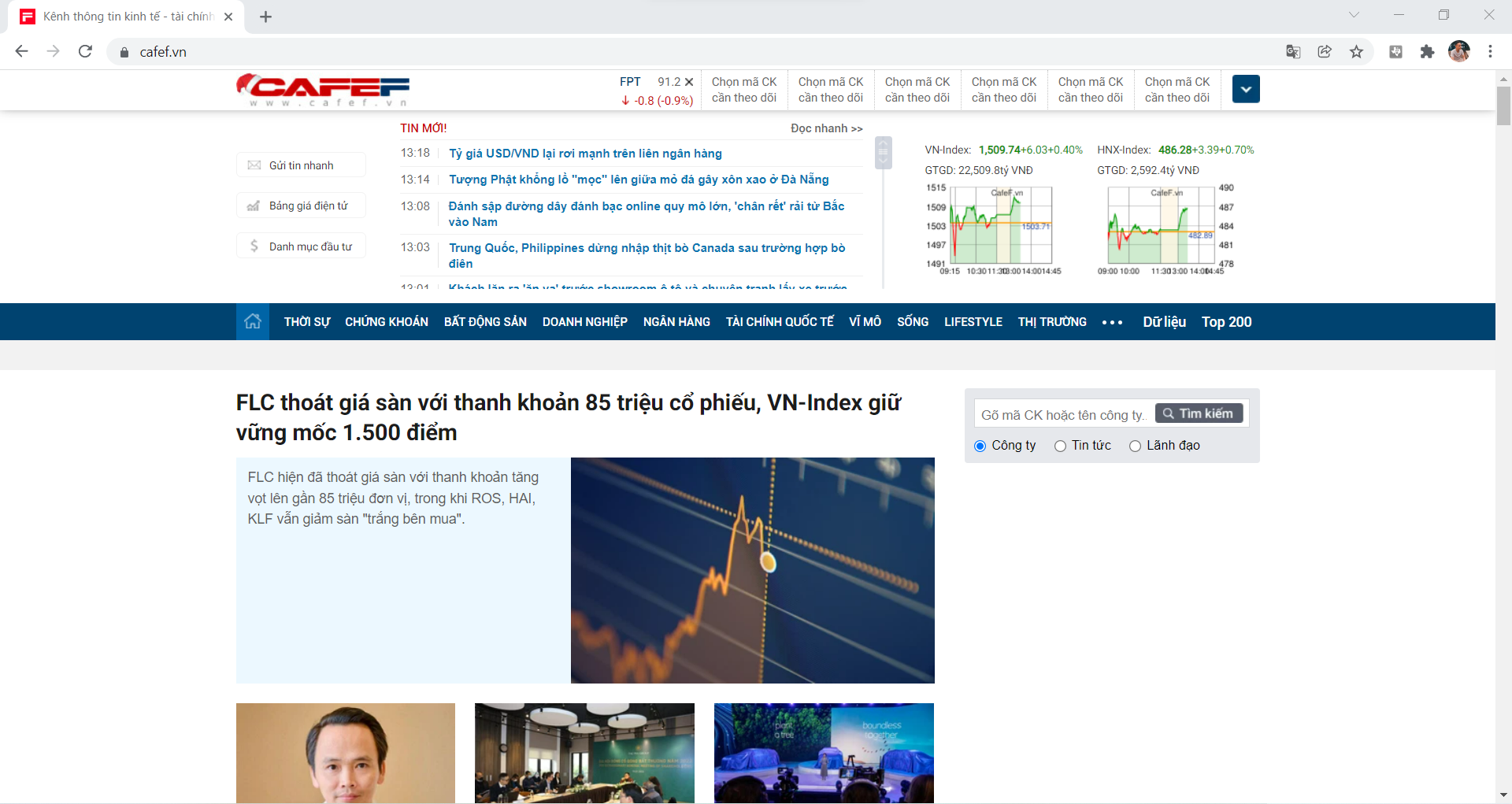
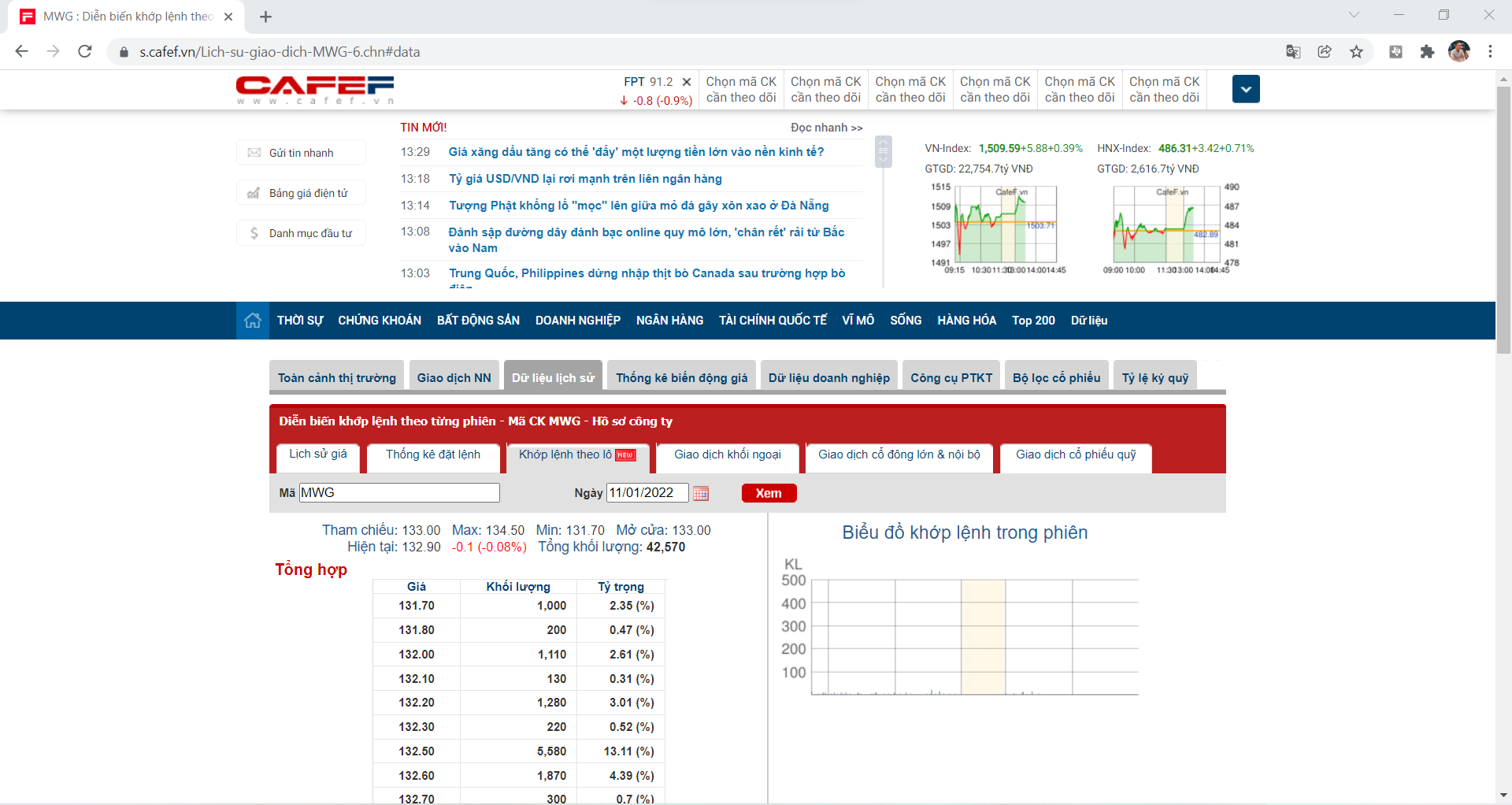
# **CRAWLING DỮ LIỆU GIAO DỊCH THEO LÔ CỦA TỪNG CỔ PHIẾU**

## **I.KHẢO SÁT TRANG WEB HTTP://CAFEF.VN**

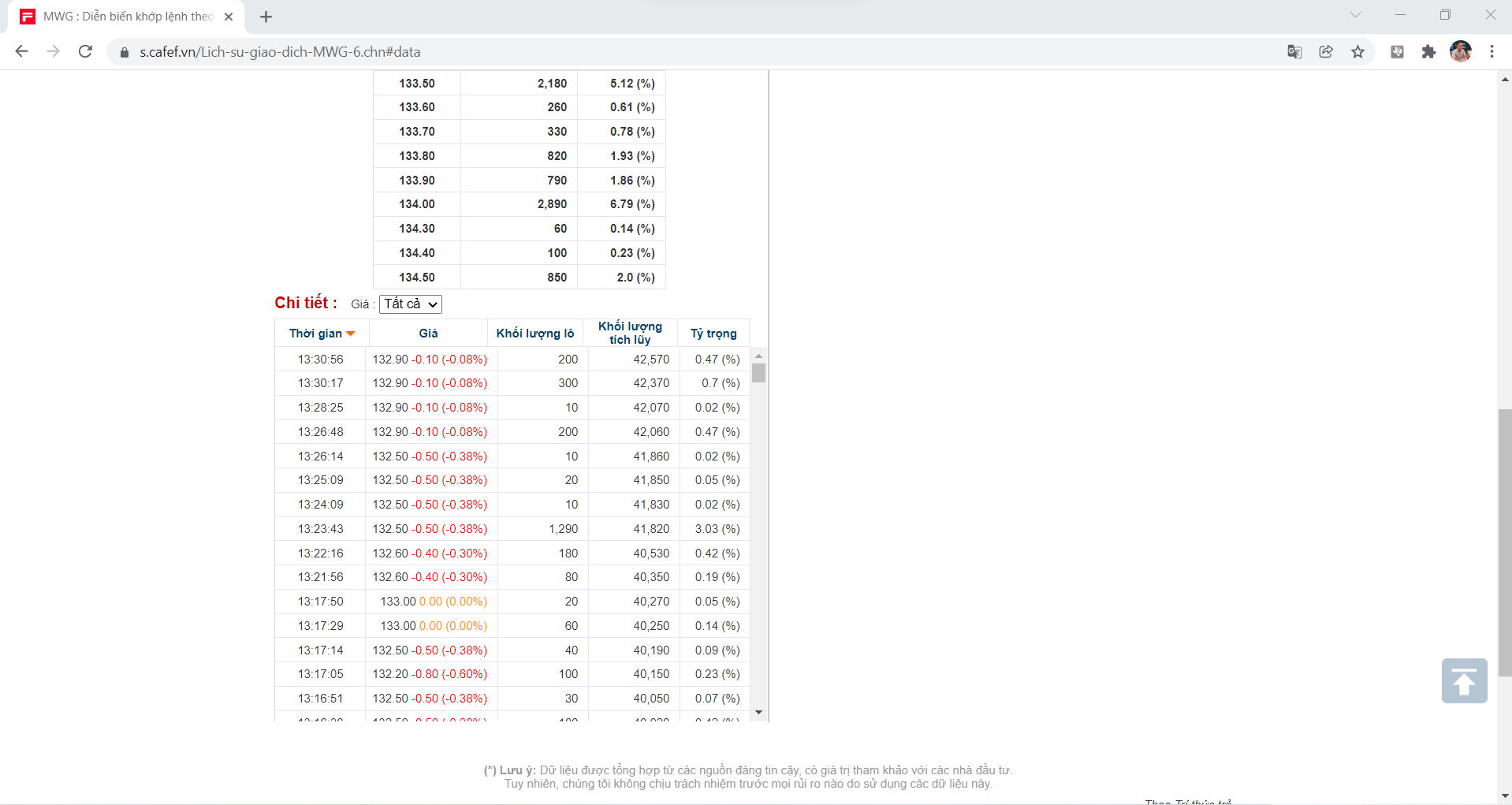
Có nhiều trang Web cung cấp dữ liệu này một cách công khai. Trang web https:\\cafef.vn là một trong số đó. Ta chọn trang web này để crawl dữ liệu giao dịch theo lô của một cổ phiếu trong nhiều ngày và lưu vào Database (PostgreSQL). Sau đó dữ liệu này được chuyển cho đội phân tích (DA, BA) để họ phân tích và đưa ra các quyết định.



Trang chủ trang web cafef.vn

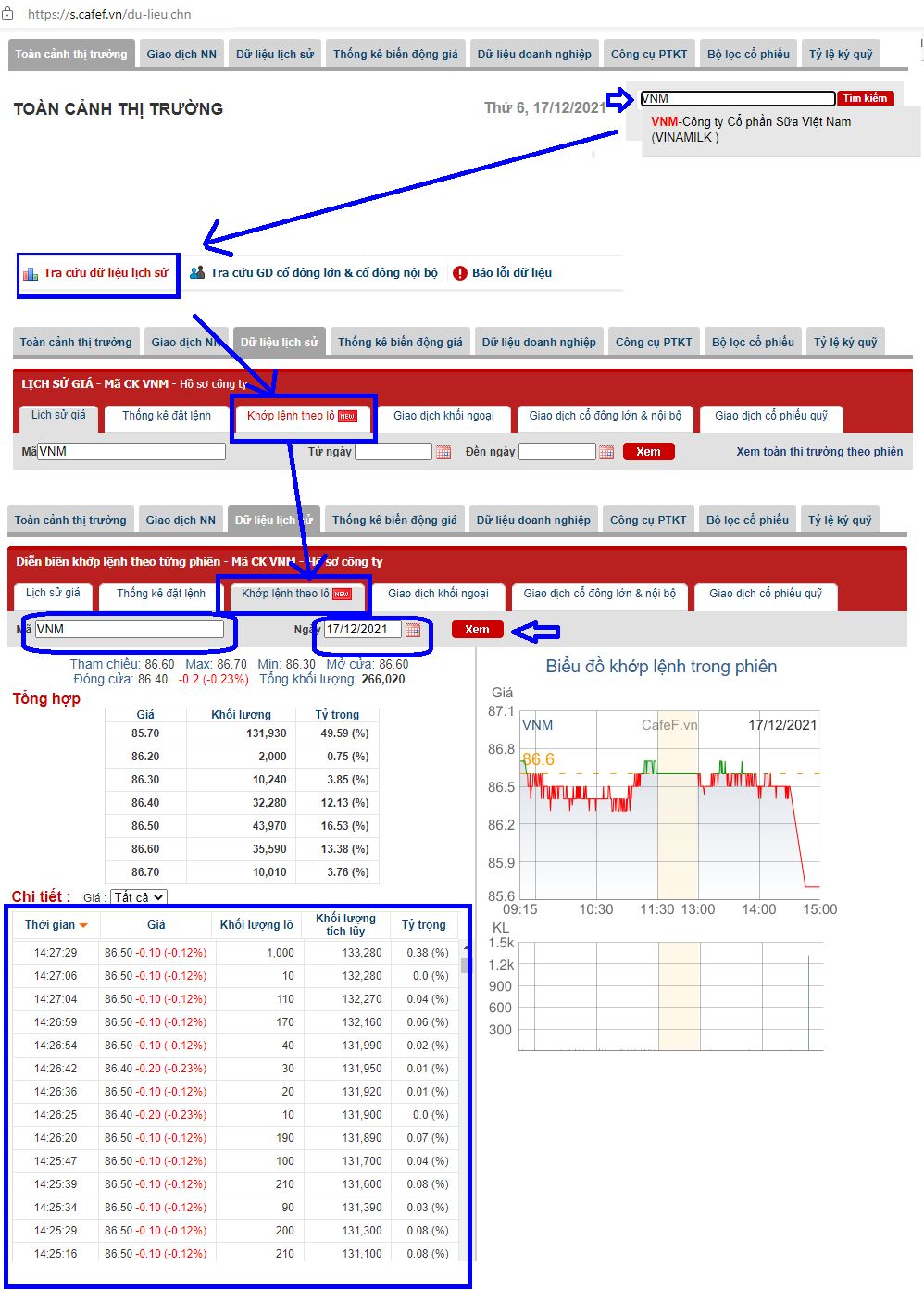


Trang dữ liệu giao dịch theo lô (trang dữ liệu)



Bảng dữ liệu giao dịch theo lô trong một ngày của một cổ phiếu MWG (Công ty CP Thế giới di động) trong ngày 11/01/2022.

Để đến được trang dữ liệu này ta có 2 cách:

* **Cách 1**: Mở trình duyệt, truy cập địa chỉ: https://cafef.vn và thực hiện các bước theo thứ tự như sau:

Ta được thống kê giao dịch theo lô của cổ phiếu VNM trong ngày (hình ảnh được chụp từ ngày 17/12/2021).

* **Cách 2**: Truy cập thẳng đến địa chỉ <https://s.cafef.vn/Lich-su-giao-dich-VNM-6.chn#data>

Với VNM là tên cổ phiếu của công ty Vinamilk, ta có thể thay đổi tên này để trực tiếp chuyển đến trang dữ liệu của cổ phiếu cần tìm. Trang web sẽ hiện thị ngày có giao dịch gần nhất (thứ 7, chủ nhật và các ngày nghỉ lễ sẽ không có giao dịch).

Sau khi đã biết được dữ liệu nào cần lấy, và nơi có thể crawl dữ liệu, ta viết tool để tự động hóa quá trình này và lưu trữ dữ liệu vào Database. Tool được viết bằng ngôn ngữ Python với sự hỗ trợ của thư viện Selenium.

[Selenium](https://selenium-python.readthedocs.io/index.html) là một công cụ kiểm thử phần mềm hoặc website tự động (Automation Test Tools). Với công cụ này, ta có thể mô phỏng hầu hết các thao tác của người dùng với trình duyệt như: click, cuộn, đóng mở tab, nhập text, …). Nhờ đó ta có thể tự động mở trình duyệt và di chuyển đến các trang dữ liệu.

Thư viện Selenium trong Python có các hàm viết sẵn để mô phỏng các thao tác của người dùng với trình duyệt. Tuy nhiên bạn cần phải nắm được khái niệm Inspect Element của trình duyệt.

Sau đây ta sẽ liệt kê một số hàm cần dùng và việc kết hợp nó với Inspect Element để thực hiện đúng thao tác ta muốn:

Mở trình duyệt Chrome: **webdriver.Chrome()**, truy cập đến trang dữ liệu bằng method **driver.get()** với url là địa chỉ trang web: [https://s.cafef.vn/Lich-su-giao-dich-MWG-6.chn#data](https://s.cafef.vn/Lich-su-giao-dich-MWG-6.chn%23data). Object **driver** lúc này là trình duyệt Chrome. Options được thêm vào để loại bớt các thông báo không cần thiết.

options = webdriver.ChromeOptions()

    options.add\_argument('--ignore-certificate-errors')

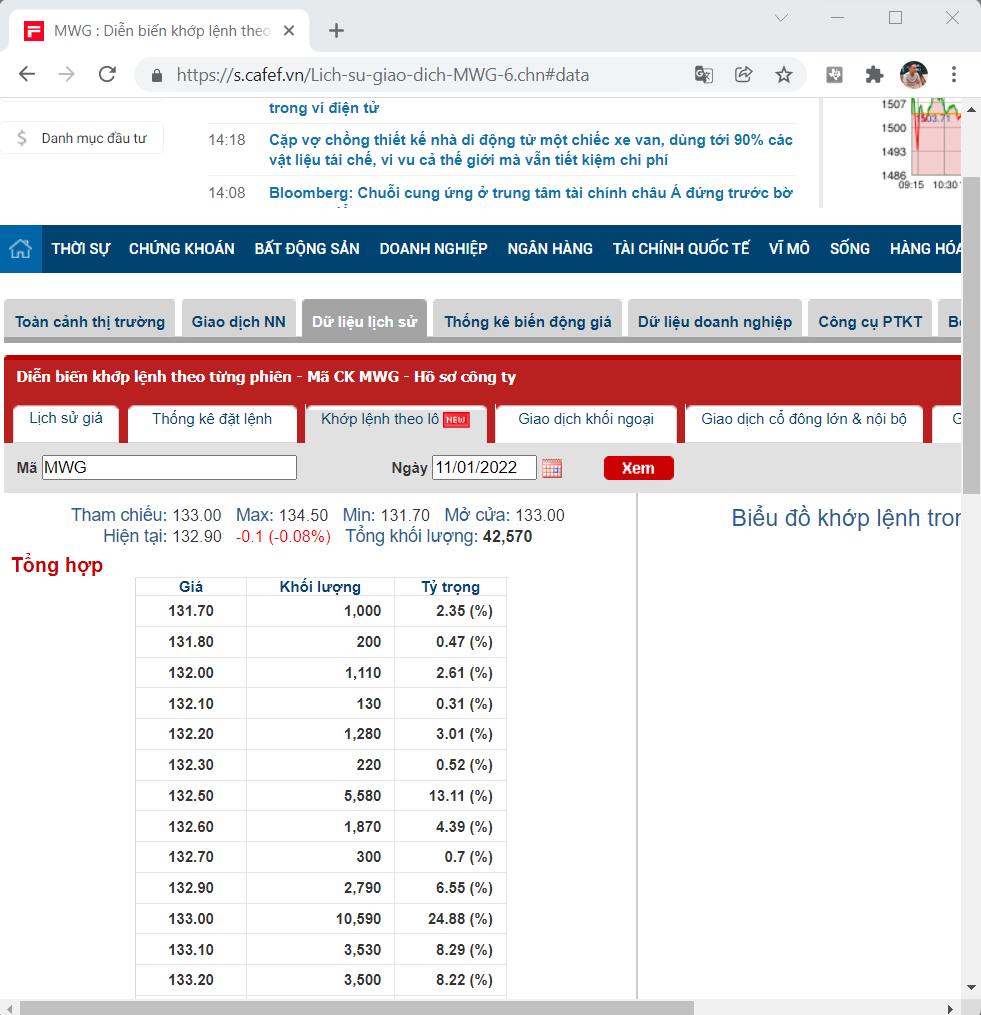
    options.add\_argument('--ignore-ssl-errors')

    options.add\_experimental\_option('excludeSwitches', ['enable-logging'])

    driver = webdriver.Chrome(options=options)

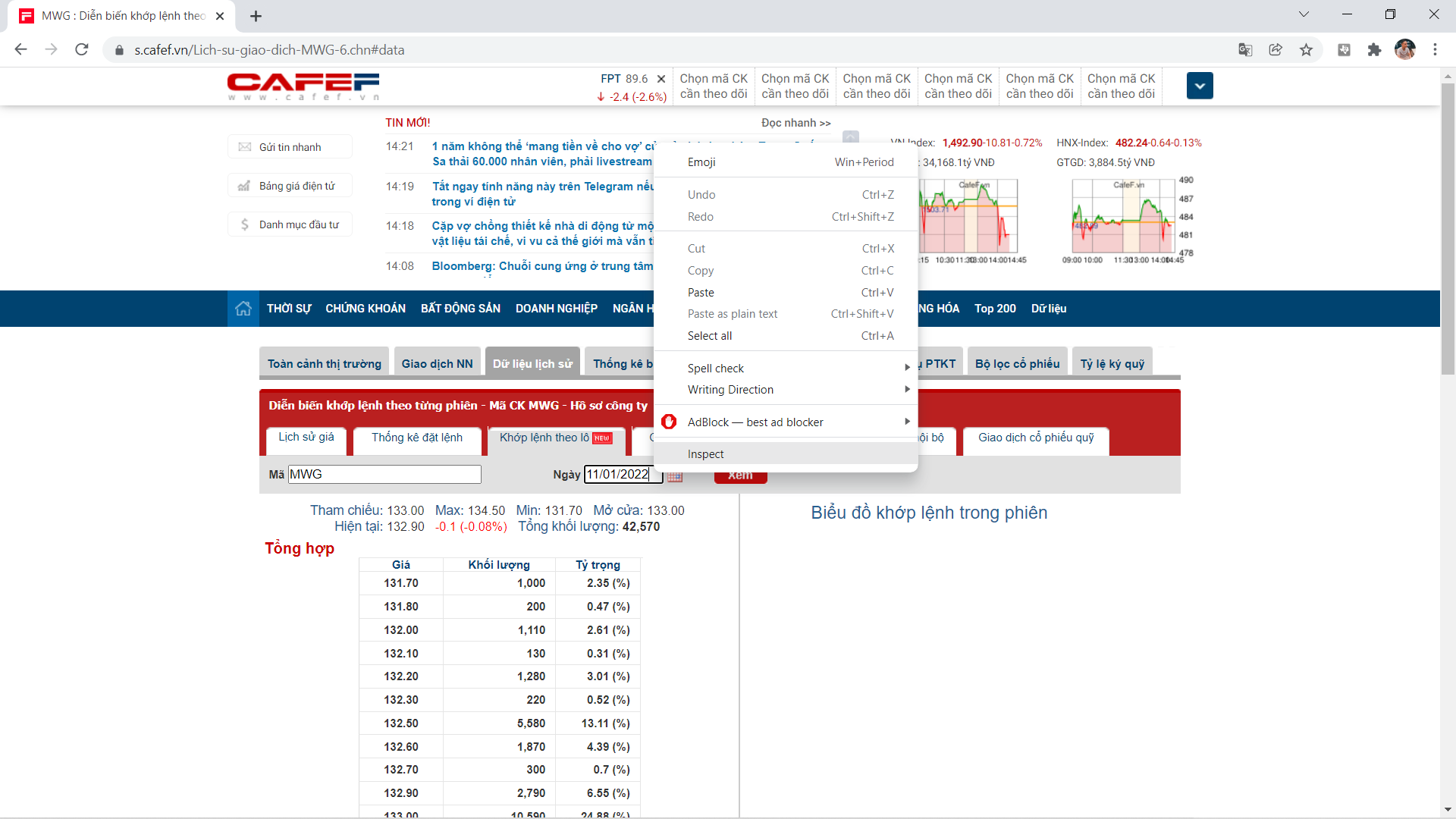
    driver.get(url=url)

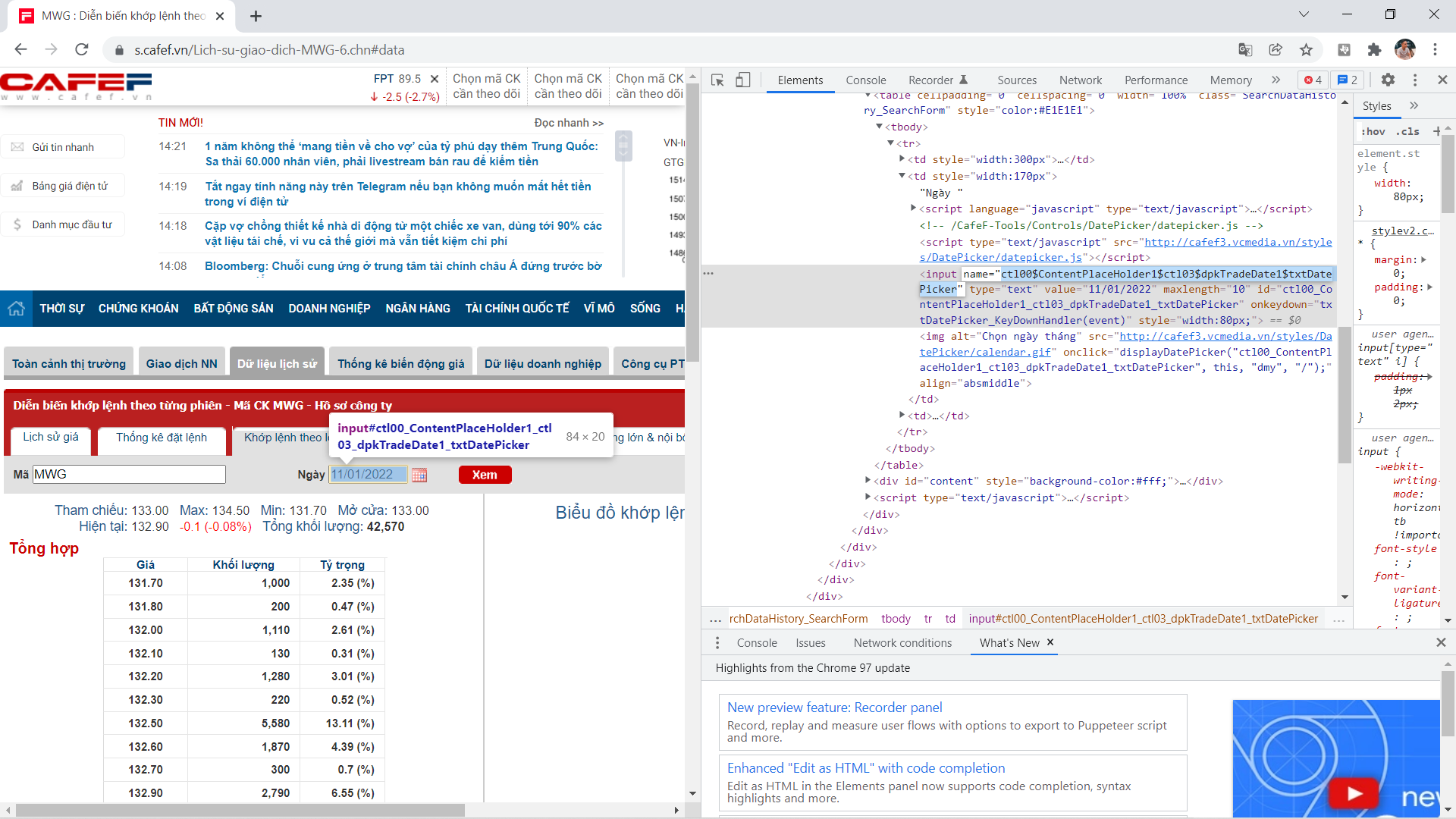
Khi chạy chương trình, máy tính sẽ mở trình duyệt và truy cập đến địa chỉ ở trên. Ta được:



Lúc này ta cần trỏ đến ô ngày tháng để nhập lại ngày cần tìm, ví dụ ngày **05/01/2021**. Để mô phỏng các thao tác này bằng Selenium, trước đó ta cần mở trang dữ liệu này lên và xác định tên của Input này trong phần Inspect một cách thủ công như sau:

Đưa con trỏ đến vị trí ô nhập ngày tháng và click chuột phải, chọn mục **Inspect**.





Ta copy lại tên của input để dùng và dán vào method **driver.find\_element\_by\_name()** để Selenium tìm đến đúng vị trí ô nhập này.

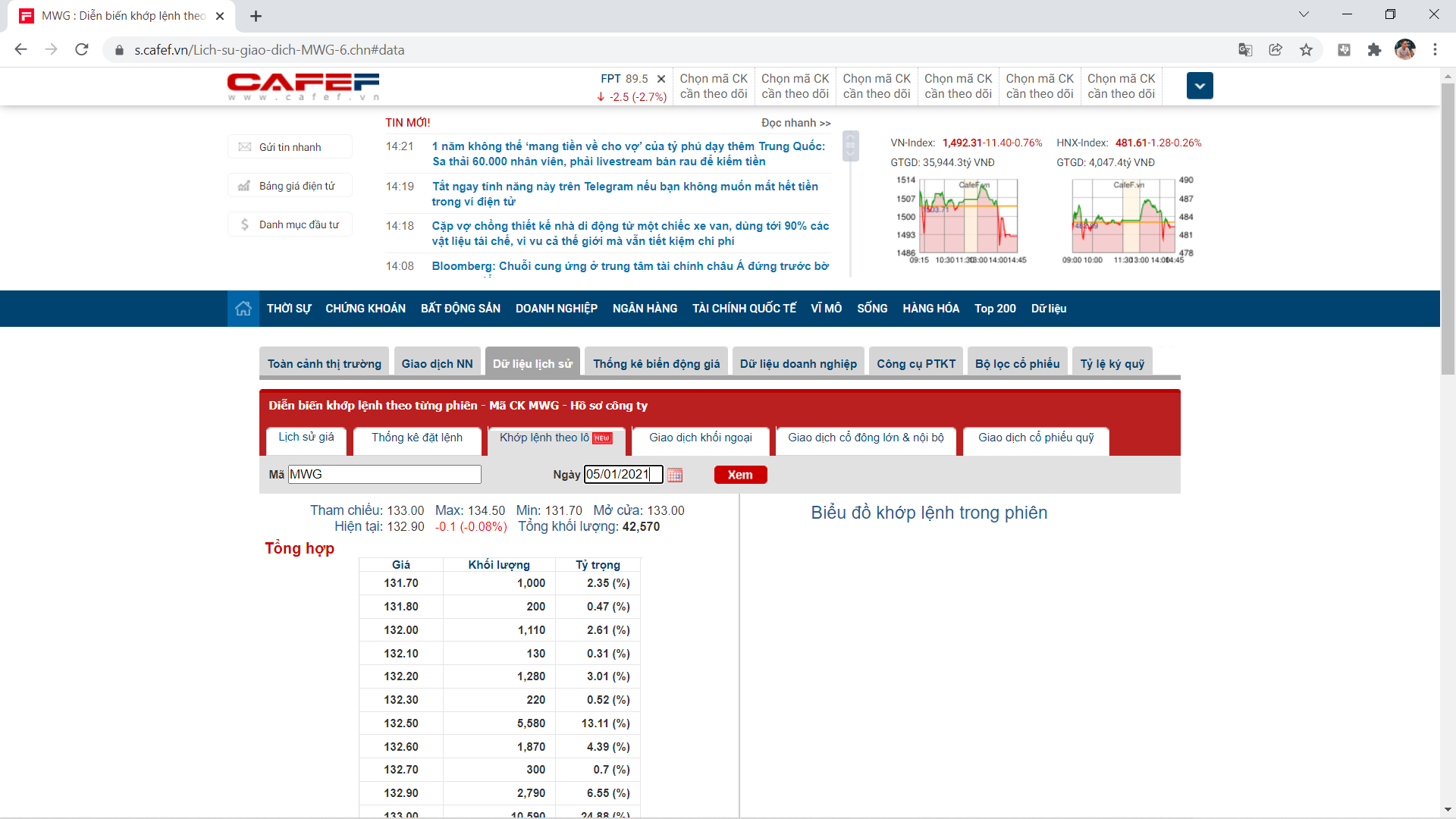
element = driver.find\_element\_by\_name("ctl00$ContentPlaceHolder1$ctl03$dpkTradeDate1$txtDatePicker")

Xóa ngày hiện tại

element.clear()

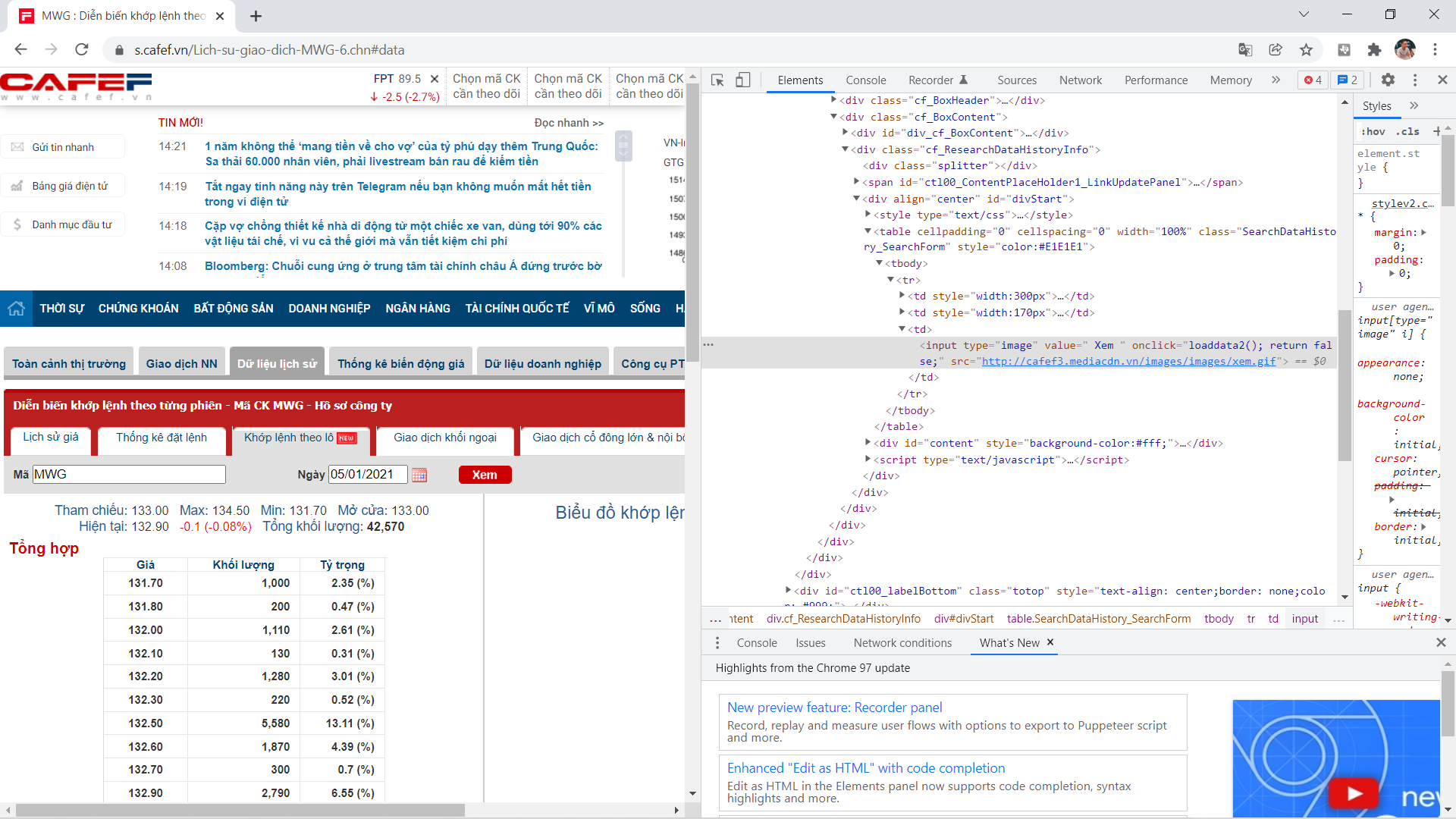
Nhập vào ngày mới

element.send\_keys(‘05/01/2021’)

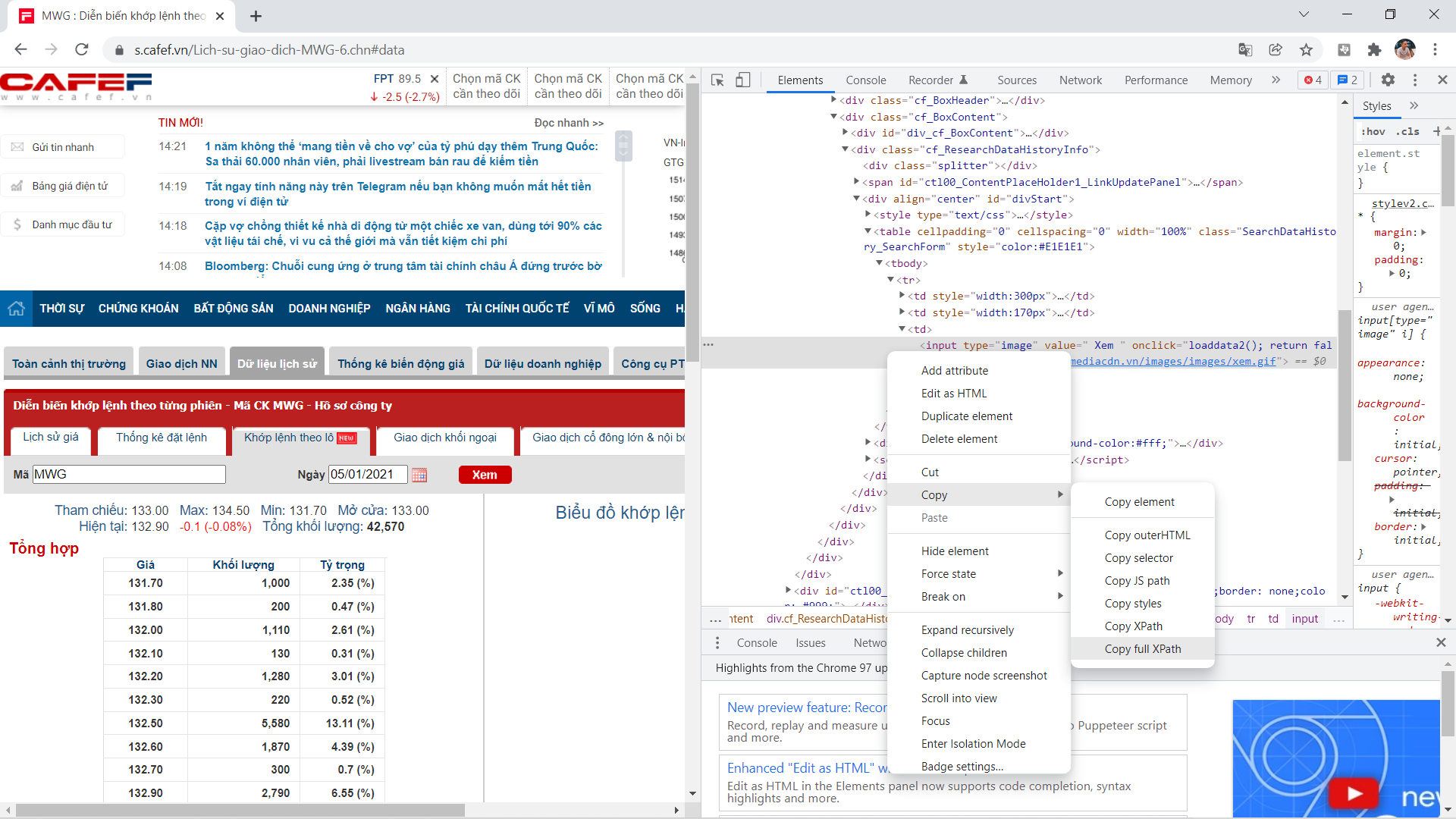


Khi đã nhập ngày, ta cần Selenium nhấn nút xem để trình duyệt load dữ liệu giao dịch vào ngày 05/01/2021.

Ta cũng cần kiểm tra Inspect của nút **‘Xem’** này giống như trên.



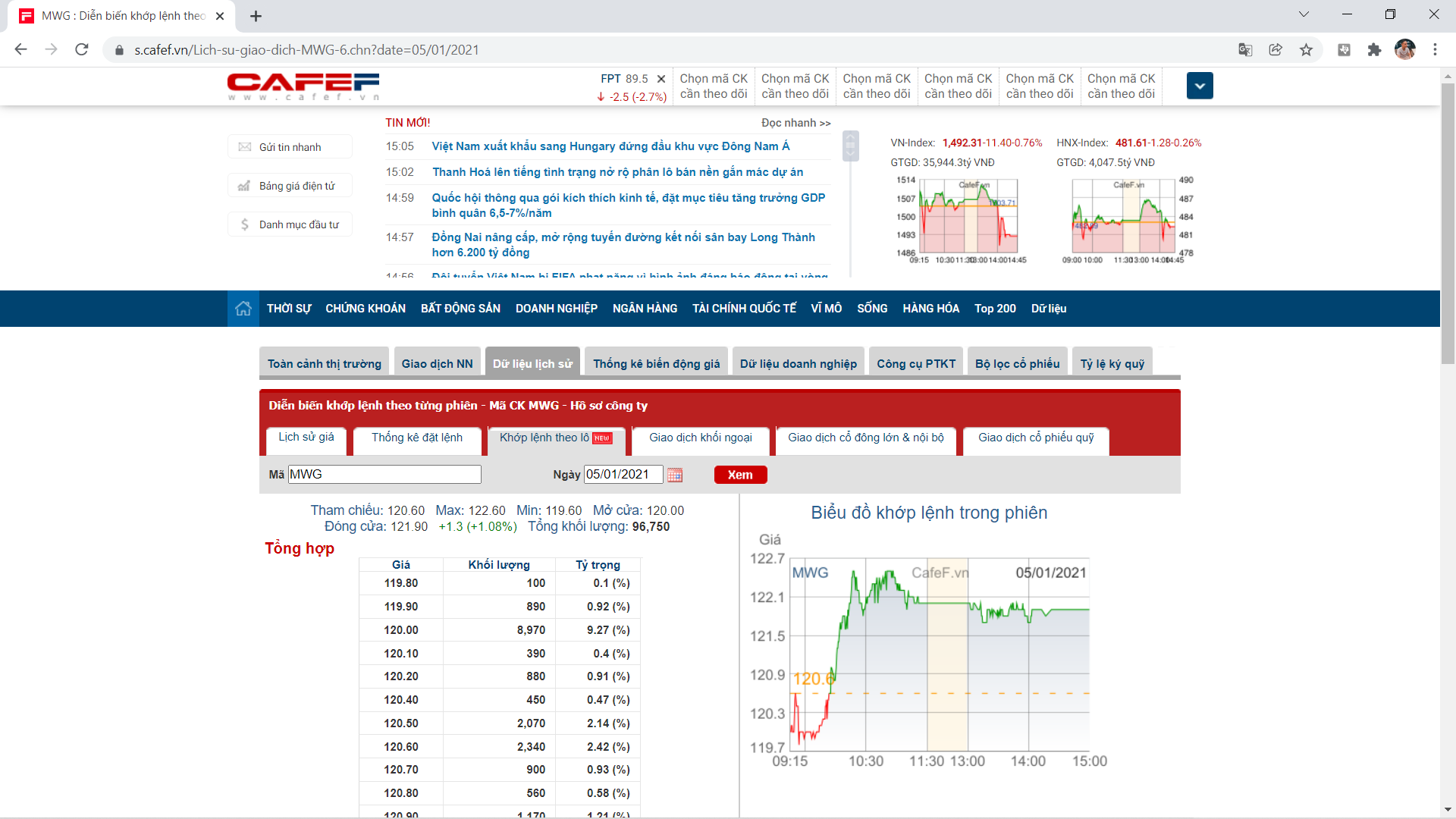
Tuy nhiên, input này không có các giá trị như **name**, **id**. Ta phải tìm nó bằng method **driver.find\_element\_by\_xpath().** Click chuột phải và chọn copy full XPATH.



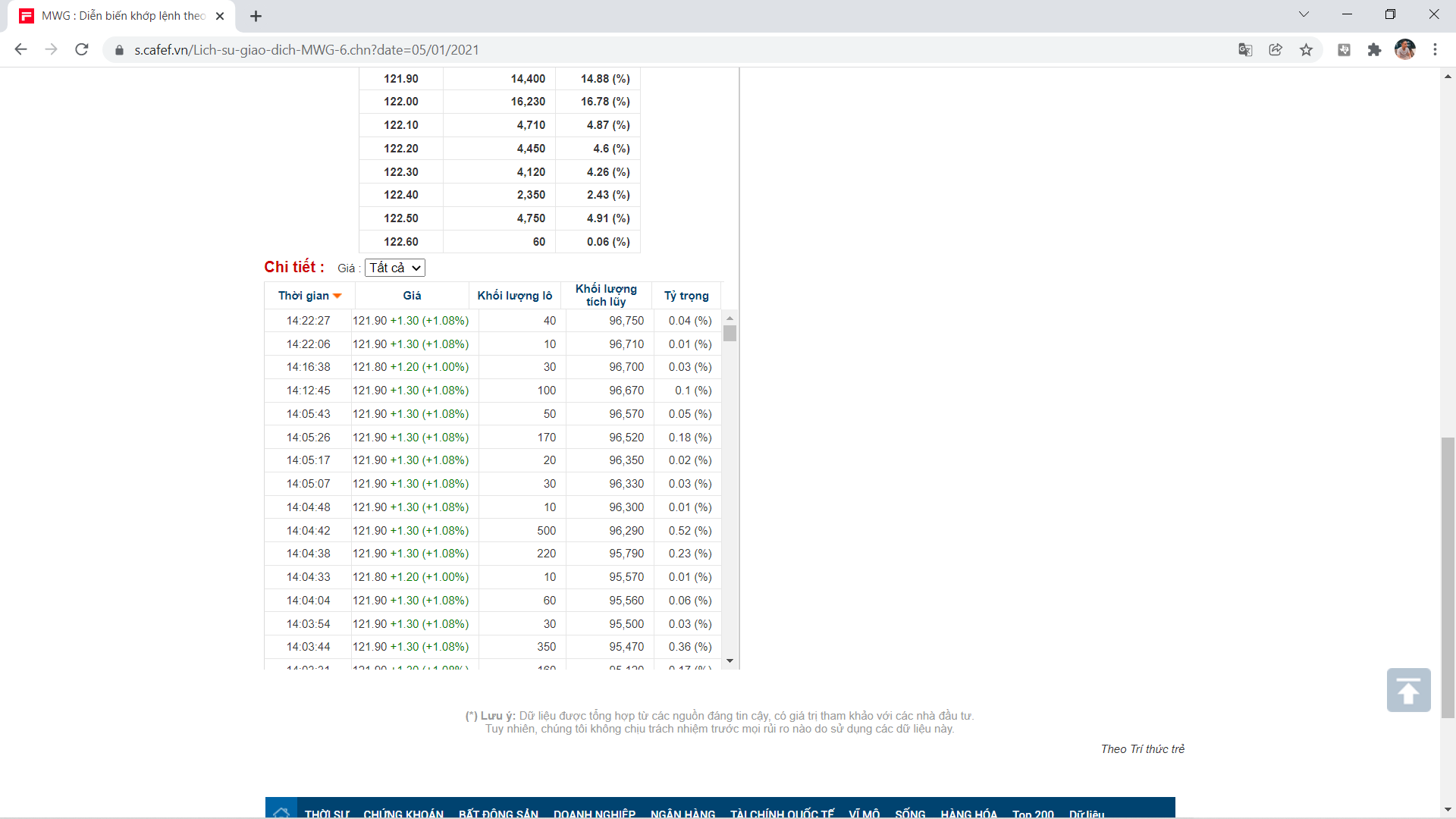
search = driver.find\_element\_by\_xpath("/html/body/form/div[3]/div/div[2]/div[2]/div[1]/div[2]/div/div/div[2]/div[2]/div[2]/table/tbody/tr/td[3]/input")

Click vào nút:

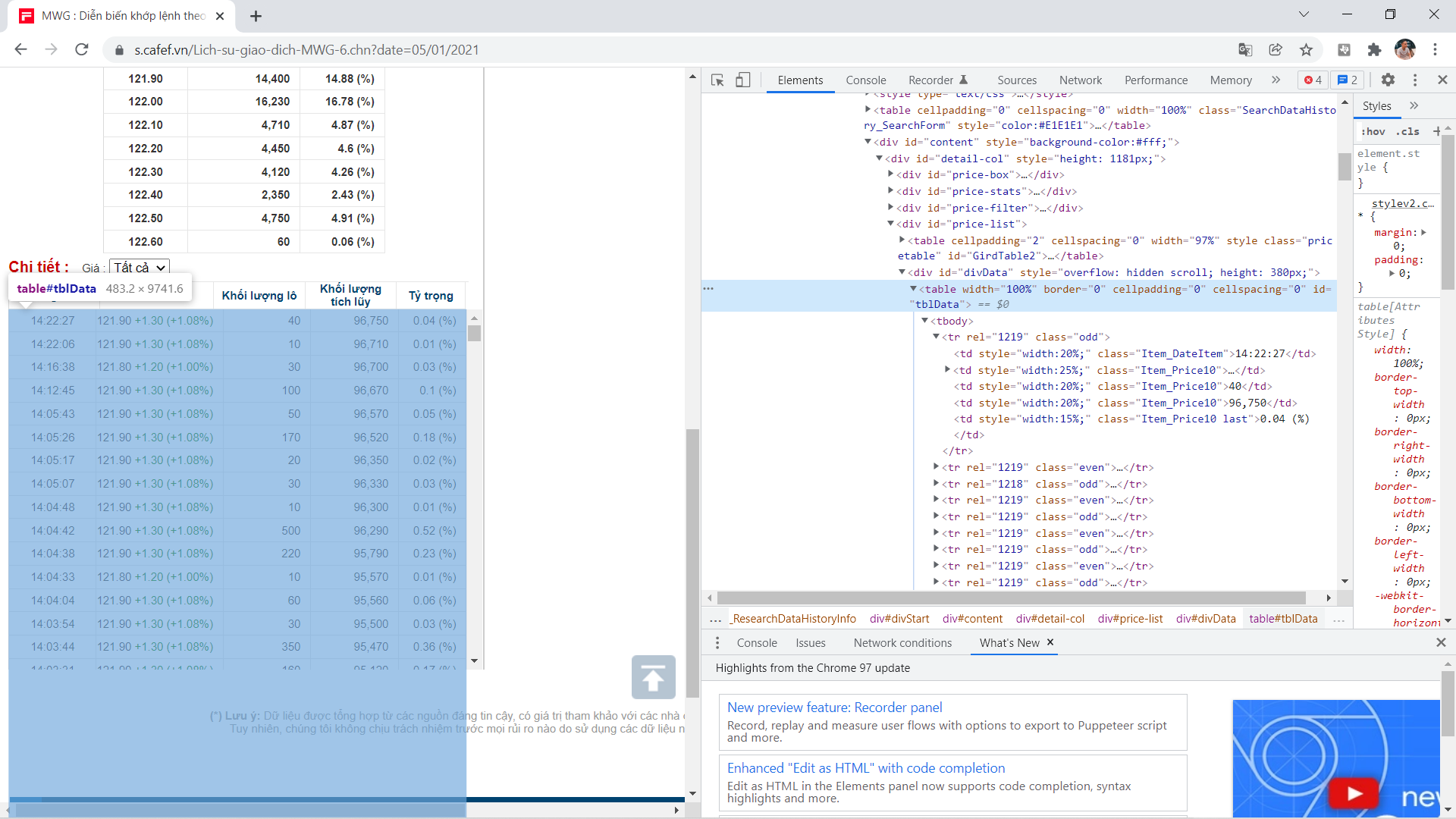
search.click()



Sau khi đã load được trang dữ liệu của ngày cần tìm, ta sẽ xử lý xử liệu giao dịch được hiển trị trong bảng Chi tiết.



Tiếp tục xem **Inspect**,



Bảng này có **id = tblData**, tìm bảng này bằng method **driver.find\_element\_by\_id()**

table = driver.find\_element\_by\_id("tblData")

Như vậy là quá trình tự động mở trình duyệt và xác định bảng dữ liệu bằng Selenium đã hoàn tất, ta lấy dữ liệu trong bảng này bằng cách lấy text của bảng.

data = table.text

Dữ liệu trả về dưới dạng chuỗi, với mỗi dấu xuống hàng là thông tin của 1 lô giao dịch.

Link Gitlab của dự án chứa file demo và file hướng dẫn cài đặt thư viện để chạy chương trình:

<https://gitlab.com/ecodata2m/ecodata2m.git>

## **II.CÀI ĐẶT WINDOWS TASK SCHEDULER ĐỂ CHẠY TOOL TỰ ĐỘNG**

**Windows Task Scheduler** là công cụ giúp người dùng có thể chạy một ứng dụng trong thời gian được thiết lập trước, và quá trình chạy tự động này có thể lặp đi lặp lại nhiều lần tùy theo việc chúng ta cài đặt. Trong hệ điều hành Linux cũng có một công cụ tương tự được gọi là **Crontab**.

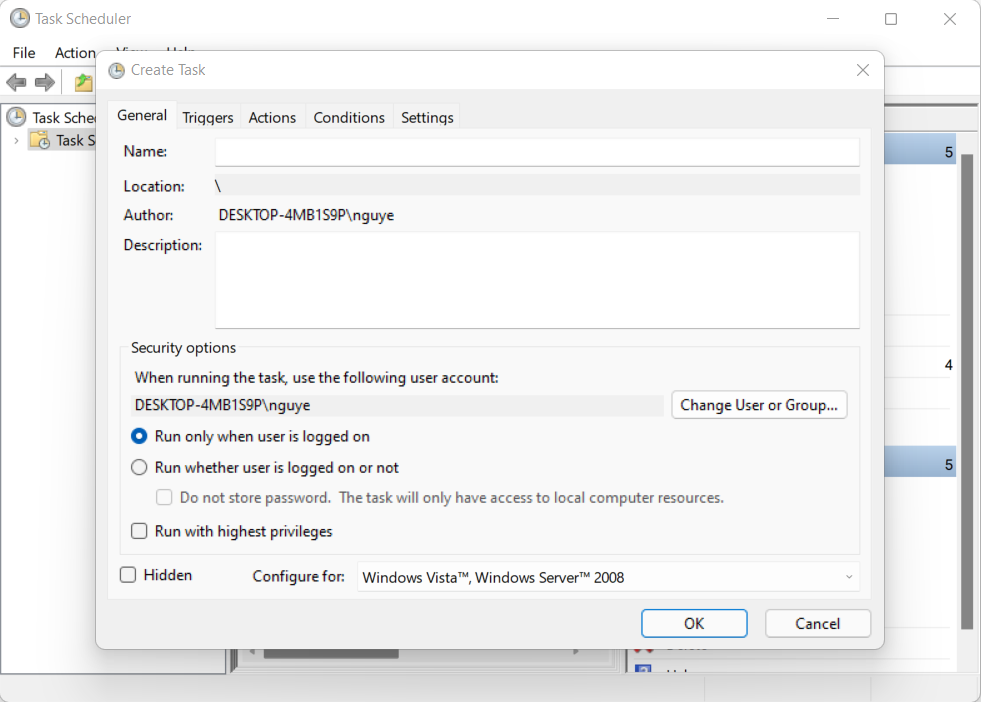
Trong tài liệu này, ta sẽ nói rõ hơn về Task Scheduler của Windows.

Vào mục tìm kiếm, search **Task Scheduler** và khởi chạy ứng dụng.

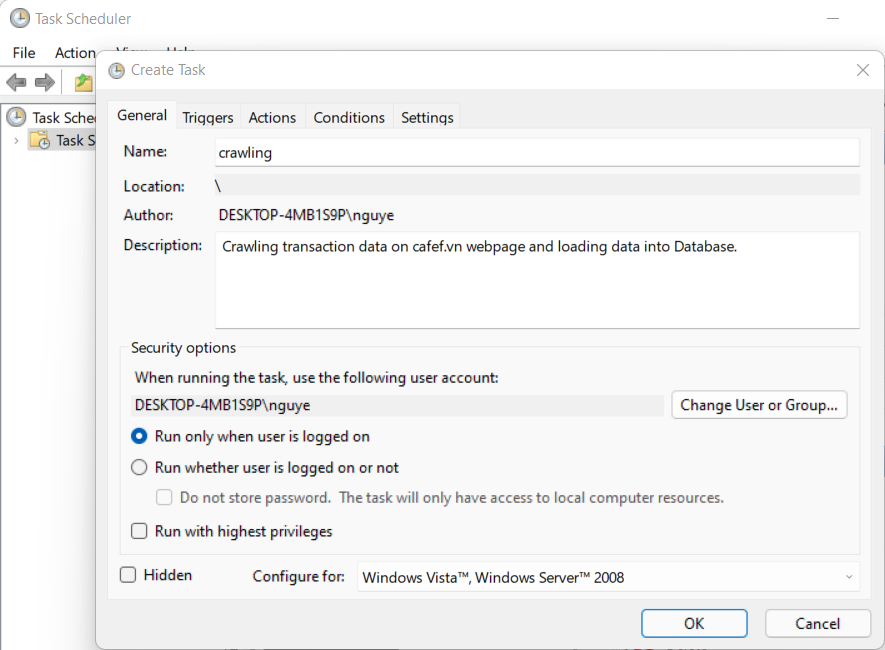


Giao diện ứng dụng Windows Task Scheler

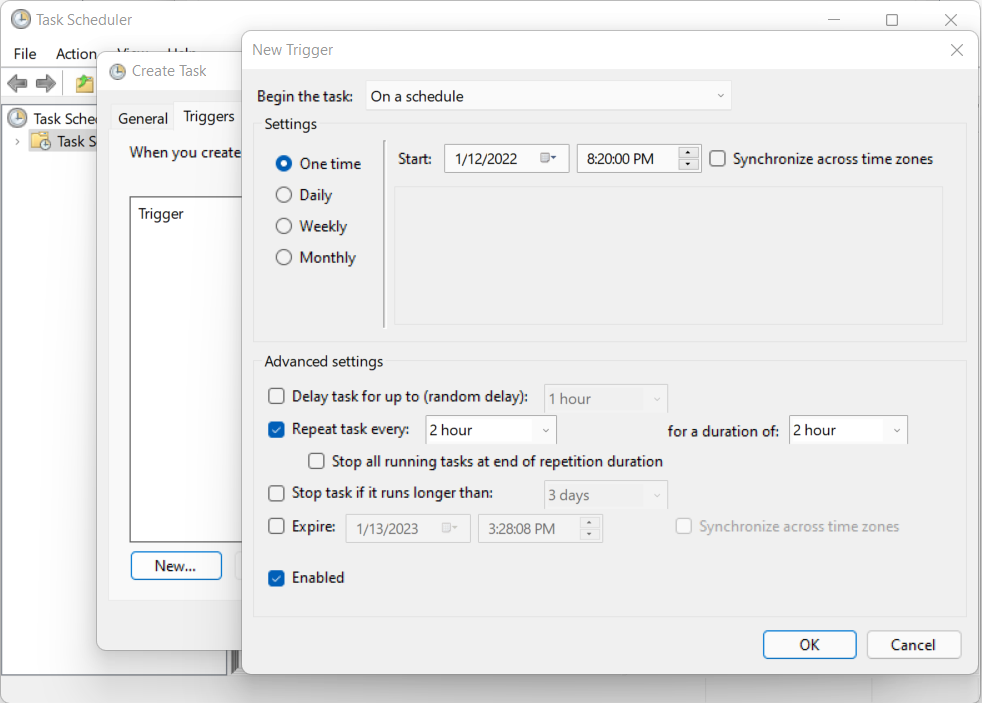
Chọn **Create Task** để tạo 1 task mới.



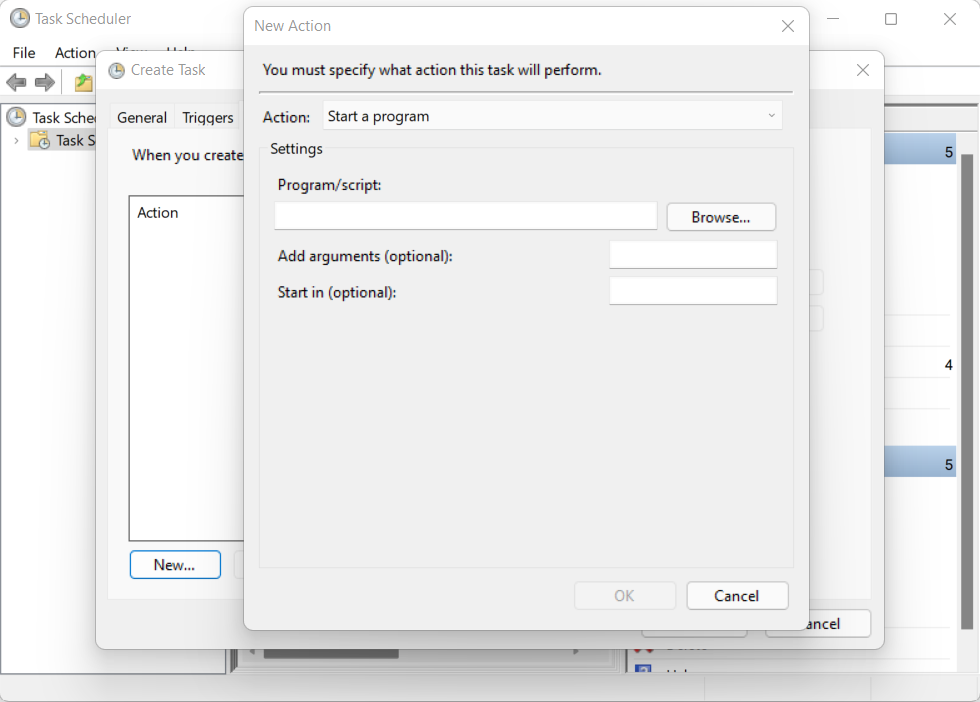
Nhập tên và mô tả cơ bản về Task mà bạn muốn tạo.



Mục **Trigger** là nơi bạn chọn cách mà bạn muốn khởi chạy ứng dụng (thời gian bắt đầu và kết thúc, lặp lại, …). Chọn Trigger, chọn **New** để cài đặt.



Tiếp đến là mục **Action**, là mục mà bạn cài đặt ứng dụng cần chạy. Tương tự chọn Action, chọn **New**.



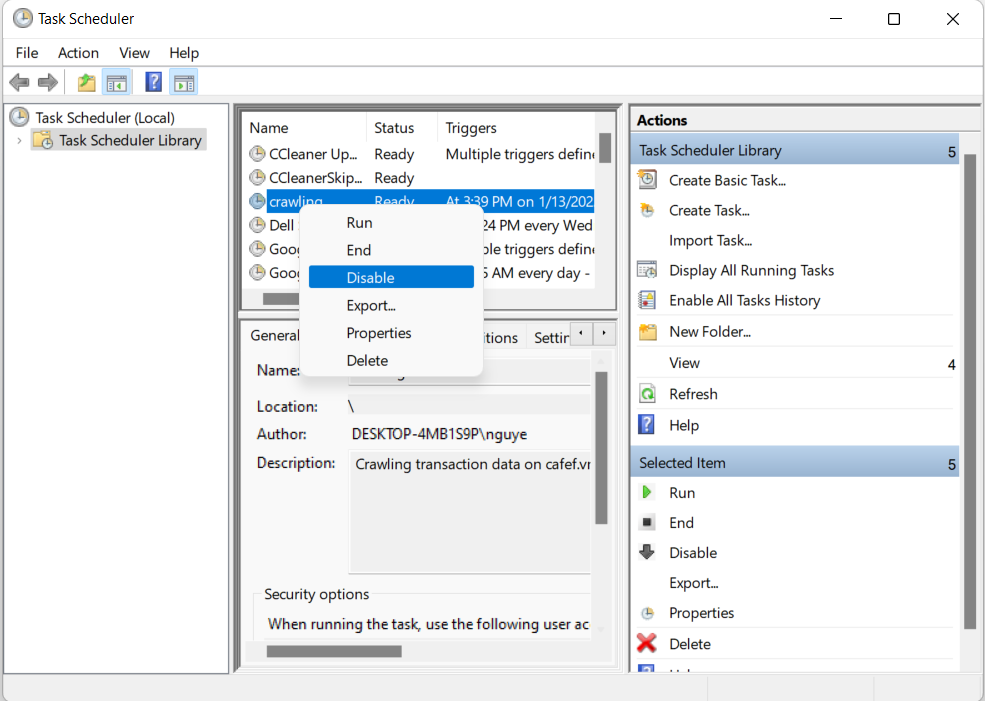
Ô Program/script là đường dẫn đến file **python.exe**. Ví dụ: **C:\Users\nguye\AppData\Local\Programs\Python\Python37\python.exe**

**Arguments** là tên ứng dụng mà bạn muốn khởi chạy. Ví dụ: **main.py**

**Start in** là nơi đặt ứng dụng. Ví dụ: **D:\Code\test\_task\_schedule**

Sau khi đã điền đầy đủ thông tin ở 3 mục trên, về cơ bản đã có thể tạo được 1 Task chạy tự động. Chọn **OK**.

Như vậy ta đã tạo Scheduler để chạy file main.py mỗi 2 tiếng một lần, bắt đầu vào lúc 20:20:00 ngày 12/01/2022. Để tắt đi tính năng này, ta chọn vào Task và chọn **Disable**.



Trên đây chỉ là một vài thiết lập cơ bản để tự động hóa 1 Task đơn giản. Khi đã tạo xong Task bạn có thể **Export** các cài đặt này thành file **.xml**, một khi muốn sử dụng Task tương tự trên máy khác, chỉ cần **import** file này vào là xong.