STAGE 1. 정적페이지와 동적페이지

기존의 requets와 BeautifulSoup을 이용해서 데이터를 수집할 때는 보통

stores = html.select("div.lsnx")

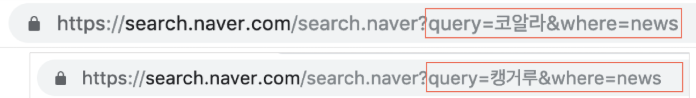
이런 식으로 컨테이너를 수집한다.

하지만 출력해보면 빈 리스트가 출력된다.

이는 네이버 지도가 동적페이지이기 때문이다.

정적페이지는 URL에 따라 정해진 페이지를 불러온다.

정적페이지의 URL에는 데이터의 정보가 포함되어있다



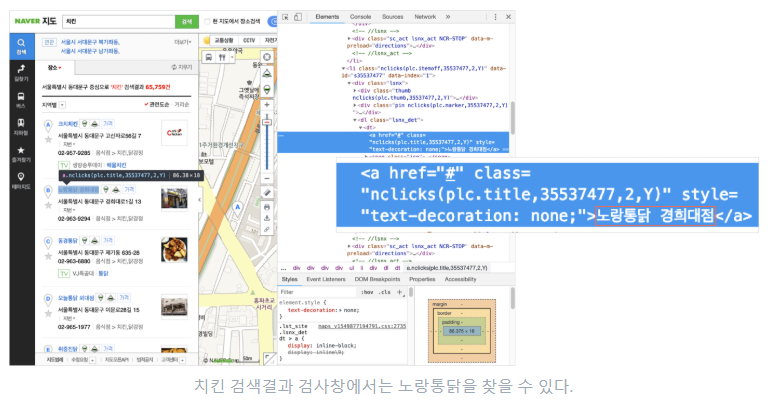
동적페이지는 URL에 관계없이 웹페이지 위에서 데이터를 불러온다.

동적페이지인 네이버 지도는 검색창에

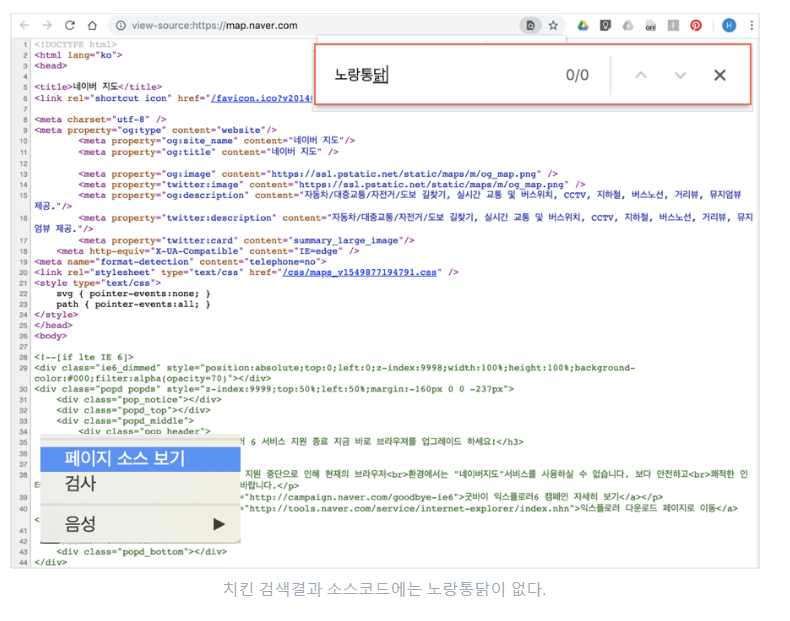


치킨을 쳐도 URL이 변하지 않는다

<동적페이지에서 데이터를 수집할 수 없는 이유>

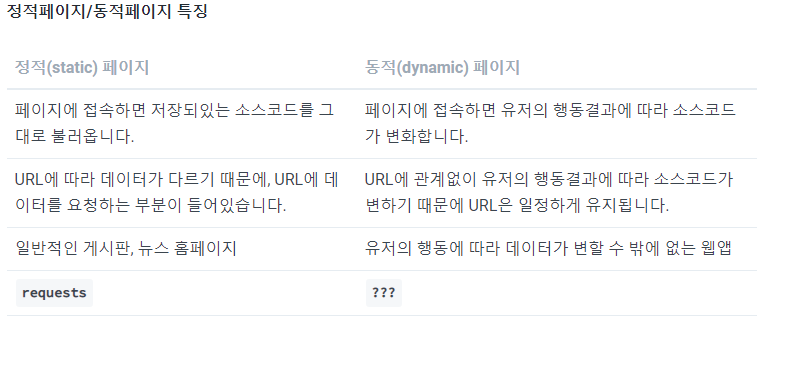


네이버지도에서 검색을 한 후 검사창을 이용하면 소스코드에 해당하는 웹페이지 요소의 위치를 확인할 수 있지만, 검사창 위의 소스코드는 이미 데이터가 가져와진 소스코드이다.



하지만 동적페이지로 페이지를 구성하는 경우 실제로 서버에 저장되어 있는 소스코드에는 우리가 원하는 데이터가 저장되어있지 않는다. requests 방법으로는 URL

(개인적으로 느끼기엔 컴파일타임, 런타임 생각하면 이해가 쉬울 듯? C/C++에서 동적할당은 런타임 시간에 메모리 할당이 이루어지고 동적페이지는 런타임 시간에 데이터를 가져온다는 느낌으로?)



Stage 2. Selenium을 통해 데이터 수집하기

셀레니움은 브라우저 자동화 도구

1. 셀레니움 설치

2. webdriver 설치 >> 크롬 버전에 따라서 설치

3. webdriver(chromedriver)를 작업하려는 소스코드와 같은 위치에 놓기

우리가 네이버 지도에 들어가서 검색창에 목적지를 입력하는 과정을 구체적으로 절차화해보면

1) 웹브라우저를 켜고

2) 네이버지도에 들어가고

3) 네이버 지도 검색창에 목적지를 입력하고

4) 검색

의 과정을 거친다.

이를 코드로 나타내면 다음과 가다

# 네이버 지도 데이터 수집하기

from selenium import webdriver

# 1) 웹브라우저를 켜고

driver = webdriver.Chrome("./chromedriver")

# 2) 구버전 네이버지도 접속

driver.get("https://v4.map.naver.com")

# 3) 검색창에 검색어 입력하기 // 검색창: input#search-input

search\_box = driver.find\_element\_by\_css\_selector("input#search-input")

search\_box.send\_keys("치킨")

# 4) 검색버튼 누르기 // 검색버튼: button.spm

search\_button = driver.find\_element\_by\_css\_selector("button.spm")

search\_button.click()

여기서 find\_element\_by\_css\_selector는 requests에서 select\_one과 비슷하고 find\_elements\_by\_css\_selector는 selet와 비슷하다.(원하는 요소를 리스트로 저장해줌)

텍스트 입력은 send.keys를 이용하고 클릭가능한 요소를 클릭하고 싶을땐 .click함수를 사용하면 된다



<검색결과 수집하기?

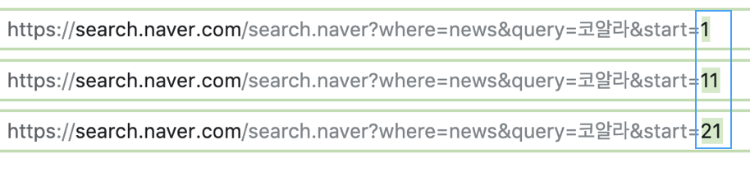
검색결과를 수집하는건 requests때 처럼 컨테이너를 수집하고 >> 세부데이터를 수집하는 방향으로 가면 된다. 방법은 거의 같고 select/ select\_one만 find\_elements\_by\_css\_selector, find\_element\_by\_css\_selector로 바꿔주면 된다

주의할 점은 데이터가 완전히 불러오기전에 데이터를 읽어오려할 때가 있기 때문에 지연시간을 주어 데이터가 완전히 로드된 이후 읽어와야한다. time 라이브러리의 sleep함수를 쓰면된다

수집이 완료됬으면 드라이버를 닫아주는데 driver.close()함수를 쓰면 자동으로 닫아준다

Stage3. 페이지 변경하며 데이터 수집하기

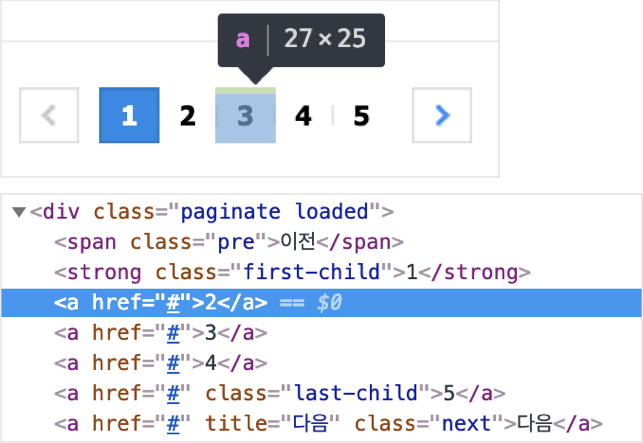
정적페이지에서는 URL의 페이지를 나타내는 숫자부분을 변경해서 데이터를 수집했는데



정적페이지에선 URL이 변하지 않기 때문에 이 방법을 사용할 수 없다.

그래서 검색결과 하단의 페이지 버튼을 클릭하여 데이터를 수집해야한다.

페이지 버튼을 표현하고 있는 부분을 찾아보면 <div class = “paginate loaded”>…</div>에 저장되어있음을 알 수 있다. 다만 페이지버튼의 코드가 <a>와 <strong>이 섞여있어 어떤 태그를 선택해야하는지 애매하다.



이때, div.paginate > \* 를 선택자로 이용하면 div.paginate의 모든 태그를 선택할 수 있다. 그 후 find\_elements\_by\_css\_selector로 버튼들을 리스트 형태로 저장해준다.

<2페이지 이동하기>

page\_bar = driver.find\_elements\_by\_css\_selector("div.paginate > \*")

page\_bar[2].click()

위와 같이 page\_bar를 정의해주면 page\_bar 리스트의 0번째에는 <(이전) 버튼이, 1번째에는 1버튼이 …6번째에는 >(다음)버튼의 태그들이 저장된다

2페이지로 이동하려면 2페이지 버튼을 클릭해주면 된다.

<5페이지까지 반복하기>

반복문을 사용해서 1페이지부터 5페이지까지의 데이터를 모두 수집할 수 있다.

for n in range(1, 5):

    time.sleep(1)

    # 컨테이너(가게 정보) 수

    stores = driver.find\_elements\_by\_css\_selector("div.lsnx")

    for store in stores:

        # 세부 데이터 수집

        name = store.find\_element\_by\_css\_selector("dt > a").text

        addr = store.find\_element\_by\_css\_selector("dd.addr").text

        phone = store.find\_element\_by\_css\_selector("dd.tel").text

        print(name, addr, phone)

    page\_bar = driver.find\_elements\_by\_css\_selector("div.paginate > \*")

    page\_bar[n+1].click()

Stage4. Selenium 데이터 수집기 완성하기

Stage3에서 반복문을 활룡해 데이터수집기를 완성했으나 이 방식으로는 6페이지 이후의 데이터를 수집할 수 없다.



6페이지부터 데이터를 수집해주려면 >(다음페이지)버튼을 클릭해서 6 7 8 9 10가 있는 페이지로 넘어가줘야 한다.

이를 위해서 %(나머지 연산자)를 활용하면 페이지가

1. 현재페이지n가 5의 배수가 아닌경우

2. 현재페이지n가 5의 배수인 경우

의 두가지로 나누어질 수 있다.

1번의 경우 n+1페이지 버튼을 클릭해주고 2번의 경우 >(다음페이지)버튼을 클릭해준다.

경우를 나누어서, 즉 분기문을 이용해서, 즉 if문을 사용하면 현재 페이지에 따라 다른 코드를 실행 할 수 있다.

     # 현재 페이지(n)가 5의 배수가 아닌 경우 n%5+1 페이지 버튼 클릭

    if n%5 != 0:

        page\_bar[n%5+1].click()

    # 현재 페이지(n)가 5의 배수인 경우 다음페이지 버튼을 클릭

    else:

        page\_bar[6].click()

<전화번호가 없는 경우 에러처리>

stage3에서 만들었던 데이터수집기의 문제는 한가지 더 있다.

공부했던 사례는 전화번호가 없는 경우 dd.tel을 찾을 수 없어서 no such element 에러가 발생한다는 것이다. 일반화해보면 페이지에 수집하려는 데이터가 없어서 선택자를 이용해서 데이터를 가져올 수 없는 문제이다.

이 경우 try-except문을 이용해 문제를 해결할 수 있다.

**# 전화번호 수집을 시도합니다.**

try:

    phone = store.find\_element\_by\_css\_selector("dd.tel").text

**# 전화번호 수집에서 에러가 발생하면 phone에 "전화번호 없음"을 저장합니다.**

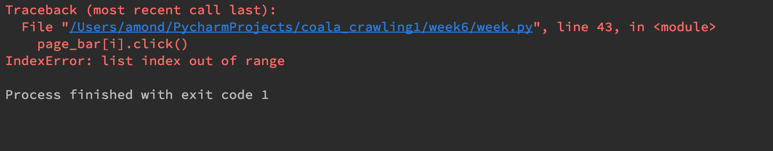
except:

    phone = "전화번호 없음"

다만 except문에 어떤 Error인지 명시해주지 않으면 No such element Error뿐만이 아니라 모든 에러에 대해서 excpet문 이하가 실행되게 된다. 이 경우

**except** NoSuchElementException: 로 바꿔주면 No such element Error에 대해서만 예외처리가 가능하다.

<다음 페이지가 없는 경우 에러처리>

페이지를 넘어가면서 데이터를 수집하게 되면 언젠가는 페이지의 끝에 도달하게 된다. 

이렇게 다음페이지를 클릭하려는 부분에서 list index out of range에러가 뜬다.

역시 try-except문을 이용해서 예외처리를 해줄 수 있다.

**try** :  
 **if** n % 5 != 0:  
 page\_bar[n%5 + 1].click()  
 **else** :  
 page\_bar[6].click()  
**except** IndexError:  
 print(**"수집완료"**)  
 **break**

Index out of range Error만을 골라서 예외처리해주기 위해 except IndexError를 사용했다.