**Bước 1: Cài đặt pip**

sudo apt-get update

sudo apt install python3-pip

**Bước 2: Cập nhật phiên bản pip mới nhất**

sudo pip install --upgrade pip

**Bước 3: Cài đặt thư viện Scrapy**

sudo -H pip install scrapy

Bước 4: Cài đặt scrapy-selenium

pip3 install scrapy-selenium

Bước 5: Cài đặt các thư viện cần thiết

pip3 install scrapy-selenium selenium scrapy webdriver-manager

Bước 6: Di chuyển đến thư mục trong phần location để thay đổi tập tin middlewares.py

cd /home/<tennguoidung>/.local/lib/python3.10/site-packages

cd scrapy\_selenium

vi middlewares.py

thay đổi nội dung file thành

"""This module contains the ``SeleniumMiddleware`` scrapy middleware"""

from importlib import import\_module

from scrapy import signals

from scrapy.exceptions import NotConfigured

from scrapy.http import HtmlResponse

from selenium.webdriver.support.ui import WebDriverWait

from .http import SeleniumRequest

class SeleniumMiddleware:

"""Scrapy middleware handling the requests using selenium"""

def \_\_init\_\_(self, driver\_name, driver\_executable\_path,

browser\_executable\_path, command\_executor, driver\_arguments):

"""Initialize the selenium webdriver"""

webdriver\_base\_path = f'selenium.webdriver.{driver\_name}'

driver\_klass\_module = import\_module(f'{webdriver\_base\_path}.webdriver')

driver\_klass = getattr(driver\_klass\_module, 'WebDriver')

driver\_options\_module = import\_module(f'{webdriver\_base\_path}.options')

driver\_options\_klass = getattr(driver\_options\_module, 'Options')

driver\_options = driver\_options\_klass()

if browser\_executable\_path:

driver\_options.binary\_location = browser\_executable\_path

for argument in driver\_arguments:

driver\_options.add\_argument(argument)

driver\_kwargs = {

'executable\_path': driver\_executable\_path,

f'{driver\_name}\_options': driver\_options

}

# locally installed driver

if driver\_executable\_path is not None:

self.driver = driver\_klass(\*\*driver\_kwargs)

# remote driver

elif command\_executor is not None:

from selenium import webdriver

capabilities = driver\_options.to\_capabilities()

self.driver = webdriver.Remote(command\_executor=command\_executor,

desired\_capabilities=capabilities)

# webdriver-manager

else:

from selenium import webdriver

from webdriver\_manager.chrome import ChromeDriverManager

from selenium.webdriver.chrome.service import Service as ChromeService

if driver\_name and driver\_name.lower() == 'chrome':

self.driver = webdriver.Chrome(options=driver\_options,

service=ChromeService(ChromeDriverManager().install()))

@classmethod

def from\_crawler(cls, crawler):

"""Initialize the middleware with the crawler settings"""

driver\_name = crawler.settings.get('SELENIUM\_DRIVER\_NAME')

driver\_executable\_path = crawler.settings.get('SELENIUM\_DRIVER\_EXECUTABLE\_PATH')

browser\_executable\_path = crawler.settings.get('SELENIUM\_BROWSER\_EXECUTABLE\_PATH')

command\_executor = crawler.settings.get('SELENIUM\_COMMAND\_EXECUTOR')

driver\_arguments = crawler.settings.get('SELENIUM\_DRIVER\_ARGUMENTS')

if driver\_name is None:

raise NotConfigured('SELENIUM\_DRIVER\_NAME must be set')

if (driver\_name.lower() != 'chrome') and (driver\_executable\_path is None and command\_executor is None):

raise NotConfigured('Either SELENIUM\_DRIVER\_EXECUTABLE\_PATH or SELENIUM\_COMMAND\_EXECUTOR must be set')

middleware = cls(

driver\_name=driver\_name,

driver\_executable\_path=driver\_executable\_path,

browser\_executable\_path=browser\_executable\_path,

command\_executor=command\_executor,

driver\_arguments=driver\_arguments

)

crawler.signals.connect(middleware.spider\_closed, signals.spider\_closed)

return middleware

def process\_request(self, request, spider):

"""Process a request using the selenium driver if applicable"""

if not isinstance(request, SeleniumRequest):

return None

self.driver.get(request.url)

for cookie\_name, cookie\_value in request.cookies.items():

self.driver.add\_cookie({

'name': cookie\_name,

'value': cookie\_value

})

if request.wait\_until:

WebDriverWait(self.driver, request.wait\_time).until(request.wait\_until)

if request.screenshot:

request.meta['screenshot'] = self.driver.get\_screenshot\_as\_png()

if request.script:

self.driver.execute\_script(request.script)

body = str.encode(self.driver.page\_source)

request.meta.update({'driver': self.driver})

return HtmlResponse(

self.driver.current\_url,

body=body,

encoding='utf-8',

request=request

)

def spider\_closed(self):

"""Shutdown the driver when spider is closed"""

self.driver.quit()

Bước 7: Clone project

Cd ~

git clone -b dev\_Vuong <https://github.com/Jucky901/BigData_Food.git>

Bước 8: Bắt đầu đào

scrapy crawl shopee -o doAn.csv