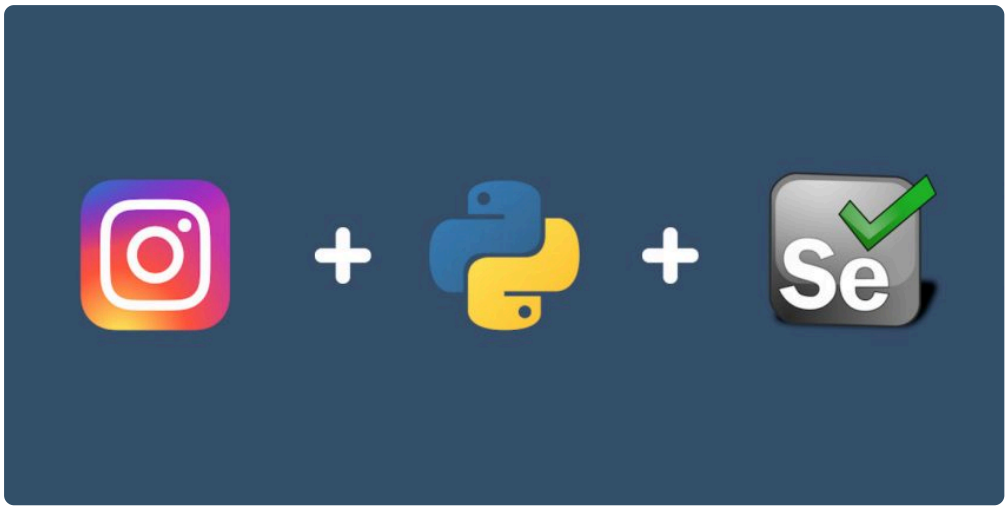


TUTORIAL

Scraping do Instagram com Python

30 março 2020 Post a comment

Já tentou extrair informações de algum site mas na hora de olhar o código fonte se deparou com algo parecido com isso?

Código fonte de um perfil do instagram

Isso acontece porque muitos sites hoje em dia, como o Instagram, utilizam *frameworks* em que o conteúdo não vem na requisição da página html, e sim por requisições futuras em javascript. Dessa maneira as páginas ficam mais "leves" e são carregadas mais rapidamente.

Então como obter essas informações? 🤔

Uma biblioteca como a requests nos dá apenas o código fonte da página, antes da execução do javascript. Precisamos de algo que

renderize a página como um navegador faz. Uma das opções mais utilizadas é a biblioteca Selenium!

> **Atenção**: Este artigo é de caráter educacional e não deve ser utilizado para obter conteúdos protegidos por direitos autorais.

Instagram

Perfil oficial do Instagram

Vamos ver então como obter as seguintes informações a partir de um nome de usuário utilizando Python, a biblioteca Selenium e o driver do Chrome:

- Números – publicações, seguidores e seguindo
- Informações dos posts – foto, localização e descrição

Nesse projeto, faremos algumas considerações:

- Os perfis acessados devem ser públicos assim não precisamos nos autenticar
- Usaremos a abordagem do melhor esforço (*best-effort*) para obter as informações porque alguns posts não possuem localização ou comentários
- Caso as informações não existam ou ocorra algum erro os atributos terão valores padrão

Instalação

Esta instalação foi testada e homologada no seguinte ambiente:

- Ubuntu 18.04
- Python 3.6
- Google Chrome 80.0
- Chrome Driver

Para instruções detalhadas da instalação você pode dar uma olhada no README.md deste projeto no github.

Código

Antes de mais nada, vamos usar o `ArgumentParser` para que possamos adicionar argumentos ao nosso script `main.py`.

```
from argparse import ArgumentParser

parser = ArgumentParser()
parser.add_argument('-u', '--username', dest='use
parser.add_argument('-d', '--debug', dest='debug'
args = parser.parse_args()
```

Criando modelos

Vamos criar duas classes para armazenar as informações de um perfil e de um post em um arquivo `models.py`.

Essas classes não são essenciais e você poderia guardar as informações em listas ou dicionários, mas a orientação a objetos sempre nos ajuda a manter um código mais limpo e organizado.

Criamos também um método `save()`, em cada classe, para salvar os dados em arquivos de texto.

```
class Profile:

    def __init__(self, username):
        self.username = username

        self.name = ''
        self.num_posts = 0
        self.num_followers = 0
        self.num_following = 0

        self.directory = 'data/' + username
```

```
        if not os.path.exists(self.directory):
            os.makedirs(self.directory)

        open('{}/posts.csv'.format(self.directory

    def save(self):

        with open('{}/data.csv'.format(self.direc

            f.write('"{}"; "{}"\n'.format('name',
            f.write('"{}"; "{}"\n'.format('num_po
            f.write('"{}"; "{}"\n'.format('num_fo
            f.write('"{}"; "{}"\n'.format('num_fo
```

```
class Post:

    def __init__(self, profile):
        self.profile = profile
        self.id_ = 0
        self.url = ''

        self.lat = 0
        self.lng = 0
        self.desc = ''

    def save(self):

        with open('{}/posts.csv'.format(self.prof
            f.write('"{}"; "{}"; "{}"; "{}"\n'.fo
                self.id_, self.lat, self.lng, sel
```

Instanciando o Chrome

Agora podemos inicializar uma instância do Chrome para obter as informações e acessar o perfil do usuário.

```
    options = Options()
    options.add_argument('--headless')
    options.add_argument('--window-size=1920x1080')
    driver = '/usr/bin/chromedriver'

    chrome = Chrome(
        chrome_options=options,
```

```
9       executable_path=driver
10  )
11
12  chrome.get('https://www.instagram.com/' + args.us
    if args.debug: print('Chrome running at ', chrome
```

Você vai reparar que temos alguns argumentos a serem passados. São eles:

- **headless**: para que o navegador inicialize sem uma interface, possibilitando rodar em um Linux Server;
- **window-size**: para que o site seja renderizado em modo desktop e todos os elementos que querermos estejam visíveis na página; e
- **driver**: o caminho para o chrome driver baixado anteriormente.

Bom, agora que já temos uma instância do chrome aberta no perfil do Instagram, vamos começar a selecionar os elementos HTML de onde queremos extrair informações.

O Selenium nos proporciona diversas maneiras de obter dados dos elementos (id, classe, nome, seletor CSS, XPath, ...).

Vamos utilizar os seletores CSS mas poderíamos escolher qualquer outro em que fosse possível localizar os elementos.

Função de copiar seletores do Google Chrome

Uma dica muito boa para ter uma ideia do seletor CSS é abrir o Google Chrome, na aba de inspecionar o código, clicar como botão direito e ir no menu `Copy`.

Uma nota sobre os seletores CSS: evite utilizar nomes de classes como `g47SY` ou `-vDIg` em seus seletores pois provavelmente elas são geradas automaticamente por algum framework e podem mudar

a qualquer momento, quebrando o seu código. Procure utilizar seletores formados pela estrutura hierárquica do site, pois ela tem menos chances de mudar ao longo do tempo.

Vamos ver os seletores dos objetos que queremos:

```
selectors = {
    'name': 'header h1',
    'num_posts': 'header ul li:nth-child(1) span'
    'num_followers': 'header ul li:nth-child(2) s
    'num_following': 'header ul li:nth-child(3) s

    'posts': 'main article a',
    'desc': 'article ul li[role=menuitem] div spa
    'img': 'main article > div img',
    'local': 'header a[href*=locations]',
    'lat': 'meta[property*=latitude]',
    'lng': 'meta[property*=longitude]',
}
```

Obtendo dados do perfil

Para obter os dados do perfil precisamos apenas dos seletores que já descobrimos para usar o método `chrome.find_element_by_css_selector()` e remover as virgulas, as strings “mil” e “milhão” para limpar os números.

```
    profile = Profile(args.username)

    name_el = chrome.find_element_by_css_selector(sel
    profile.name = name_el.text

    num_posts_el = chrome.find_element_by_css_selecto
    profile.num_posts = int(num_posts_el.text.replace

    num_followers_el = chrome.find_element_by_css_sel
    profile.num_followers = int(num_followers_el.text

    num_following_el = chrome.find_element_by_css_sel
    profile.num_following = int(num_following_el.text
```

```
    if args.debug: print('Saved', profile)
    profile.save()
```

Obtendo os posts

Aqui temos um problema, o Instagram só carrega os primeiros posts por padrão e para ver os próximos o usuário precisa rolar (*scroll*) a página para baixo. Então vamos precisar simular essa ação usando javascript.

Precisamos rolar a página até que o número de posts que aparecem seja igual ao número total que já obtemos.

```
urls = []
while len(urls) < profile.num_posts:

    if args.debug: print('Scroll down... ', end='
    chrome.execute_script('window.scrollTo(0, doc
    sleep(1)

    for a in chrome.find_elements_by_css_selector
        href = a.get_attribute('href')
        if href not in urls:
            urls.append(href)

    if args.debug: print('found', len(urls), 'of'
```

Note que ao mesmo tempo que novos posts aparecem conforme rolamos a página, os antigos vão sendo removidos. O Instagram provavelmente faz isso para deixar a página mais leve.

Agora que temos as url dos posts podemos acessar cada página e obter suas informações:

```
    for i, url in enumerate(urls):

        post = Post(profile)
        post.id_ = i
        post.url = url
```

```
        chrome.get(url)
        if args.debug: chrome.get_screenshot_as_file(

        # pega a descrição do post que é o primeiro c
        desc_el = chrome.find_elements_by_css_selecto
        if len(desc_el) > 0:
            post.desc = desc_el[0].text.replace('\n',

        # pega a imagem
        img_el = chrome.find_element_by_css_selector(
        post.download_img(img_el.get_attribute('src')

        # pega o link do local, e depois a latitude e
        local_el = chrome.find_elements_by_css_select
        if len(local_el) > 0:
            href = local_el[0].get_attribute('href')

            if 'locations' in href:
                chrome.get(href)
                if args.debug: chrome.get_screenshot_

                lat_metas = chrome.find_elements_by_c
                if len(lat_metas) > 0:
                    post.lat = lat_metas[0].get_attri

                lng_metas = chrome.find_elements_by_c
                if len(lng_metas) > 0:
                    post.lng = lng_metas[0].get_attri

        if args.debug: print('Saved', post)
        post.save()
```

Falta apenas implementarmos o método `download_img()` da classe `Post` que, como o nome já diz, faz o download da imagem.

```
class Post:

    def download_img(self, src):
        filename = '{}/{}.png'.format(self.profil
        urllib.request.urlretrieve(src, filename)
```

Por fim, não podemos esquecer de finalizar a execução do nosso driver do chrome:

```
1   chrome.quit()
```

Pronto! Agora já podemos testar nosso código!

```
$ python main.py --username=usuario --debug=True
```

Conclusão

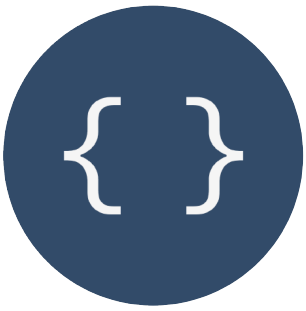
A biblioteca Selenium facilita, e muito, o trabalho de *scraping* em sites muito dinâmicos como o Instagram.

Tenha sempre em mente que quando estamos lidando com *scraping* de páginas da internet somos reféns de qualquer mudança ou atualização que possa ocorrer.

Por isso é importante sempre manter seu código atualizado e realizar testes.

Código

O código completo deste artigo está disponível no github



instagram-scraper
by umcodigo

0 Subscribers 0 Watchers 0 Forks

Check out this repository on GitHub.com

Referências

1. Documentação oficial do Selenium.
2. Download do Chrome Driver.

CSS Instagram JavaScript Python Selenium

Previous Post
Cloud9 – Porque usar uma Cloud IDE

Next Post
Obtenha preços de ações da bolsa de valores no Google Sheets

One thought on “Scraping do Instagram com Python”

Reply

Sheldon

23 fevereiro 2022 at 09:08

É possível requisitar dados internos como o insigth do Instagram , como seria para efetuar o log na conta

Deixe um comentário

O seu endereço de e-mail não será publicado. Campos obrigatórios são marcados com *

Comentário *

Nome *

E-mail *

Site

Salvar meus dados neste navegador para a próxima vez que eu comentar.

PUBLICAR COMENTÁRIO

ARITGOS RECENTES

Como utilizar o Vuex com classes TypeScript

25 agosto 2020

Autenticação com Google no Django

3 agosto 2020

Autenticação com Facebook no Django

20 julho 2020

REDES SOCIAIS

CATEGORIAS

Ferramenta (1)

Tutorial (9)

DESCONTOS

Um código - 2020