

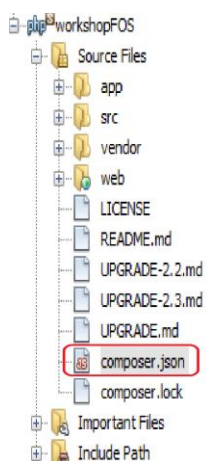
Workshop n°2 : FOS UserBundle

Objectif

Le but de ce workshop est l'installation de FOS UserBundle destiné à la gestion des utilisateurs d'une application web.

<http://symfony.com/doc/current/bundles/FOSUserBundle/index.html>

1. La première étape consiste à enlever le caractère « ; » les deux variables suivant : **hp_openssl.dll** et **extension=php_curl.dll** dans le fichier **php.ini** (**C:\wamp\bin\php\php5.4.16\php.ini**)
Openssl : une bibliothèque qui possède une dépendance à l'exécution, dans notre cas elle va autoriser le téléchargement de composer.
2. Créer un nouveau projet sous le répertoire : **C:\wamp\www\workshopFOS**.
3. Ajouter FOSUserBundle au fichier **composer.json**:
Composer : est une bibliothèque de gestion de dépendances pour PHP qui sert à télécharger les bundles externe



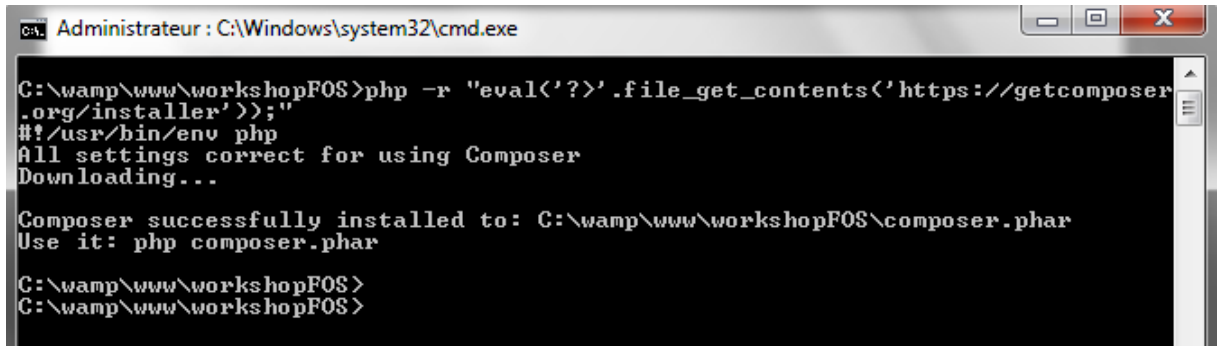
```
4  "type": "project",
5  "description": "The \"Symfony Standard Edition\" distribution",
6  "autoload": {
7      "psr-0": { "": "src/" }
8  },
9  "require": {
10     "php": ">=5.3.3",
11     "symfony/symfony": "2.3.*",
12     "doctrine/orm": ">=2.2.3,<2.4-dev",
13     "doctrine/doctrine-bundle": "1.2.*",
14     "twig/extensions": "1.0.*",
15     "symfony/assetic-bundle": "2.3.*",
16     "symfony/swiftmailer-bundle": "2.3.*",
17     "symfony/monolog-bundle": "2.3.*",
18     "sensio/distribution-bundle": "2.3.*",
19     "sensio/framework-extra-bundle": "2.3.*",
20     "sensio/generator-bundle": "2.3.*",
21     "incenteev/composer-parameter-handler": "~2.0",
22     "friendsofsymfony/user-bundle": "~2.0@dev"
23 }
```

"friendsofsymfony/user-bundle": "~2.0@dev"

Ensuite, il faut ouvrir une fenêtre de commande et se placer à la racine du projet
Et copier la commande suivante :

php -r "eval('?'>'.file_get_contents('https://getcomposer.org/installer'))';"

(vous aller avoir maintenant un fichier composer.phar sous le dossier)



```
Administrateur : C:\Windows\system32\cmd.exe

C:\wamp\www\workshopFOS>php -r "eval('?'>'.file_get_contents('https://getcomposer.org/installer'))';"
#!/usr/bin/env php
All settings correct for using Composer
Downloading...

Composer successfully installed to: C:\wamp\www\workshopFOS\composer.phar
Use it: php composer.phar

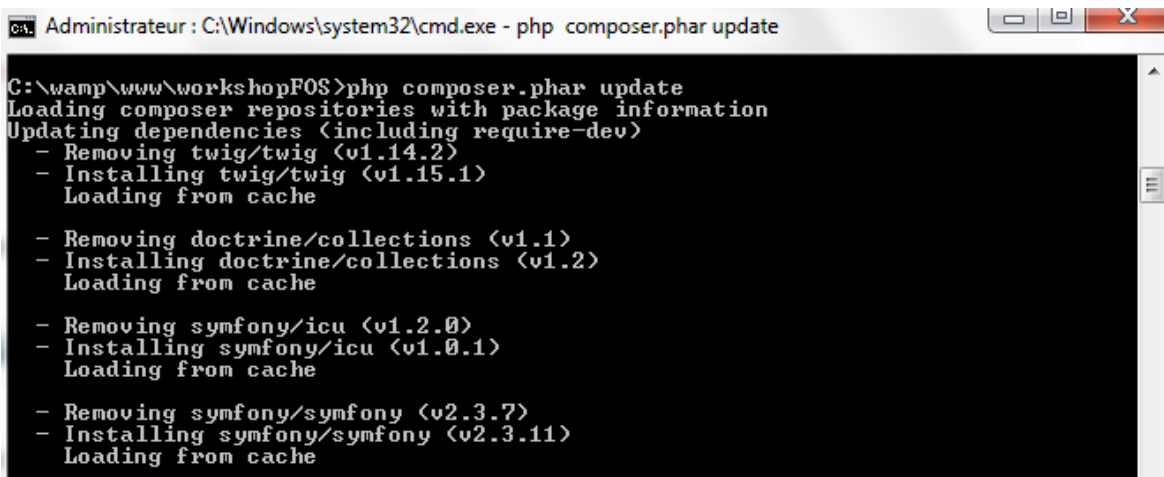
C:\wamp\www\workshopFOS>
C:\wamp\www\workshopFOS>
```

Si tout se passe bien la récupération du fichier devrait se lancer.

Une fois le fichier est récupéré, on lance le téléchargement des bundles depuis le terminal avec la commande suivante :

php composer.phar update

→Mettre à jours la bibliothèque composer et télécharger FOSUserBundle.



```
Administrateur : C:\Windows\system32\cmd.exe - php composer.phar update

C:\wamp\www\workshopFOS>php composer.phar update
Loading composer repositories with package information
Updating dependencies (including require-dev)
- Removing twig/twig (v1.14.2)
- Installing twig/twig (v1.15.1)
  Loading from cache

- Removing doctrine/collections (v1.1)
- Installing doctrine/collections (v1.2)
  Loading from cache

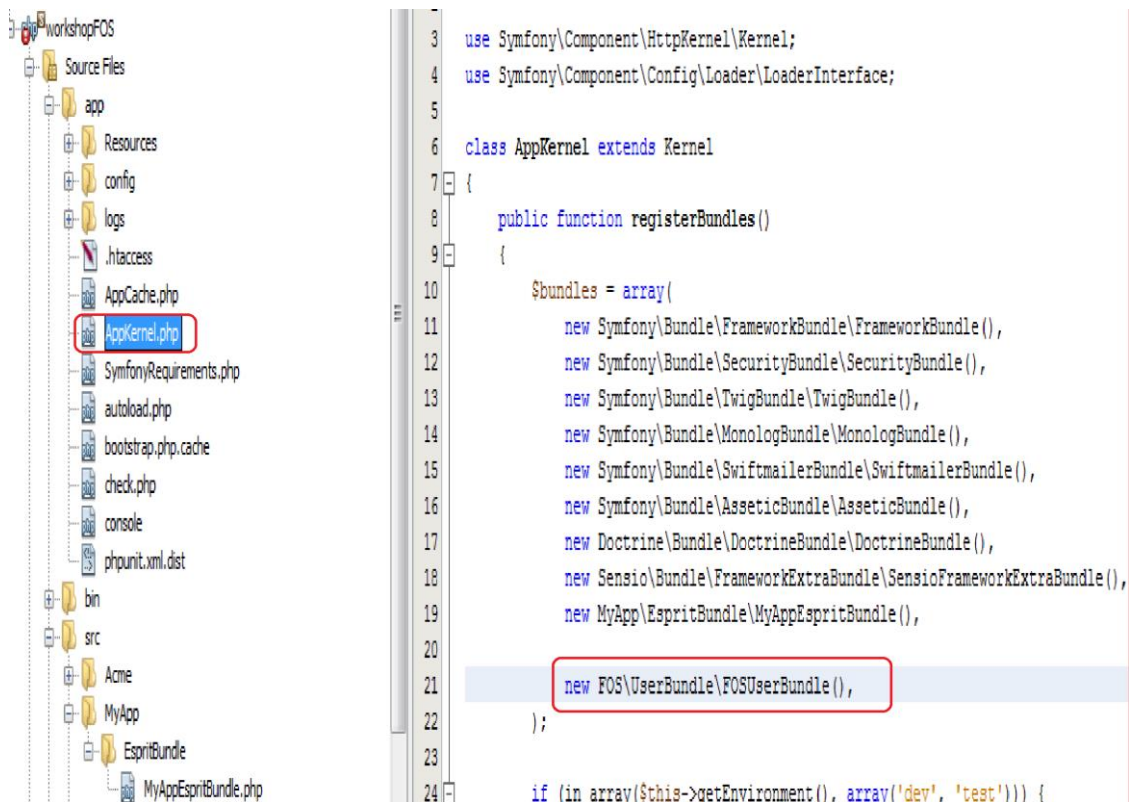
- Removing symfony/icu (v1.2.0)
- Installing symfony/icu (v1.0.1)
  Loading from cache

- Removing symfony/symfony (v2.3.7)
- Installing symfony/symfony (v2.3.11)
  Loading from cache
```

Une fois que le téléchargement est effectué avec succès, nous allons trouver dans le dossier vendor : notre bundle friendsofsymfony.

4. L'étape suivante consiste à créer un bundle : **MyApp\UserBundle**.

5. Activer le bundle dans le kernel : **new FOS\UserBundle\FOSUserBundle()**,



6. Une fois que notre bundle est prêt à être utiliser, nous créons un dossier **Entity** sous **UserBundle** puis **User.php** :

```
<?php

namespace MyApp\UserBundle\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
    }
}

?>
```

7. Afin d'assurer la sécurité de l'application, nous devons alors configurer notre

App\Config\security.yml

NB : Il faut remplacer le contenu du fichier security.yml par le fichier ci-dessous : (en respectant bien sur les exigences yaml).

```
security:
  encoders:
    FOS\UserBundle\Model\UserInterface: sha512

  role_hierarchy:
    ROLE_ADMIN:       ROLE_USER
    ROLE_SUPER_ADMIN: ROLE_ADMIN

  providers:
    fos_userbundle:
      id: fos_user.user_provider.username

  firewalls:
    main:
      pattern: ^/
      form_login:
        provider: fos_userbundle
        csrf_provider: form.csrf_provider
      logout:      true
      anonymous:    true

  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 }
```

Firewalls: système de sécurité du projet, détermine si un utilisateur doit ou ne doit pas être authentifié

Pour plus de détails voir ici :

<http://symfony.com/fr/doc/current/book/security.html>

8. La configuration de notre bundle se fait dans le fichier **App\Config.yml**.

```
fos_user:
  db_driver: orm # other valid values are 'mongodb', 'couchdb' and 'propel'
  firewall_name: main
  user_class: MyApp\UserBundle\Entity\User
```

9. La configuration du fichier service.yml :

```
services:
  fos_user.doctrine_registry:
    alias: doctrine
```

10. La configuration des routing se fait dans : **App\routing.yml**

Importer les fichiers de routing de FosUserBundle.

```
fos_user:
  resource: "@FOSUserBundle/Resources/config/routing/all.xml"
```

11. Configurer notre base de données :

Donner le nom de la base de données dans le fichier parameters.yml

```
Php app/console doctrine:database:create
php app/console doctrine:schema:create
```

12. Personnaliser les formulaires d'inscription et d'authentification en français.

Pour changer personnaliser l'interface d'authentification ainsi que l'interface de d'inscription.

a- Dans le fichier **parameters.yml** changer la langue en anglais :

```
parameters:
    database_driver: pdo_mysql
    database_host: 127.0.0.1
    database_port: null
    database_name: TestFosUser
    database_user: root
    database_password: null
    mailer_transport: smtp
    mailer_host: 127.0.0.1
    mailer_user: null
    mailer_password: null
    locale: fr
    secret: ThisTokenIsNotSoSecretChangeIt
```

b- Dans le fichier **config.yml**, enlever le caractère « # » la ligne de traduction.

framework:

#esi:

~

translator: { **fallback:** "%locale%" }

On va tester si notre bundle a été installé avec succès en tapant l'adresse :

localhost/TestFOS/web/app_dev.php/register : pour l'inscription

localhost/TestFOS/web/app_dev.php/login : pour l'authentification

13.Modification du formulaire d'inscription.

13.1 Si nous souhaitons changer le formulaire d'inscription, en ajoutant par exemple un champ pour le nom et un champ pour le prenom nous devons.

a- Modifier la classe User :

```
/**
 *
 * @ORM\Column(type="string", length=255)
 *
 */
protected $nom;

/**
 *
 * @ORM\Column(type="string", length=255)
 *
 */
protected $prenom;
```

NB : Ajouter les getters et les setters

b- Modifier le formulaire d'inscription (RegistrationFormType) sous Vendor\FOS\UserBundle\Form\Type

```
->add('nom')
->add('prenom')
```

13.2 Modifier le formulaire de l'inscription de telle sorte permettre l'utilisateur de choisir un rôle lors de l'inscription.

a- Créer un dossier Form dans votre Bundle puis créer la classe « StringToArrayTransformer » comme suit :

```
namespace MyApp\UserBundle\Form ;
use Symfony\Component\Form\DataTransformerInterface;
use Symfony\Component\Form\Exception\TransformationFailedException;

class StringToArrayTransformer implements DataTransformerInterface
```

```
{
    public function transform($array)
    {
        return $array[0];
    }

    public function reverseTransform($string)
    {
        return array($string);
    }
}
```

b- Modifier le formulaire d'inscription (RegistrationFormType) sous Vendor\FOS\UserBundle\Form\Type.

```
namespace FOS\UserBundle\Form\Type;

use MyApp\UserBundle\Form\StringToArrayTransformer;
use FOS\UserBundle\Util\LegacyFormHelper;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\OptionsResolver\OptionsResolverInterface;

class RegistrationFormType extends AbstractType
{
    private $class;

    public function __construct($class)
    {
        $this->class = $class;
    }

    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $transformer = new StringToArrayTransformer();
        $builder
            ->add('email', LegacyFormHelper::getType('Symfony\Component\Form\Extension\Core\
Type\EmailType'), array('label' => 'form.email', 'translation_domain' => 'FOSUserBundle'))
            ->add('username', null, array('label' => 'form.username', 'translation_domain'
=> 'FOSUserBundle'))
            ->add('plainPassword', LegacyFormHelper::getType('Symfony\Component\Form\Extens
ion\Core\Type\RepeatedType'), array(
                'type' => LegacyFormHelper::getType('Symfony\Component\Form\Extension\Core\
Type>PasswordType'),
                'options' => array('translation_domain' => 'FOSUserBundle'),
                'first_options' => array('label' => 'form.password'),
                'second_options' => array('label' => 'form.password_confirmation'),
                'invalid_message' => 'fos_user.password.mismatch',
            ))
            ->add('nom')
            ->add('prenom')
            ->add($builder->create('roles', 'choice', array(
                'label' => 'Rôle:',
                'mapped' => true,
                'expanded' => true,
                'multiple' => false,
                'choices' => array(
                    'ROLE_AGENT' => 'Agent',
                    'ROLE_CLIENT' => 'Client',
                )
            ))->addModelTransformer($transformer))
        ;
    }
}
```

```

}

public function configureOptions(OptionsResolver $resolver)
{
    $resolver->setDefaults(array(
        'data_class' => $this->class,
        'csrf_token_id' => 'registration',
        // BC for SF < 2.8
        'intention' => 'registration',
    ));
}

// BC for SF < 2.7
public function setDefaultOptions(OptionsResolverInterface $resolver)
{
    $this->configureOptions($resolver);
}

// BC for SF < 3.0
public function getName()
{
    return $this->getBlockPrefix();
}

public function getBlockPrefix()
{
    return 'fos_user_registration';
}
}

```

c- Modifier le fichier security.yml.

```

role_hierarchy:
    ROLE_CLIENT:      ROLE_USER
    ROLE_AGENT:       ROLE_USER
    ROLE_SUPER_ADMIN: ROLE_ADMIN

```

14.Les contrôles d'accès

Vous pouvez créer vos pages et les restreindre en modifiant le fichier **app/config/security.yml** en agissant sur la partie **access_control** (en supposant bien sûr que vous avez déjà créé une nouvelle route nommée Affichage)

```

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 }
    - { path: ^/affichage, role: ROLE_CLIENT }

```


15. Intégration d'un Template a FosUserBundle

a- Surcharger fosuserBundle

Ajouter la fonction getParent à la classe : EspritUserBundle

```
public function getParent()
{
    return 'FOSUserBundle';
}
```

b- Copiez vos dossiers **CSS, images et js** sous le répertoire **Web/**

b- Créez une nouvelle vue **layout.html.twig** sous le répertoire

Resources/views/ de votre Bundle

c- Copier le contenu de du page index de votre template dans le fichier layout.html.twig

d- Utiliser **asset** pour faire appel à nos images et au fichier css.

e- Réserver un block dans la page layout.html.twig pour les formulaires d'inscription et d'authentification en utilisant : **{% block formulaire %}**

f- Copier tous les dossiers qui se trouvent sous : vendor\friendsofsymfony\user-bundle\ Resources\views, dans votre nouveau Bundle (**UserBundle**) sous Resources\views.

g- On va tester l'intégration de template sur la page de login :

Dans le fichier login.html.twig, ajouter le nécessaire pour pouvoir hériter de la page layout.html.twig

```
{% extends "MyAppUserBundle::layout.html.twig" %}

{% trans_default_domain 'FOSUserBundle' %}

{% block formulaire %}
{% if error %}
    <div>{{ error|trans }}</div>
{% endif %}

<form action="{{ path('fos user security check') }}"
```

Donc vous allez avoir comme résultat en tapant l'adresse :

localhost/TestFOS/web/app_dev.php/login

16.Redirection après authentification.

a- Redirection simple :

Pour rediriger l'utilisateur après authentification il faut changer le fichier index.html.twig :

```
{% block formulaire %}
{% if error %}
    <div>{{ error.messageKey|trans(error.messageData, 'security') }}</div>
{% endif %}

<form action="{{ path('Affichage') }}" method="post">
    <input type="hidden" name="_csrf_token" value="{{ csrf_token }}" />

    <label for="username">{{ 'security.login.username'|trans }}</label>
    <input type="text" id="username" name="_username" value="{{ last_username }}" />

    <label for="password">{{ 'security.login.password'|trans }}</label>
```

Vous pouvez trouver d'autres cas ici !

symfony.com/doc/current/cookbook/security/form_login.html

a- Redirection Après authentification selon le rôle

Dans cette partie nous allons supposer que notre application contient deux types d'utilisateur : Agent et client. Chaque rôle doit renvoyer vers un espace réservé à chacun deux lors de la connexion.

Ci-dessous les étapes :

- Dans un premier temps nous allons créer le dossier suivant :
MyApp/UserBundle/Redirection/.
- Après Nous créons sous Redirection, le fichier AfterLoginRedirection.php qui sera appelé juste après l'authentification.

```
<?php

namespace MyApp\UserBundle\Redirection;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\RouterInterface;
use Symfony\Component\Security\Core\Authentication\Token\TokenInterface;
use Symfony\Component\Security\Http\Authentication\AuthenticationSuccessHandlerInterface;
;

class AfterLoginRedirection implements AuthenticationSuccessHandlerInterface
{
    /**
     * @var \Symfony\Component\Routing\RouterInterface
     */
    private $router;

    /**
     * @param RouterInterface $router
     */
    public function __construct(RouterInterface $router)
    {
        $this->router = $router;
    }

    /**
     * @param Request $request
     * @param TokenInterface $token
     * @return RedirectResponse
     */
    public function onAuthenticationSuccess(Request $request, TokenInterface $token)
    {
        // Get list of roles for current user
        $roles = $token->getRoles();
        // Tranform this list in array
        $rolesTab = array_map(function($role){
            return $role->getRole();
        }, $roles);
        // If is a admin or super admin we redirect to the backoffice area
        if (in_array('ROLE_CLIENT', $rolesTab, true) )
            $redirection = new RedirectResponse($this->router->generate('Client'));
        // otherwise, if is a commercial user we redirect to the crm area
        elseif (in_array('ROLE_AGENT', $rolesTab, true))
            $redirection = new RedirectResponse($this->router->generate('Agent'));
        // otherwise we redirect user to the member area
        else
            $redirection = new RedirectResponse($this->router->generate('Affichage'));

        return $redirection;
    }
}
```

- NB : Nous supposons que nous avons déjà les routes : 'Client' , 'Agent' et 'Affichage'.
- Une fois que nous avons le fichier : afterloginredirection, nous devons le transformer en services Symfony, ensuite l'utiliser dans la configuration du firewall de notre application.
- Pour cela, nous allons modifier notre fichier **services.yml** qui se trouve toujours dans notre bundle:

```
services:
  redirect.after.login:
    class: MyApp\UserBundle\Redirect\AfterLoginRedirection
    arguments: [@router]
```

- Nous injectons le service du **@router** pour pouvoir générer nos URL dans nos classes de redirections.
- Maintenant que notre service est configuré et fonctionnel, nous allons pouvoir modifier le firewall de notre application pour que Symfony utilise les bonnes redirections.
- Pour ce faire nous devons modifier notre fichier security.yml pour ajouter le success_handler comme ceci:

```
firewalls:
  main:
    pattern: ^/
    form_login:
      provider: fos_userbundle
      csrf_provider: form.csrf_provider
      success_handler: redirect.after.login
    logout: true
    anonymous: true
```