



Web Scraping Instagram avec Selenium et l'implémentation dans Django

Réalisé par :

EL MAHFOUD RADOUANE
NOUALI TAHA
AMBAKANE DIT BREHIMA GUINDO

Encadré par :

Kamal Idrissi

Filière : Data Engineer

2022-2023



Contents

1 Introduction

1.1	Web Scraping	3
1.2	Outils Web Scraping	5

2 Web Scraping sous Selenium Python

2.1	La bibliothèque Selenium	6
2.2	Documentation du Selenium	8

3 Projet : Web Scraping Instagram avec Selenium

3.1	Démo sur les Scripts selenium	15
3.2	Implémentation web : django	19





1. Web Scraping

Le *web scraping* (parfois appelé *harvesting*) est une technique d'extraction du contenu de **sites Web**, via un **script** ou un **programme**, dans le but de le transformer pour permettre son utilisation dans un autre contexte comme l'enrichissement de **bases de données**, le **référencement** ou l'**exploration de données**.

Le Web scraping sélectionne des textes sur les sites Internet afin d'obtenir des informations et de les enregistrer. Ce processus est comparable à un copier-coller automatique. Pour la recherche d'images, la dénomination du processus est encore plus précise et s'intitule image scraping.

Le scraping comprend différentes fonctionnalités, mais on opère généralement une distinction entre le scraping manuel et automatique. Le scraping manuel désigne le fait de copier et insérer manuellement des informations et des données. On peut le comparer avec le fait de découper et rassembler des articles de journaux. Le scraping manuel est uniquement effectué lorsque l'on souhaite trouver et enregistrer des informations de façon sporadique. Il s'agit d'un processus très laborieux qui est rarement appliqué pour de grandes quantités de données.

Dans le cas du scraping automatique, on utilise un logiciel ou un algorithme qui explore plusieurs sites Internet afin d'extraire des informations. Un logiciel spécifique est utilisé en fonction de la nature du site Internet et du contenu. Dans le scraping automatique, on distingue différentes méthodes :



- **Les analyseurs syntaxiques** : un analyseur syntaxique est utilisé pour convertir le texte en une nouvelle structure. Dans le cas de l'analyse d'un HTML par exemple, le logiciel lit le document HTML et enregistre les informations. L'analyse d'un **DOM** utilise l'affichage des contenus dans le navigateur côté client pour extraire les données.
- **Les robots** : un robot est un logiciel réalisant des tâches spécifiques et les automatisant. Dans le Web harvesting, les robots sont utilisés pour explorer automatiquement des sites Internet et collecter des données.
- **Le texte** : les personnes sachant utiliser la Command Line peuvent utiliser les instructions Unix **grep** pour explorer le Web à la recherche de certains termes dans Python ou Perl. Il s'agit d'une méthode très simple pour obtenir des données qui requiert toutefois davantage de travail que lorsqu'on utilise un logiciel.



2. Outils Web Scraping

Il existe de multiples frameworks et bibliothèques logicielles, certains reposent sur l'émulation d'une instance d'un **navigateur web** afin de réaliser des actions sur des pages web, d'autres frameworks et bibliothèques reposent sur l'analyse du code **HTML** de la page obtenu en réalisant une **requête HTTP**.

Les bibliothèques et frameworks les plus populaires pour le web scraping sont :

Nom	Langage
Beautiful soup	Python
Puppeteer	JavaScript (Node.js)
Goutte	PHP (Symfony)
Scrapy	Python
Selenium	Multiples
woob	Python
PhantomJS	JavaScript

L'utilisation d'**interfaces de programmation** est une bonne alternative aux bibliothèques et frameworks pour les **développeurs** souhaitant accélérer le développement de leurs applications de web scraping.

De nombreuses sociétés proposent des API de web scraping, généralement payantes, dont voici une liste non exhaustive des options les plus populaires :

- ScraperAPI
- ScrapingBee
- Scrapfly
- ScrapingFish
- Apify
- Bright Data
- Scraping bot
- Diffbot



1. La bibliothèque Selenium

Selenium est un framework développé en java, qui offre des passerelles pour s'exécuter avec différents langages comme Python et PHP. Il s'agit d'un outil puissant pour contrôler les navigateurs web grâce à des programmes et effectuer l'automatisation du navigateur. Il prend en compte tous les navigateurs, tous les principaux systèmes d'exploitation et ses scripts sont écrits dans différents langages comme Python, java, C# ...

Selenium vous permet de tester votre application web. En d'autres termes, il permet une automatisation efficace des tests de l'interface graphique des applications Web. Il est composé principalement de 4 composants à savoir : Selenium IDE, Selenium RC, Selenium Webdriver et Selenium GRID.

Selenium Python offre plusieurs avantages pour scraper les données d'un site web de façon structurée. L'apprendre est un grand atout si l'on souhaite effectuer du WebScraping. Les avantages de Selenium Python sont qu'il :

- est un framework open source et portable ;
- fonctionne avec de nombreux langages de programmation.
- peut être utilisé avec de nombreux navigateurs et plateformes différents.
- est alimenté par une grande communauté.
- peut explorer un site Web à l'aide d'un navigateur spécifique : bien que de nombreux logiciels de scraping



Web Scraping sous Selenium Python

de sites Web utilise un véritable navigateur Web pour l'extraction de données, dans la plupart des cas, le navigateur qu'ils utilisent est WebBrowser Control, c'est-à-dire Internet Explorer. Selenium, cependant, fonctionne non seulement avec Internet Explorer, mais aussi avec une variété de navigateurs tels que Google Chrome, Firefox, Opera, HtmlUnit et même Android et iOS.

- peut extraire des pages Web complexes au contenu dynamique : parfois, les données que vous devez extraire ne se trouvent pas dans le HTML brut que vous avez obtenu après avoir effectué une requête HTTP. Elles peuvent être générées de manière dynamique (en utilisant AJAX et JavaScript). Selenium Python permet l'extraction de données même qui sont générées dynamiquement.
- Selenium Python est capable de faire des captures d'écran de la page que vous scraper.



2. Documentation du Selenium sous Python

Selenium Python sous Windows

Si vous utilisez Windows, vous pouvez effectuer les étapes suivantes :

Etape #1 : installer Python

Tout d'abord nous allons installer python sur votre ordinateur. Cliquez sur le lien pour télécharger et installer python.

Etape #2 : installer Selenium WebDriver

Ensuite, installez Selenium WebDriver à l'aide de Pip, qui est le gestionnaire officiel de package de python.

Ouvrez votre invite de commande et saisissez le code :

```
pip install selenium
```

Etape #3 : intégrer Selenium python avec un navigateur

Après l'installation, il faut que vous téléchargiez un pilote qui s'intègre au navigateur de votre choix. Ce pilote permettra à Selenium de contrôler le navigateur et d'automatiser les commandes que vous écrivez dans vos scripts. Ici on va utiliser le pilote de chrome.



Web Scraping sous Selenium Python

Après le téléchargement, vous devez dézipper le fichier et placer l'exécutable sur votre stockage à un endroit de votre choix.

Après cela, copiez le chemin vers l'exécutable que vous ajoutez à votre variable d'environnement.

Etape #4 : vérifier les installations

Ensuite, vous pouvez effectuer un test dans votre invite de commande pour vérifier. Comme nous avons utilisé chrome, dans l'invite de commande

```
Administrateur : Invite de commandes - chromedriver
D:\>chromedriver
Starting ChromeDriver 94.0.4606.41 (333e85df3c9b656b518b5f1add5ff246365b6c24-refs/branch-heads/4606@{#845}) on port 9515
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
```

Etape #5 : installer Visual Studio code (par exemple)

Après cela, vous pouvez installer **Visual Studio Code**.

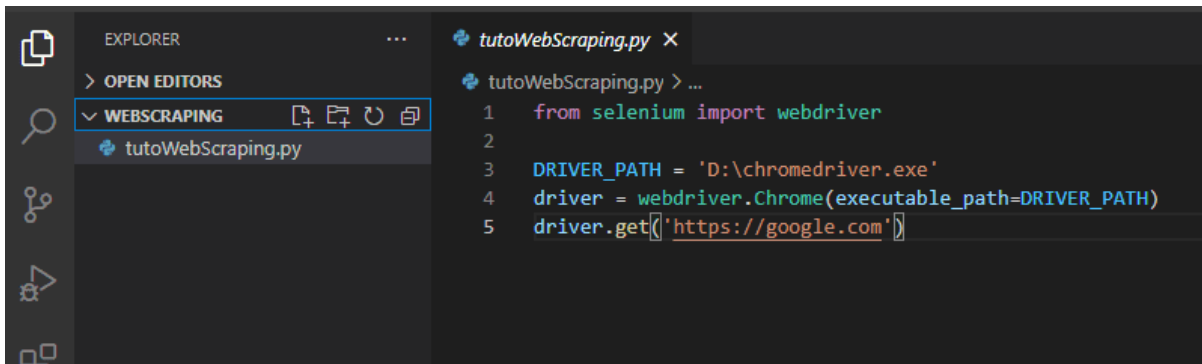
Etape #6 : créer un projet

Pour scrapper un site web avec Selenium Python, nous avons essentiellement 6 étapes à suivre :

- Trouver l'URL du site à scrapper ;
- Inspecter la page ;
- Trouver les données que nous voulons extraire ;
- Coder le script de scraping ;
- Exécuter le script et extraire les données ;

Web Scraping sous Selenium Python

- Stocker les données sous le format requis.

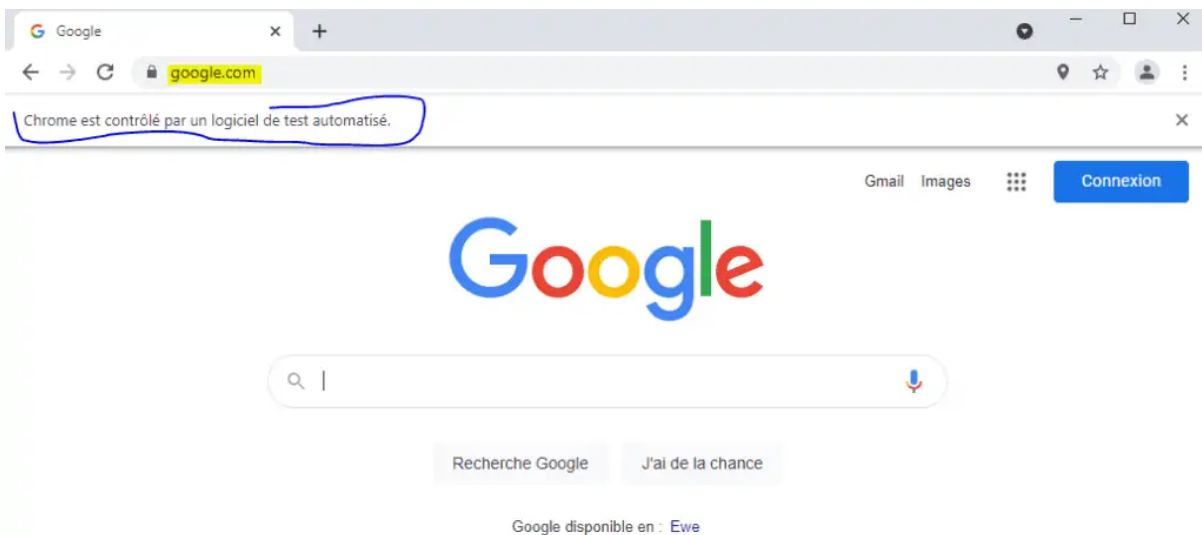


```
EXPLORER
> OPEN EDITORS
  WEBSCRAPING
    tutoWebScraping.py

tutoWebScraping.py x
tutoWebScraping.py > ...
1  from selenium import webdriver
2
3  DRIVER_PATH = 'D:\chromedriver.exe'
4  driver = webdriver.Chrome(executable_path=DRIVER_PATH)
5  driver.get('https://google.com')
```

La ligne 1 nous permet d'importer le WebDriver. Le chemin vers l'exécutable de chrome est visible à la ligne 3. Nous voyons par la suite, à la ligne 4 le code qui permet de lancer le navigateur. Enfin, on demande la page de google à la ligne 5.

Après lancement de ce bout de code, nous aurons le lancement du navigateur :



Vous devez voir un message indiquant que le navigateur est contrôlé par un logiciel automatisé.

Etape #7 : localiser un élément à scraper



Web Scraping sous Selenium Python

Maintenant, il faut que nous localisions l'élément que nous voulons scraper. Il existe de nombreuses méthodes disponibles dans l'API Selenium pour sélectionner des éléments sur la page. On peut utiliser :

- Nom de la balise
- Nom du site web
- ID
- XPath
- Sélecteurs CSS

Le moyen le plus simple de localiser un élément est d'ouvrir vos outils de développement Chrome et d'inspecter l'élément dont vous avez besoin.

Donc, après lancement du navigateur allez sur le de lws, par exemple.

Il existe de nombreuses façons de localiser un élément dans le sélénium. Disons que nous voulons localiser la balise title sur notre site web. Il suffit d'ajouter à notre précédent code ce bout de code :

```
7 title = driver.find_element_by_name('title')
```

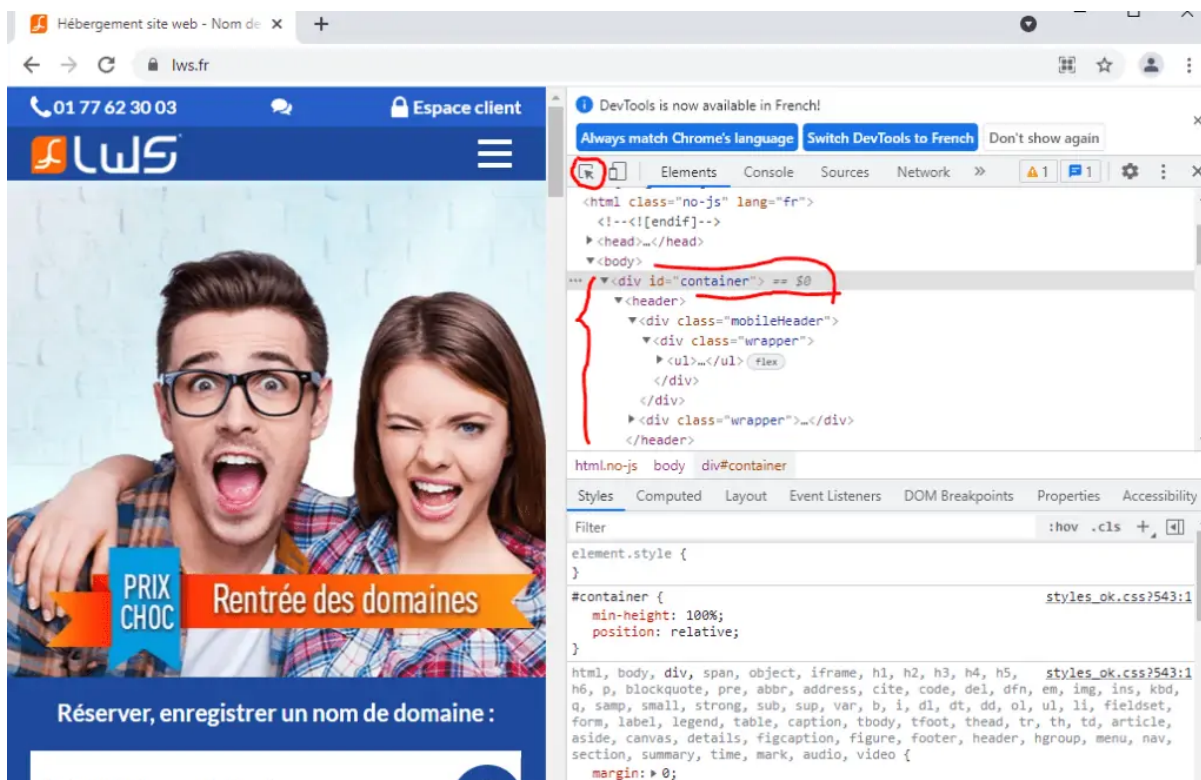
Après exécution, nous pouvons voir dans notre console :

```
\remote\webdriver.py", line 496, in find_element_by_name
\remote\webdriver.py", line 976, in find_element
\remote\webdriver.py", line 321, in execute
\remote\errorhandler.py", line 242, in check_response
t: {"method": "css selector", "selector": "[name='title']"}
```

Etape #8 : extraire des données

Jusque-là, nous avons juste localisé un élément. Maintenant, supposons que nous voulons extraire certaines données de notre site web. On fera l'exemple avec le site lws

En premier lieu, nous allons visiter le site et inspecter l'élément que nous allons extraire.



Supposons que nous voulons l'information contenue dans la balise `div` dont l'id est `container`. Nous allons utiliser le

XPath dans cet exemple pour localiser notre élément. C'est un moyen puissant d'extraire n'importe quel élément d'une page, en fonction de sa position absolue sur le DOM ou par rapport à un autre élément.

```
tutoWebScraping.py • tutoWebScraping2.py X
tutoWebScraping2.py > ...
1  from selenium import webdriver
2
3  DRIVER_PATH = 'D:\chromedriver.exe'
4  driver = webdriver.Chrome(executable_path=DRIVER_PATH)
5  driver.get('https://www.lws.fr/')
6
7  aPropos_details = []
8  aPropos_details = driver.find_elements_by_xpath('//*[@id="container"]')
9  for e in aPropos_details:
10     print(e.text)
```

Voici ce que fait notre code ci-dessus

De la ligne 3 à la ligne 5, nous l'avons expliqué plus haut. Ce qui a changé c'est juste le lien de la page visité (lws). Dans l'exemple précédent, c'était celui de google.

À la ligne 7 nous faisons une déclaration de tableau.

Ensuite, à la ligne 8, nous affectons les éléments de notre div grâce à la méthode `find_elements_by_xpath` de Sélénium en lui passant l'id de notre div. Les lignes suivantes, nous faisons juste un parcours du tableau pour afficher les données localisées.

Voici une partie des informations obtenues dans la console :



Web Scraping sous Selenium Python

```
DevTools listening on ws://127.0.0.1:60414/devtools/browser/00a65a57-048c-4ffd-8bb3-73933b64c1b1
Hebergeur web, nom de domaine, serveur VPS et serveur dédié - LWS
Avis clients :
Mon panier :
0 article(s)
Tél : 01 77 62 30 03
Espace client
Contact
L'hébergement web accessible à tous
PLUS DE 280 000 SITES HÉBERGÉS
DOMAINE
HÉBERGEMENT
SERVEUR
EMAILS
```

```
DevTools listening on ws://127.0.0.1:60414/devtools/browser/00a65a57-048c-4ffd-8bb3-73933b64c1b1
Hebergeur web, nom de domaine, serveur VPS et serveur dédié - LWS
Avis clients :
Mon panier :
0 article(s)
Tél : 01 77 62 30 03
Espace client
Contact
L'hébergement web accessible à tous
PLUS DE 280 000 SITES HÉBERGÉS
DOMAINE
HÉBERGEMENT
SERVEUR
EMAILS
```

Tout ce qui est texte sur notre page web contenu dans notre div est affiché dans la console.



1. Démo sur les Scripts

📁 > pweb

	Nom	Date	Type	Taille	Mots clés
	📁 django_web	02/01/2023 22:35	Dossier de fichiers		
	📁 videos	05/01/2023 00:29	Dossier de fichiers		
ements	📄 chromedriver	19/10/2022 19:46	Application	11 903 Ko	
is	📄 InfluencersList	01/01/2023 22:09	Fichier source Jup...	4 Ko	
	📄 instagram	02/01/2023 22:32	Fichier source Pyt...	2 Ko	
thon	📄 instagram_scraping	04/01/2023 22:24	Microsoft Excel C...	43 Ko	
	📄 names_dj	04/01/2023 00:05	Document texte	13 Ko	
	📄 username_scraped	03/01/2023 21:44	Microsoft Excel C...	18 Ko	

"InfluencerList.ipynb" c'est script pour scraping d'un site web en utilisant selenium pour extraire Top 1000 username des plus gros influenceurs Instagram au monde.

```
#Selenium imports here
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait
```




Web Scrapping sous Selenium Python

Login

```
#specify the path to chromedriver.exe (download and save on your computer)
driver = webdriver.Chrome('chromedriver.exe')
login_url='https://stargage.com/plus/en-us/login'
url='https://stargage.com/plus/en-us/influencer/ranking'
#open the webpage
driver.get(login_url)

email = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.CSS_SELECTOR,"input[type='email']")))
password = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.CSS_SELECTOR,"input[type='password']")))

in1=input("entrer ton email : ")
in2=input("entrer ton password : ")
email.clear()
email.send_keys(in1)
password.clear()
password.send_keys(in2)

button = WebDriverWait(driver, 2).until(EC.element_to_be_clickable((By.CSS_SELECTOR, "button[class='el-button rounded-pill btn-bl
```

```
names=[]
for k in range(10):
    try:
        url='https://stargage.com/plus/en-us/influencer/ranking?page='+str(k+1)
        driver.get(url)
        l=driver.find_elements(By.XPATH,('//a[@class="link color-pink text-break"]'))
        names=names+l
    except : pass

names=[k.text for k in names]
```

Python

```
print(len(names))
```

Python

1000

```
print(names)
```

Python

```
username=pd.DataFrame(names,columns=['username'])
```

Python

```
username.to_csv('username_scraped.csv')
```

Python

et voilà le résultat de ce script :

Web Scraping sous Selenium Python

```
InfluencersList.ipynb  username_scraped.csv X
username_scraped.csv
1 ,username
2 0,@cristiano
3 1,@leomessi
4 2,@kyliejenner
5 3,@selenagomez
6 4,@therock
7 5,@arianagrande
8 6,@kimkardashian
9 7,@beyonce
10 8,@khloekardashian
11 9,@justinbieber
12 10,@kendalljenner
13 11,@taylorswift
14 12,@jlo
15 13,@virat.kohli
16 14,@kourtneykardash
17 15,@nickiminaj
18 16,@neymarjr
19 17,@mileycyrus
20 18,@katyperry
```

Maintenant on passe au Script principal du Web Scraping avec Selenium sur la plateforme Instagram :

```
instagram.py
instagram.py > ...
1  ### importing Libraries
2
3  import undetected_chromedriver as uc
4  from selenium import webdriver
5  from webdriver_manager.chrome import ChromeDriverManager
6  from selenium.webdriver.common.by import By
7  import time
8  import pandas as pd
9
10 if __name__ == '__main__':
11
12     browser=uc.Chrome()
13
14     browser.maximize_window()
15
16     time.sleep(3)### waiting for three seconds
17
18     profiles=pd.read_csv('username_scraped.csv')#loading profiles already scraped
19     profiles=list(profiles["username"].astype("str"))
20     liste=map(lambda x: x.replace("@",""),profiles)
21
22     prf=list(liste)
```

Web Scrapping sous Selenium Python

```
for compte in prf:
    try:

        url = f"https://www.instagram.com/{compte}?hl=fr"

        #sending request to url
        browser.get(url)
        #extracting data from the source as this example using CSS_SELECTOR:
        """
[

|                                                               |   |               |        |   |            |      |   |                  |                                 |
|---------------------------------------------------------------|---|---------------|--------|---|------------|------|---|------------------|---------------------------------|
| '11M'                                                         | , | 'followers',' | 3,221' | , | 'suivis',' | 10K' | , | 'publications',' | -','Voir','les','photos','et',' |
| 'vidéos','Instagram','de','Heart','Evangelista','(@iamhearte) |   |               |        |   |            |      |   |                  |                                 |


"""
        data = browser.find_element(By.CSS_SELECTOR,("meta[property='og:description']"))
        text = data.get_attribute('content').split()

        ##getting the DATA


        username=text[-1][2:-1]#getting the username(@iamhearte) and removing @ and parentheses
        name=" ".join(text[-3:-1]) # getting and joining the name together
        number_followers=text[0] #extracting number of followers
        number_following=text[2]#extracting number of following
        number_publi=text[4]#extracting number of publications



## Loading the Data in a csv file
with open('instagram_scrapping.csv','a') as myFile:
    myFile.write(name +","+username+","+number_following+","+number_followers+","+number_publi+"\n")

except:
    pass
```

et voilà le résultat :

```
instagram_scrapping.csv X
instagram_scrapping.csv
1 name,username,Following,Followers,Posts
2 Cristiano Ronaldo,cristiano,533,567M,"3,416"
3 Leo Messi,leomessi,294,447M,980
4 Kylie ❤️,kyliejenner,96,407M,"7,035"
5 Selena Gomez,selenagomez,236,400M,"1,869"
6 Dwayne Johnson,therock,641,384M,"7,055"
7 Ariana Grande,arianagrande,605,371M,"4,988"
8 Kim Kardashian,kimkardashian,217,363M,"5,768"
9 Beyoncé,beyonce,0,312M,"2,054"
10 Khloé Kardashian,khloekardashian,111,317M,"4,205"
11 Justin Bieber,justinbieber,740,296M,"7,400"
12 Kendall,kendalljenner,237,291M,642
13 Taylor Swift,taylorswift,0,256M,563
14 Jennifer Lopez,jlo,"1,494",245M,220
15 Virat Kohli,virat.kohli,269,244M,"1,491"
16 Barbie,nickiminaj,671,221M,"6,451"
17 Kourtney Kardashian Barker,kourtneykardash,147,227M,"4,433"
18 NJ BR,neymarjr,"1,747",207M,"5,402"
19 Miley Cyrus,mileycyrus,317,206M,"1,202"
20 KATY PERRY,katyperry,754,198M,"2,104"
21 Zendaya,zendaya,"1,800",174M,"3,536"
22 Kevin Hart,kevinhart4real,971,173M,"8,480"
```



2. Implémentation web : django

On a créé tout d'abord un environnement virtuel : **"py_env"**
et aussi le projet django sous le nom **"project_python"**

Ce PC > Bureau > pweb > django_web

Nom	Modifié le	Type	Taille
project_python	04/01/2023 22:14	Dossier de fichiers	
py_env	02/01/2023 23:07	Dossier de fichiers	

et voilà notre application sous le nom **"myapp"**

Bureau > pweb > django_web > project_python

Nom	Modifié le	Type	Taille
myapp	03/01/2023 22:49	Dossier de fichiers	
project_python	02/01/2023 23:18	Dossier de fichiers	
db.sqlite3	03/01/2023 12:34	Fichier SQLITE3	128 Ko
manage	02/01/2023 23:15	Fichier source Pyt...	1 Ko

Bureau > pweb > django_web > project_python > myapp

Nom	Modifié le	Type	Taille
__pycache__	03/01/2023 23:40	Dossier de fichiers	
migrations	03/01/2023 00:22	Dossier de fichiers	
templates	04/01/2023 22:03	Dossier de fichiers	
__init__	03/01/2023 00:22	Fichier source Pyt...	0 Ko
admin	03/01/2023 00:22	Fichier source Pyt...	1 Ko
apps	03/01/2023 00:22	Fichier source Pyt...	1 Ko
models	03/01/2023 00:22	Fichier source Pyt...	1 Ko
tests	03/01/2023 00:22	Fichier source Pyt...	1 Ko
urls	03/01/2023 22:47	Fichier source Pyt...	1 Ko
views	03/01/2023 23:40	Fichier source Pyt...	2 Ko

Web Scraping sous Selenium Python

```
urls.py

django_web > project_python > myapp > urls.py > urlpatterns
1  from django.urls import path
2  from . import views
3
4  urlpatterns=[
5      path('table/',views.table,name="table"),
6  ]
```

```
views.py

django_web > project_python > myapp > views.py > ...
1  from django.shortcuts import render
2  import pandas as pd
3  import json
4  from django.contrib.auth.models import User
5  from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
6  def table(request):
7      df=pd.read_csv("C:/Users/user/Desktop/pweb/django_web/project_python/myapp/templates/instagram_scrapping.csv")
8      df.dropna(inplace=True)
9      # parsing the DataFrame in json format.
10     json_records = df.to_json(orient='records')
11     data = []
12     data = json.loads(json_records)
13
14     page = request.GET.get('page', 1)
15
16     paginator = Paginator(data, 90)
17     try:
18         data = paginator.page(page)
19     except PageNotAnInteger:
20         data = paginator.page(1)
21     except EmptyPage:
22         data = paginator.page(paginator.num_pages)
23     context = {'d': data}
24     return render(request,"table.html",context)
```

Bureau > pweb > django_web > project_python > project_python

Nom	Modifié le	Type	Taille
__pycache__	03/01/2023 22:51	Dossier de fichiers	
__init__	02/01/2023 23:15	Fichier source Pyt...	0 Ko
asgi	02/01/2023 23:15	Fichier source Pyt...	1 Ko
settings	03/01/2023 22:19	Fichier source Pyt...	4 Ko
urls	03/01/2023 22:51	Fichier source Pyt...	2 Ko
wsgi	02/01/2023 23:15	Fichier source Pyt...	1 Ko

```
urls.py
django_web > project_python > project_python > urls.py > ...
16 from django.contrib import admin
17 from django.urls import path, include
18
19
20 #dans localhost:8000
21 #directory here = project_python <=> localhost:8000
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('', include('myapp.urls')),
25 ]
26
```

```
urls.py settings.py X
django_web > project_python > project_python > settings.py > STATICFILES_DIRS
5/
58 TEMPLATES = [
59     {
60         'BACKEND': 'django.template.backends.django.DjangoTemplates',
61         'DIRS': [os.path.join(BASE_DIR, "myapp/templates")],
62         'APP_DIRS': True,
63         'OPTIONS': {
64             'context_processors': [
65                 'django.template.context_processors.debug',
66                 'django.template.context_processors.request',
67                 'django.contrib.auth.context_processors.auth',
68                 'django.contrib.messages.context_processors.messages',
69             ],
70         },
71     },
72 ]
73
```



Web Scrapping sous Selenium Python

```
<> table.html X
<> table.html > ...
162     <body>
163         <div style="background: url(img3.jpg)" class="jumbotron bg-cover text-white">
164             <div class="container py-5 text-center">
165                 </a>
166             </p>
167         </div>
168
169         <div class="container" >
170             <center><h1 color>Instagram Influencers</h1></center>
171             <table id="table">
172                 <thead>
173                     <tr>
174                         <th>name</th>
175                         <th>username</th>
176                         <th>Following</th>
177                         <th>Followers</th>
178                         <th>Posts</th>
179                     </tr>
180                 </thead>
181                 <tbody>
```

```

                    <tbody>
                    {% if d %}
                    {% for i in d %}
                    <tr>
                        <td>{{i.name}}</td>
                        <td>{{i.username}}</td>
                        <td>{{i.Following}}</td>
                        <td>{{i.Followers}}</td>
                        <td>{{i.Posts}}</td>
                    </tr>
                    {% endfor %}
                    {% endif %}
                    {% if d.has_other_pages %}

<center><ul class="pagination">
    {% if d.has_previous %}
        <a href="?page={{ d.previous_page_number }}">&laquo;</a>
    {% else %}
        <a class="disabled"><span>&laquo;</span></a>
    {% endif %}
    {% for i in d.paginator.page_range %}
        {% if d.number == i %}
            <a class="active"><span>{{ i }} <span class="sr-only">(current)</span></span></a>
        {% else %}
            <a href="?page={{ i }}">{{ i }}</a>
        {% endif %}
    {% endfor %}
    </ul>
```



```
        <a class="disabled"><span>&raquo;</span></a>
    {% endif %}
</ul>
{% endif %}
</center>
    </tbody>
</table>
</div>
</div>
</body>
</html>
```




Web Scrapping sous Selenium Python

Et voila notre site web :

localhost:8000/table/

Instagram Influencers

« 1 (current) 2 3 4 5 6 7 8 9 10 11 12 »

name	username	Following	Followers	Posts
Cristiano Ronaldo	cristiano	533	567M	3,416
Leo Messi	leomessi	294	447M	980
Kylie ♥	kyliejenner	96	407M	7,035
Selena Gomez	selenagomez	236	400M	1,869
Dwayne Johnson	therock	641	384M	7,055
Ariana Grande	arianagrande	605	371M	4,988

« 1 2 3 4 5 6 (current) 7 8 9 10 11 12 »

name	username	Following	Followers	Posts
Bretman (Da Baddest) Rock	bretmanrock	328	19M	1,793
Sidharth Malhotra	sidmalhotra	201	23M	1,150
Auron	auronplay	343	19M	802
LaTripleT	tinistoessel	974	21M	2,919
Mustafa Hosny مصطفى حسني	mustafahosnyofficial	12	20M	10K
Álvaro Morata	alvaromorata	2,073	20M	1,376