Instituto Federal Goiano – IF Goiano
Curso Bacharelado em Ciência da Computação
RPA e Web Scraping com Python e Selenium
Notas de Aula

05/09/2022 – Web Scraping com Selenium (Métodos Essenciais)

Prof. Jesmmer da Silveira Alves e-mail: jesmmer.alves@ifgoiano.edu.br

1. Introdução

Algumas ações são bastante comuns durante o processo de *web scraping* em um site, tais como: trocar de páginas; acessar janela modal; arrastar e soltar; rolar e redimensionar a página; e executar script. Nesta aula vamos ver como executar estas tarefas com o selenium.

2. Métodos Essenciais

Esta seção descreve a criação de alguns métodos dentro da classe Principal que usamos nas aulas anteriores. Veja na Figura 1 todas as importações necessárias para que todos métodos funcionem.

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver import ActionChains
from selenium.webdriver.common.action_chains import ActionChains
from selenium import webdriver
import time
import os
import tkinter
import tkinter.messagebox

class Principal(webdriver.Chrome):
    def __init__(self, driver_path=r"chromedriver.exe"):
        self.driver_path = driver_path
        os.environ['PATH'] += self.driver_path
        super(Principal, self).__init__()
        self.implicitly_wait(15)
        self.maximize_window()
```

Figura 1. Importações necessárias.

2.1 Trocando a página ou frame

É bastante comum precisar mudar de página durante o processo de *scraping*. No exemplo da Figura 2, o método do javascript *window.open*() é usado para abrir uma nova janela no navegador e o método *switch_to.window*() é usado para acessar outra janela no navegador. As janelas são identificadas com índices, o que facilita a troca de janelas.

```python
def trocando_pagina(self):
    self.get("https://www.google.com")
    self.execute_script("window.open()")
    self.switch_to.window(self.window_handles[1])
    self.get("https://www.youtube.com")
    time.sleep(5)
    self.execute_script("window.open()")
    self.switch_to.window(self.window_handles[2])
    self.get("https://python.org")
    self.switch_to.window(self.window_handles[1])
    time.sleep(5)
    self.switch_to.window(self.window_handles[0])
    time.sleep(5)
    self.quit()
```

Figura 2. Navegando nas páginas.

Um processo similar e bem comum é acessar uma janela modal. O processo é basicamente o mesmo, no entanto, primeiro identificamos o *iframe* e, em seguida, usamos o método *switch_to.frame*() para acessar uma janela modal. O exemplo da Figura 3 descreve este processo interagindo com uma janela modal no site *https://jqueryui.com/dialog/#modal-form*.

```python
def acessar_modal(self):
    self.maximize_window()
    self.get("https://jqueryui.com/dialog/#modal-form")
    iframe = self.find_element(By.XPATH,"//*[@id='content']/iframe")
    time.sleep(3)
    self.switch_to.frame(iframe)
    botao = self.find_element(By.XPATH, "//*[@id='create-user']")
    botao.click()
    nome = self.find_element(By.XPATH, "//*[@id='name']")
    nome.clear()
    nome.send_keys("Juaquina")
    email = self.find_element(By.XPATH, "//*[@id='email']")
    email.clear()
    email.send_keys("Juaquina@gmail.com")
    senha = self.find_element(By.XPATH, "//*[@id='password']")
    senha.clear()
    senha.send_keys("Juaquina123")
    criar_conta = self.find_element(By.XPATH, "/html/body/div[2]/div[3]/div/button[1]")
    time.sleep(3)
    criar_conta.click()
    time.sleep(3)
    self.close()
```

Figura 3. Acessando uma janela modal.

2.2 Executando um script

Em algum momento pode ser interessante executar um script no navegador que o robô está trabalhando. No selenium pode-se usar o método *execute_script*() para executar um código Javascript durante o processo de scraping. O exemplo na Figura 4 executa um *alert,* usa o método *switch_to.alert.accept*() para clicar no botão Ok da janela do alert e, em seguida, usa o método Javascript *window.open*() para abrir uma página em uma nova janela.

```
def executar_script(self):
    self.get("https://www.google.com")
    self.execute_script("alert('Mudando de página!')")
    time.sleep(3)
    try:
        WebDriverWait(self, 5).until (EC.alert_is_present())
        self.switch_to.alert.accept()
    except:
        print("Não achei o alert")
    self.execute_script("window.open('https://www.terra.com', 'new tab')")
    time.sleep(3)
    self.quit()
```

Figura 4. Executando um script.

A Figura 5 descreve outro exemplo utilizando o Javascript. Neste caso, o robô de software acessa o site do *wikipedia* e rola a página até o final.

```
def rolando_pagina(self):
    self.get("https://pt.wikipedia.org")
    self.execute_script("window.scrollTo(0,document.body.scrollHeight)")
    time.sleep(3)
    self.close()
```

Figura 5. Rolando a página.

2.3 Segurar e soltar

Vamos agora ver como segurar e arrastar elementos na página. Os métodos *click_and_hold*() e *drag_and_drop*() podem ser usados para esta função. Já o método *drag_and_drop_by_offset*() pode ser uma boa opção quando queremos definir a posição na tela para soltar o elemento (veja na Figura 6).

```
def segurar_e_soltar(self):
    self.maximize_window()
    self.get("https://jqueryui.com/droppable/")
    self.refresh()
    iframe = self.find_element(By.XPATH,"//*[@id='content']/iframe")
    self.switch_to.frame(iframe)
    source = self.find_element(By.XPATH, "//*[@id='draggable']")
    target= self.find_element(By.XPATH, "//*[@id='droppable']")
    action = ActionChains(self)
    #action.drag_and_drop(source, target).perform()
    action.click_and_hold(source).pause(3).move_to_element(target).release(target).perform()
    #action.drag_and_drop_by_offset(source, 200, 60).perform()
    self.close()
```

Figura 6. Arrastar e soltar.

2.4 Upload de arquivos

Fazer o upload de um arquivo em uma página web é uma tarefa bastante comum, no entanto, muito fácil. Primeiro precisamos identificar o endereço do arquivo. Para isto, copie o endereço do arquivo no sistema e salve em uma variável (trocando a "/" por "\\"). Depois identificamos o elemento na página com tipo "file", para enviar o endereço copiado. O Figura 7 descreve este procedimento.

```
def teste_upload(self):
    self.get('http://the-internet.herokuapp.com/upload')
    arquivo = 'C:\\Users\\silve\\OneDrive\\Área de Trabalho\\Cursos MAPA\\Aula 4\\logo-python.png'
    self.find_element(By.ID, 'file-upload').send_keys(arquivo)
    time.sleep(3)
    self.find_element(By.ID, 'file-submit').click()
```

Figura 7. Upload de um arquivo.

Instituto Federal Goiano – IF Goiano
Curso Bacharelado em Ciência da Computação
RPA e Web Scraping com Python e Selenium
Notas de Aula

3. Atividades

Crie um robô de software para acessar a plataforma SEI, consultar o processo '92.000114/2020-20' e anexar um arquivo *pdf* externo neste processo. Selecione as seguintes opções ao fazer o upload do arquivo externo:

- Tipo de documento: Anexo
- Data do Documento: 26/05/2022
- Formato: Nato-digital
- Nível de Acesso: Público

Acesse a plataforma SEI pelo endereço:

https://sipseiteste.fiocruz.br/login.php?sigla_orgao_sistema=FIOCRUZ&sigla_sistema=SEI

e utilize o usuário = teste e senha = teste.

Referências

BAIJU MUTHUKADAN. **Selenium with Python.** https://selenium-python.readthedocs.io/index.html

NumPy Developers. **Numpy API reference**. https://numpy.org/doc/stable/reference/index.html

PLIGHTBO, SIMON M., HBCCHAI, JRHUGGINS, *et al*. **Selenium 4.1.0 documentation**. https://www.selenium.dev/selenium/docs/api/py

The pandas development team. **Pandas API reference**. https://pandas.pydata.org/docs/index.html