Symfony

Table des matières

Documentation

1. IDE

PHPStorm dispose de **plugins** pour le développement avec **Symfony :**

- Framework MVC Structure Support
- Symfony Plugin

Chercher dans « settings » … « plugins »

2. Installation

Il faut télécharger le fichier (installateur) qui permettra de créer des projets Symfony

```
php –r "readfile('http://symfony.com/installer');" > symfony
```

Le fichier « symfony » devrait apparaitre dans le dossier courant de l'invite au moment de l'exécution de la commande. Cela peut être par exemple « c:\users\[user] »

3. Créer un projet Symfony

a. Placer le fichier « symfony » dans le dossier parent qui contiendra le futur projet Symfony. Cela peut être par exemple « c:\wamp\www » si on crée un projet avec WAMP.
b. Depuis une invite de commande, naviguer jusqu'à ce dossier et entrer la commande

```
php symfony new <nom_du_projet>
```

Ou **choix de la version de Symfony utilisée**

```
php symfony new <nom_du_projet> <version>
```

Exemple

```
php symfony new myproject 2.8
```

a. Les commandes

- Avec Symfony 3 « console » se trouve dans le dossier « bin » les commandes seront donc de la forme « php bin/console … ».
- Avec Symfony 2.8 « console » se trouve dans le dossier « app » donc les commandes seront de la forme « php app/console …»

Exemple Obtenir la **liste des commandes**

Avec Symfony 3.0

```
php bin/console
```

Avec Symfony 2.8

```
php app/console
```

b. Pour afficher le site

Se rendre à « http://localhost/[dossier_du_site] /web/ » Ou
« http://localhost/[dossier_du_site] /web/app_dev.php »

4. Créer un nouveau bundle

Un bundle est une sorte de module, on va pouvoir découper l'application en plusieurs modules.

Pour créer un nouveau bundle…Depuis une invite de commande, naviguer jusqu'au dossier du projet (cd …) puis

```
php bin/console generate:bundle
```

- Donner un **nom au bundle** : doit finir par **Bundle** (exemple « ArticlesBundle »)

… Le bundle est ajouté au **dossier** « **src** »

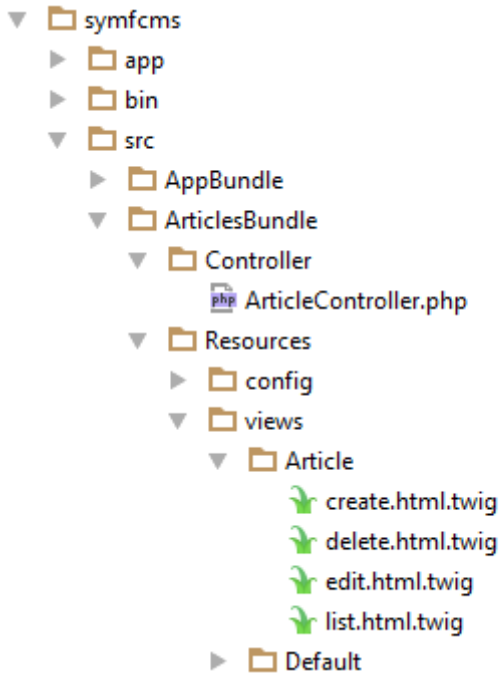
5. Créer un contrôleur

Documentation

```
php bin/console generate:controller
```

- Donner un nom au contrôleur, précéder celui-ci du nom du bundle : ArticlesBundle :Article pour générer le contrôleur « ArticleController »
- Actions du contrôleur : donner le nom pour chaque action (exemple : « listAction »), la route, le nom du template

… le contrôleur avec les actions définies et les vues sont générées dans le bundle choisi.



a. Routes

Documentation

Les routes sont désormais définies en commentaire au dessus des actions et du contrôleur

- On définit un nom à la route si elle est utilisée dans une vue par un lien (avec la méthode « path ») ou pour une redirection depuis un contrôleur par exemple.

```php
/**
 * Affiche le détail d'un article
 *
 * @Route("/view/{id}", name="articles_view")
 */
public function viewAction(Article $article)
{
  // etc.
}
```

- Route avec paramètre

```php
/**
 * Affiche le détail d'un article
 *
 * @Route("/view/{id}", name="articles_view")
 */
public function viewAction(Article $article)
{
  // etc.
}
/**
 * Edition de l'article
 *
 * @Route("/edit/{id}", name="articles_edit")
 * @Method({"GET", "POST"})
 */
public function editAction(Request $request, Article $article)
{
 // etc.
}
```

Paramètre « id » passé dans la route

Récupération de l'entité

Après request

- On peut définir la ou les méthodes appliquées pour une action

```php
/**
 *
 * @Route("/create", name="articles_create")
 * @Method({"GET", "POST"})
 */
public function createAction(Request $request)
{
 // etc.
}
```

- Une route définie au dessus d'un contrôleur s'applique à toutes les routes des actions. Les routes seront automtiquement « /articles/list », « articles/create », etc.

```php
/**
 * Article controller.
 *
 * @Route("/articles")
 */
class ArticleController extends Controller
{
```

b. Afficher une vue

Une action peut permettre d'afficher une **vue**

Vue dans le bundle

```
/**
 * Affiche la liste des articles
 *
 * @Route("/list")
 */
public function listAction()
{
    return $this->render('ArticlesBundle:Article:list.html.twig');
}
```

Vue dans « **app/Resources/views** »

```
return $this->render('default/index.html.twig');
```

Passage de **paramètres à la vue**

```
return $this->render('ArticlesBundle:Article:view.html.twig',
array('article' => $article));
```

c. Retourner une **réponse**

Documentation

Importer

```
use Symfony\Component\HttpFoundation\Response;
```

```
    /**
     * Affiche un message
     *
     * @Route("/{name}")
     */
    public function helloAction($name){
        return new Response('Bonjour '.$name);
    }
```

On définit la route désormais en commentaire. On peut de plus définir la méthode (Get, Post, etc.), donner un nom à cette route.

6. Créer une entité avec l'ORM Doctrine2

a. Modifier les informations de base de données

dans le fichier « **parameters.yml** » du dissier « app/config »

parameters.yml

```
# This file is auto-generated during the composer install
parameters:
    database_host: localhost
    database_port: null
    database_name: symfdb
    database_user: root
    database_password: null
    mailer_transport: smtp
    mailer_host: 127.0.0.1
    mailer_user: null
    mailer_password: null
    secret: ef78085c351d58718590596293d30fd75e86ad73
```

b. Création d'une **entité**

```
php bin/console generate:doctrine:entity
```

- Nom de l'entité : « NomBundle :NomEntité » (exemple « ArticlesBundle : Article »)
- Fields : donner pour chaque champ le nom, le type, la taille, unique

… L'**entité** est **générée** dans le dossier « Entity », le **repository** dans un dossier « Repository » du bundle.

a. Création de la **table correspondante**

(On peut créer la base de données si elle n'existe pas)

```
php bin/console doctrine:database:create
```

Création des tables correspondantes aux entités

```
php bin/console doctrine:schema:create
```

c. Entité avec relation

Exemple un article a un auteur (en fait correspond à l'utilisateur connecté qui rédige l'article) et un auteur peut avoir rédigé plusieurs articles.

On rajoute dans l'entité

```
/**
 * @ORM\ManyToOne(targetEntity="UsersBundle\Entity\User")
 * @ORM\JoinColumn(name="user_id", nullable=false)
 */
private $user;
```

Et le getter/ setter

```php
/**
 * Set user
 *
 * @param  User $user
 *
 * @return Article
 */
public function setUser($user)
{
    $this->user = $user;

    return $this;
}

/**
 * Get user
 *
 * @return User
 */
public function getUser()
{
    return $this->user;
}
```

Dans le **contrôleur**

```php
/**
 *
 * @Route("/create", name="articles_create")
 * @Method({"GET", "POST"})
 */
public function createAction(Request $request)
{
    $article = new Article();
    $user = $this->getUser();
    $article->setUser($user);

    $form = $this->createForm('ArticlesBundle\Form\ArticleType', $article);
    $form->handleRequest($request);

    if($request->isMethod('POST') && $form->isSubmitted() && $form->isValid()){

        $manager = $this->get('articles_article_manager');
        $article = $manager->persist($form->getData());

        $session = $this->get('session');
        $session->getFlashBag()->add('success','L\'article a été ajouté.');

        return $this->redirectToRoute('articles_view', array('id' => $article->getId()));
    }

    return $this->render('ArticlesBundle:Article:create.html.twig', array('form'=> $form->createView()));
}
```

On récupère l'utilisateur connecté et on utilise le setter de l'entité pour définir l'utisateur rédigeant l'article

Les utilisateurs seront **automatiquement récupérés** lors de la **lecture** d'articles.

```
    /**
     * Affiche la liste des articles
     *
     * @Route("/", name="articles_list")
     * @Method("GET")
     */
    public function listAction()
    {
        $manager = $this->get('articles_article_manager');
        $articles = $manager->getAll();

        return $this->render('ArticlesBundle:Article:list.html.twig', array('articles'
=> $articles));
    }
```

Dans la **vue** on affiche simplement les infomations de l'utilisateur

```
{{ article.user.username }}
```

d. Requêtes en invite de commande

Depuis une invite de commande, naviguer jusqu'au dossier du projet (cd …) puis

Exemple « Insert »

```
php bin/console doctrine:query:sql "insert into article(title,content) values('Premier article', 'Contenu du premier article article')"
```

« Select »

```
php bin/console doctrine:query:sql "select * from article"
```

e. Création de fonctions dans le repository

```
<?php
namespace AppBundle\Repository;

class ArticleRepository extends \Doctrine\ORM\EntityRepository
{
    public function getAll(){
        $qb = $this->createQueryBuilder('s');
        $result = $qb->getQuery()->execute();
        return $result;
    }
    public function getOne($id){
        $qb = $this->createQueryBuilder('s');
        $query = $qb
            ->where('s.id=:id')
            ->setParameter('id',$id);
        $result = $query->getQuery()->execute();
        return $result;
    }
}
```

Utilisation, par exemple depuis un contrôleur

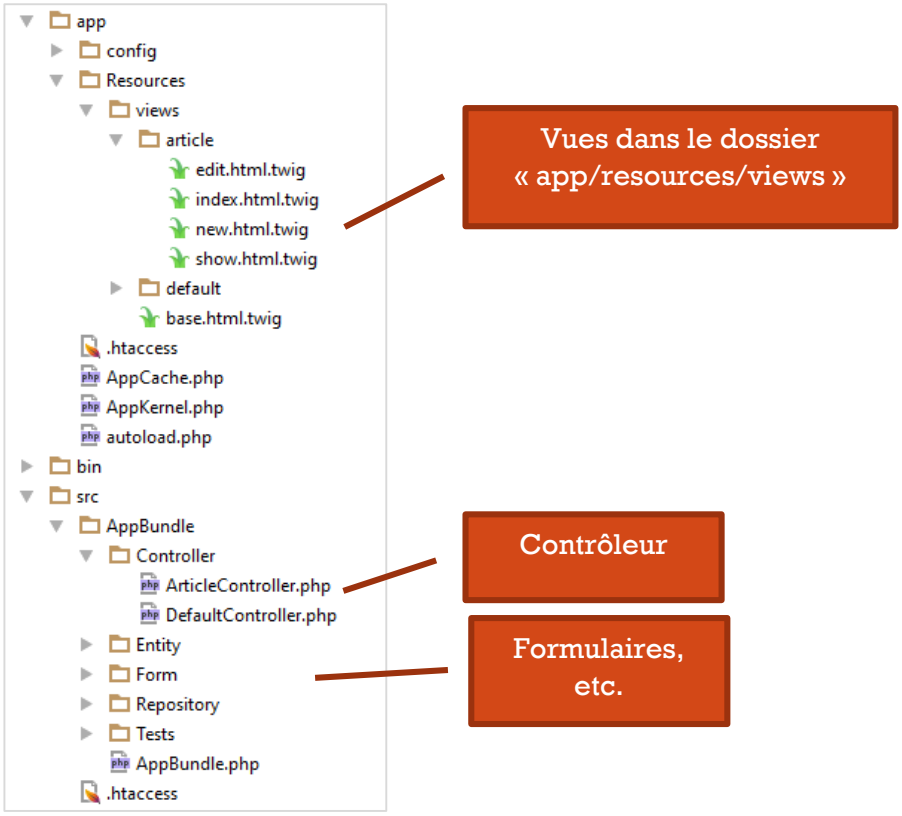
```
$em = $this->getDoctrine()->getManager();
$repository = $em->getRepository('AppBundle:Article');

$articles = $repository->getAll();
$article = $repository->getOne(1);
```

f. Crud Generator

Il est possible de générer tout le code, vues pour une entité

```
php bin\console doctrine:generate:crud
```

Sont générés



Vue liste

Vues pour l'ajout et l'édition avec formulaire

Pour installer l'aide

```
php bin\console assets:install --help
```

```
php bin\console assets:install web
```

7. Lecture et modification de données de la base

a. Avec un service

```php
<?php

namespace ArticlesBundle\Services;

use Doctrine\ORM\EntityManager;
use Doctrine\ORM\EntityRepository;

class ArticleManager
{
    private $em;
    private $repository;

    public function __construct(EntityManager $em)
    {
        $this->em = $em;
        $this->repository = $em->getRepository('ArticlesBundle:Article');
    }

    public function getAll()
    {
        return $this->repository->findAll();
    }

    public function findOne($id)
    {
        return $this->repository->find($id);
    }

    public function persist($article)
    {
        $this->em->persist($article);
        $this->em->flush();
        return $article;
    }

    public function delete($article)
    {
        $this->em->remove($article);
        $this->em->flush();
    }
}
```

Injection de dépendance

```yaml
#app\config\services.yml
parameters:

services:
    articles_article_manager:
        class: ArticlesBundle\Services\ArticleManager
        arguments: ["@doctrine.orm.entity_manager"]
```

Obtenir le service depuis un contrôleur

```php
$manager = $this->get('articles_article_manager');
```

Puis on utilise les méthodes du service pour la lecture et la modification d'articles.

b. Sans service

Il est possible de se passer d'un service. Exemple depuis les actions d'un contrôleur

Lecture

```
$em = $this->getDoctrine()->getManager();
$articles = $em->getRepository('ArticlesBundle:Article')->findAll();
```

Récupérer une ligne

```
$em = $this->getDoctrine()->getManager();
$article = $em->getRepository('ArticlesBundle:Article')->find(1);
```

Insertion

```
$article = new Article();
$article->setTitle('Le titre de l\'article');
$article->setContent('Contenu');

$em = $this->getDoctrine()->getManager();
$em->persist($article);
$em->flush();
```

Modification

```
$em = $this->getDoctrine()->getManager();
$article = $em->getRepository('ArticlesBundle:Article')->find(1);

$article->setTitle('Nouveau titre de l\'article');

$em = $this->getDoctrine()->getManager();
$em->persist($article);
$em->flush();
```

Suppression

```
$em = $this->getDoctrine()->getManager();
$article = $em->getRepository('ArticlesBundle:Article')->find(1);
$em = $this->getDoctrine()->getManager();
$em->remove($article);
$em->flush();
```

8. Création de formulaire

Documentation

a. Générer un formulaire à partir d'un modèle

Exemple génération du formulaire « ArticleType » dans un dossier « Form » du bundle à partir de l'entité Article

```
php bin\console generate:doctrine:form ArticlesBundle:Article
```

On peut ensuite personnaliser le formulaire généré

```php
<?php

namespace ArticlesBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type\TextareaType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;


class ArticleType extends AbstractType
{
    /**
     * @param FormBuilderInterface $builder
     * @param array $options
     */
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('title', TextType::class, array('label'=>'Titre','required'=>false))
            ->add('content', TextareaType::class,
array('label'=>'Contenu','required'=>false))
        ;
    }

    /**
     * @param OptionsResolver $resolver
     */
    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' => 'ArticlesBundle\Entity\Article'
        ));
    }
}
```

Pour cacher l'attribut HTML5 « required » ajouté aux champs de formulaire on définit « required à false »

Pour utiliser le formulaire depuis un contrôleur. Exemple

```
/**
 * Edition de l'article
 *
 * @Route("/edit/{id}", name="articles_edit")
 * @Method({"GET", "POST"})
 */
public function editAction(Request $request, Article $article)
{
    $form = $this->createForm('ArticlesBundle\Form\ArticleType', $article);
    $form->handleRequest($request);

    if($form->isSubmitted() && $form->isValid()){
        $manager = $this->get('articles_article_manager');
        $article = $manager->persist($form->getData());

        $session = $this->get('session');
        $session->getFlashBag()->add('success','L\'article a été modifié.');

        return $this->redirectToRoute('articles_view', array('id' => $article->getId()));
    }

    return $this->render('ArticlesBundle:Article:edit.html.twig', array('form'=> $form-
>createView()));
}
```

b. formulaire sans générer un « type »

Exemple dans une action d'un contrôleur

```
$form = $this->createFormBuilder(new Article())
        ->add('title', TextType::class, array('label'=>'Titre'))
        ->add('content', TextareaType::class, array('label'=>'Contenu'))
        ->add('submit', SubmitType::class, array('label' => 'Publier'))
        ->getForm();

$form->handleRequest($request);
if($request->isMethod('post') && $myform->isValid()){
        // etc.
}
```

Ne pas oublier les imports pour les tyes de champs

```
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type\TextareaType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
```

9. Messages Flash

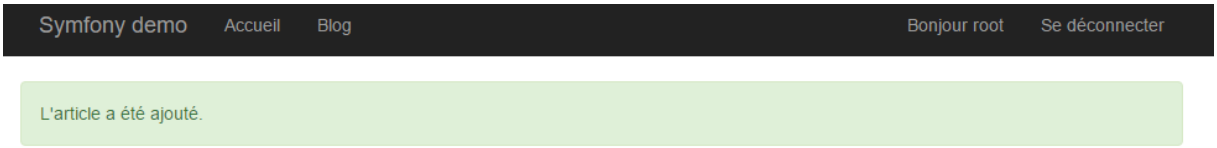
Un message qui ne sera affiché qu'une seule fois après redirection. Si on recharge la page, le message flash aura disparu.

Exemple dans une action

```
$session = $this->get('session');
$session->getFlashBag()->add('success','L\'article a été ajouté.');
```

Afficher le message (code dans la Master Page ou le header)

```
<div class="container">
    {% if(app.session.flashbag.has('success'))  %}
        <div class="alert alert-success">
            {{ app.session.flashbag.get('success')[0] }}
        </div>
    {% endif %}

    {% if(app.session.flashbag.has('error'))  %}
        <div class="alert alert-danger">
            {{ app.session.flashbag.get('error')[0] }}
        </div>
    {% endif %}
</div>
```



L'article a été ajouté.

10. CRUD (Exemple de contrôleur)

```php
<?php

namespace ArticlesBundle\Controller;

use ArticlesBundle\Entity\Article;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;
use Symfony\Component\HttpFoundation\Request;

/**
 * Articles controller.
 *
 * @Route("/articles")
 */
class ArticleController extends Controller
{
    /**
     * Affiche la liste des articles
     *
     * @Route("/", name="articles_list")
     * @Method("GET")
     */
    public function listAction()
    {
        $manager = $this->get('articles_article_manager');
        $articles = $manager->getAll();
        return $this->render('ArticlesBundle:Article:list.html.twig', array('articles' =>
$articles));
    }

    /**
     * Affiche le détail d'un article
     *
     * @Route("/view/{id}", name="articles_view")
     */
    public function viewAction(Article $article)
    {
        $manager = $this->get('articles_article_manager');
        $article = $manager->findOne($article->getId());
        return $this->render('ArticlesBundle:Article:view.html.twig', array('article' =>
$article));
    }

     /**
     *
     * @Route("/create", name="articles_create")
     * @Method({"GET", "POST"})
     */
    public function createAction(Request $request)
    {
        $article = new Article();
        $user = $this->getUser();
        $article->setUser($user);

        $form = $this->createForm('ArticlesBundle\Form\ArticleType', $article);
        $form->handleRequest($request);

        if($request->isMethod('POST') && $form->isSubmitted() && $form->isValid()){
```

```
            $manager = $this->get('articles_article_manager');
            $article = $manager->persist($form->getData());

            $session = $this->get('session');
            $session->getFlashBag()->add('success','L\'article a été ajouté.');

            return $this->redirectToRoute('articles_view', array('id' => $article-
>getId()));
        }

        return $this->render('ArticlesBundle:Article:create.html.twig', array('form'=>
$form->createView()));
    }

    /**
     * Edition de l'article
     *
     * @Route("/edit/{id}", name="articles_edit")
     * @Method({"GET", "POST"})
     */
    public function editAction(Request $request, Article $article)
    {
        $form = $this->createForm('ArticlesBundle\Form\ArticleType', $article);
        $form->handleRequest($request);

        if($form->isSubmitted() && $form->isValid()){
            $manager = $this->get('articles_article_manager');
            $article = $manager->persist($form->getData());

            $session = $this->get('session');
            $session->getFlashBag()->add('success','L\'article a été modifié.');

            return $this->redirectToRoute('articles_view', array('id' => $article-
>getId()));
        }

        return $this->render('ArticlesBundle:Article:edit.html.twig', array('form'=>
$form->createView()));
    }

    /**
     * Supprime l'article
     *
     * @Route("/{id}", name="articles_delete")
     * @Method("GET")
     */
    public function deleteAction(Request $request, Article $article)
    {
        $manager = $this->get('articles_article_manager');
        $manager->delete($article);

        $session = $this->get('session');
        $session->getFlashBag()->add('success','L\'article a été supprimé.');

        return $this->redirectToRoute('articles_list');
    }
}
```

11. Vues

Documentation Symfony, documentation moteur de vues TWIG

a. Master Page

Définition de la **Master Page** « **base.html.twig** » dans « **app/Resources/views** »

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}{% endblock %}
        <link rel="icon" type="image/x-icon" href="{{
asset('favicon.ico') }}" />
        <link rel="stylesheet" type="text/css" href="{{
asset('bootstrap/bootstrap.css') }}" />
        <link rel="stylesheet" type="text/css" href="{{
asset('css/style.css') }}" />
    </head>
    <body>
        {% include('default/header.html.twig') %}
        <section class="container">
        {% block body %}{% endblock %}
        </section>
        {% block javascripts %}{% endblock %}
    </body>
</html>
```

Utilisation de la Master Page dans les **vues d'un bundle**.

```
{% extends "::base.html.twig" %}

{% block title %}Le titre{% endblock %}

{% block body %}
<p>Le contenu</p>
{% endblock %}
```

b. Inclure des fichiers

Exemple dans la Master Page on inclut la vue « header.html.twig » du dossier « app/Resources/views/default/ »

```
{% include('default/header.html.twig') %}
```

Variables entre accolades {{ }} , Blocs de code entre {% %} , filtres

Boucles

```
{% if articles|length > 0 %}
    {% for article in articles %}
        <article>
          <h2><a href="{{ path('articles_view',{ 'id': article.id  })}</a></h2>
          <p>{{ article.content  }}</p>
        </article>
    {%  endfor %}
{%  endif %}
```

c. Faire des liens

```
<a href="{{ path('homepage') }}">Accueil</a>
<a href="{{ path('articles_list') }}">Blog</a>
```

Avec des paramètres passés

```
<a href="{{ path('articles_edit', { 'id': article.id })}}">Modifier</a>
<a href="{{ path('articles_delete', { 'id': article.id })}}">Supprimer</a>
```

Les actions du contrôleur

```
/**
 * Affiche le détail d'un article
 *
 * @Route("/view/{id}", name="articles_view")
 */
public function viewAction(Article $article)
{
  // etc.
}
/**
 * Edition de l'article
 *
 * @Route("/edit/{id}", name="articles_edit")
 * @Method({"GET", "POST"})
 */
public function editAction(Request $request, Article $article)
{
 // etc.
}
```

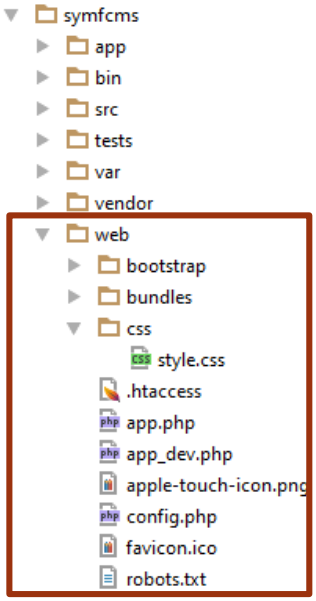
Paramètre « id » passé dans la route

Récupération de l'entité

Après request

d. Dossier « web »

On peut ajouter ses feuilles Styles, images, etc. dans le dossier « web ».



Pour récupérer des assets dans la **Master Page** « base.html.twig »

```
<link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}" />
<link rel="stylesheet" type="text/css" href="{{ asset('bootstrap/bootstrap.css') }}" />
<link rel="stylesheet" type="text/css" href="{{ asset('css/style.css') }}" />
```

12. Bundles pour Symfony

packagist (bundles à installer avec Composer), knpbundles

(Les bundles seront installés au dossier vendor)

13. Sécurité avec FOSUserBundle

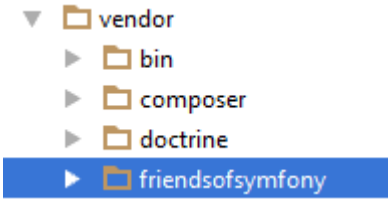
Documentation avec Symfony

a. Installation du bundle

Depuis une invite de commande…naviguer jusqu'au dossier du projet (cd …) puis

```
composer require friendsofsymfony/user-bundle "~2.0@dev"
```

« friendsofsymfony » est ajouté au dossier « vendor »



b. Configuration

On peut créer un bundle dédié à l'authentification ("UsersBundle" par exemple)

Activer la translation dans "**config.yml**"

```
#app\config\config.yml
parameters:
    locale: fr

framework:~
    translator:       { fallbacks: ["%locale%"] }
```

Décommenter (on peut aussi changer la langue locale)

Activer le bundle dans "**AppKernel.php**" du dossier "app". Ajouter la ligne :

```
#app\AppKernel.php
public function registerBundles()
{
    $bundles = [
        new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
        // etc.
        new FOS\UserBundle\FOSUserBundle()
    ];

    return $bundles;
}
```

Créer une **entité « User »**

```
<?php

namespace UsersBundle\Entity;

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

/**
 * @ORM\Entity
 * @ORM\Table(name="user")
 *
 * Class User
 * @package AppBundle\Entity
 */
```

```php
class User extends BaseUser
{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     * @var
     */
    protected $id;
    public  function __construct()
    {
        parent::__construct();
    }
}
```

Configurer "security.yml"

Supprimer tout le code de "security.yml" du dossier "app/config" et le remplacer par

```yaml
#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
                # if you are using Symfony < 2.8, use the following
config instead:
                # csrf_provider: form.csrf_provider

            logout:       true
            anonymous:    true

    access_control:
        - { path: ^/login$, role: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/register, role: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/resetting, role: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/admin/, role: ROLE_ADMIN }
        - { path: ^/articles/create$, role: IS_AUTHENTICATED_REMEMBERED }
        - { path: ^/articles/edit$, role: IS_AUTHENTICATED_REMEMBERED }
        - { path: ^/articles/delete$, role: IS_AUTHENTICATED_REMEMBERED }

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

services:
```

Route vers entité User

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

Import du **routing de FOSUserBundle** dans "routing.yml"

```
#app\config\routing.yml
fos_user:
    resource: "@FOSUserBundle/Resources/config/routing/all.xml"
```

Mettre à jour la base de données avec Doctrine

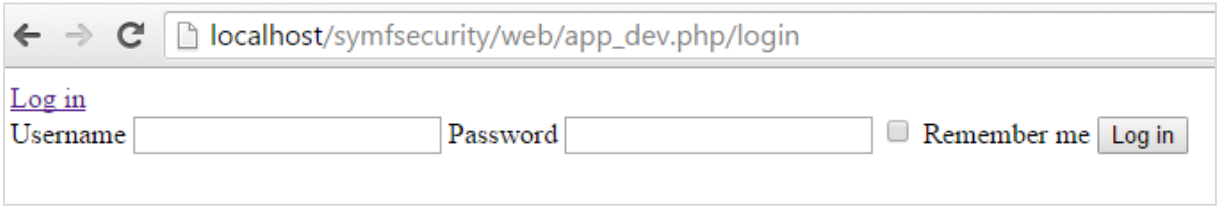
```
php bin/console doctrine:schema:update --force
```

La table est ajoutée dans la base de données

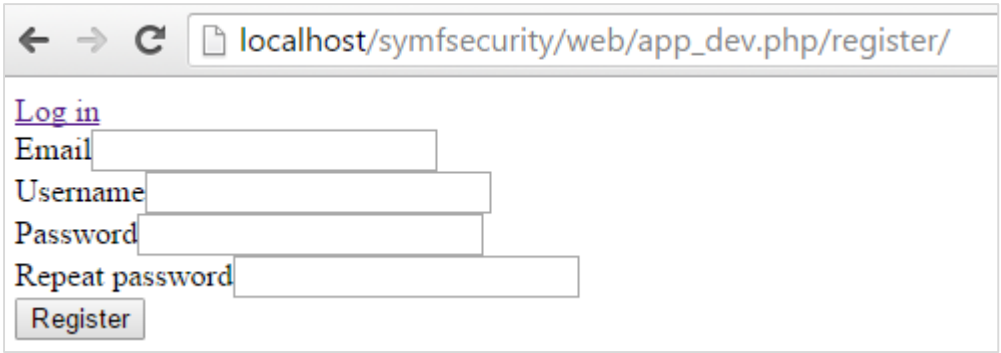
Afficher | Structure | SQL | Rechercher | Insérer | Exporter | Importer

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	id	int(11)			Non	*Aucune*	AUTO_INCREMENT
2	username	varchar(255)	utf8_unicode_ci		Non	*Aucune*	
3	username_canonical	varchar(255)	utf8_unicode_ci		Non	*Aucune*	
4	email	varchar(255)	utf8_unicode_ci		Non	*Aucune*	
5	email_canonical	varchar(255)	utf8_unicode_ci		Non	*Aucune*	
6	enabled	tinyint(1)			Non	*Aucune*	
7	salt	varchar(255)	utf8_unicode_ci		Non	*Aucune*	
8	password	varchar(255)	utf8_unicode_ci		Non	*Aucune*	
9	last_login	datetime			Oui	*NULL*	
10	locked	tinyint(1)			Non	*Aucune*	
11	expired	tinyint(1)			Non	*Aucune*	
12	expires_at	datetime			Oui	*NULL*	
13	confirmation_token	varchar(255)	utf8_unicode_ci		Oui	*NULL*	
14	password_requested_at	datetime			Oui	*NULL*	
15	roles	longtext	utf8_unicode_ci		Non	*Aucune*	
16	credentials_expired	tinyint(1)			Non	*Aucune*	
17	credentials_expire_at	datetime			Oui	*NULL*	

On peut déjà naviguer jusqu'à la page de connexion



Et d'inscription



Astuce **afficher toutes les routes** du projet dans la console :

```
php bin/console debug:router
```

Des **commandes** permettent d'enregistrer des utilisateurs depuis la **console**

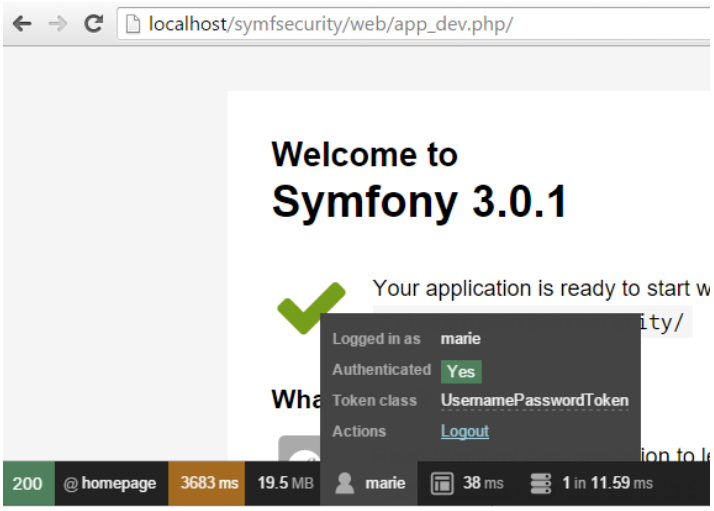
```
fos
 fos:user:activate                Activate a user
 fos:user:change-password         Change the password of a user.
 fos:user:create                  Create a user.
 fos:user:deactivate              Deactivate a user
 fos:user:demote                  Demote a user by removing a role
 fos:user:promote                 Promotes a user by adding a role
```

Exemple création d'un utilisateur

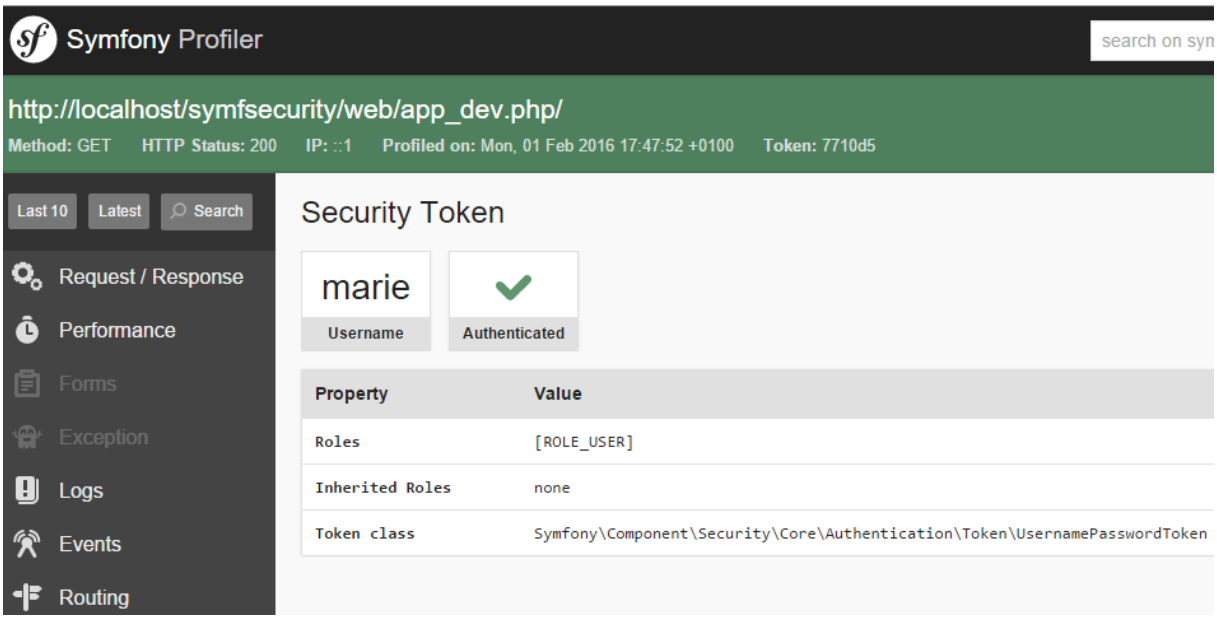
```
php bin/console fos:user:create
```

```
C:\wamp\www\symfsecurity>php bin/console fos:user:create
Please choose a username:marie
Please choose an email:mb3@hotmail.com
Please choose a password:
Created user marie
```

Debug d'un utilisateur connecté



… en cliquant sur l'utilisateur



c. Insérer les formulaires de connexion et d'inscription de FOSUserBundle dans ses vues

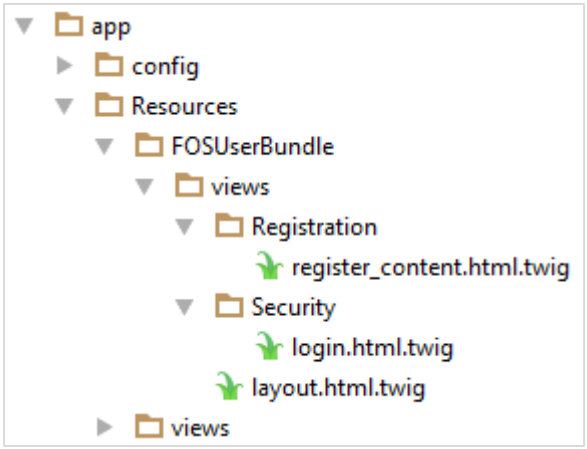
Pour afficher un formulaire d'inscription, de connexion, etc. dans une page

```
{{ render(controller('FOSUserBundle:Security:login')) }}
{{ render(controller('FOSUserBundle:Registration:register')) }}
```

d. Personnaliser un formulaire généré par FOSUserBundle

Les formulaires générés par défaut ne sont pas top pour la mise en forme. On va surcharger le formulaire de connexion par exemple.

Pour cela, on crée un dossier « FOSUserBundle » dans « app/Resources » puis on copie le chemin des vues de FOSUserBundle. Regarder les contrôleurs, les vues du bundle « friendsofsymfony » pour bien comprendre.



Ensuite on copie le contenu des vues du bundle FOSUser que l'on colle dans la vue que l'on vient de créer pour finalement personnaliser le code.

Exemples « layout.html.twig »

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
    </head>
    <body>

        <div>
            {% block fos_user_content %}
            {% endblock fos_user_content %}
        </div>
    </body>
</html>
```

« login.html.twig »

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

{% trans_default_domain 'FOSUserBundle' %}

{% block fos_user_content %}
    <div class="col-md-offset-4 col-md-4">
        <h3>Se connecter</h3>
        <hr>
        <form action="{{ path("fos_user_security_check") }}" method="post" class="form-
horizontal" autocomplete="off">
            <input type="hidden" name="_csrf_token" value="{{ csrf_token }}" />

            <div class="form-group">
                <label for="username">Nom d'utilisateur</label>
                <input type="text" id="username" name="_username" value="{{
last_username }}" class="form-control" required="required" />
            </div>
            <div class="form-group">
                <label for="password">Mot de passe</label>
                <input type="password" id="password" name="_password" class="form-
control" required="required" />
            </div>
            <div class="text-center form-group">
                <input type="checkbox" id="remember_me" name="_remember_me" value="on"
/>
                <label for="remember_me">Se souvenir de moi</label>
            </div>

            <div class="text-center form-group">
                <button type="submit" class="btn btn-primary">Se connecter</button>
                  ou  
                <a href="{{ path('register') }}">S'inscrire</a>
            </div>

            {% if error %}
                <div class="text-center text-danger">{{
error.messageKey|trans(error.messageData, 'security') }}</div>
            {% endif %}
        </form>
    </div>
{% endblock fos_user_content %}
```

« register_content.html.twig »

```
{% trans_default_domain 'FOSUserBundle' %}

<div class="col-md-offset-4 col-md-4">
    <h3>Inscription</h3>
    <hr>
{{ form_start(form, {'method': 'post', 'action':
path('fos_user_registration_register'), 'attr': {'class':
'fos_user_registration_register'}}) }}
    <fieldset>
        <div class="form-group">
            <label for="username">Nom d'utilisateur</label>
            <div class="controls">
                {{ form_widget(form.username, {'attr': {'class': 'form-
control', 'placeholder':'Nom d\'utilisateur'}}) }}
                {{ form_errors(form.username) }}
            </div>
        </div>
        <div class="form-group">
            <label for="email">Email</label>
            <div class="controls">
                {{ form_widget(form.email, {'attr': {'class': 'form-
control', 'placeholder':'Email'}}) }}
                {{ form_errors(form.email) }}
            </div>
        </div>
        <div class="form-group">
            <label for="password">Mot de passe</label>
            <div class="controls">
                {{ form_widget(form.plainPassword.first, {'attr':
{'class': 'form-control', 'placeholder': 'Mot de passe', 'required':
'required'}}) }}
                {{ form_errors(form.plainPassword.first) }}
            </div>
        </div>
        <div class="form-group">
            <label for="username">Confirmer le mot de passe</label>
            <div class="controls">
                {{ form_widget(form.plainPassword.second, {'attr':
{'class': 'form-control', 'placeholder': 'Confirmer le mot de passe',
'required': 'required'}}) }}
                {{ form_errors(form.plainPassword.second) }}
            </div>
        </div>
    <div class="text-center form-group">
        <button type="submit" class="btn btn-primary">S'inscrire</button>
          ou  
        <a href="{{ path('login') }}">Se connecter</a>
    </div>
    </fieldset>
{{ form_end(form) }}

</div>
```

e. Protéger une route

Obliger l'utilisateur à être authentifié pour accéder à une route .

```
access_control:
    - { path: ^/login$, role: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/register, role: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/resetting, role: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/admin/, role: ROLE_ADMIN }
    - { path: ^/articles/create$, role: IS_AUTHENTICATED_REMEMBERED }
    - { path: ^/articles/edit$, role: IS_AUTHENTICATED_REMEMBERED }
    - { path: ^/articles/delete$, role: IS_AUTHENTICATED_REMEMBERED }
```

On oblige l'utilisateur à être authentifié pour la création, modiifcation et suppression d'article

f. Rediriger automatiquement un utilisateur authentifié vers une route

```
firewalls:
    main:
        pattern: ^/
        form_login:
            provider: fos_userbundle
            csrf_token_generator: security.csrf.token_manager
            default_target_path: /profile
            login_path: /
```

« default_target_path »

L'utilisateur se connecte puis est redirigé automatiquement vers une page de profil

g. Contrôleur permettant d'afficher ses propres vues

```
<?php
namespace UsersBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Component\HttpFoundation\Request;
use UsersBundle\Entity\User;

class AuthController extends Controller
{
    /**
     * @Route("/register", name="register")
     */
    public function registerAction(Request $request)
    {
        return $this->render('UsersBundle:Auth:register.html.twig');
    }

    /**
     * @Route("/login", name="login")
     */
    public function loginAction()
    {
        return $this->render('UsersBundle:Auth:login.html.twig');
    }
    /**
     * Intercepte "/register/confirmed" de FOSUserBundle pour
redirection au lieu d'afficher la page confirmed
     *
     * @Route("/register/confirmed", name="redirection_confirmed")
     */
    public function redirectRegisteredAction()
```

Interception et redirection après l'inscription

```
    {
        return $this->redirectToRoute('articles_list');
    }
}
```

14. Connexion avec les réseaux sociaux

HWIOAuthBundle ne supporte actuellement pas Symfony 3 … Documentation

On peut utiliser avec Symfony 3

- OAuth2-Client Bundle
- Avec OAuth2-Google
- Et OAuth2-Facebook

a. Installation

```
composer require league/oauth2-client
composer require league/oauth2-facebook
composer require league/oauth2-google
```

b. Configuration

On peut définir les identifiants de connexion dans « **parameters.yml** »

```
facebook_client_id:  565…1
facebook_client_secret: 3271d6bc…d4f9a44ddeae
google_client_id:  92644…95ts5mphl.apps.googleusercontent.com
google_client_secret: ASG310x571…DkrgHt0ZL9
```

Dans « **services.yml** »

```
services:
#    etc.

    app.facebook_provider:
        class: League\OAuth2\Client\Provider\Facebook
        arguments:
                    -
                        clientId: %facebook_client_id%
                        clientSecret: %facebook_client_secret%
                        graphApiVersion: v2.3
                        redirectUri:
"http://localhost/symfcms/web/app_dev.php/connect/facebook-check"

    app.google_provider:
        class: League\OAuth2\Client\Provider\Google
        arguments:
                    -
                        clientId: %google_client_id%
                        clientSecret: %google_client_secret%
                        redirectUri:
"http://localhost/symfcms/web/app_dev.php/connect/google-check"
```

c. Mise à jour de la table User

a. Ajout de champs à l'entité User :

- provider : facebook, google ,etc.
- social_id l'id récupéré depuis le réseau social

```
    /**
     * @ORM\Column(type="string")
     */
    protected  $provider;

    /**
     * @ORM\Column(type="string")
     */
    protected $social_id;

    /**
     * @return mixed
     */
    public function getProvider()
    {
        return $this->provider;
    }

    /**
     * @param mixed $provider
     */
    public function setProvider($provider)
    {
        $this->provider = $provider;
    }

    /**
     * @return mixed
     */
    public function getSocialId()
    {
        return $this->social_id;
    }

    /**
     * @param mixed $social_id
     */
    public function setSocialId($social_id)
    {
        $this->social_id = $social_id;
    }
```

b. Mise à jour de la base

```
php bin/console doctrine:schema:update --force
```

c. Contrôleur

Dans le contrôleur « **AuthController** » on ajoute pour **Facebook**

```
/**
 * @Route("/connect/facebook", name="auth_facebook_connect")
 */
public function oauthFacebook()
{
    $provider= $this->get('app.facebook_provider');
    $url = $provider->getAuthorizationUrl([
        'scopes' => ['public_profile', 'email'],
    ]);

    return $this->redirect($url);
}

/**
 * @Route("/connect/facebook-check", name="auth_facebook_callback")
 */
public function oauthFacebookCallback(Request $request)
{
    $provider= $this->get('app.facebook_provider');
    $accessToken = $provider->getAccessToken('authorization_code', [
        'code' => $_GET['code']
    ]);

    // FacebookUser
    $user = $provider->getResourceOwner($accessToken);

    $em = $this->getDoctrine()->getManager();
    $userCheck = $em->getRepository('UsersBundle:User')->findOneBy(array('email' =>
$user->getEmail()));
    if(empty($userCheck))
    {
        // register
        $newUser = new User;
        $newUser->setEmail($user->getEmail());
        $parts = explode("@", $user->getEmail());
        $username = $parts[0];
        $newUser->setUsername($username);
        $newUser->setProvider("facebook");
        $newUser->setSocialId($user->getId());
        $newUser->setPassword(crypt(str_shuffle("abcdefgh123456")));

        $em->persist($newUser);
        $em->flush();

        $this->logIn($request, $newUser);

        $session = $this->get('session');
        $session->getFlashBag()->add('success','Vous êtes connecté');

        return $this->redirectToRoute('articles_list');
    }
    else
    {
        if($userCheck->getProvider() == "facebook")
        {
            $this->logIn($request, $userCheck);

            $session = $this->get('session');
```

```
            $session->getFlashBag()->add('success','Vous êtes connecté');

            return $this->redirectToRoute('articles_list');
        }
        else
        {
            $session = $this->get('session');
            $session->getFlashBag()->add('success','Vous êtes connecté');
            return $this->redirect('/login');
        }
    }
}
```

Pour **Google**

Créer une application avec

- API Google + activée
- Créer des identifiants « ID Client Oauth » avec par exemple
  - Origine http://localhost
  - Redirection
    « http://localhost/symfcms/web/app_dev.php/connect/google-check »

```
/**
 * @Route("/connect/google", name="auth_google_connect")
 */
public function oauthGoogle()
{
    $provider= $this->get('app.google_provider');
    $url = $provider->getAuthorizationUrl();
    return $this->redirect($url);
}

/**
 * @Route("/connect/google-check", name="auth_google_callback")
 */
public function oauthGoogleCallback(Request $request)
{

    $provider= $this->get('app.google_provider');
    $accessToken = $provider->getAccessToken('authorization_code', [
        'code' => $_GET['code']
    ]);

    // GoogleUser
    $user = $provider->getResourceOwner($accessToken);

    $em = $this->getDoctrine()->getManager();
    $userCheck = $em->getRepository('UsersBundle:User')->findOneBy(array('email' =>
$user->getEmail()));
    if(empty($userCheck))
    {
        // register
        $newUser = new User;
        $newUser->setEmail($user->getEmail());
        $parts = explode("@", $user->getEmail());
        $username = $parts[0];
        $newUser->setUsername($username);
        $newUser->setProvider("google");
```

```
            $newUser->setSocialId($user->getId());
            $newUser->setPassword(crypt(str_shuffle("abcdefgh123456")));

            $em->persist($newUser);
            $em->flush();

            $this->logIn($request, $newUser);

            $session = $this->get('session');
            $session->getFlashBag()->add('success','Vous êtes connecté');

            return $this->redirectToRoute('articles_list');
        }
        else
        {
            // verifier provider
            if($userCheck->getProvider() == "google")
            {
                $this->logIn($request, $userCheck);

                $session = $this->get('session');
                $session->getFlashBag()->add('success','Vous êtes connecté');

                return $this->redirectToRoute('articles_list');
            }
            else
            {
                $session = $this->get('session');
                $session->getFlashBag()->add('success','Vous êtes connecté');
                return $this->redirect('/login');
            }
        }
}
```

La fonction permettant de connecter l'utilisateur

Firewall

```
private function logIn(Request $request, User $user)
{
    $token = new UsernamePasswordToken($user, $user->getPassword(), "public", $user-
>getRoles());
    $this->get("security.token_storage")->setToken($token);


    $event = new InteractiveLoginEvent($request, $token);
    $this->get("event_dispatcher")->dispatch("security.interactive_login", $event);
}
```

Les imports

```
use Symfony\Component\Security\Core\Authentication\Token\UsernamePasswordToken;
use Symfony\Component\Security\Http\Event\InteractiveLoginEvent;
```

d. Vue

Dans la vue « login.html.twig », on peut ajouter deux liens

```
<a href="{{ path('auth_facebook_connect') }}">Facebook</a>
<a href="{{ path('auth_google_connect') }}">Google</a>
```


Symfony demo
Accueil
Blog
S'inscrire
Se connecter
Se connecter avec
Facebook
Google+
Ou
Nom d'utilisateur
root
Mot de passe
Se souvenir de moi
Se connecter
ou
S'inscrire