Selenium Web driver

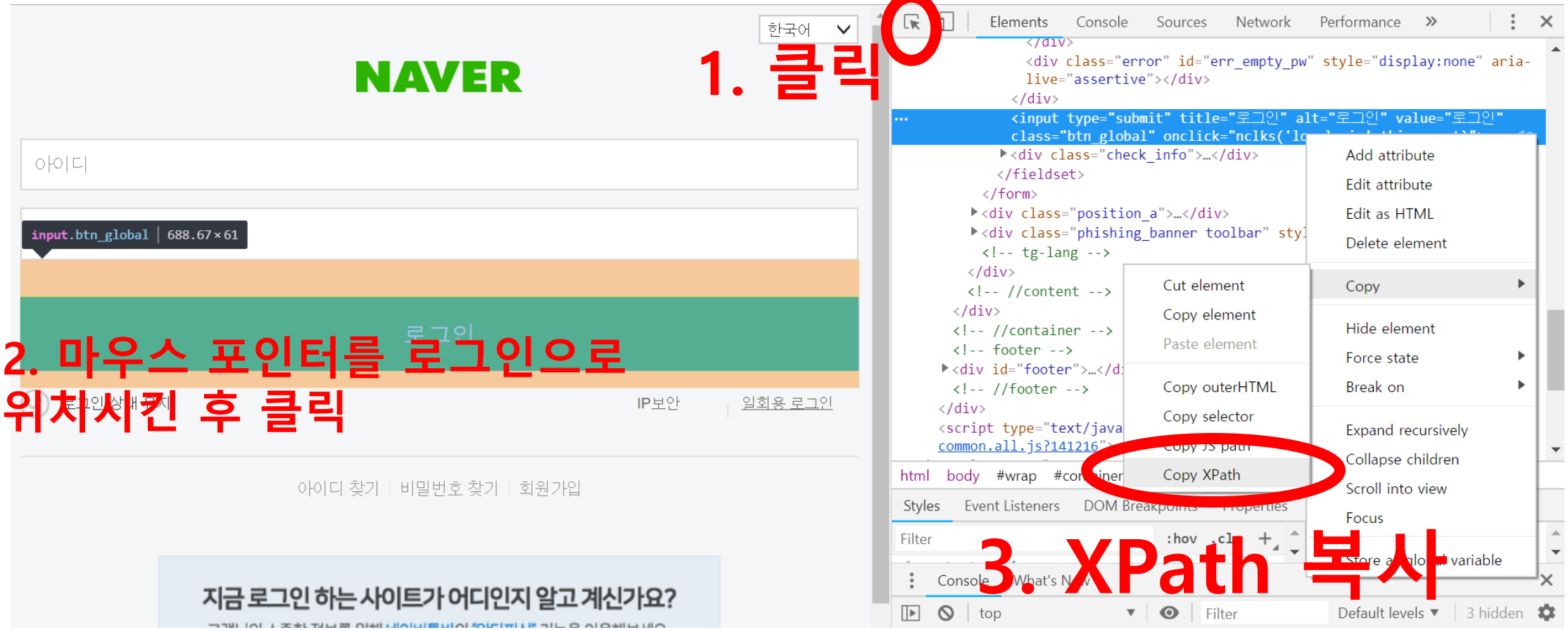
2019. 2. 12

1. 키보드, 마우스 제어

- 키보드 마우스 제어를 이용해서 네이버 로그인
  1. Selenium Webdriver를 이용하여 네이버 접속
  2. Naver Login 클릭 (마우스 제어)
  3. 로그인 페이지에서 ID, PW 입력 (키보드 제어)
  4. 로그인 클릭 (마우스 제어)

- XPath를 이용해서 네이버 로그인
- BeautifulSoup(bs4)은 XPath를 지원하지 않지만, Selenium을 포함한 파이썬의 많은 라이브러리에서 XPath를 지원함

1. 키보드, 마우스 제어


NAVER
1. 클릭
2. 마우스 포인터를 로그인으로 위치시킨 후 클릭
3. XPath 복사
input.btn_global | 688.67×61
Copy XPath

1. 키보드, 마우스 제어 (XPath 사용)

```
1   # Naver Login Using xpath
2   from selenium import webdriver
3
4   url = 'https://naver.com'
5
6   driver = webdriver.Chrome('chromedriver')
7   driver.get(url)
8
9   driver.find_element_by_xpath('//*[@id="account"]/div/a').click()
10
11  driver.find_element_by_xpath('//*[@id="id"]').send_keys('your_ID')
12  driver.find_element_by_xpath('//*[@id="pw"]').send_keys('your_password')
13
14  submit_login = driver.find_element_by_xpath('//*[@id="frmNIDLogin"]/fieldset/input')
15  submit_login.click()
```

각 XPath를 붙여넣기

1. 키보드, 마우스 제어

- 편의점 주소 크롤링
  1. Selenium Webdriver를 이용하여 네이버 지도 접속
  2. 네이버 지도 검색 클릭 (마우스 제어)
  3. '씨유' 입력 후 검색 (키보드 제어)
  4. 주소 크롤링

- 키보드의 특수키를 입력하기 위해선 Keys 사용
- 특수키 종류는 파이썬으로 배우는 웹 크롤러 (박정태)
  P279-280 참고

```python
# Selenium을 이용한 키보드 제어, 마우스 제어
# 편의점 주소 크롤링
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time

driver = webdriver.Chrome('chromedriver')
url = 'https://map.naver.com'
driver.get(url)

search = driver.find_element_by_css_selector('input#search-input')
search.send_keys('씨유')
search.send_keys(Keys.ENTER)

time.sleep(0.5)

for j in range(0,5):
    time.sleep(2)
    page = driver.find_elements_by_css_selector('div.paginate a')
    targets = driver.find_elements_by_class_name('lsnx_det')

    for target in targets:
        names = target.find_element_by_css_selector('dt a').text
        address = target.find_element_by_css_selector('dd.addr').text
        print(names, address)

    print('\n')
    page[j].click()
```

```python
count = 0
while count < 2:
    for j in range(1,6):
        time.sleep(2)
        page = driver.find_elements_by_css_selector('div.paginate a')
        targets = driver.find_elements_by_class_name('lsnx_det')

        for target in targets:
            names = target.find_element_by_css_selector('dt a').text
            address = target.find_element_by_css_selector('dd.addr').text
            print(names, address)
        print('\n')

        page[j].click()
    count += 1
```

결과

```
DevTools listening on ws://127.0.0.1:63221/devtools/browser/0a04c8e4
CU 코리아나호텔점 서울특별시 중구 세종대로 135 코리아나호텔 지번
CU 중구세종대로점 서울특별시 중구 세종대로 135-9 지번
CU 중구무교점 서울특별시 중구 다동길 16 지번
CU 시청광장점 서울특별시 중구 을지로 6 재능빌딩 지번
CU 광화문광장점 서울특별시 종로구 새문안로 103-1 지번
CU SK서린사옥점 서울특별시 종로구 종로 26 SK빌딩 지하1층 지번
CU 한화빌딩점 서울특별시 중구 세종대로 92 지번
CU 광화문D타워점 서울특별시 종로구 종로3길 17 D타워 지번
CU 종로1가점 서울특별시 종로구 종로 19 르메이에르종로타운1 지번
CU 중구시청역점 서울특별시 중구 세종대로 83 지번
```

2. 저장된 로그인 쿠키 사용

- **쿠키**(cookie)란 인터넷 사용자가 어떠한 웹사이트를 방문할 경우 그 사이트가 사용하고 있는 서버를 통해 인터넷 사용자의 컴퓨터에 설치되는 작은 기록 정보 파일을 일컫는다.(위키백과)

- 네이버의 경우, 로그인 시도가 많아지면 자동 로그인 방지를 위해 문자 입력 요구

- 로그인 된 쿠키를 사용함으로써, 문자 입력 요구없이 로그인 상태로 접속

3. Wait 기능 사용

- 같은 코드를 실행하더라도, 웹 페이지의 로딩 속도 차이에 따라 크롤링이 안되는 경우가 발생
- Selenium의 wait 기능을 이용해서, 웹 페이지가 모두 로드된 후 크롤링 시작
- 또는 Python의 time 라이브러리의 sleep 함수를 이용해서 웹 페이지가 로드되는 일정 시간 동안 코드를 멈추게 한 뒤, 그 후에 크롤링 시작
- Implicit Wait는 작동이 잘 안되고, Explicit Wait는 코드가 복잡함. time의 sleep 함수 사용이 가장 간편하고 적용이 쉬움

3. Wait 기능 사용

```python
# 롯데시네마
from selenium import webdriver
import time

driver = webdriver.Chrome('chromedriver')
url = "http://www.lottecinema.co.kr/LCHS/Contents/Cinema/Cinema-Detail.aspx?divisionCode=1&detailDivisi


driver.get(url)
#driver.implicitly_wait(3)
#time.sleep(3)

items = driver.find_element_by_class_name('time_inner').text
print(items)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
15극한직업
2D
5관
16:45
327석 / 332석
4관
17:15
132석 / 132석
1관
18:00
```

```python
# 롯데시네마
from selenium import webdriver
import time

driver = webdriver.Chrome('chromedriver')
url = "http://www.lottecinema.co.kr/LCHS/Contents/Cinema/Cinema-Detail.aspx?divisionCode=1&detailDivisi


driver.get(url)
#driver.implicitly_wait(3)
#time.sleep(3)

items = driver.find_element_by_class_name('time_inner').text
print(items)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
상영시간이 조회되지 않았습니다.
```

같은 코드라도 웹 페이지 로딩 속도에 따라 결과가 다르게 나타남

3. Explicit Wait

```
1   # 롯데시네마
2   from selenium import webdriver
3   from selenium.webdriver.common.by import By
4   from selenium.webdriver.support.ui import WebDriverWait
5   from selenium.webdriver.support import expected_conditions as EC
6   #import time
7
8   driver = webdriver.Chrome('chromedriver')
9   url = "http://www.lottecinema.co.kr/LCHS/Contents/Cinema/Cinema-Detail.aspx?divisionCode=1&detailDivisi
10
11  driver.get(url)
12
13  #time.sleep(3)
14
15  items = WebDriverWait(driver, 3) \
16          .until(EC.presence_of_element_located((By.CSS_SELECTOR, 'div.time_inner')))
17  print(items.text)
```

1. 여러가지 라이브러리 Import

2. Selenium Webdriver를 이용하여 롯데시네마 접속

3. Div 태그의 time_inner 클래스를 찾을 때까지 3초 대기

3. time.sleep 이용

```
1   # 롯데시네마
2   from selenium import webdriver
3   import time
4
5   driver = webdriver.Chrome('chromedriver')
6   url = "http://www.lottecinema.co.kr/LCHS/Contents/Cinema/Cinema-Detail.aspx?divisionCode=1&detailDivisi
7
8   driver.get(url)
9   time.sleep(3)
10
11  items = driver.find_element_by_class_name('time_inner')
12  print(items.text)
```

time 라이브러리 import

코드를 3초간 정지시킨 뒤, 다시 실행시킴

4. Headless Chrome (Phantom JS)

- Selenium Webdriver를 사용하면 웹 브라우저(크롬)를 띄운 뒤 Selenium 동작
- Headless Chrome을 사용하면 웹 브라우저를 띄우지 않고도 Selenium 사용 가능
- Phantom JS도 Headless Chrome과 같은 기능을 하는 도구이지만, 2018년 기준으로 더 이상 개발되지 않음

  (http://phantomjs.org/download.html에서 다운해서 사용)

4. Headless Chrome (Phantom JS)

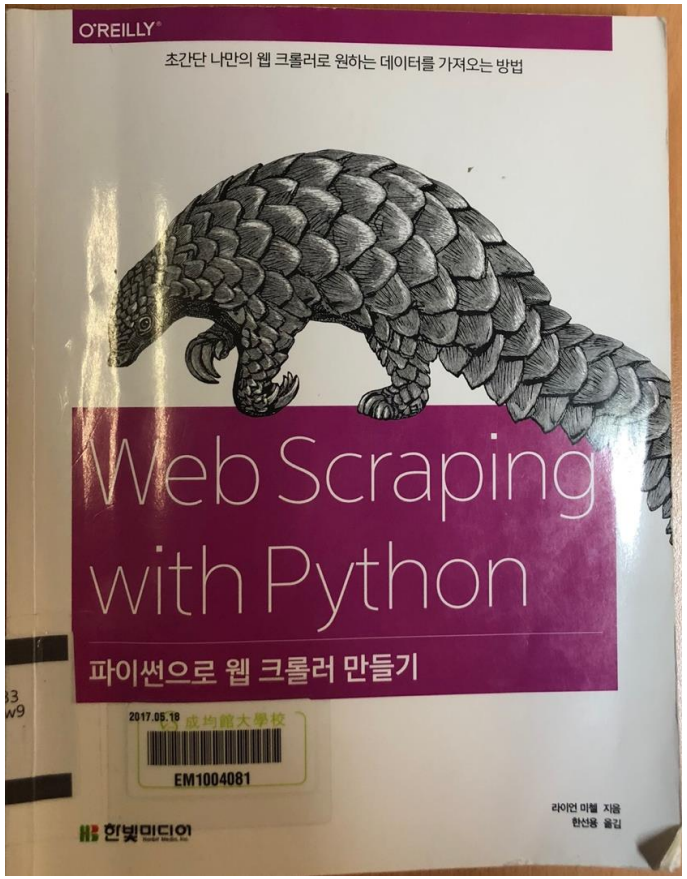
```
1   # 롯데시네마 Phantom JS 사용
2   from selenium import webdriver
3
4   driver = webdriver.PhantomJS()
5
6   url = "http://www.lottecinema.co.kr/LCHS/Contents/Cinema/Cinema-Detail.aspx?divisionCode=1&detailDivis
7   driver.get(url)
8
9   items = driver.find_element_by_class_name('time_inner')
10  print(items.text)
11
12  driver.close()
```

Webdriver.Chrome 대신
Webdriver.PhantomJS 사용

5. 참고 자료

- 파이썬으로 웹 크롤러 만들기(라이언 미첼), 2016, 한빛미디어

- P191~197: Phantom JS, Explicit Wait, XPath에 관한 간략한 설명 포함
- P225~227: 쿠키 처리에 관한 간략한 설명



5. 참고 웹 사이트

- Selenium with Python:

  https://selenium-python.readthedocs.io/

- XPath 사용

  https://wkdtjsgur100.github.io/selenium-xpath/

- Implicit Wait, Explicit Wait

  https://beomi.github.io/2017/10/29/HowToMakeWebCrawler-ImplicitWait-vs-ExplicitWait/

- Headless Chrome

  https://beomi.github.io/2017/09/28/HowToMakeWebCrawler-Headless-Chrome/