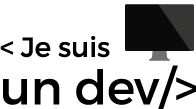
[[](http://www.jesuisundev.fr/)Je fais des trucs sympas, des fois je partage. Y'a des gifs marrants aussi.](http://www.jesuisundev.fr/" \o "Je suis un dev)



**[Sign-in & Login Facebook / Google / Twitter / Email via Symfony2](http://www.jesuisundev.fr/sign-in-et-login-facebook-google-twitter-email-via-symfony2/)**

7 mai 2015 [7 comments](http://www.jesuisundev.fr/sign-in-et-login-facebook-google-twitter-email-via-symfony2/#disqus_thread) Article

Si vous affichez un formulaire de trois kilomètres de long à un utilisateur pour s’inscrire, il va vous faire une syncope tout de suite. C’est terrible, les statistiques le montre, tout le monde à la flemme et du coup personne ne s’inscrit.

Mais proposez-lui une inscription en deux clics depuis un compte Google ou Facebook (ça tombe bien en plus il est déjà connecté sur son navigateur) et ça sera joie / bonheur / exaltation dans son cœur. Et vous, vous récupérerez un utilisateur de plus. Les fainéants sont légion et il nous faut toujours plus de monde sur nos applis !

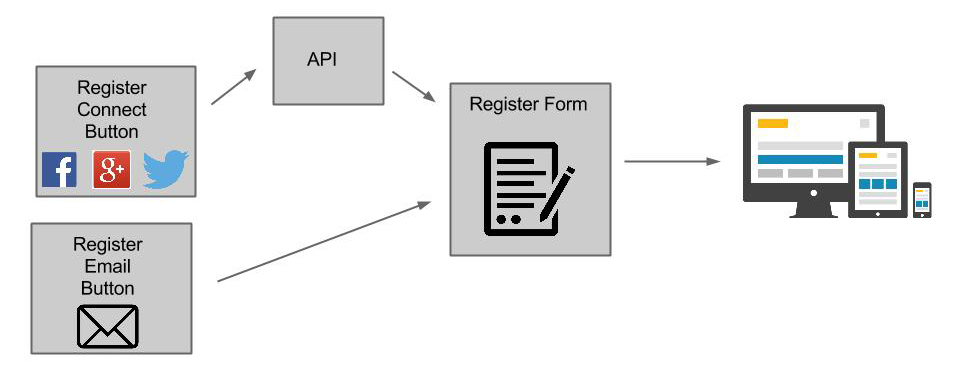
Cependant, je vous vois venir avec votre remarque. En effet, certains vont prendre peur en voyant des réseaux sociaux et ne s’inscriront jamais chez vous. On va donc également proposer une alternative email classique.

Pour ce faire on va se baser sur deux bundles, le premier étant [FOSUserBundle](https://github.com/FriendsOfSymfony/FOSUserBundle), poids lourd de la gestion d’utilisateurs dans le milieu symfony2.  Le second [HWIOAuthBundle](https://github.com/hwi/HWIOAuthBundle" \t "_blank) permettant une intégration simple du OAuth2 pour les réseaux sociaux.

Le tout va nous permettre un système de sign-in & login comme on les aime.

**Un beau dessin**

Alors avant de commencer on va déjà se mettre d’accord sur où on va. Pour ce faire au lieu de faire un gros pavé, on va faire un gros dessin.

[](http://www.jesuisundev.fr/wp-content/uploads/2015/03/form_register.jpg)

Il est pas beau mon schéma ? Ho oui c’est de toute beauté! Bon OK, ça ne m’a pas pris un temps fou mais c’est quand même bien sympa  !

Alors du coup c’est assez simple, que l’utilisateur veuille passer par les réseaux sociaux ou via le mail classique on le renverra vers un formulaire intermédiaire. A la différence que via les réseaux sociaux il passera par l’API correspondante et on aura déjà les infos !

**Installation**

Commencons par installer nos chers bundles

**composer.son**



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | "require": {          ...          "friendsofsymfony/user-bundle": "~1.3",          "hwi/oauth-bundle": "0.4.\*@dev",          ...      }, |

Et on met à jour



|  |  |
| --- | --- |
| 1 | php composer.phar update |

Je ne reviens pas sur l’installation de FOS user, tout est très bien expliqué sur [la page officiel](https://github.com/FriendsOfSymfony/FOSUserBundle/blob/1.3.x/Resources/doc/index.md) !

**Move fast and break things**

Ho oui comme disait Mark Zuckerberg a une certaine époque, on passe aux choses sérieuses, il y à de nombreux fichiers à configurer pour arriver à nos fins.

Alors dans l’ordre :

**Acme/UserBundle/Entity/User.php**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | <?php  namespace Acme\UserBundle\Entity;    use FOS\UserBundle\Entity\User as BaseUser;  use Doctrine\ORM\Mapping as ORM;  use Doctrine\Common\Collections\ArrayCollection;    /\*\*  \* @ORM\Entity  \* @ORM\Table(name="fos\_user")  \*/    class User extends BaseUser  {      public function \_\_construct()      {          parent::\_\_construct();          $this->media = new \Doctrine\Common\Collections\ArrayCollection();      }        /\*\*       \* @ORM\Id       \* @ORM\Column(type="integer")       \* @ORM\GeneratedValue(strategy="AUTO")       \*/      protected $id;        /\*\*       \* @var string       \*       \* @ORM\Column(name="facebook\_id", type="string", nullable=true)       \*/      protected $facebook\_id;        /\*\*       \* @var string       \*       \* @ORM\Column(name="google\_id", type="string", nullable=true)       \*/      protected $google\_id;        /\*\*       \* @var string       \*       \* @ORM\Column(name="twitter\_id", type="string", nullable=true)       \*/      protected $twitter\_id;    } |

Ici on ajuste notre entité utilisateur à nos besoins. Je rajoute l’id unique et indispensable pour connexion de chaque réseaux sociaux. On les met bien évidement en non obligatoire pour nos amis hater des réseaux sociaux pour qu’ils puissent se connecter sans ces derniers. Hé oui on aime tout le monde !

[](http://www.jesuisundev.fr/wp-content/uploads/2015/05/LOVE_YOU.gif)

N’oubliez pas de faire un petit : **php app/console doctrine:generate:entities AcmeUserBundle** pour créer les getters et setters.

**App/config/config.yml**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | fos\_user:      db\_driver: orm      firewall\_name: main      user\_class: Acme\UserBundle\Entity\User        registration:          form:              type: acme\_user\_registration          confirmation:              enabled: false    hwi\_oauth:      connect:          confirmation: true          registration\_form: fos\_user.registration.form      firewall\_name: main      fosub:          username\_iterations: 30          properties:              facebook: facebook\_id              google: google\_id              twitter: twitter\_id      resource\_owners:          # Facebook Access          facebook:              type:                facebook              client\_id:           [ClientID]              client\_secret:       [ClientIDSecret]              scope:               "email"              options:                  display: popup          # Google Access          google:              type:                google              client\_id:           [ClientID]              client\_secret:       [ClientIDSecret]              scope:               "https://www.googleapis.com/auth/userinfo.email https://www.googleapis.com/auth/userinfo.profile"              options:                  display: popup          # Twitter Access          twitter:              type:                twitter              client\_id:           [ClientID]              client\_secret:       [ClientIDSecret]              scope:               "" |

Le fichier de config !

Ici je me contente de Facebook, Google et Twitter.

Pour accéder aux infos de chaque réseaux sociaux, il faudra créer une application concernant votre site sur le réseau social en question. Cette application vous permettra de faire communiquer votre site avec les données de l’utilisateur sur le réseau social. Concernant les deux id : ‘client\_id’ et ‘client\_secret’ il s’agit des deux clefs d’authentification de votre application.  
Voici les liens par réseaux pour créer une application et obtenir les deux clefs d’API nécessaire à la connexion.

– [Facebook](https://developers.facebook.com/docs/facebook-login/v2.3)

– [Google](https://console.developers.google.com)

– [Twitter](https://dev.twitter.com/docs/auth/oauth)

Une fois sur ces liens suivez les explications vous allez pouvoir créer vos applications en peu de temps.

**App/config/routing.yml**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | # HWIOAuthBundle  hwi\_oauth\_security:      resource: "@HWIOAuthBundle/Resources/config/routing/login.xml"      prefix: /connect  hwi\_oauth\_redirect:      resource: "@HWIOAuthBundle/Resources/config/routing/redirect.xml"      prefix:   /connect    # We override this part of the routing  hwi\_oauth\_connect:       resource: "@AcmeAcmeBundle/Resources/config/routing/connect.xml"       prefix: /connect    facebook\_login:      pattern: /connect/check-facebook    google\_login:      pattern: /connect/check-google    twitter\_login:      pattern: /connect/check-twitter    # FOSUser    fos\_user\_security:      resource: "@FOSUserBundle/Resources/config/routing/security.xml"    fos\_user\_register:      resource: "@FOSUserBundle/Resources/config/routing/registration.xml"      prefix: /register    fos\_user\_resetting:      resource: "@FOSUserBundle/Resources/config/routing/resetting.xml"      prefix: /resetting    fos\_user\_change\_password:      resource: "@FOSUserBundle/Resources/config/routing/change\_password.xml"      prefix: /profile |

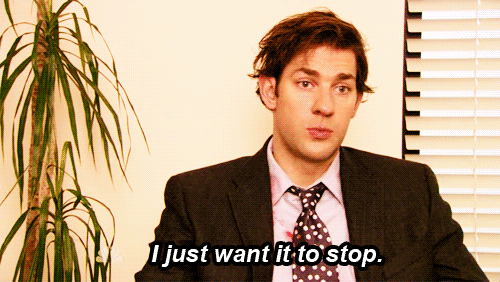
Le fichier de routing. On rajoute nos routes pour chaque réseaux sociaux pour différencier l’appel vers les apis correspondantes, mais surtout on surcharge une partie du routing par notre fichier qui traitera, l’inscription par les réseaux sociaux.

**AcmeAcmeBundle/Resources/config/routing/connect.xml**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?xml version="1.0" encoding="UTF-8" ?>    <routes xmlns="http://symfony.com/schema/routing"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:schemaLocation="http://symfony.com/schema/routing http://symfony.com/schema/routing/routing-1.0.xsd">        <route id="hwi\_oauth\_connect\_service" pattern="/service/{service}">          <default key="\_controller">HWIOAuthBundle:Connect:connectService</default>      </route>        <route id="hwi\_oauth\_connect\_registration" pattern="/registration/{key}">          <default key="\_controller">AcmeAcmeBundle:Connect:registration</default>      </route>  </routes> |

Le fichier de route qui nous permettra de faire ce qu’on veut, on aime ça faire ce qu’on veut, lors de l’inscription d’une personne sur notre appli. Je détaillerai le contenu du controller registration plus bas. Mais avant, ENCORE de la config… C’est bientôt fini !

[](http://www.jesuisundev.fr/wp-content/uploads/2015/05/stop.gif)

**app/config/security.yml**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | security:      encoders:          FOS\UserBundle\Model\UserInterface: sha512        role\_hierarchy:          ROLE\_ADMIN:       ROLE\_USER          ROLE\_SUPER\_ADMIN: [ROLE\_ADMIN, ROLE\_ALLOWED\_TO\_SWITCH]        providers:          fos\_userbundle:              id: fos\_user.user\_provider.username\_email        firewalls:          main:              pattern: ^/              form\_login:                  provider: fos\_userbundle                  csrf\_provider: form.csrf\_provider                  login\_path: /login                  check\_path: /login\_check              oauth:                  resource\_owners:                      facebook:           "/connect/check-facebook"                      google:             "/connect/check-google"                      twitter:            "/connect/check-twitter"                  login\_path:        /connect                  failure\_path:      /connect                    oauth\_user\_provider:                      service: hwi\_oauth.user.provider.fosub\_bridge              logout:       true              anonymous:    true            login:              pattern:  ^/login$              security: false              remember\_me:                  key: "%secret%"                  lifetime: 31536000                  path: /                  domain: ~        access\_control:          - { path: ^/login$, role: IS\_AUTHENTICATED\_ANONYMOUSLY }          - { path: ^/register, role: IS\_AUTHENTICATED\_ANONYMOUSLY }          - { path: ^/resetting, role: IS\_AUTHENTICATED\_ANONYMOUSLY } |

Le fichier de sécurité indispensable aux connexions.

**Ok Google**

Overdose de config je suis d’accord mais pas le choix si on veut faire tourner la machine nickel ! Et c’est pas la première, ni la dernière fois que vous configurez un truc hein ! Bref on passe au controller.

Alors on surchage le controller qui se trouve la [HWIOAuthBundle:Connect](https://github.com/hwi/HWIOAuthBundle/blob/master/Controller/ConnectController.php" \t "_blank),  on va le chercher et on en fait une copie dans notre bundle à AcmeAcmeBundle:Connect.  
Vous l’aurez remarqué on ne surcharge que la méthode registration, cette méthode nous permet d’avoir un contrôle total lors de l’inscription via réseau sociaux, avant et après soumission du formulaire:

**src/Acme/AcmeBundle/Controller/ConnectController.php**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131 | <?php  namespace Acme\AcmeBundle\Controller;    use HWI\Bundle\OAuthBundle\OAuth\ResourceOwnerInterface;  use HWI\Bundle\OAuthBundle\Security\Core\Authentication\Token\OAuthToken;  use HWI\Bundle\OAuthBundle\Security\Core\Exception\AccountNotLinkedException;  use Symfony\Component\DependencyInjection\ContainerAware;  use Symfony\Component\Form\FormInterface;  use Symfony\Component\HttpFoundation\RedirectResponse;  use Symfony\Component\HttpFoundation\Request;  use Symfony\Component\HttpFoundation\Response;  use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;  use Symfony\Component\Security\Core\Exception\AccessDeniedException;  use Symfony\Component\Security\Core\Exception\AccountStatusException;  use Symfony\Component\Security\Core\SecurityContext;  use Symfony\Component\Security\Core\User\UserInterface;  use Symfony\Component\Security\Http\Event\InteractiveLoginEvent;  use Symfony\Component\Security\Http\SecurityEvents;    class ConnectController extends ContainerAware  {      public function registrationAction(Request $request, $key)      {          $connect = $this->container->getParameter('hwi\_oauth.connect');          if (!$connect) {              throw new NotFoundHttpException();          }            $hasUser = $this->container->get('security.context')->isGranted('IS\_AUTHENTICATED\_REMEMBERED');          if ($hasUser) {              throw new AccessDeniedException('Cannot connect already registered account.');          }            $session = $request->getSession();          $error = $session->get('\_hwi\_oauth.registration\_error.'.$key);          $session->remove('\_hwi\_oauth.registration\_error.'.$key);            if (!($error instanceof AccountNotLinkedException) || (time() - $key > 300)) {              throw new \Exception('Cannot register an account.');          }            $userInformation = $this              ->getResourceOwnerByName($error->getResourceOwnerName())              ->getUserInformation($error->getRawToken())          ;            // enable compatibility with FOSUserBundle 1.3.x and 2.x          if (interface\_exists('FOS\UserBundle\Form\Factory\FactoryInterface')) {              $form = $this->container->get('hwi\_oauth.registration.form.factory')->createForm();          } else {              $form = $this->container->get('hwi\_oauth.registration.form');          }            $formHandler = $this->container->get('hwi\_oauth.registration.form.handler');            if ($formHandler->process($request, $form, $userInformation)) {              // Connect user              $this->container->get('hwi\_oauth.account.connector')->connect($form->getData(), $userInformation);                // Authenticate the user              $this->authenticateUser($request, $form->getData(), $error->getResourceOwnerName(), $error->getRawToken());                // Getting user              $user = $this->container->get('security.context')->getToken()->getUser();                // Getting social network source              $source = $userInformation->getResourceOwner()->getName();                // Updating user by source              switch ($source) {                  case 'facebook':                      $user = $this->handleFacebookResponse($userInformation, $user);                      break;                  case 'google':                      $user = $this->handleGoogleResponse($userInformation, $user);                      break;                  case 'twitter':                      $user = $this->handleTwitterResponse($userInformation, $user);                      break;              }                // Saving User              $em = $this->container->get('doctrine.orm.entity\_manager');              $em->persist($user);              $em->flush();                // Redirect the user to homepage              $url = $this->container->get('router')->generate(                  'acme\_acme\_homepage'              );              return new RedirectResponse($url);          }            // reset the error in the session          $key = time();          $session->set('\_hwi\_oauth.registration\_error.'.$key, $error);            return $this->container->get('templating')->renderResponse('HWIOAuthBundle:Connect:registration.html.' . $this->getTemplatingEngine(), array(              'key' => $key,              'form' => $form->createView(),              'userInformation' => $userInformation,          ));      }        public function handleFacebookResponse($response, $user) {          // User is from Facebook : DO STUFF HERE \o/          // All data from Facebook          $data = $response->getResponse();          // His profile image : file\_get\_contents('https://graph.facebook.com/' . $response->getUsername() . '/picture?type=large')            return $user;      }        public function handleGoogleResponse($response, $user) {          // User is from Google: DO STUFF HERE \o/          // All data from Google          $data = $response->getResponse();          // His profile image : file\_get\_contents($data['picture'])            return $user;      }        public function handleTwitterResponse($response, $user) {          // User is from Twitter: DO STUFF HERE \o/          // All data from Twitter          $data = $response->getResponse();          // His profile image : file\_get\_contents($data['profile\_image\_url'])            return $user;      }  } |

Ici je ne fais que changer la fin du traitement d’inscription, en différenciant les sources de réseaux sociaux pour personnaliser le traitement des informations pour chaque cas.

A noter que si vous voulez changer le formulaire d’inscription, il vous suffira de changer le formulaire d’inscription de FOSUser,  il en va de même pour les templates !

[](http://www.jesuisundev.fr/wp-content/uploads/2015/05/epic-gif.gif)

**Epilogue**

Je me contente ici de trois réseaux sociaux mais si vous en voulez d’autre HWIOAuthBundle en proprose un gros paquet [ici](https://github.com/hwi/HWIOAuthBundle/blob/master/Resources/doc/2-configuring_resource_owners.md).

A vous ensuite d’adapter le code pour chaque réseaux sociaux dans les fichiers de configs et dans le controller !

Happy coding !

**Categories:** [PHP](http://www.jesuisundev.fr/category/php-2/), [Symfony2](http://www.jesuisundev.fr/category/symfony2-2/)

Tags: [Connect](http://www.jesuisundev.fr/tag/connect/), [FOSUserBundle](http://www.jesuisundev.fr/tag/fosuserbundle/), [HWIOAuthBundle](http://www.jesuisundev.fr/tag/hwioauthbundle/), [Login](http://www.jesuisundev.fr/tag/login/), [Sign-in](http://www.jesuisundev.fr/tag/sign-in/)

Copyright - Je suis un dev - 2015

[Email](mailto:metrik41@gmail.com) [Rss](http://www.jesuisundev.fr/feed/)