1 Objectif

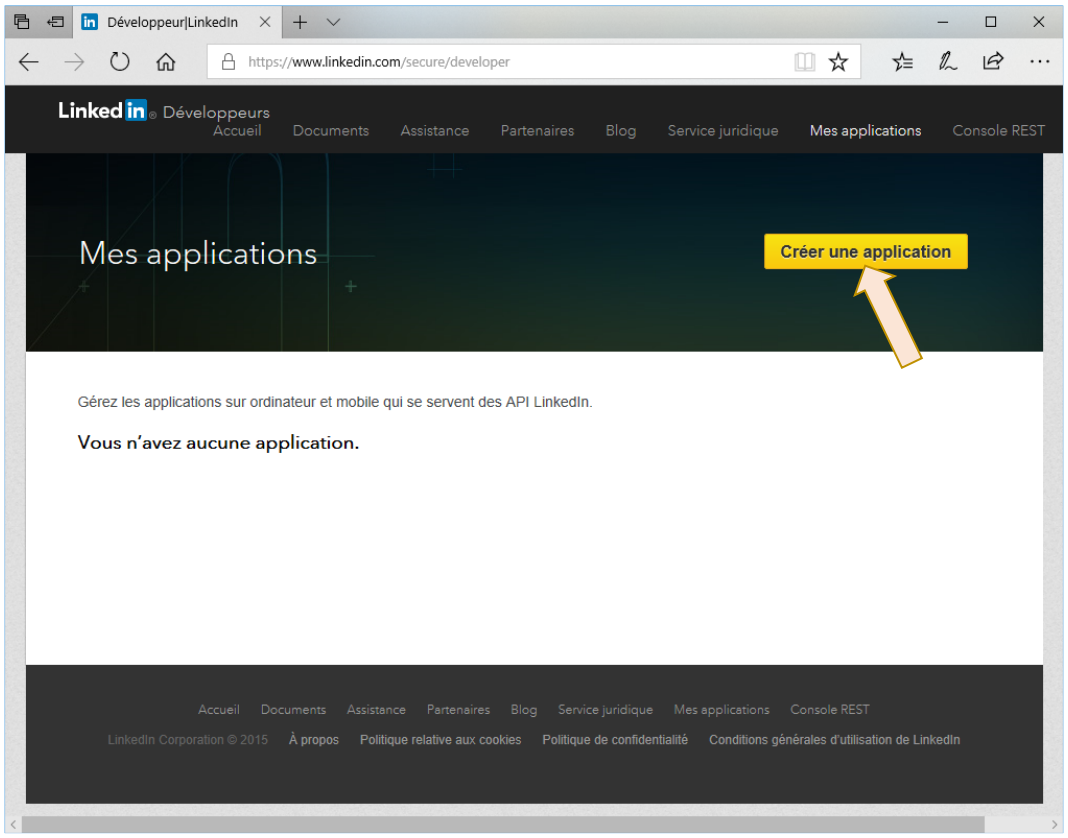
Utilisation de l'API de LinkedIn pour accéder aux données relatives aux profils professionnels sous Python. Le package python-linkedin pour Python.

LinkedIn est un réseau social professionnel. Il permet de mettre en ligne notre CV, établir des contacts, entretenir des relations, accéder ou diffuser des offres d'emploi, chercher de l'aide, indiquer ses aspirations, etc. C'est un instrument privilégié pour construire notre identité virtuelle professionnelle. L'outil est particulièrement prisé des recruteurs (Wikipédia).

Dans ce tutoriel, nous étudierons comment écrire un programme en Python permettant d'accéder aux fonctionnalités de l'API LinkedIn via le package python-linkedin. J'ai utilisé un compte LinkedIn fictif. Il s'agit avant tout de montrer la faisabilité de la chose. Nous nous attarderons sur deux étapes essentielles du dispositif : l'activation de l'API LinkedIn qui n'est pas évidente ; le package n'est pas inclus dans la distribution standard Anaconda (Python 3.6) que j'utilise, il m'a fallu l'installer et surtout le configurer spécifiquement pour qu'il fonctionne.

2 Activation de l'API LinkedIn

Création d'une application – Etape 1.



Il faut disposer d'un compte LinkedIn. Nous nous connectons sur le site dédié aux développeurs (https://www.linkedin.com/secure/developer). Nous devons créer et configurer une application.

La création d'une application nous délivre un **ID CLIENT** et un code **SECRET CLIENT**. Ces informations sont confidentielles. Elles sont masquées dans la copie d'écran.

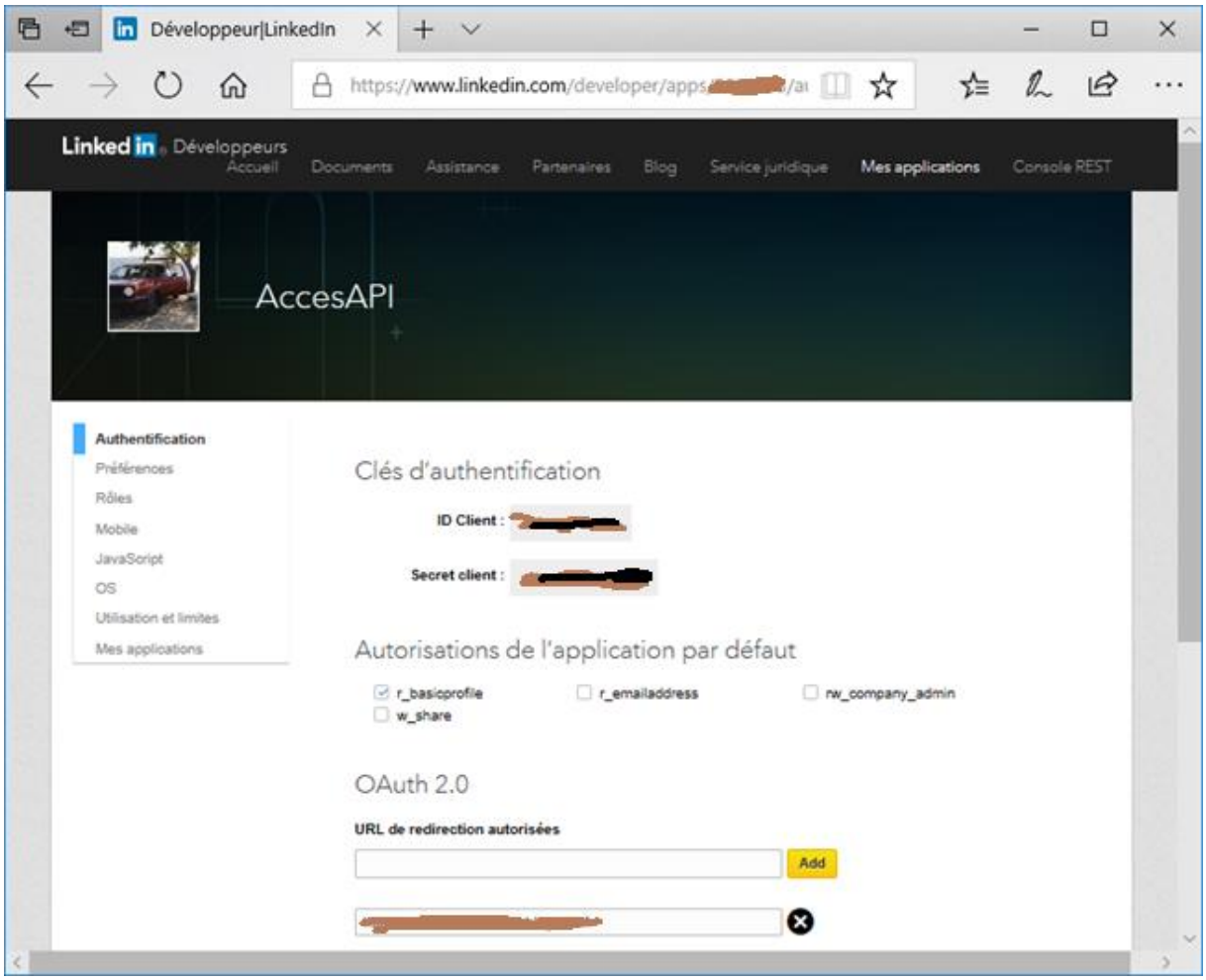


Figure 1 - ID CLIENT et SECRET CLIENT

Récupération d'un code d'autorisation – Etape 2. Les accès à l'API sont authentifiés avec le mécanisme OAuth. Nous utilisons OAuth 2.0 dans ce tutoriel. Nous devons indiquer à cet effet une URL (**URL de redirection autorisées**) permettant de rediriger les informations qui nous seront transmises par LinkedIn (code d'autorisation et token). L'adresse que j'ai utilisée est également masquée dans la copie d'écran ci-dessus. Nous devons, c'est impératif, indiquer une URL valide. La démarche complète est détaillée sur le site « **Authenticating with OAuth 2.0** » : https://developer.linkedin.com/docs/oauth2.

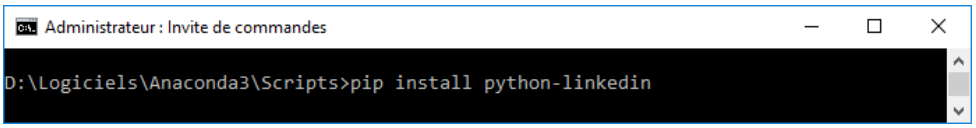
Nous en sommes à l'étape 2 à ce stade (Step 2 : Request an Authorization Code). Nous sollicitons le site https://www.linkedin.com/oauth/v2/authorization en lui passant une série d'informations, notamment notre ID CLIENT (appelé également API KEY). Il nous renvoie – sur notre URL de redirection – le code d'autorisation. Attention, sa durée de vie est très courte, nous devons l'exploiter très rapidement dans l'étape suivante.

Récupération d'un token (code d'authentification) – Etape 3. Cette étape (Step 3 — Exchange Authorization Code for an Access Token) consiste à convertir le code d'autorisation en code d'authentification (**Acess Token**) qui présente une durée de vie plus longue (60 jours). Il s'agit de solliciter le site https://www.linkedin.com/oauth/v2/accessToken cette fois-ci, en lui passant d'autres types d'information, dont le code d'autorisation obtenu à l'étape précédente, l'URL de redirection, le CLIENT ID et le mot de passe SECRET CLIENT (API SECRET).

3 Installation du package linkedin sous Python

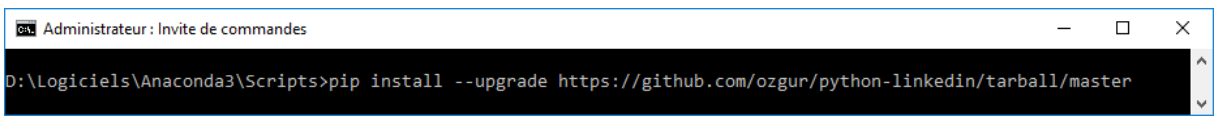
J'utilise la distribution Anaconda pour Windows (Anaconda 5.0.1 pour Python 3.6). Cette précision est importante car le package linkedin (https://pypi.python.org/pypi/python-linkedin/4.0) n'est pas installé par défaut.

Installation du package. L'installation du package ne pose pas de difficultés particulières. Nous ouvrons un terminal de commande sous Windows (invoqué en mode administrateur en ce qui me concerne). Nous allons dans le dossier « Scripts » d'Anaconda, nous faisons appel à l'utilitaire d'installation **pip**.



Modification du package. En l'état, le package semble fonctionner sous Python 2, mais pose problème sous Python 3. J'avoue y avoir passé un petit moment. En fait, il faut apporter un correctif dont j'ai réussi à trouver la description sur un forum (ouf !) (https://stackoverflow.com/questions/29435205/error-when-importing-the-linkedin-api-in-python).

Nous insérons l'instruction suivante dans le terminal de commande, toujours avec l'utilitaire **pip**.



Maintenant, et maintenant seulement, nous pouvons travailler sous Python.

4 Piloter LinkedIn sous Python

Importation de la librairie. Dans notre code Python, nous commençons en important la classe `linkedin` du package éponyme. Nous en affichons les membres.

```
#importation de la librairie et de la classe
from linkedin import linkedin

#membres de la classe linkedin
print(dir(linkedin))
```

Nous obtenons :

```
['AccessToken',
 'ENDPOINTS',
 'LinkedInApplication',
 'LinkedInAuthentication',
 'LinkedInDeveloperAuthentication',
 'LinkedInError',
 'LinkedInInvitation',
 'LinkedInMessage',
 'LinkedInSelector',
 'NETWORK_UPDATES',
 'OAuth1',
 'PERMISSIONS',
 'StringIO',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'contextlib',
 'enum',
 'hashlib',
 'json',
 'quote',
 'quote_plus',
 'raise_for_error',
 'random',
 'requests',
 'to_utf8',
 'unicode_literals']
```

Accès à l'application. Pour accéder aux informations, nous devons instancier une application via la classe `LinkedInApplication()`. Nous lui passons en paramètre le token (ACESS TOKEN) obtenu sur le site de LinkedIn (section 2).

```
#token
ACCESS_TOKEN = '... utilisez votre token ...'
#instanciation d'une application
application = linkedin.LinkedInApplication(token=ACCESS_TOKEN)
```

```
#membres de l'application
print(dir(application))
```

Les propriétés et procédures sont nombreuses :

```
['BASE_URL',  '__class__',  '__delattr__',  '__dict__',  '__dir__',  '__doc__',
'__eq__',  '__format__',  '__ge__',  '__getattribute__',  '__gt__',  '__hash__',
'__init__',  '__init_subclass__',  '__le__',  '__lt__',  '__module__',  '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__',       '__subclasshook__',       '__weakref__',       'authentication',
'comment_on_update',     'comment_post',     'follow_company',     'get_companies',
'get_company_by_email_domain',    'get_company_products',    'get_company_updates',
'get_connections', 'get_group', 'get_job', 'get_job_bookmarks', 'get_memberships',
'get_network_status',         'get_network_update',         'get_network_updates',
'get_picture_urls', 'get_post_comments', 'get_posts', 'get_profile', 'join_group',
'leave_group',   'like_post',   'like_update',   'make_request',   'search_company',
'search_job',       'search_profile',       'send_invitation',       'send_message',
'submit_company_share', 'submit_group_post', 'submit_share', 'unfollow_company']
```

Accéder au profil. Nous utilisons la commande `get_profile()` pour disposer des informations basiques relatives à notre profil.

```
#obtention du profil
application.get_profile()
```

Je le répète encore une fois, ce profil n'existe pas en réalité…

```
{'firstName': 'Ricco',
 'headline': 'Data Scientist chez Master SISE',
 'id': 'xtNZ_T3sFn',
 'lastName': 'Rakotomalala',
 'siteStandardProfileRequest':                                          {'url':
'https://www.linkedin.com/profile/view?id=...&authType=name&authToken=...&trk=api*
...*...*'}}
```

Remarque 1 : Nous nous en tiendrons à ce simple exercice de style dans ce tutoriel. En effet, nous avions uniquement spécifié la lecture du profil basique (`r_basicprofile`) lors de la demande du token d'authentification (Figure 1). Si nous souhaitons aller plus loin, il nous faudrait reproduire la démarche (de demande de token) en indiquant une palette plus large de permissions. Cela doit pouvoir se faire par programme comme le détaille l'auteur du package (https://pypi.python.org/pypi/python-linkedin/4.0). Par rapport à la création manuelle du token (section 2), les étapes sont schématiquement les mêmes : demande d'authentification à l'aide du CLIENT ID et CLIENT SECRET, obtention d'un code temporaire qui permet de créer le token.

Remarque 2 : Néanmoins, je n'ai pas moi-même testé cette possibilité. Je crains d'ailleurs qu'il y ait des problèmes si nous cherchons à le faire. Il semble en effet que LinkedIn ait sérieusement limité les possibilités pour les développeurs, sauf à adhérer à leur programme de partenariat (https://developer.linkedin.com/blog/posts/2015/developer-program-changes).

5 Conclusion

L'API LinkedIn nous donne accès par programme aux fonctionnalités du réseau social professionnel LinkedIn. Dans ce tutoriel, nous montrons comment les exploiter dans un programme Python via le package python-linkedin d'Ozgur Vatansever.

L'idée était avant tout de montrer la faisabilité de la chose. Elle est susceptible de nous ouvrir des perspectives particulièrement enthousiasmantes en tant que *data miner*. En effet, comme tout réseau social, les données s'enrichissent au fil de notre utilisation. Les interactions se multiplient. Forcément, au bout d'un certain temps, certaines régularités d'usage apparaissent, correspondant à des comportements types (ex. quels sont les caractéristiques clés des personnes qui constituent mon réseau, etc.). Le traitement statistique de ces informations nous permet de mieux comprendre ces phénomènes, et éventuellement d'en tirer parti. Nous nous situons vraiment dans le cadre typique de la *data science* (science des données) où les outils numériques récents d'interaction sociale (on parle aussi de Web Social) sont producteurs de données inexistantes avant l'avènement du web, et nous offrent de nouvelles opportunités d'analyse.

6 Références

Russel M.A., « Mining the Social Web », O'Reilly, 2014 ; chapitre 3.

Vatansever O., « python-linkedin 4.0 », https://pypi.python.org/pypi/python-linkedin/4.0

Wikipédia, « LinkedIn », consulté le 16 décembre 2017.