BÀI 6

Lấy dữ liệu thống kê về dịch Covid-19 bằng phân tích cú pháp mã nguồn HTML

Chúng ta đã biết, các trang web sẽ hiển thị giao diện và các thông tin như văn bản, hình ảnh, liên kết... bằng mã HTML. Trong bài học này, chúng ta sẽ tìm hiểu cách quan sát mã nguồn của một trang web và lấy ra thông tin bất kỳ từ trang web đó. Thao tác lấy thông tin cần được thực hiện theo hai bước thực hiện trong tập tin views.py:

Bước 1: Lấy tệp tin mã nguồn hiển thị giao diện HTML của trang web.

Bước 2: Tìm vị trí dữ liệu cần lấy nằm ở đâu trong mã nguồn và tách lấy số liệu.

Website của chúng ta cần lấy các thông tin về dịch Covid-19:

- Thống kê số liệu về dịch Covid-19:
 - + Tổng số ca nhiễm ở Việt Nam và trên thế giới
 - + Số ca khỏi ở Việt Nam và trên thế giới
 - + Số ca tử vong ở Việt Nam và trên thế giới
- Tin tức về dịch Covid-19
 - + Các tin tức về dịch Covid-19 ở Việt Nam

Chúng ta sẽ cần tìm những trang báo điện tử có những thông tin này, ví dụ số liệu thống kê ở Việt Nam có thể tổng hợp ở trang https://vnexpress.net/covid-19/covid-19-viet-nam, tin tức có thể lấy ở trang https://tuoitre.vn/dich-covid-19-e576.htm. Khi mới bắt đầu các bạn nên thực hành theo 2 trang web này, sau khi đã thành thạo các bạn có thể sử dụng bất kỳ trang web nào mong muốn.

6.1. Lấy tệp tin mã nguồn hiển thị giao diện HTML của trang web



Công bố hôm qua +142

' Số ca nhiễm bao gồm cả trong nước và nhập cảnh

Công bố hóm qua +33.034



Đầu tiên, chúng ta sẽ lấy thống kê số liệu của dịch Covid-19. Để lấy được mã nguồn, chúng ta sẽ sử dụng thư viện selenium và chromedriver_py để mô phỏng một trình duyệt gửi yêu cầu truy cập đến một website và nhận lại được mã nguồn. Các bạn cài đặt hai thư viện trên bằng lệnh pip install selenium và pip install chromedriver_py, sau đó cài đặt các module cần thiết vào chương trình như dòng 3, 4, 5. Vai trò cụ thể các module này tương đối phức tạp, các bạn có thể dựa vào tên module để tự tìm hiểu sau.

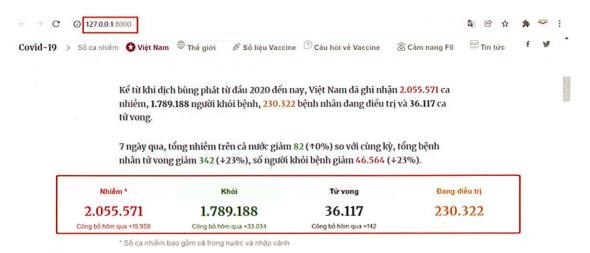
```
from django.shortcuts import render
   from django.http import HttpResponse
2
   from selenium import webdriver
   from chromedriver py import binary path
   def index(request):
       return render(request, 'web_covid.html')
```

Tiếp theo, chúng ta sẽ sử dụng các thư viện này để truy cập vào website cần lấy thông tin và lưu lại kết quả, chúng ta sẽ lấy tin tức trước. Lệnh ở dòng 6, 7, 8 dùng để thiết lập trình điều khiển (driver) của trình duyệt Chrome, lệnh ở dòng 9 dùng để sử dụng trình điều khiển này truy cập vào trang web https://vnexpress.net/covid-19/covid-19-viet-nam như một trình duyệt thông thường. Tại dòng 10, tập tin mã nguồn HTML của trang web được lưu vào biến html_page. Ngoài ra, trong dòng 12 chúng ta sử dụng lại hàm HttpResponse() để hiển thị kết quả thu được lên website.

```
from chromedriver py import binary path
5
  options = Options()
  web driver = webdriver.Chrome(executable path=binary path,
                                              options=options)
                                                     viet-nam")
10 html page = web_driver.page_source
11 def index (request):
       return HttpResponse(html page)
12
```

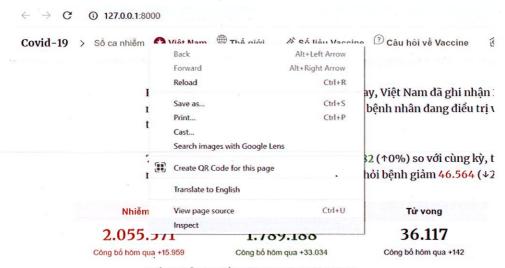
Các bạn chạy dự án bằng lệnh python manage.py runserver để quan sát kết quả. Lần đầu chạy sẽ mất khá nhiều thời gian, các bạn đợi cho đến khi link localhost xuất hiện và nhấn vào link đó. Sau khi nhấn, các bạn thấy giao diện website của chúng ta

chứa nội dung giống với website gốc, như vậy là chúng ta đã lấy mã nguồn của website gốc thành công, tuy nhiên có một số thành phần khác biệt như banner quảng cáo, những thành phần này sử dụng một số cách đặc biệt để hiển thị chứ không có trong mã nguồn, đây là nội dung nâng cao và sẽ không được phân tích chi tiết, chúng ta chỉ cần quan tâm những số liệu về dịch Covid-19 đã được hiển thị, tức là đã được lấy kèm cùng với mã nguồn.

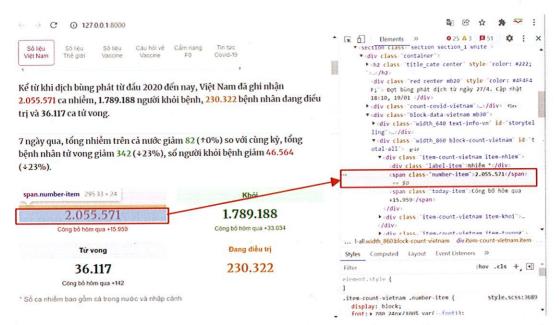


6.2. Tìm vị trí dữ liệu cần lấy nằm ở đâu trong mã nguồn

Tiếp theo chúng ta chuyển sang bước 2. Để biết vị trí của một nội dung bất kỳ trên giao diện nằm ở đâu trong mã nguồn, các bạn nhấn phải chuột vào nội dung đó và chọn Inspect. Mã nguồn của nội dung đó sẽ được hiển thị, các bạn di chuyển con trỏ chuột lên phần nào thì hình ảnh phần đó sẽ được đánh dấu trên giao diện để chúng ta có thể xác định được mã nguồn đó là của phần nào.



* Số ca nhiễm bao gồm cả trong nước và nhập cảnh



Ví dụ với thông tin số ca nhiễm, chúng ta thấy giá trị số ca nhiễm là 2.055.571 nằm bên trong thẻ .

```
▼ <div class="item-count-vietnam item-nhiem">
   <div class="label-item">Nhiem *</div>
   <span class="number-item">2.055.571</span>
   <span class="today-item">Công bố hôm qua
   +15.959</span>
 </div>
```

Chúng ta cần tìm thẻ này trong mã nguồn, qua đó lấy được giá trị văn bản (text) của thẻ. Khi tìm kiếm một thẻ bất kỳ, chúng ta thường gặp 2 trường hợp chính:

- Thẻ có thuộc tính "ID": Vì giá trị thuộc tính ID là duy nhất và dùng để định danh các thẻ, nên chúng ta có thể sử dụng thông tin này để tìm kiếm thẻ mà không bị nhầm lẫn với thẻ cùng loại khác.
- Thẻ không có thuộc tính "ID": Ví dụ với thẻ trên có thuộc tính class giá trị "number-item" và không có ID, trong mã nguồn có thể có nhiều thẻ khác như vậy.

Với trường hợp thẻ không có thuộc tính ID thì chúng ta có nhiều cách để tìm, ví dụ như tìm toàn bộ danh sách các thẻ như vậy và xem thẻ mình cần xuất hiện ở vị trí thứ mấy trong danh sách, hoặc chúng ta cũng có thể tìm thông qua thẻ cha, ví dụ thẻ trên nằm bên trong thẻ <div> có class = "item-count-vietnam" và "item-nhiem", chúng ta có thể tìm thẻ <div> này, từ đó sẽ lấy được thẻ .

Để trích xuất dữ liệu từ HTML một cách dễ dàng và hiệu quả, chúng ta sử dụng thêm thư viện BeautifulSoup, các bạn cài đặt thư viện bằng lệnh pip install bs4, import thư



viện bằng lệnh ở dòng 6, sau đó phân tích cú pháp của mã nguồn html_page và tạo một đối tượng **BeautifulSoup()** (ví dụ: parser) như dòng 12.

```
from chromedriver_py import binary_path
from bs4 import BeautifulSoup

poptions = Options()

html_page = web_driver.page_source
parser = BeautifulSoup(html_page, 'html.parser')
```

Đối tượng parser sẽ lưu mã nguồn đã được phân tích cú pháp và có thể thực hiện các việc tìm kiếm các thẻ theo loại thẻ, thuộc tính... Như phân tích ở trên, chúng ta có thể tìm toàn bộ danh sách các thẻ và xem thẻ chứa thông tin số ca nhiễm nằm ở vị trí thứ mấy, tuy nhiên cách này không khả thi vì số lượng thẻ trong mã nguồn thường rất nhiều và không cố định, vì trang web có thể có các nội dung quảng cáo chứa thẻ có sự thay đổi mỗi lần lấy dữ liệu. Chúng ta thu hẹp phạm vi tìm kiếm bằng các tìm các thẻ nhưng có thông tin class = "number-item".

Để tìm kiếm một thẻ bất kỳ, ta sử dụng hàm **find_all()**, hàm sẽ trả về danh sách các thẻ phù hợp. Chúng ta có thể truyền vào tên thẻ để tìm tất cả các thẻ đó trong mã nguồn, ngoài ra có thể truyền thêm thuộc tính của thẻ để thu hẹp phạm vi tìm kiếm.

```
# list_tag là danh sách tất cả các the <span>
list_tag = parser.find_all('span')

# list_tag_with_class là danh sách tất cả các thẻ <span> có thuộc
tính class là 'item'
list_tag_with_class = parser.find_all('span', class_='item')
```

Khi các bạn có kinh nghiệm làm việc với HTML sẽ đưa ra được cách tìm kiếm theo các thẻ tối ưu hơn. Chúng ta thử tìm kiếm theo với thuộc tính class = "numberitem", sau đó in ra màn hình Terminal danh sách các thẻ này. Các bạn thêm các lệnh từ dòng 13 – 18 và chạy dự án, trong đó dòng 14 và 18 in ra chuỗi các dấu gạch ngang dùng để phân biệt với các thông tin khác được hiển thị trên màn hình, dòng 15 in ra toàn bộ danh sách list_span. Chúng ta thường xuyên sử dụng những lệnh print() này để hiển thị thông tin lên màn hình, qua đó theo dõi được chương trình hoạt động đúng hay không.

```
12 parser = BeautifulSoup(html_page, 'html.parser')
13 list_span = parser.find_all('span', class_='number-item')
14 print('-----')
15 print(list_span)
```

```
16 print(list_span[0])
17 print('so luong: ' + str(len(list_span)))
18 print('----')
```

list_span là một danh sách lưu các thẻ cần tìm. Danh sách (list), hay còn gọi là mảng (array), là một kiểu dữ liệu. Thay vì lưu trữ một thông tin duy nhất như biến, nó lưu trữ tập hợp nhiều thông tin, tạo thành một danh sách. Thông tin có thể là một số, một ký tự, một đoạn văn bản, một đối tượng..., danh sách list_span trong trường hợp này lưu các đối tượng thẻ và có thể coi là các xâu ký tự.

Mỗi thông tin trong danh sách được gọi là một phần tử (**item**). Các phần tử lưu trữ trong danh sách đều có vị trí (**index**) xác định, hay còn được gọi là chỉ mục và được bắt đầu từ **0**. Chúng ta truy cập đến phần tử bất kỳ trong danh sách qua gặp dấu ngoặc vuông [] với cú pháp **<danh sách>[<vị trí>]**, ví dụ lệnh ở dòng 16 giúp hiển thị phần tử ở vị trí đầu tiên (vị tri 0) trong danh sách. Hàm **len(<danh sách>)** trả về kích thước (số lượng phần tử) của một danh sách bất kỳ như được sử dụng trong dòng 17.

Dựa vào kết quả trên, ta thấy các thông tin Số ca nhiễm, Số ca khỏi, Số ca tử vong nằm ở 3 phần tử đầu tiên có vị trí 0, 1, 2 trong danh sách.

Nhiễm *	Khòi	Từ vong	Đang điều trị
2.088.221	1.794.924	36.446	256.907
Công bố hôm qua +15.935		Công bố hôm qua +177	Công bổ hôm qua +15.758

```
span class="number-item">2.088.221</span>, <span class="number-item">1.794.924</span>
<span class="number-item">36.446</span> <span class="number-item"</pre>
>256.907</span>, <span class="number-item">512.422</span>, <span class="number-item">20<
/span>, <span class="number-item">20.138</span>, <span class="numbe
r-item" id="item-vaccine-city">100,0%</span>]
so luona: 8
<span class="number-item">2.088.221
```

Để tách ra được số liệu, ta sử dụng lệnh .text như mô tả dưới đây. Các bạn tạo thêm 3 biến để lưu 3 thông tin trên (ví dụ: VN_case, VN_recovery, VN_death), việc đưa thông tin này vào giao diện web sẽ được hướng dẫn chi tiết sau. Lúc này các bạn có thể xóa các lệnh từ dòng 14 – 18, thay vào đó là các lệnh lưu và hiển thị thông tin mới. Các bạn chạy thử và thấy chúng ta đã tách ra thành công các số liệu cần tìm.

```
13 list_span = parser.find_all('span', class = 'number-item')
14 VN case = list span[0].text
15 VN recovery = list span[1].text
16 VN death = list span[2].text
17 print (VN case)
18 print (VN_recovery)
19 print (VN death)
```

```
[0122/031104.003:INFO:CONSOLE(3)] "Unrecognized feat
.1.js(3)
[0122/031104.190:INFO:CONSOLE(1)] "WARNING: Multiple
[0122/031104.266:INFO:CONSOLE(1)] "WARNING: Multiple
2.088.221
1.794.924
System check identified no issues (O silenced).
```



Tóm tắt lý thuyết và bài tập thực hành

Trong bài học này, chúng ta đã tìm hiểu thư viện selenium để lấy dữ liệu và BeautifulSoup để phân tích cú pháp HTML.

Bài tập 1. Em hãy nêu trang web nguồn chúng ta lấy số liệu dịch Covid-19 tại Việt Nam.

Bài tập 2. Em hãy tìm trang web có thông tin dịch Covid-19 trên thế giới, sau đó phân tích và lập trình lấy số liệu số ca nhiễm, số ca khỏi và số ca tử vong tương ứng.

108