|  |  |
| --- | --- |
| **logo esprit.png** | Année Universitaire : 2016-2017  **Workshops Framework Symfony2**  **UP -Web** |
| **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>

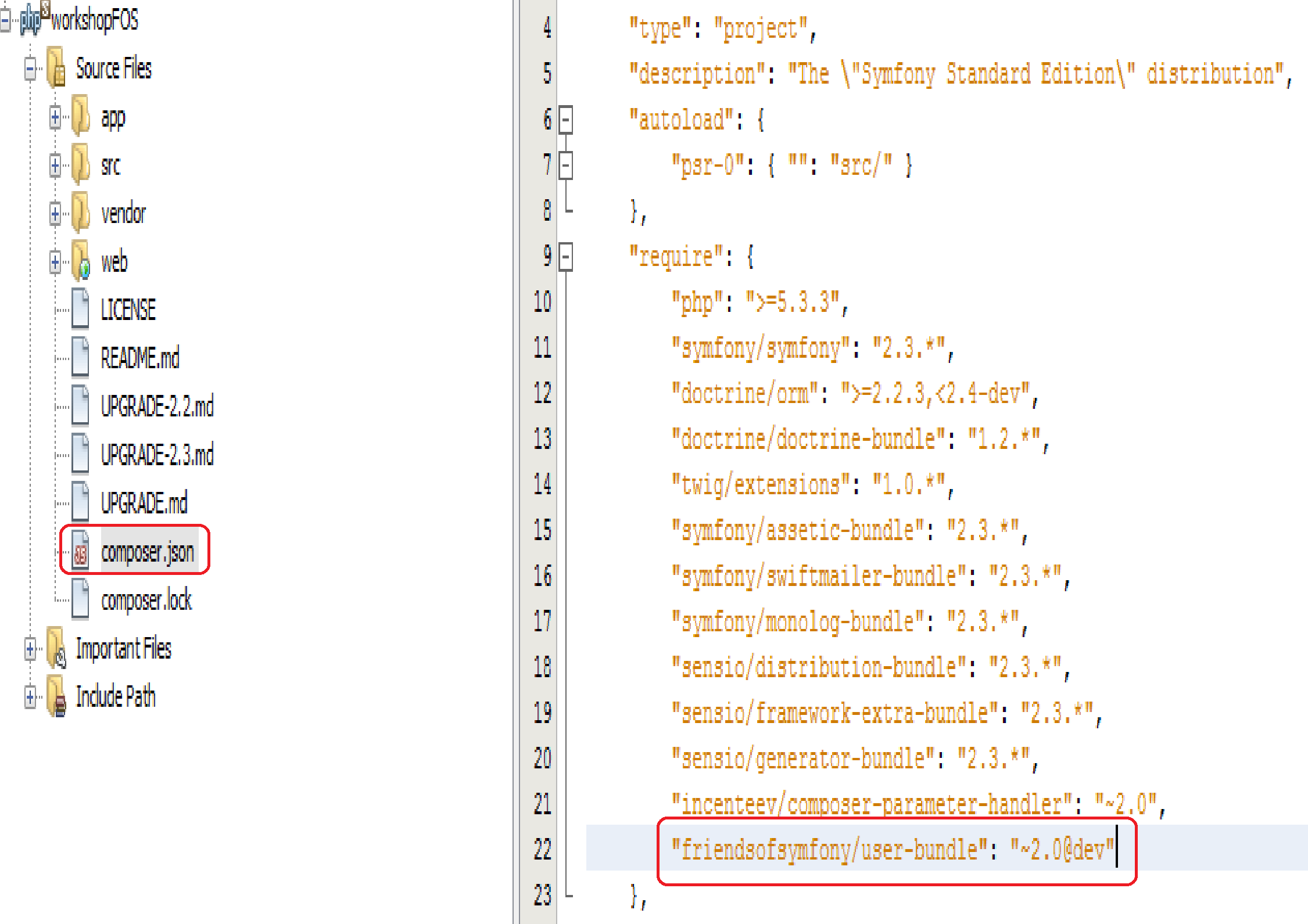
1. La première étape consiste à enlever le caractère « ; » les deux variables suivant :

**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.

1. Créer un nouveau projet sous le répertoire : **C:\wamp\www\workshopFOS.**
2. Ajouter FOSUserBundle au fichier **composer.json:**

**Composer :** est une bibliothèque de gestion de dépendances pour PHP qui sert à télécharger les bundles externes.



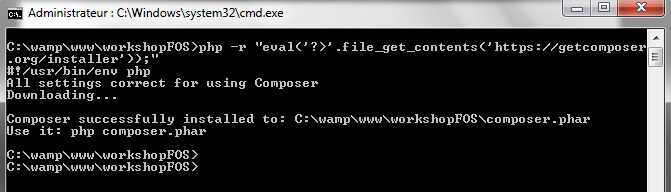
**"friendsofsymfony/user-bundle": "~2.0@dev"**

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 )**

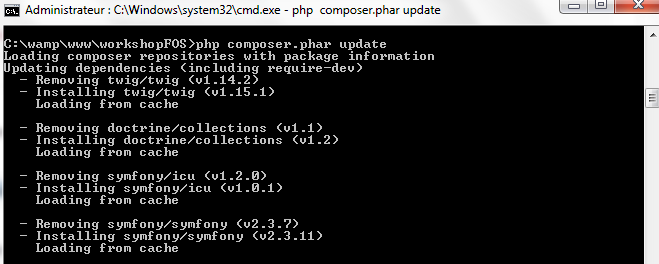
****

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 pour avoir la dernière version.

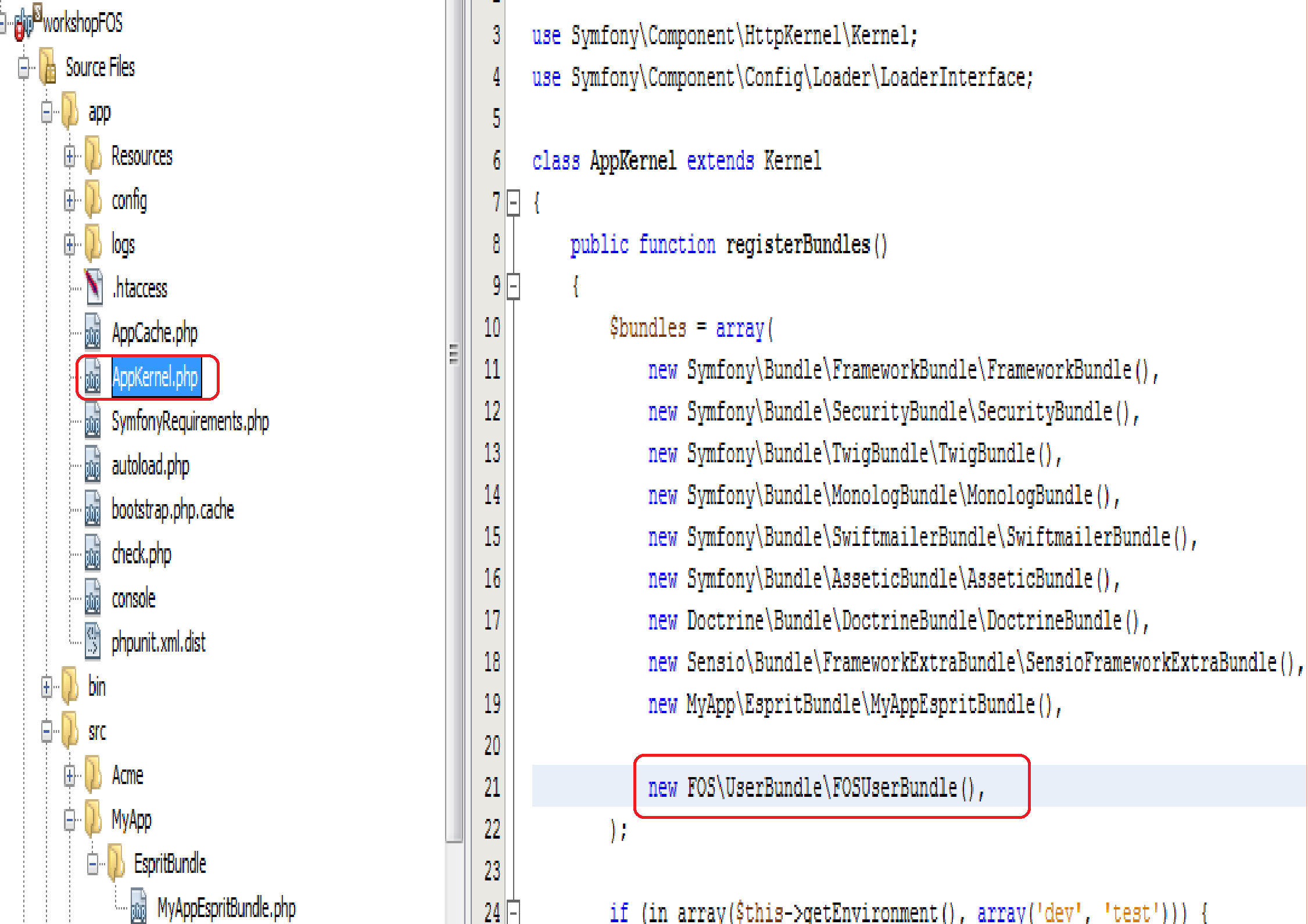


**php composer.phar update friendsofsymfony/user-bundle**

* Télécharger le bundle fosUserBundle

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

1. L’étape suivante consiste à créer un bundle : **MyApp\UserBundle**.
2. Activer le bundle dans le kernel : **new** FOS\UserBundle\FOSUserBundle(),



1. 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="utilisateur ")

\*/

class User extends BaseUser

{

/\*\*

\* @ORM\Id

\* @ORM\Column(type="integer")

\* @ORM\GeneratedValue(strategy="AUTO")

\*/

protected $id;

public function \_\_construct()

{

parent::\_\_construct();

// your own logic

}

}

?>

1. Afin d’assurer la sécurité de l’application, nous devons alors configurer notre **App\Config\security.yml**

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

*# app/config/security.yml*

security:

encoders:

FOS\UserBundle\Model\UserInterface: bcrypt

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\_token\_generator: security.csrf.token\_manager

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

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

|  |
| --- |
| fos\_user:  db\_driver: orm # other valid values are 'mongodb', 'couchdb' and 'propel'  firewall\_name: main  user\_class: MyApp\UserBundle\Entity\User |

1. La configuration du fichier service.yml : **App\Config**

|  |
| --- |
| services:  fos\_user.doctrine\_registry:  alias: doctrine |

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

Importer les fichiers de routing de FosUserBundle.

fos\_user:

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

1. Configurer notre base de données :
2. Donner le nom de la base de données dans le fichier parameters.yml
3. Exécuter les commandes suivantes afin de générer la base de données.

php app/console doctrine : database : create

php app/console doctrine:schema :create

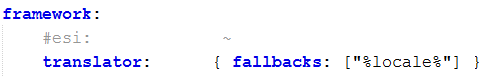
1. Personnaliser les formulaires d’inscription et d’authentification en français.

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

1. Dans le fichier **config.yml** changer la langue en anglais :

|  |
| --- |
|  |

1. Dans le fichier **config.yml**, enlever le caractère « # »de la ligne suivante.

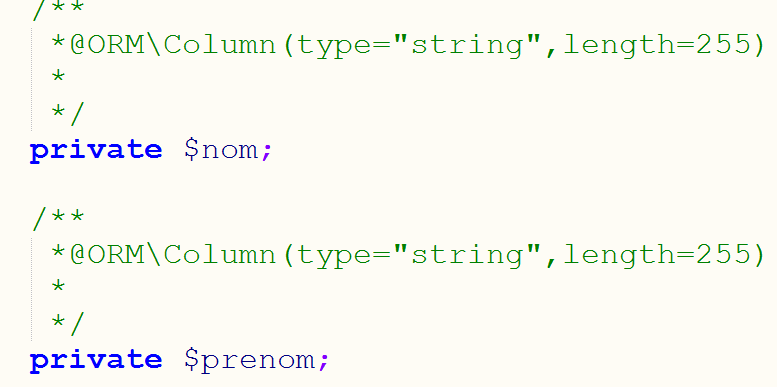


On va tester si notre bundle a été installé avec succès en tapant l’adresse :

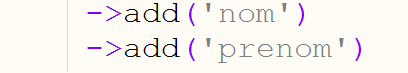
localhost/TestFOS/web/app\_dev.php/register : pour l’inscription

localhost/TestFOS/web/app\_dev.php/login : pour l’authentifcation

1. Modification du formulaire d’inscription.
   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.
2. Modifier la classe User :



1. Modifier le formulaire d’inscription (RegistrationFormType) sous Vendor\FOS\UserBundle\Form\Type



* 1. Modifier le formulaire de l’inscription de telle sorte permettre l’utilisateur de choisir un rôle lors de l’inscription.

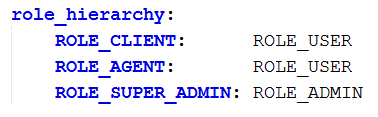
1. Modifier le formulaire d’inscription (RegistrationFormType) sous Vendor\FOS\UserBundle\Form\Type.

|  |
| --- |
| ->add(**'roles'**, ChoiceType::***class***, **array**(**'label'** => **'Type '**,  **'choices'** => **array**(**' AGENT'** => **'ROLE\_AGENT'**,  **'CLIENT'** => **'ROLE\_CLIENT'**),  **'required'** => **true**, **'multiple'** => **true**,)) |

NB : n’oublier pas d’ajouter

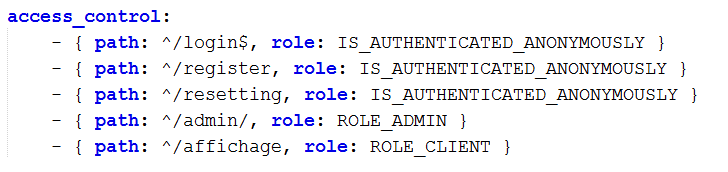
use Symfony\Component\Form\Extension\Core\Type\ChoiceType;

1. Modifier le fichier security.yml.



1. 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)



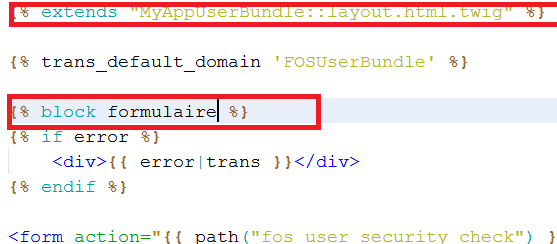
1. Intégration d’un Template a FosUserBundle
2. Surcharger fosuserBundle

Ajouter la fonction getParent à la classe : EspritUserBundle

|  |
| --- |
| **public function getParent()**  **{**  **return 'FOSUserBundle';**  **}** |

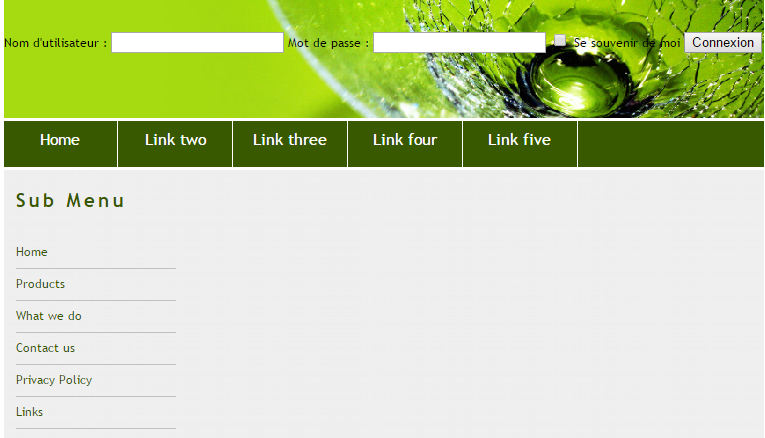
1. Copiez vos dossiers **CSS**, **images** et **js** sous le répertoire **Web/**
2. Créez une nouvelle vue **layout.html.twig** sous le répertoire **Resources/views/** de votre Bundle
3. Copier le contenu de du page index de votre template dans le fichier layout.html.twig
4. Utiliser **asset** pour faire appel à nos images et au fichier css.
5. Réserver un block dans la page layout.html.twig pour les formulaires d’inscription et d’authentification en utilisant : **{% block formulaire %}**
6. Copier tous les dossiers qui se trouvent sous : vendor\ friendsofsymfony\user-bundle\ Resources\views, dans votre nouveau Bundle (**UserBundle**) sous Resources\views.
7. 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



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

localhost/TestFOS/web/app\_dev.php/login



1. Redirection après authentification.
2. Redirection simple :

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



Vous pouvez trouver d’autres cas ici !

symfony.com/doc/current/cookbook/security/form\_login.html

1. 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 Redirection sous : MyApp/UserBundle/Redirection/.
* Après Nous créons sous Redirection, le fichier [AfterLoginRedirection.php](https://gist.github.com/FlYos/9828431#file-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(Page\_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(Page\_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](https://gist.github.com/FlYos/9828431" \l "file-services-yml" \t "_blank)** qui se trouve toujours dans notre bundle:

|  |
| --- |
| C:\Users\Marwen\Desktop\Capture.PNG |

* Nous injections 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](https://gist.github.com/FlYos/9828431" \l "file-security-yml" \t "_blank) pour ajouter le  success\_handler comme ceci:

|  |
| --- |
| C:\Users\Marwen\Desktop\1e.PNG |

Bonus :

Vous pouvez enlever l’algorithme decryptage utilisé par symfony avec les etapes suivantes :

1. Sous App\Confi\Security.yml changer le encoders avec plaintext.



1. Sous FOS\UserBundle\Model\User, enlever l’algorithme utilisé pour l’attribut salt dans le constructeur.

