Claroline & O365

Documentation

Table des matières

[Introduction 3](#_Toc414451377)

[Principe 3](#_Toc414451378)

[Paramétrer Azure 4](#_Toc414451379)

[Se connecter 4](#_Toc414451380)

[Ajouter l’application 4](#_Toc414451381)

[Type d’application 6](#_Toc414451382)

[Nom de l’application 6](#_Toc414451383)

[Sign-on URL et Application ID URI 7](#_Toc414451384)

[Fin de l’ajout 7](#_Toc414451385)

[Client ID et Key 8](#_Toc414451386)

[Connexion 10](#_Toc414451387)

[Bouton de connexion 10](#_Toc414451388)

[Connexion Office 365 11](#_Toc414451389)

[Retour vers Claroline 12](#_Toc414451390)

[Implémentation 13](#_Toc414451391)

[Paramètres 13](#_Toc414451392)

[Graph 13](#_Toc414451393)

[AuthorizationHelperForGraph 14](#_Toc414451394)

[GraphServiceAccessHelper 15](#_Toc414451395)

[Bouton de connexion 16](#_Toc414451396)

[Contrôleur 16](#_Toc414451397)

[O365Controller 17](#_Toc414451398)

[Définition du routage 17](#_Toc414451399)

[Définition des services 18](#_Toc414451400)

[Méthode de réception du token 18](#_Toc414451401)

[Méthode de connexion 19](#_Toc414451402)

[Objet JSON 21](#_Toc414451403)

# Introduction

## Principe

La connexion entre Claroline et Office 365 se réalise en plusieurs étapes, décrites ci-dessous :

1. La plateforme Claroline effectue une première demande à Office 365 : l’authentification de l’utilisateur à O365 à l’aide d’une URL particulière ;
2. Si les données de connexion sont correctes (et donc si l’utilisateur est correctement authentifié), il est redirigé vers Claroline avec un token de validation dans la réponse ;
3. A l’aide de ce token, une seconde requête est effectuée vers O365 pour récupérer certaines informations concernant l’utilisateur en question ;
4. Ces informations sont renvoyées vers Claroline sous forme de JSON ;

1

O365

Claroline

2

4

3

# Paramétrer Azure

## Se connecter

Connectez-vous avec votre compte administrateur O365 (mydomain.onmicrosoft.com) sur <https://manage.windowsazure.com>.

## Ajouter l’application

Divers onglets se trouvent sur la gauche. Cliquez sur « Active Directory », tout en bas, et cliquez ensuite sur votre domaine.

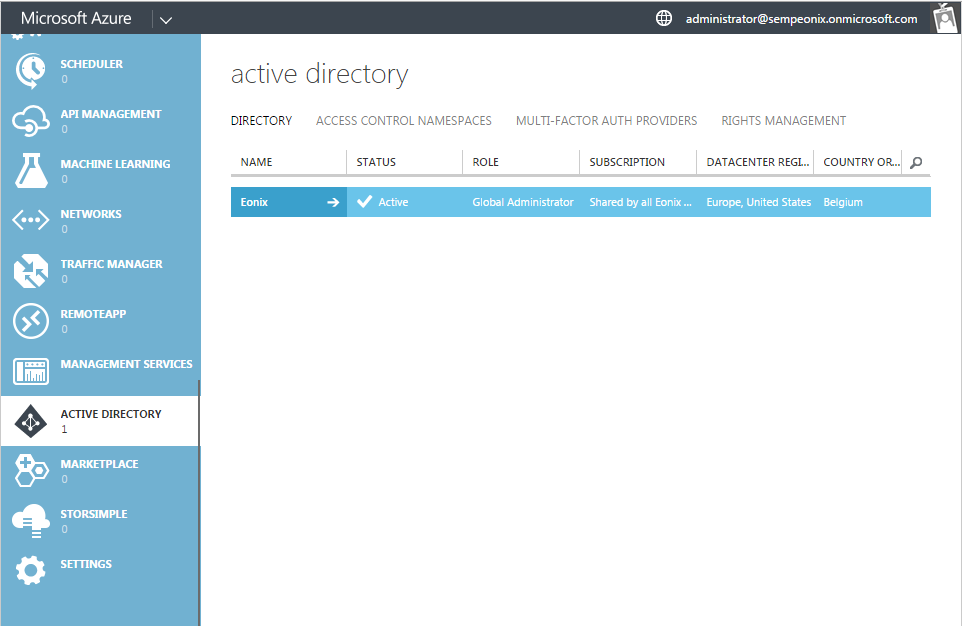


Image 1

Des onglets apparaîtront en haut de la page, cliquez sur « Applications ».

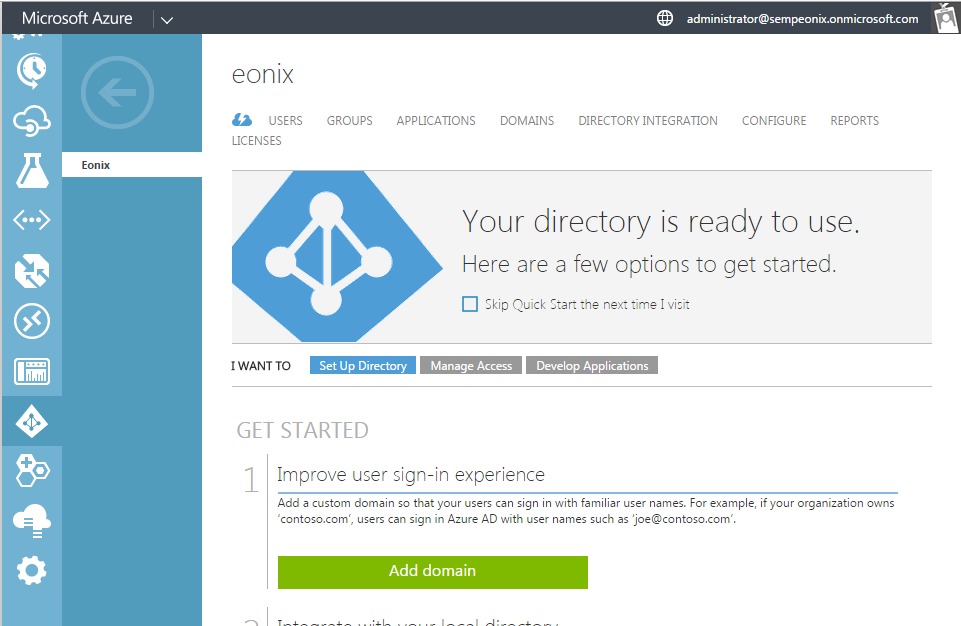


Image 2

Une fois dans ce menu, vous apercevrez les applications déjà existantes, comme Exchange Online ou Sharepoint Online. A l’aide du bouton d’ajout situé au bas de la page, créons une nouvelle application.

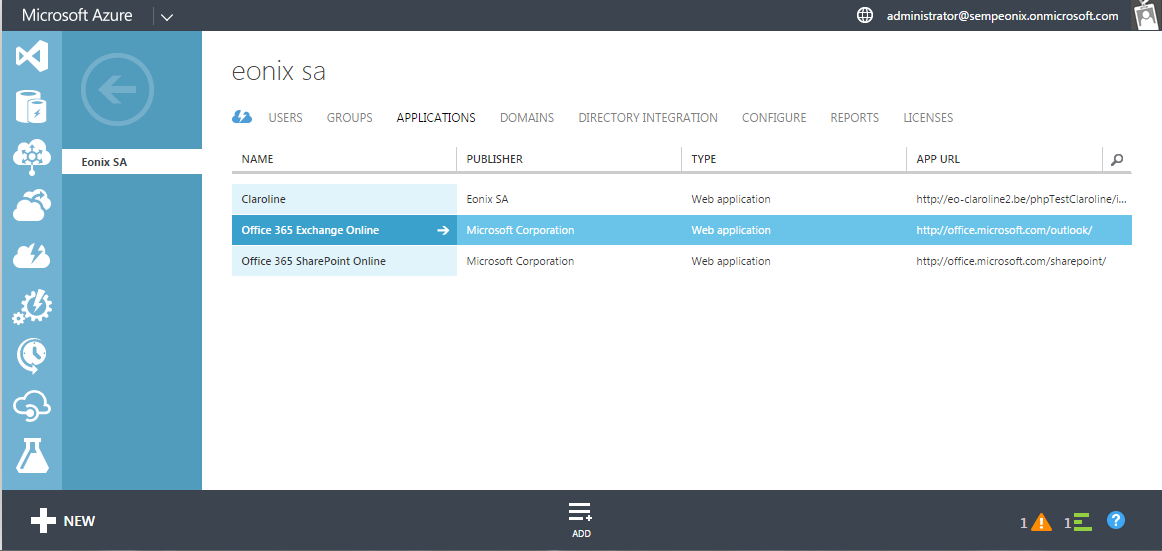


Image 3

### Type d’application

Sélectionnons l’option « Add an application my organization is developing ».

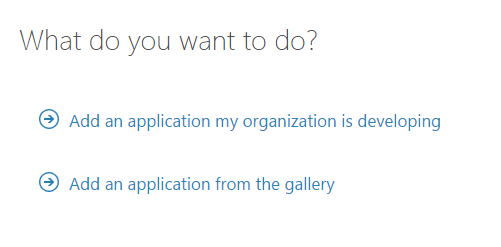


Image 4

### Nom de l’application

Fournissons-lui un nom, tout simplement « Claroline ».

*Remarque*: afin de réaliser ce tutoriel, nous avons appelé cette nouvelle application « Claroline-Demo », l’application *« Claroline » était déjà créée.*

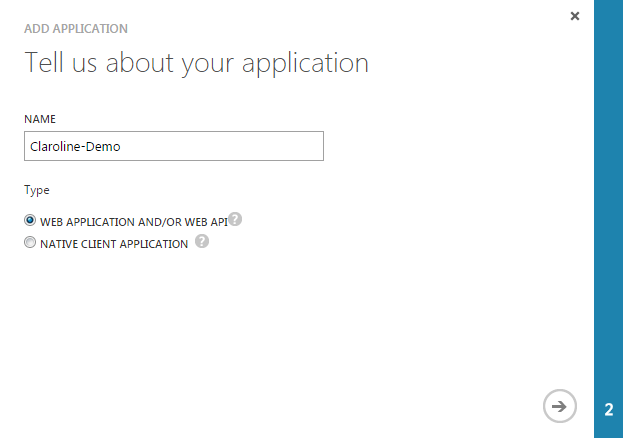


Image 5

### Sign-on URL et Application ID URI

L’application ID URI n’est pas utilisée ici, vous pouvez tout de même la définir avec une adresse comme <http://monadresse.be/app.php>.

Remarque : cette adresse par défaut est l’adresse de la page web d’accueil de Claroline.

La sign-on URL, par contre, est la première « Reply URL » définie dans l’application.  
Ces URL sont les adresses vers lesquelles la page de connexion O365 sera autorisée à se rediriger.   
Si l’adresse de redirection n’est pas enregistrée dans O365, un message d’erreur sera affiché.

Définissons à <http://monadresse.be/app.php/o365/login>.

### Fin de l’ajout

Voilà, votre application a bien été ajoutée.

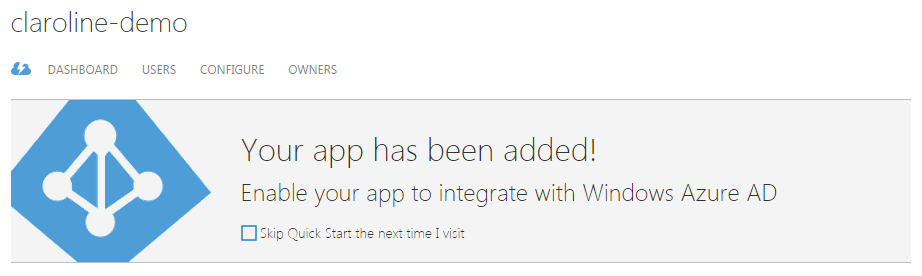


Image 6

## Client ID et Key

En cliquant sur « Configure » sur le menu supérieur, vous accéderez aux paramètres de configuration de votre application.

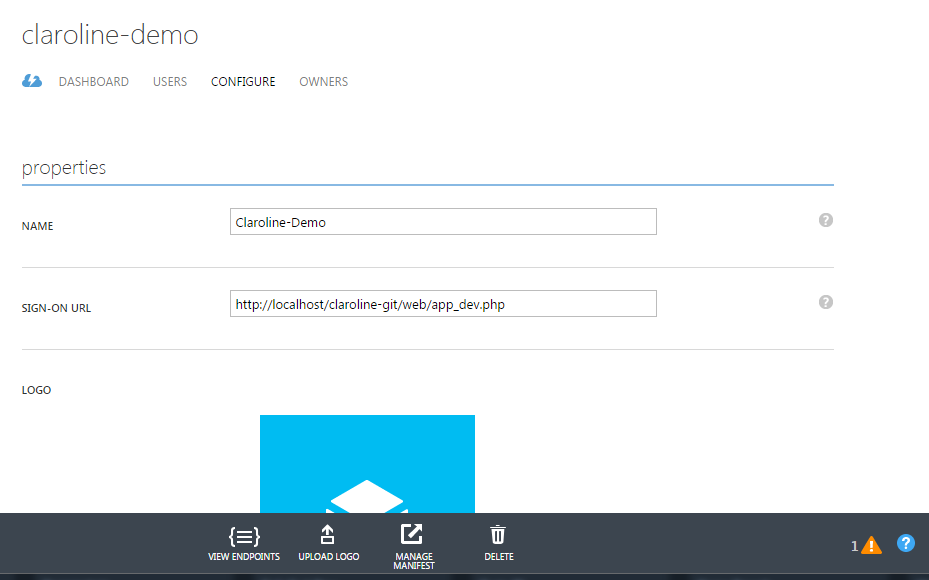


Image 7

Vous trouverez plus bas les deux dernières informations nécessaires, à savoir le Client ID et la clé permettant l’utilisation de votre l’application.

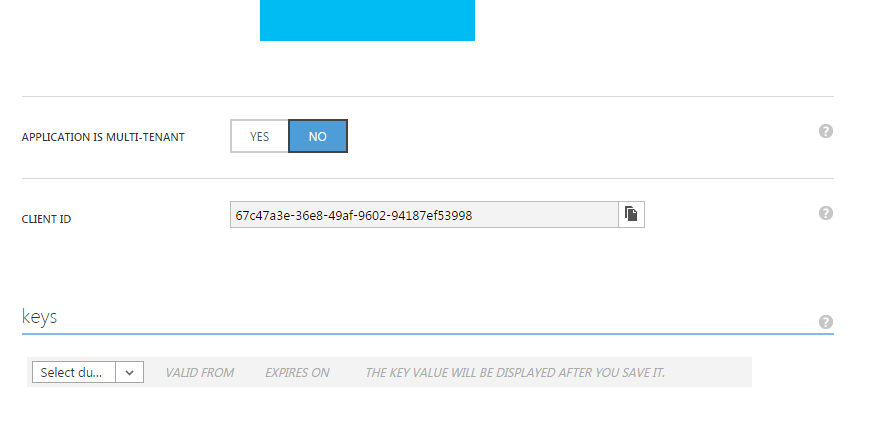


Image 8

La clé a été générée à la création de l’application, et peut directement être copiée dans Claroline (cf. Paramétrer Claroline).

La clé doit être créée : définissez tout d’abord une durée de validation (une année, ou deux), et une fois la durée de vie de la clé sélectionnée, O365 génère une nouvelle clé une fois les modifications sauvegardées.

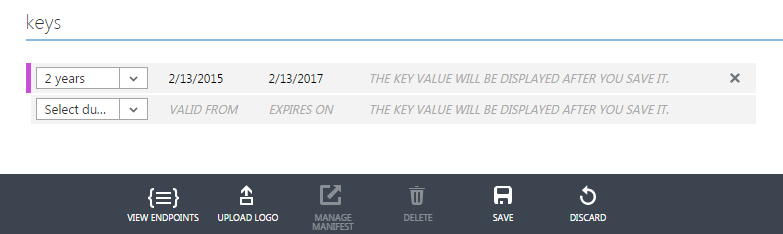


Image 9

Attention, il faut copier la clé une fois celle-ci générée, au risque de ne plus pouvoir y accéder.

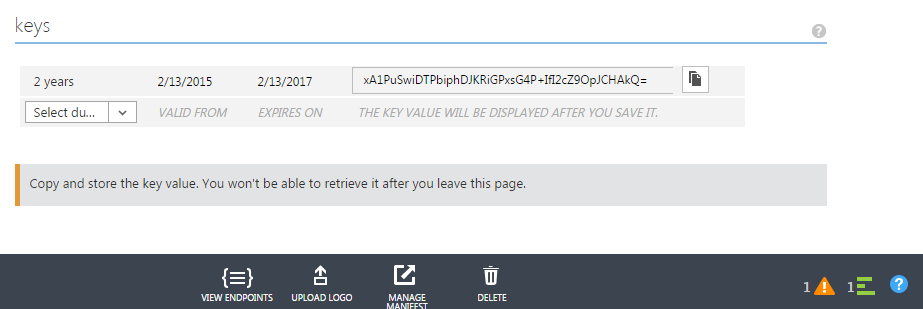
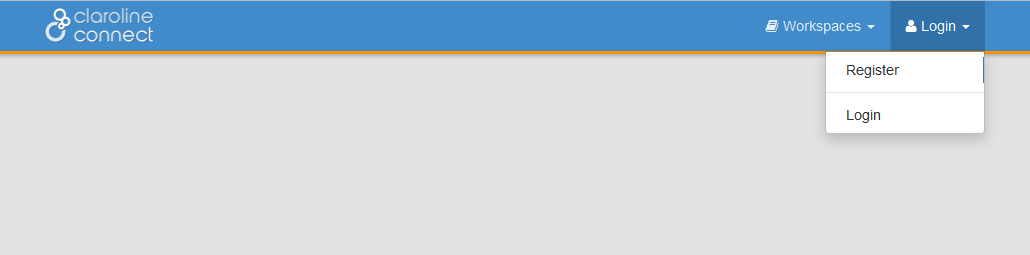


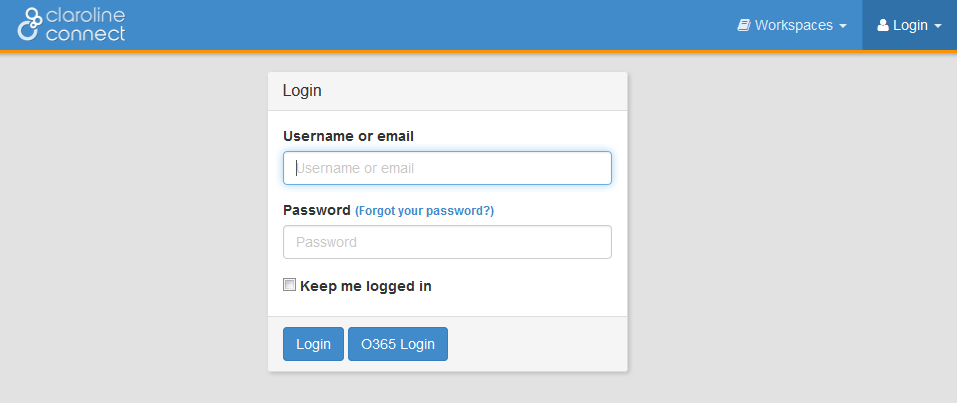
Image 10

# Connexion

Sur la page principale (correspondant à l’adresse [http://monadresse/app.php](http://monadresse/app.php/o365/login)), le menu de connexion permet d’accéder à la page d’authentification.

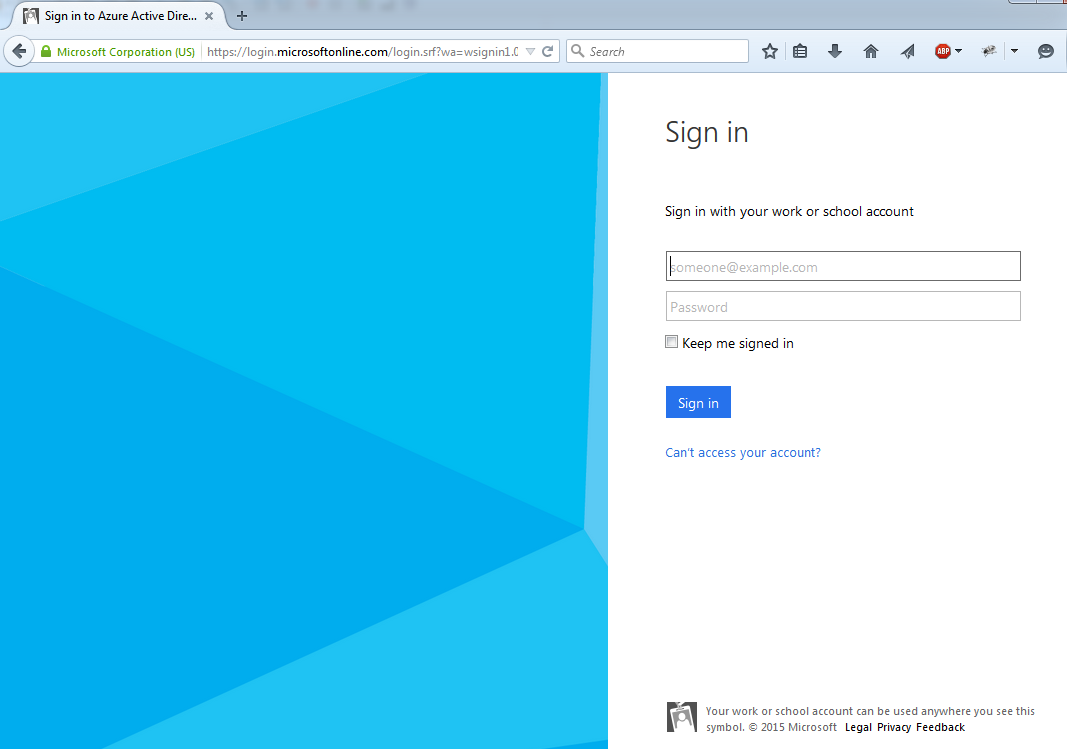


## Bouton de connexion



Sur cette page, un bouton a été ajouté pour permettre la connexion avec Office 365. En cliquant sur ce bouton, vous êtes redirigé vers Azure Active Directory lié à votre Office 365.

## Connexion Office 365



Introduisez vos informations de connexion, et validez-les. Une fois celles-ci, vous êtes redirigé vers Claroline.

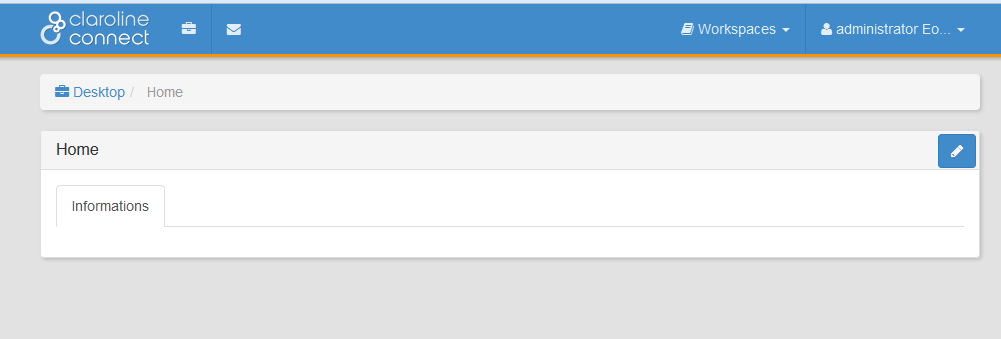
Remarque : votre adresse peut être une adresse de n’importe quel domaine de votre forêt Office 365. Par exemple, [claroline@mondomaineO365.be](mailto:claroline@mondomaineO365.be).

Si vous étiez déjà connecté sur Office 365, cette connexion sera utilisée et vous serez automatiquement redirigé vers Claroline.

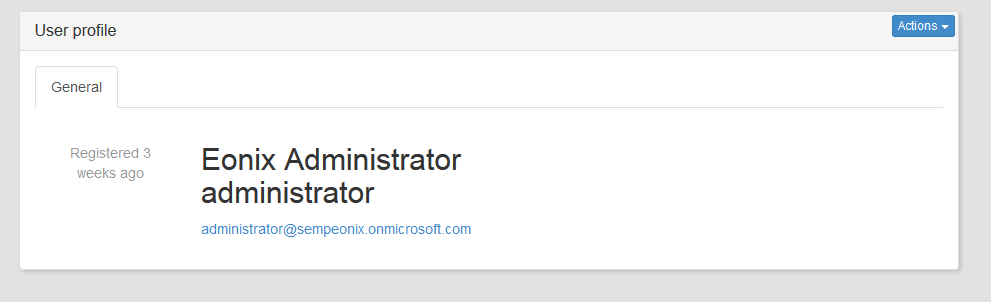
## Retour vers Claroline

Si votre utilisateur n’existe pas dans la base de données de Claroline et que les paramètres l’autorisent, l’utilisateur est automatiquement créé.

Ensuite, cet utilisateur est recherché en base de données, et chargé en session.



Vous voilà connecté à Claroline avec votre compte Office 365.



# Implémentation

Les explications présentées dans cette section seront à revoir, l’équipe de développement de Claroline compte effectuer une refonte du système de connexion prochainement, mais les fondements de cette connexion seront identiques.

## Paramètres

Une classe Settings sert de container pour les définir les paramètres qui seront utilisés pour se connecter à Office 365, et se présente comme suit :

<?php

namespace Claroline\CoreBundle\Library\Security\OfficeAuth;

class Settings

{

public static $clientId = 'cbbbdd52-XXXX-XXXX-XXXX-dbd97c40b03e';

public static $password = thisIsTheLongPassword=';

public static $redirectURI = 'http://monadresse.be/app.php/o365/login';

public static $appTenantDomainName = 'mondomaineO365.onmicrosoft.com';

public static $isAutoCreated = true;

public static $resourceURI = 'https://graph.windows.net';

public static $apiVersion = 'api-version=2013-11-08';

}

Les quatre premières informations sont à paramétrer :

* La propriété « clientId » définit dans l’application sur le site d’Azure (cf. Client ID et Key) ;
* La propriété « password » définit dans l’application sur le site d’Azure (cf. Client ID et Key) ;
* La propriété « redirectURI » définit l’URL vers laquelle Office 365 renverra le token certifiant la bonne authentification de l’utilisateur. Cette URL doit être définie correctement par Azure (cf. Sign-on URL et Application ID URI) ;
* La propriété « appTenantDomainName » : le nom de votre Office 365 ;
* La propriété « isAutoCreated » définit si l’utilisateur qui se connecte avec Office 365 est créé s’il n’existe pas en base de données.

## Graph

Ces paramètres seront utilisés lors de la construction du lien de la requête vers Office 365. Cette construction est réalisée à l’aide des classes AuthorizationHelperForGraph.php et GraphServiceAccessHelper.php, toutes deux situées dans le répertoire C:\wamp\www\claroline-git\vendor\claroline\core-bundle\Claroline\CoreBundle\Library\Security\OfficeAuth.

Ces méthodes contiennent les procédés de connexion à Azure Active Directory en utilisant la [Graph API pour Azure](https://msdn.microsoft.com/en-us/library/azure/hh974476.aspx).

### AuthorizationHelperForGraph

<?php

namespace Claroline\CoreBundle\Library\Security\OfficeAuth;

class AuthorizationHelperForGraph

{

// Get Authorization URL

public static function getAuthorizatonURL(){

$authUrl = "https://login.windows.net/common/oauth2/authorize". "?" .

"response\_type=code" . "&" .

"client\_id=" . Settings::$clientId . "&" .

"resource=" . Settings::$resourceURI . "&" .

"redirect\_uri=" . Settings::$redirectURI;

return $authUrl;

}

// Use the code retrieved from authorization URL to get the authentication

token that will be used to talk to Graph

public static function getAuthenticationHeaderFor3LeggedFlow($code)

{

$authenticationRequestBody =

"grant\_type=authorization\_code&client\_id=".

urlencode(Settings::$clientId)."&".

"redirect\_uri=".Settings::$redirectURI."&".

"client\_secret=".urlencode(Settings::$password)."&".

"code=".$code;

$ch = curl\_init();

$stsUrl = 'https://login.windows.net/common/oauth2/token';

curl\_setopt($ch, CURLOPT\_URL, $stsUrl);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

curl\_setopt($ch, CURLOPT\_POST, 1);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $authenticationRequestBody);

curl\_setopt($ch, CURLOPT\_SSL\_VERIFYPEER, false);

$output = curl\_exec($ch);

curl\_close($ch);

$tokenOutput = json\_decode($output);

$tokenType = $tokenOutput->{'token\_type'};

$accessToken = $tokenOutput->{'access\_token'};

$tokenScope = $tokenOutput->{'scope'};

$\_SESSION['token\_type']=$tokenType;

$\_SESSION['access\_token']=$accessToken;

$\_SESSION['tokenOutput'] = $tokenOutput;

}

}

### GraphServiceAccessHelper

<?php

namespace Claroline\CoreBundle\Library\Security\OfficeAuth;

class GraphServiceAccessHelper

{

// Constructs a Http GET request to fetch the entry for the current user.

// Returns the json decoded respone as the objects that were recieved in feed.

public static function getMeEntry(){

// initiaze curl which is used to make the http request

$ch = curl\_init();

// Add authorization and other headers. Also set some common settings.

self::addRequiredHeadersAndSettings($ch);

// Create url for the entry based on the feedname and the key value

$feedURL = self::getUri(true);

curl\_setopt($ch, CURLOPT\_URL, $feedURL);

//Enable fiddler to capture request

//curl\_setopt($ch, CURLOPT\_PROXY, '127.0.0.1:8888');

// $output contains the output string

$output = curl\_exec($ch);

// close curl resource to free up system resources

curl\_close($ch);

$jsonOutput = json\_decode($output);

return $jsonOutput;

}

// Add required headers like Authorization, Accept, Content-Type etc.

public static function addRequiredHeadersAndSettings($ch)

{

//Generate the authentication header

$authHeader = 'Authorization:' . $\_SESSION['token\_type'].' '.$\_SESSION['access\_token'];

curl\_setopt($ch,

CURLOPT\_HTTPHEADER,

array($authHeader,

'Accept:application/json;odata=minimalmetadata',

'Content-Type:application/json'));

// Set the option to recieve the response back as string.

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

// By default https does not work for CURL.

curl\_setopt($ch, CURLOPT\_SSL\_VERIFYPEER, false);

}

Et également la méthode statique permettant de générer les adresses de connexion :

private static function getUri($withTenant)

{

if ($withTenant){

return Settings::$resourceURI.

"/".Settings::$appTenantDomainName.

"/me/?".Settings::$apiVersion ;

}

else {

return Settings::$resourceURI."/me?".Settings::$apiVersion;

}

}

}

Ces méthodes seront appelées durant le processus de connexion, par le contrôleur, que nous décrivons plus loin.

## Bouton de connexion

L’ajout du bouton de connexion spécialement prévu pour la connexion Office 365 a été réalisé à l’aide du code suivant, dans le fichier login.html.twig, situé dans le répertoire C:\wamp\www\claroline\vendor\claroline\core-bundle\Claroline\CoreBundle\Resources\views\Authentication :

<a class="btn btn-primary o365" href="<http://monadresse.be/app.php/o365/token>">

O365 Login

</a>

Le lien vers lequel le bouton de connexion redirige est ici placé en dur dans un but de démonstration et démontre l’utilisation du contrôleur Office 365.

## Contrôleur

Ce contrôleur est une classe nommée O365Controller, héritant de la classe Controller. Elle a été créée afin de gérer les méthodes de connexion et de réception du token.

L’effet de ce contrôleur est qu’il est appelé à l’aide de l’adresse web.

Afin de mieux comprendre l’utilisation de ce type de classe, nous présenterons la définition de cette classe très simplement, nous continuerons par les étapes servant à la paramétrer et nous terminerons ce point par la description de ses méthodes.

### O365Controller

Cette classe a été créée spécialement pour gérer :

* La connexion via Office 365 et la réception du token d’authentification ;
* La requête d’information sur l’utilisateur courant ;

Celle-ci hérite de la classe Controller et est définie comme suit :

<?php

namespace Claroline\CoreBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

use Symfony\Component\Security\Core\Authentication\Token\UsernamePasswordToken;

use Sensio\Bundle\FrameworkExtraBundle\Configuration as EXT;

use Claroline\CoreBundle\Library\Security\OfficeAuth\AuthorizationHelperForGraph;

use Claroline\CoreBundle\Library\Security\OfficeAuth\Settings;

use Claroline\CoreBundle\Library\Security\OfficeAuth\O365ResponseUser;

use Claroline\CoreBundle\Library\Security\OfficeAuth\GraphServiceAccessHelper;

use Symfony\Component\HttpFoundation\RedirectResponse;

use Claroline\CoreBundle\Entity\User;

use Claroline\CoreBundle\Library\Security\PlatformRoles;

class O365Controller extends Controller

{

// code

}

Elle se nomme O365Controller.php et situe dans le répertoire C:\wamp\www\claroline-git\vendor\claroline\core-bundle\Claroline\CoreBundle\Controller.

Cette classe contient deux méthodes, que nous détaillerons après avoir expliqué comment paramétrer ce contrôleur dans le projet Claroline.

### Définition du routage

Ce contrôleur a été défini avec l’extension /o365 dans la définition des routes dans PHP. Cette définition a été réalisée dans le fichier routing.yml situé dans le répertoire C:\wamp\www\claroline\vendor\claroline\core-bundle\Claroline\CoreBundle\Resources\config., comme suit :

claro\_o365:

ressource: "@ClarolineCoreBundle/Controller/O365Controller.php"

prefix: o365

type: annotation

L’annotation /o365 permet donc d’accéder au contrôleur. Comme par exemple, à l’aide de l’adresse <http://monadresse.be/app.php/o365>.

### Définition des services

Ce contrôleur a également été défini comme service dans le fichier services.yml situé dans le répertoire C:\wamp\www\claroline\vendor\claroline\core-bundle\Claroline\CoreBundle\Resources\config.

# add controller for O365

claroline.o365controller:

class: Claroline\CoreBundle\Controller\O365Controller

public: true

Cette configuration permet d’utiliser cette classe comme un service (concept lié au Framework Symfony).

### Méthode de réception du token

Dans la classe O365Controller se trouve une première méthode, servant à récupérer l’adresse en provenance de la classe AuthorizationHelperForGraph (laquelle est reconstruite à l’aide des paramètres de la classe Settings que nous avons vus précédemment) et rediriger l’utilisateur vers cette adresse.

/\*\*

\* @EXT\Route(

\* "/token",

\* name="claro\_o365\_get\_token"

\* )

\*

\* @return Response

\*/

public function getTokenAction()

{

$url = AuthorizationHelperForGraph::getAuthorizatonURL();

return new RedirectResponse($url);

}

#### Routage

Ici, une extension du routage a été définie en /token, et la méthode a été nommée en service par claro\_o365\_get\_token.

Cela signifie que cette méthode sera appelée à l’aide de l’adresse /o365/token ([http://monadresse.be/app.php/o365](http://localhost/claroline-git/web/app_dev.php/o365)/token).

Cette adresse pourra également être recréée à l’aide du code suivant :

$this->generateUrl('claro\_o365\_get\_token')

### Méthode de connexion

Une deuxième méthode permet de récupérer les données de l’utilisateur connecté à Azure Active Directory.

/\*\*

\* @EXT\Route(

\* "/login",

\* name="claro\_o365\_login"

\* )

\*

\* @return Response

\*/

public function loginAction()

{

AuthorizationHelperForGraph::GetAuthenticationHeaderFor3LeggedFlow(

$\_GET['code']);

$jsonResponse = GraphServiceAccessHelper::getMeEntry();

$userResponse = new O365ResponseUser($jsonResponse);

$userManager = $this->get('claroline.manager.user\_manager');

$email = $userResponse->getEmail();

$user = $userManager->getUserByEmail($email);

if ($user === null)

{

$user = new User();

$user->setFirstName($userResponse->getNickname());

$user->setLastName($userResponse->getRealName());

$user->setUsername($userResponse->getEmail());

$user->setPlainPassword($userResponse->getEmail());

$user->setMail($userResponse->getEmail());

$roleName = PlatformRoles::USER;

$userManager->createUser($user, $roleName);

}

$userRepo = $this->get('doctrine.orm.entity\_manager')->getRepository('ClarolineCoreBundle:User');

$securityContext = $this->get('security.context');

$userLoaded = $userRepo->loadUserByUsername($user->getUsername());

$providerKey = 'main';

$token = new UsernamePasswordToken($userLoaded, $userLoaded->getPassword(), $providerKey, $userLoaded->getRoles());

$securityContext->setToken($token);

return new RedirectResponse($this->generateUrl('claro\_desktop\_open'));

}

Une fois ces informations récupérées, nous encapsulons le JSON récupéré et vérifions si l’utilisateur existe ou non en base de données sur base de l’adresse email :

$userResponse = new O365ResponseUser($jsonResponse);

$email = $userResponse->getEmail();

$user = $userManager->getUserByEmail($email);

Si l’utilisateur n’existe pas, nous le créons :

if ($user === null)

{

$user = new User();

$user->setFirstName($userResponse->getNickname());

$user->setLastName($userResponse->getRealName());

$user->setUsername($userResponse->getEmail());

$password = self::getRandomPassword(50);

$user->setPlainPassword($password);

$user->setMail($userResponse->getEmail());

$roleName = PlatformRoles::USER;

$userManager->createUser($user, $roleName);

}

Et la méthode privée (à déplacer dans une classe utilitaire) permettant de générer un mot de passe aléatoire dont la dimension est fournie en paramètre.

/\*

\* Generates a random password for created users.

\*/

private function getRandomPassword($length)

{

$characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';

$charactersLength = strlen($characters);

$randomString = '';

for ($i = 0; $i < $length; $i++) {

$randomString .= $characters[rand(0, $charactersLength - 1)];

}

return $randomString;

}

Nous chargeons ensuite l’utilisateur en session :

$userRepo = $this->get('doctrine.orm.entity\_manager')

->getRepository('ClarolineCoreBundle:User');

$securityContext = $this->get('security.context');

$userLoaded = $userRepo->loadUserByUsername($user->getUsername());

$providerKey = 'main';

$token = new UsernamePasswordToken($userLoaded,

$userLoaded->getPassword(),

$providerKey,

$userLoaded->getRoles());

$securityContext->setToken($token);

Nous terminons la connexion en redirigeant l’utilisateur vers la page d’accueil.

return new RedirectResponse($this->generateUrl('claro\_desktop\_open'));

#### Routage

Ici, une extension du routage a été définie en /token, et la méthode a été nommée en service par claro\_o365\_login.

Cela signifie que cette méthode sera appelée à l’aide de l’adresse /o365/login (http://monadresse.be/app.php/login) qui est l’adresse définie dans les paramètres d’Azure Active Directory et dans les paramètres de la classe Settings car elle sera envoyée à AAD comme adresse de redirection.

Cette adresse pourra également être recréée à l’aide du code suivant :

$this->generateUrl('claro\_o365\_login')

## Objet JSON

Afin de faciliter l’utilisation de l’objet JSON qui contient les informations de l’utilisateur, nous l‘avons encapsulé dans un objet PHP :

<?php

namespace Claroline\CoreBundle\Library\Security\OfficeAuth;

class O365ResponseUser{

private $responseObj;

public function \_\_construct($jsonUser){

$this->responseObj = $jsonUser;

}

public function getUsername(){

return $this->responseObj->{'mail'};

}

public function getResponse() {

return $this->responseObj;

}

public function getEmail() {

return $this->responseObj->{'mail'};

}

public function getNickname() {

return $this->responseObj->{'mailNickname'};

}

public function getRealName() {

return $this->responseObj->{'displayName'};

}

}