fr
```

Ensuite le fichier services.yaml pour rajouter ces paramètres d'environnement en parameter de symfony :

config/services.yaml

```
parameters:
    locale: 'fr'
    oauth_id: '%env(OAUTH_ID)%'
    oauth_secret: '%env(OAUTH_SECRET)%'
    oauth_base: '%env(OAUTH_BASE)%'
```

Enfin il faut créer un controller qui va gérer la connexion !

Si vous utilisez FOSUserBundle :

src/Controller/UserController.php

```
<?php

namespace App\Controller;

use App\Entity\User;
use FOS\UserBundle\Model\UserManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\EventDispatcher\EventDispatcherInterface;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Session\SessionInterface;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;
use
Symfony\Component\Security\Core\Authentication\Token\Storage\TokenStora
geInterface;
use
Symfony\Component\Security\Core\Authentication\Token\UsernamePasswordTo
ken;
use Symfony\Component\Security\Http\Event\InteractiveLoginEvent;

class UserController extends AbstractController
```

```
{
    /**
     * @Route("/login", name="user_login")
     */
    public function index(Request $request, UserManagerInterface
$userManager, TokenStorageInterface $tokenStorage, SessionInterface
$session, EventDispatcherInterface $dispatcher)
    {
        $id = $this->getParameter('oauth_id');
        $secret = $this->getParameter('oauth_secret');
        $base = $this->getParameter('oauth_base');

        $client = new \OAuth2\Client($id, $secret);

        if(!$request->query->has('code')){
            $url =
$client->getAuthenticationUrl($base.'/oauth/v2/auth',
$this->generateUrl('user_login',
[],UrlGeneratorInterface::ABSOLUTE_URL));
            return $this->redirect($url);
        }else{
            $params = ['code' => $request->query->get('code'),
'redirect_uri' => $this->generateUrl('user_login',
[],UrlGeneratorInterface::ABSOLUTE_URL)];
            $resp = $client->getAccessToken($base.'/oauth/v2/token',
'authorization_code', $params);

            if(isset($resp['result']) &&
isset($resp['result']['access_token'])){
                $info = $resp['result'];

$client->setAccessTokenType(\OAuth2\Client::ACCESS_TOKEN_BEARER);
                $client->setAccessToken($info['access_token']);
                $response = $client->fetch($base.'/api/user/me');
                $data = $response['result'];

                $username = $data['username'];

                $user = $userManager->findUserByUsername($username);
                if($user === null){ // Création de l'utilisateur s'il
n'existe pas
                    $user = $userManager->createUser();
                    $user->setUsername($username);
                    $user->setPlainPassword(sha1(uniqid()));
                    $user->setEnabled(true);
                    $user->setEmail($data['email']);
                    $user->setNom($data['nom']);
                    $user->setPrenom($data['prenom']);


                    $userManager->updateUser($user);
```

```
                }

                // Connexion effective de l'utilisateur
                $token = new UsernamePasswordToken($user, null, 'main',
$user->getRoles());
                $tokenStorage->setToken($token);

                $session->set('_security_main', serialize($token));

                $event = new InteractiveLoginEvent($request, $token);
                $dispatcher->dispatch("security.interactive_login",
$event);

            }

            // Redirection vers l'accueil
            return $this->redirectToRoute('default');
        }

    }
}
```

Si vous n'utilisez pas FOS :

src/Controller/UserController.php

```
<?php

namespace App\Controller;

use App\Entity\User;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\EventDispatcher\EventDispatcherInterface;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Session\SessionInterface;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;
use
Symfony\Component\Security\Core\Authentication\Token\Storage\TokenStora
geInterface;
use
Symfony\Component\Security\Core\Authentication\Token\UsernamePasswordTo
ken;
use Symfony\Component\Security\Http\Event\InteractiveLoginEvent;
use Doctrine\Common\Persistence\ObjectManager;

class UserController extends AbstractController
{
    /**
     * @Route("/login", name="user_login")
```

```
     */
        public function index(Request $request, TokenStorageInterface
$tokenStorage, SessionInterface $session, EventDispatcherInterface
$dispatcher, ObjectManager $manager)
    {
        if($this->isGranted('ROLE_USER')){
            return $this->redirectToRoute('home');
        }


        $id = $this->getParameter('oauth_id');
        $secret = $this->getParameter('oauth_secret');
        $base = $this->getParameter('oauth_base');

        $client = new \OAuth2\Client($id, $secret);

        if(!$request->query->has('code')){
            $url =
$client->getAuthenticationUrl($base.'/oauth/v2/auth',
$this->generateUrl('user_login',
[],UrlGeneratorInterface::ABSOLUTE_URL));
            return $this->redirect($url);
        }else{
            $params = ['code' => $request->query->get('code'),
'redirect_uri' => $this->generateUrl('user_login',
[],UrlGeneratorInterface::ABSOLUTE_URL)];
            $resp = $client->getAccessToken($base.'/oauth/v2/token',
'authorization_code', $params);

            if(isset($resp['result']) &&
isset($resp['result']['access_token'])){
                $info = $resp['result'];

$client->setAccessTokenType(\OAuth2\Client::ACCESS_TOKEN_BEARER);
                $client->setAccessToken($info['access_token']);
                $response = $client->fetch($base.'/api/user/me');
                $data = $response['result'];

                $username = $data['username'];

                $user =
$this->getDoctrine()->getRepository(User::class)->findOneBy(['username'
=>$username]);
                if($user === null){ // Création de l'utilisateur s'il
n'existe pas
                    $user = new User;
                    $user->setUsername($username);
                    $user->setPassword(sha1(uniqid()));
                    $user->setEmail($data['email']);
                    $user->setLastName($data['nom']);
                    $user->setFirstName($data['prenom']);
```

```
                    $manager->persist($user);
                    $manager->flush();
                }

                // Connexion effective de l'utilisateur
                $token = new UsernamePasswordToken($user, null, 'main',
$user->getRoles());
                $tokenStorage->setToken($token);

                $session->set('_security_main', serialize($token));

                $event = new InteractiveLoginEvent($request, $token);
                $dispatcher->dispatch("security.interactive_login",
$event);

            }

            // Redirection vers l'accueil
            return $this->redirectToRoute('default');
        }

    }

}
```