



데이터 과학 기반의 파이썬 빅데이터 분석

Week5 파이썬 크롤링 – 라이브러리 이용

목차

01 정적 웹 페이지 크롤링

02 동적 웹 페이지 크롤링

학습목표

API를 제공하지 않는 웹 페이지를 크롤링할 수 있다.

BeautifulSoup 라이브러리로 정적 웹 페이지를 크롤링할 수 있다.

Selenium 라이브러리로 동적 웹 페이지를 크롤링할 수 있다.

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 준비

■ BeautifulSoup 연습하기 1

1. BeautifulSoup 라이브러리를 사용하기 위해 추가 설치작업을 실시:
명령 프롬프트 창에서 pip 명령을 사용

```
C:\W> pip install beautifulsoup4
```

2. 설치가 끝나면 파이썬 셸 창에서 BeautifulSoup을 임포트하여 사용

```
>>> from bs4 import BeautifulSoup
```

3. 연습용 html을 작성

```
>>> html = '<h1 id="title">한빛출판네트워크</h1><div class="top"><ul  
class="menu"><li><a href=http://www.hanbit.co.kr/member/login.html  
class="login">로그인 </a></li></ul><ul class="brand"><li><a href="http://www.  
hanbit.co.kr/media/>한빛미디어</li><a href="http://www.hanbit.co.kr/  
academy/">한빛아카데미</a></li></ul></div>'
```

4. BeautifulSoup 객체를 생성

```
>>> soup = BeautifulSoup(html, 'html.parser')
```

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 준비

■ BeautifulSoup 연습하기 1

5. 객체에 저장된 html 내용을 확인

```
>>> print(soup.prettify())
<h1 id="title">
한빛출판네트워크
</h1>
<div class="top">
<ul class="menu">
<li>
<a class="login" href="http://www.hanbit.co.kr/member/login.html">
로그인
</a>
</li>
</ul>
<ul class="brand">
<li>
<a href="http://www.hanbit.co.kr/media/">
한빛미디어
</a>
</li>
<li>
<a href="http://www.hanbit.co.kr/academy/">
한빛아카데미
</a>
</li>
</ul>
</div>
```

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 준비

■ BeautifulSoup 연습하기 2

1. 태그 파싱하기 - 지정된 한 개의 태그만 파싱

```
>>> soup.h1
<h1 id="title">한빛출판네트워크</h1>
>>> tag_h1 = soup.h1
>>> tag_h1
<h1 id="title">한빛출판네트워크</h1>
>>> tag_div = soup.div
>>> tag_div
<div class="top"><ul class="menu"><li><a class="login" href="http://www.hanbit.co.kr/member/login.html">로그인</a></li></ul><ul class="brand"><li><a href="http://www.hanbit.co.kr/media/">한빛미디어</a></li><li><a href="http://www.hanbit.co.kr/academy/">한빛아카데미</a></li></ul></div>
>>> tag_ul = soup.ul
>>> tag_ul
<ul class="menu"><li><a class="login" href="http://www.hanbit.co.kr/member/login.html">로그인</a></li></ul>
>>> tag_li = soup.li
>>> tag_li
<li><a class="login" href="http://www.hanbit.co.kr/member/login.html">로그인</a></li>
>>> tag_a = soup.a
>>> tag_a
<a class="login" href="http://www.hanbit.co.kr/member/login.html">로그인</a>
```

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 준비

■ BeautifulSoup 연습하기 2

2. 태그 파싱하기 - 지정된 태그를 모두 파싱

```
>>> tag_ul_all = soup.find_all("ul")
>>> tag_ul_all
[<ul class="menu"> <li> <a class="login" href="http://www.hanbit.co.kr/member/login.html">로그인
</a> </li> </ul>, <ul class="brand"> <li> <a href="http://www.hanbit.co.kr/media/">한빛미디어</a> </li> <li> <a
href="http://www.hanbit.co.kr/academy/">한빛아카데미</a> </li> </ul>]
>>> tag_li_all = soup.find_all("li")
>>> tag_li_all
[<li> <a class="login" href="http://www.hanbit.co.kr/member/login.html">로그인</a> </li>, <li> <a
href="http://www.hanbit.co.kr/media/">한빛미디어</a> </li>, <li> <a href="http://www.hanbit.co.kr/academy/">
한빛아카데미</a> </li>]
>>> tag_a_all = soup.find_all("a")
>>> tag_a_all
[<a class="login" href="http://www.hanbit.co.kr/member/login.html">로그인</a>, <a
href="http://www.hanbit.co.kr/media/">한빛미디어</a>, <a href="http://www.hanbit.co.kr/academy/">한빛아카데
미</a>]
```

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 준비

■ BeautifulSoup 연습하기 2

3. 속성을 이용하여 파싱

- ❶ attrs: 속성 이름과 속성값으로 딕셔너리 구성
- ❷ find(): 속성을 이용하여 특정 태그 파싱
- ❸ select(): 지정한 태그를 모두 파싱하여 리스트 구성 태그#id 속성값 / 태그.class 속성값

```
>>> tag_a.attrs
{'href': 'http://www.hanbit.co.kr/member/login.html', 'class': ['login']}
>>> tag_a['href']
'http://www.hanbit.co.kr/member/login.html'
>>> tag_a['class']
['login']
>>> tag_ul_2 = soup.find('ul', attrs={'class': 'brand'})
>>> tag_ul_2
<ul class="brand"><li><a href="http://www.hanbit.co.kr/media/">한빛미디어</a></li><li><a href="http://www.hanbit.co.kr/academy/">한빛아카데미</a></li></ul>
>>> title = soup.find(id="title")
>>> title
<h1 id="title">한빛출판네트워크</h1>
>>> title.string
'한빛출판네트워크'
>>> li_list = soup.select("div>ul.brand>li")
>>> li_list
[<li><a href="http://www.hanbit.co.kr/media/">한빛미디어</a></li>, <li><a href="http://www.hanbit.co.kr/academy/">한빛아카데미</a></li>]
>>> for li in li_list: [Enter]
    print(li.string) [Enter]
    [Enter]
한빛미디어
한빛아카데미
```


01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

■ 크롤링 허용 여부 확인하기

- 웹 페이지를 크롤링하기 전에 크롤링 허용 여부를 확인하기 위해 주소 창에 '크롤링할 주소/ robots.txt'를 입력
- 만약 robots.txt 파일이 없다면 수집에 대한 정책이 없으니 크롤링을 해도 된다는 의미

표시	허용 여부
User-agent: * Allow: / 또는 User-agent: * Disallow:	모든 접근 허용
User-agent: * Disallow: /	모든 접근 금지
User-agent: * Disallow: /user/	특정 디렉토리만 접근 금지

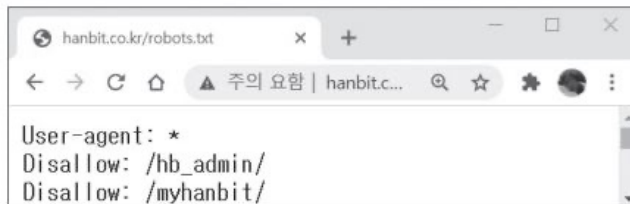


그림 6-1 특정 디렉토리만 접근 금지를 한 예

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

■ 웹 페이지 분석하기

1. 매장 정보 찾기

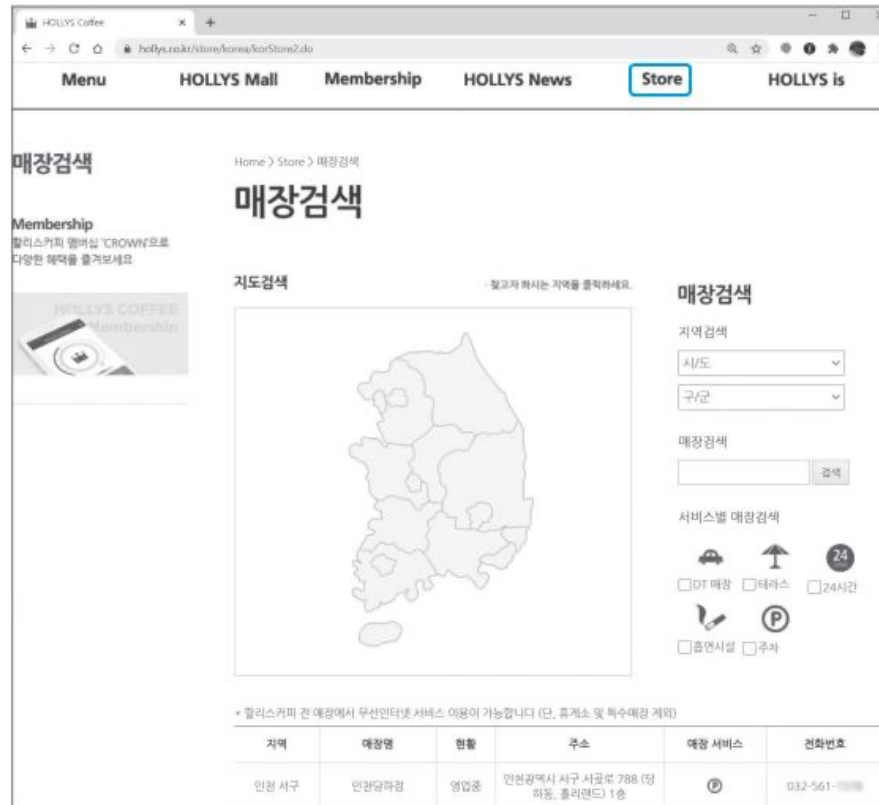


그림 6-2 매장 정보를 찾을 수 있는 매장검색 페이지

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

- 웹 페이지 분석하기
 1. URL 분석하기
 2. HTML 코드 확인하기

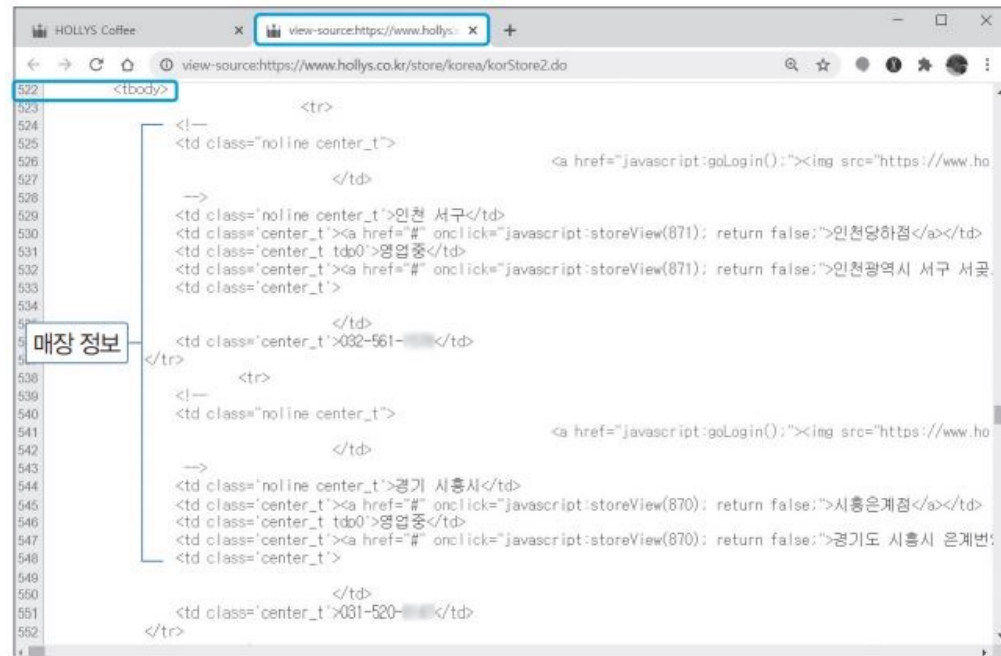


그림 6-3 매장 정보에 대한 HTML 코드

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

- 웹 페이지 분석하기
 - 3. 나머지 매장 정보 확인하기



(a) 매장 정보 2페이지



(b) 매장 정보 2페이지의 HTML 코드

그림 6-4 2페이지 이동 후 HTML 코드 확인

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

■ 웹 페이지 분석하기

4. 'pageNo=' 다음에 페이지 번호를 붙여 다음 페이지를 확인



그림 6-5 마지막 페이지 확인

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

■ 파이썬 셸 창에서 크롤링하기

1. BeautifulSoup과 urllib.request를 импорт

```
>>> from bs4 import BeautifulSoup
>>> import urllib.request
```

2. 작업 결과를 저장할 리스트를 준비

```
>>> result = []
```

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

■ 파이썬 셸 창에서 크롤링하기

3. BeautifulSoup 객체를 생성하여 파싱

- ❶ 1~58페이지까지 반복해서 url 설정 ❷ url 요청하여 응답 받은 웹 페이지 저장
- ❸ BeautifulSoup 객체 생성 ❹ tr 태그 하위의 td 태그 중에서 필요한 항목만 추출하여 result 리스트에 추가 저장

```
>>> for page in range(1,59):
    Hollys_url = 'https://www.hollys.co.kr/store/korea/korStore.do?pageNo=%d&sid0=&gugun=&store=' %page
    print(Hollys_url)
    html = urllib.request.urlopen(Hollys_url)
    soupHollys = BeautifulSoup(html, 'html.parser')
    tag_tbody = soupHollys.find('tbody')
    for store in tag_tbody.find_all('tr'):
        if len(store) <= 3:
            break
        store_td = store.find_all('td')
        store_name = store_td[1].string
        store_sido = store_td[0].string
        store_address = store_td[3].string
        store_phone = store_td[5].string
        result.append([store_name]+[store_sido]+[store_address]+[store_phone]) [Enter]
[Enter]
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=1&sid0=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=2&sid0=&gugun=&store=
...
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=58&sid0=&gugun=&store=
```

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

■ 파이썬 셸 창에서 크롤링하기

4. 크롤링된 내용을 확인

- ❶ 결과가 저장된 result의 원소 개수 확인 ❷ 첫 번째 원소 확인
- ❸ 마지막 원소 확인 ❹ 마지막 매장 정보가 저장되어 있는 store_td의 내용 확인

```
>>> len(result)
566
>>> result[0]
['성남터미널점', '경기 성남시 분당구', '경기도 성남시 분당구 성남대로925번길 16, 성남종합버스터미널 1층', '031-725-0000']
>>> result[565]
['성남점', '경기 성남시 수정구', '경기도 성남시 수정구 수정로 175, 동일빌딩1층', '031-721-0000']
>>> store_td
[<td class="noline center_t">경기 성남시 수정구</td>,<td class="center_t"><a href="#" onclick="javascript:storeView(11); return false;"> 성남점</a></td>,<td class="center_t tdp0">영업중</td>,<td class="center_t"><a href="#" onclick="javascript:storeView(11); return false;">경기도 성남시 수정구 수정로 175, 동일빌딩1층</a></td>,<td class="center_t"></td>,<td class="center_t">031-721-0000</td>]
>>> store_td[1].string
'성남점'
>>> store_td[0].string
'경기 성남시 수정구'
>>> store_td[3].string
'경기도 성남시 수정구 수정로 175, 동일빌딩1층'
>>> store_td[5].string
'031-721-0000'
```


01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

■ 파이썬 셸 창에서 크롤링하기

5. 크롤링한 데이터 저장하기

```
C:\W> pip install pandas
```

6. pandas를 임포트

```
>>> import pandas as pd
```

7. pandas를 사용하여 테이블 형태의 데이터프레임을 생성

```
>>> hollys_tbl = pd.DataFrame(result, columns = ('store', 'sido-gu', 'address', 'phone'))
```

8. 테이블을 CSV 파일로 저장

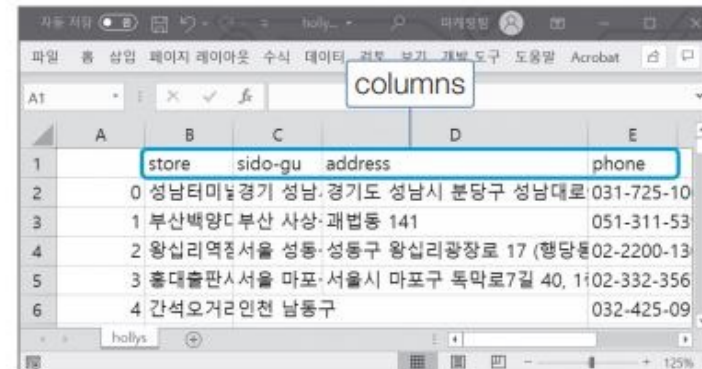
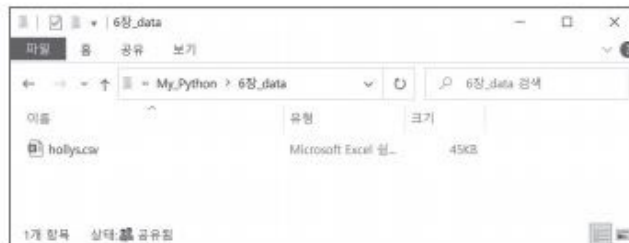
```
>>> hollys_tbl.to_csv("C:/Users/kmj/My_Python/6장_data/hollys.csv", encoding = "cp949", mode = "w", index = True)
```

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

■ 파이썬 셸 창에서 크롤링하기

- My_Python 폴더 하위의 6장_data 폴더 안에 hollys.csv 파일이 생성되었는지 확인



A screenshot of a Microsoft Excel spreadsheet. The columns are labeled 'store', 'sido-gu', 'address', and 'phone'. The data is as follows:

	store	sido-gu	address	phone
0	성남티마	경기	성남시 분당구 성남대로	031-725-10
1	부산백양대	부산	사상·괘법동 141	051-311-53
2	왕십리역	서울	성동·성동구 왕십리광창로 17 (행당동)	02-2200-13
3	홍대출판사	서울	마포·서울시 마포구 독막로7길 40, 11	02-332-356
4	간석오거리	인천	남동구	032-425-09

그림 6-6 생성된 hollys.csv 파일 확인

```
#[CODE 0]
def main():
    result = []
    print('Hollys store crawling >>>>>>>>>>>>>>>>>>>>>>')
    hollys_store(result) #[CODE 1] 호출
    hollys_tbl = pd.DataFrame(result, columns = ('store', 'sido-gu',
                                                'address','phone'))
    hollys_tbl.to_csv('C:/Users/kmj/My_Python/6장_data/hollys1.csv',
                     encoding = 'cp949', mode = 'w', index = True)
    del result[:]

if __name__ == '__main__':
    main()
```

01. 정적 웹 페이지 크롤링

■ 정적 웹 페이지 크롤링 실습

■ 파이썬 파일 작성하여 크롤링하기

- [F5]를 눌러 실행하면 파이썬 셸 창에 print문의 실행 결과가 출력되고 My_Python/6장_data 폴더에 CSV 파일이 생성



```
Python 3.7.7 Shell
File Edit Shell Debug Options Window Help
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=45&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=46&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=47&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=48&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=49&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=50&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=51&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=52&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=53&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=54&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=55&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=56&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=57&ido=&gugun=&store=
https://www.hollys.co.kr/store/korea/korStore.do?pageNo=58&ido=&gugun=&store=
>>>
Ln: 64 Col: 4
```

그림 6-7 hollysCrawler.py의 실행 결과

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 준비

1. Selenium 라이브러리 설치

```
C:\W> pip install selenium
```

2. WebDriver 다운로드하기

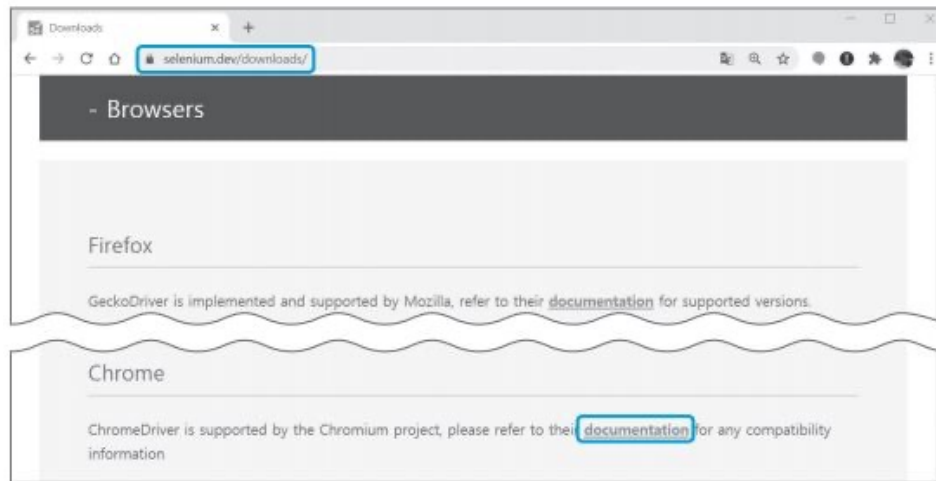


그림 6-8 WebDriver를 다운로드할 페이지 선택

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 준비

3. 다운로드 링크를 클릭

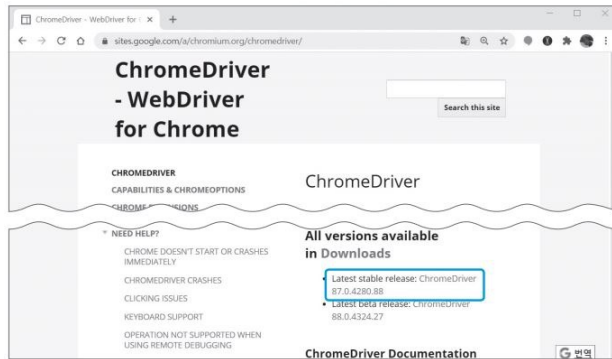


그림 6-9 크롬 웹 브라우저에서 사용할 WebDriver인 ChromeDriver 선택

4. 시스템 운영체제에 맞는 ChromeDriver를 선택하여 다운로드

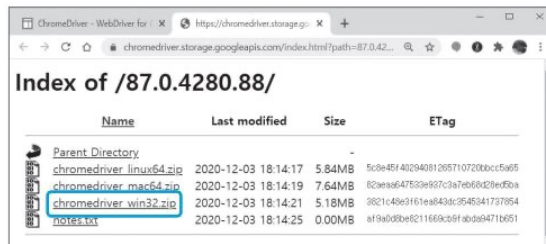


그림 6-10 시스템 운영체제에 맞는 ChromeDriver 다운로드

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 준비

5. My_Python 폴더 하위에 WebDriver 폴더를 만들고 다운로드한 압축 파일을 풀고 'chromedriver.exe' 파일을 WebDriver 폴더로 이동

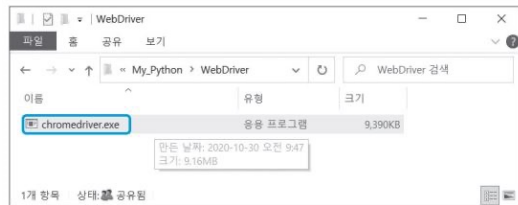


그림 6-11 다운로드한 chromedriver.exe 파일

6. Selenium 라이브러리의 WebDriver를 импорт

```
>>> from selenium import webdriver
```

7. 크롬 WebDriver 객체를 생성

```
>>> wd = webdriver.Chrome('C:/Users/kmj/My_Python/WebDriver/chromedriver.exe')
```

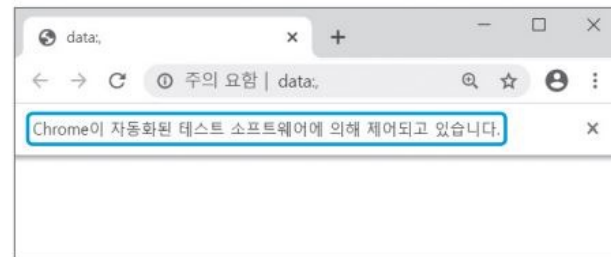
02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 준비

- WebDriver를 설치한 경로를 지정하여 크롬 WebDriver 객체를 생성하면 크롬 WebDriver가 실행



(a) 크롬 WebDriver 실행 창



(b) 크롬 웹 브라우저 창

그림 6-12 크롬 WebDriver 객체 생성

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 준비

8. 파이썬 셸 창에서 다음과 같이 입력하여 Selenium이 제어하는 크롬 창에서 웹 페이지를 열어 확인

```
>>> wd.get("http://www.hanbit.co.kr")
```

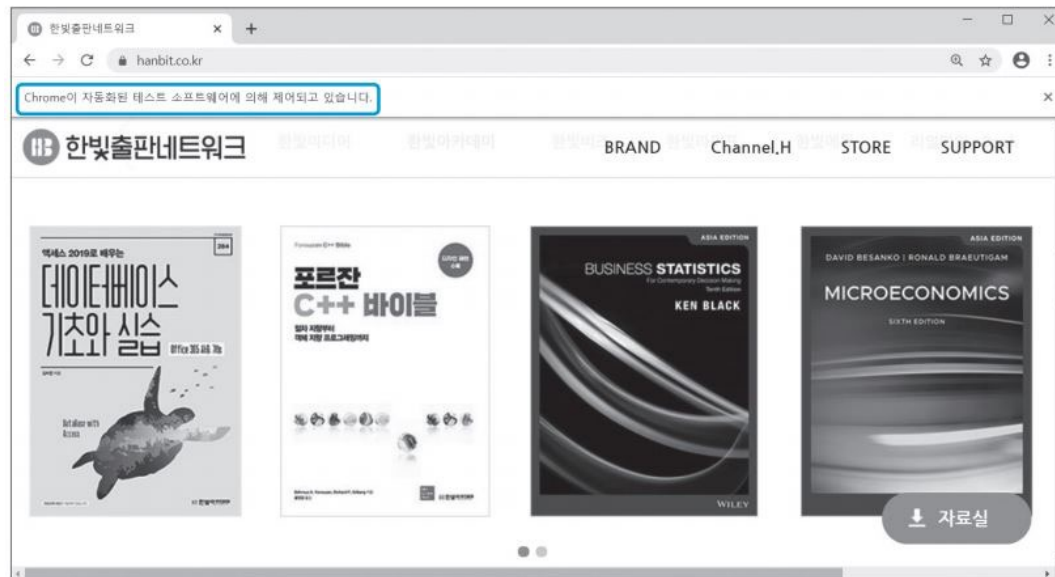


그림 6-13 Selenium이 제어하는 크롬 창에서 웹 페이지 열기

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

■ 웹 페이지 분석하기

1. 매장 정보 찾기

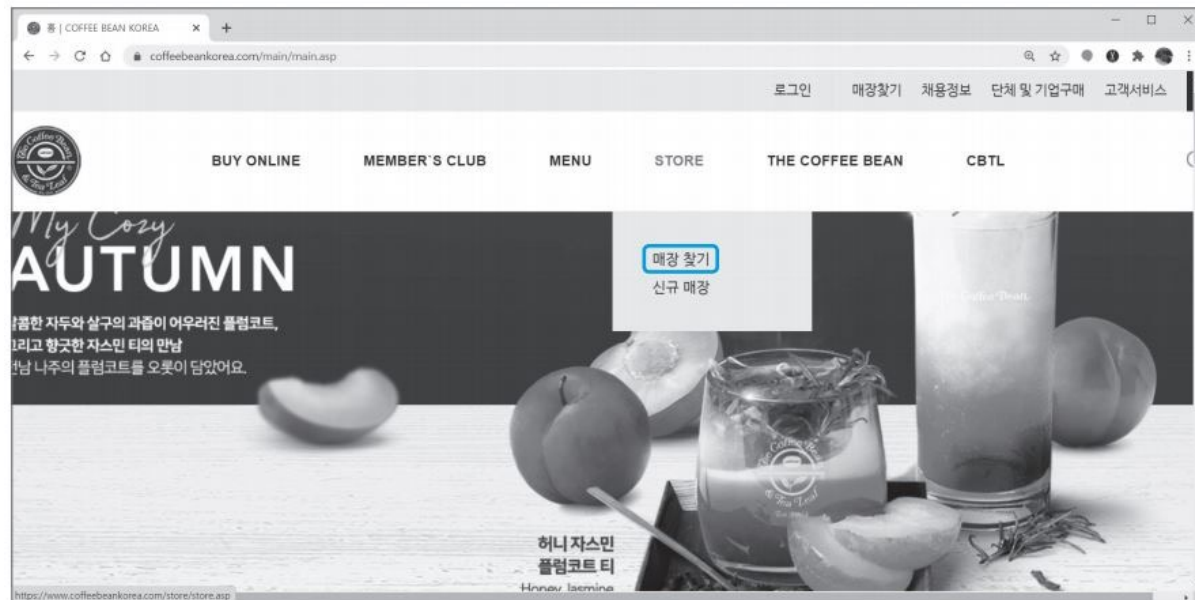


그림 6-14 매장 정보 찾기

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

■ 웹 페이지 분석하기

2. 자바스크립트의 stoneLocal2() 함수 확인하기

- [지역 검색]에서 [서울]을 선택하면 서울에 있는 매장 70개 목록이 표시

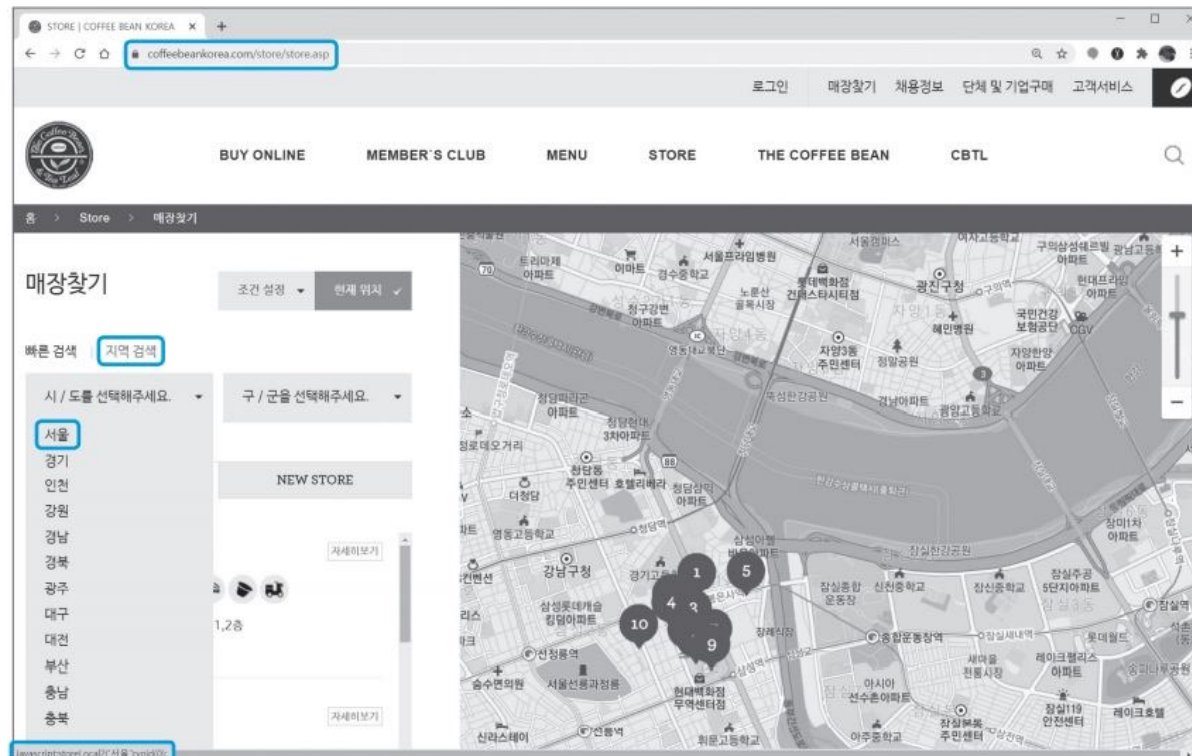


그림 6-15 자바스크립트의 storeLocal2() 함수 확인

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

■ 웹 페이지 분석하기

2. 자바스크립트의 stoneLocal2() 함수 확인하기

- [지역 검색]에서 [서울]을 선택하면 서울에 있는 매장 70개 목록이 표시

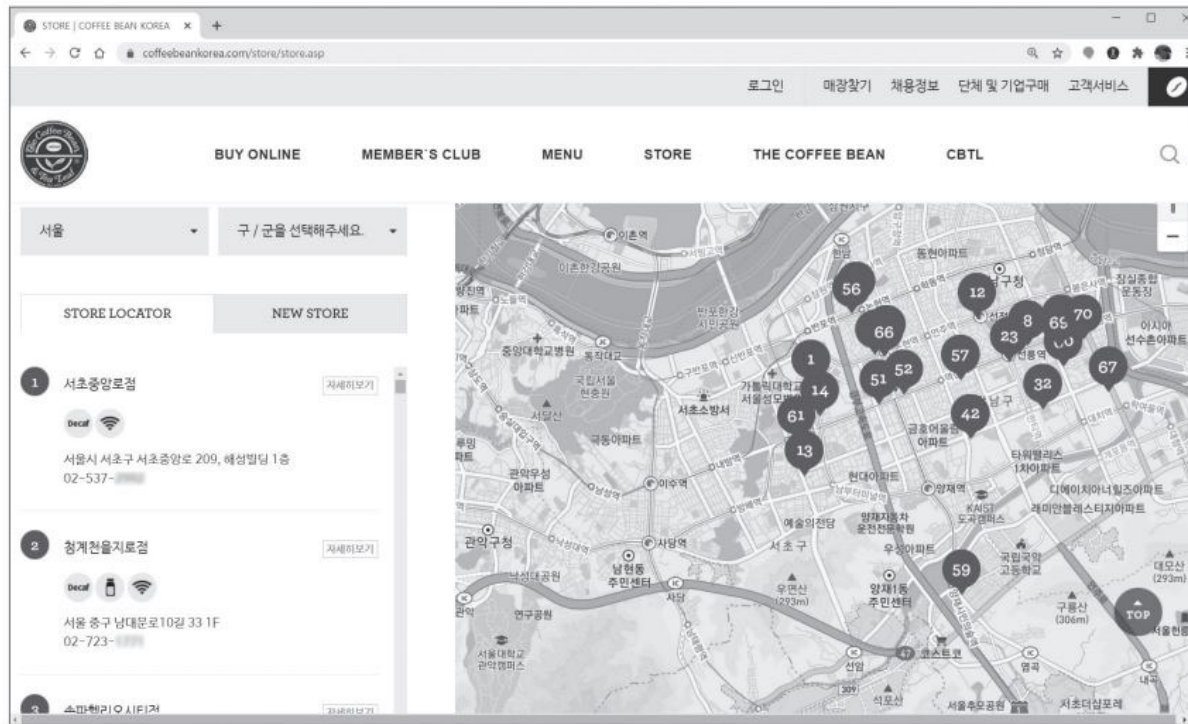


그림 6-16 서울에 있는 매장 목록

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

■ 웹 페이지 분석하기

3. HTML 소스 확인하기

- [Ctrl] + [U]를 눌러 HTML 소스를 확인하면 HTML 소스에는 조회된 매장 목록이 없음을 알 수 있음
- Selenium/WebDriver를 이용하여 파싱해야 함

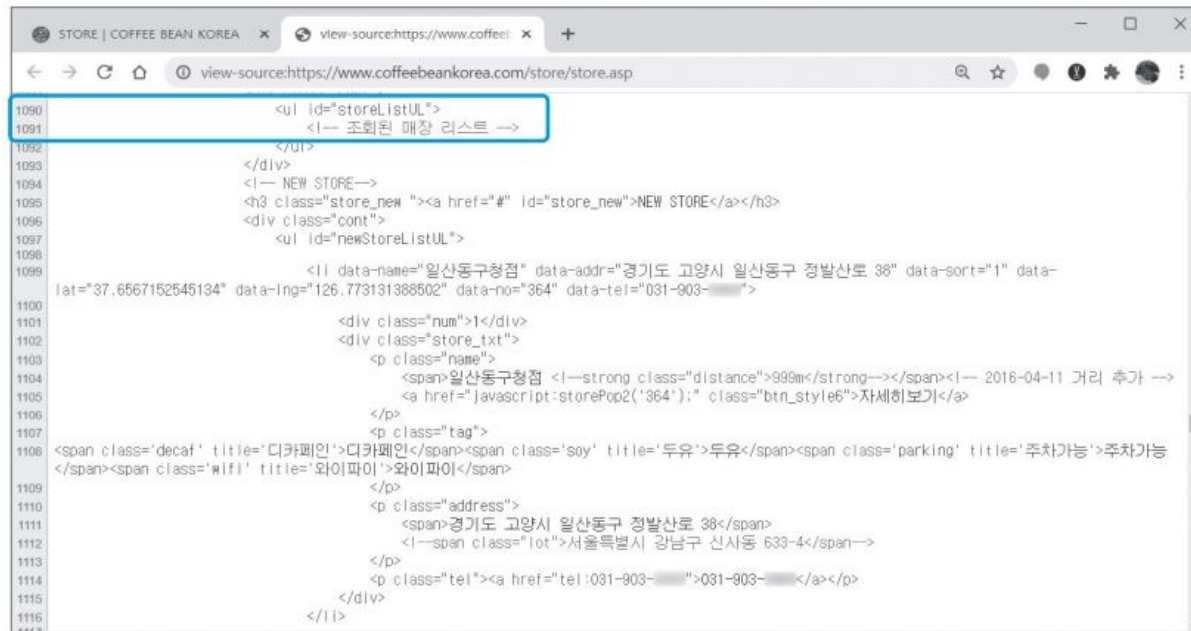


그림 6-17 웹 페이지의 HTML 소스 보기

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

■ 웹 페이지 분석하기

4. 버튼에 연결된 자바스크립트 확인하기



그림 6-18 <자세히보기> 버튼에 연결된 자바스크립트 확인



그림 6-19 자바스크립트를 이용해 팝업 창으로 표시된 매장 정보

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

■ 파이썬 셀 창에서 크롤링하기

1. Selenium 패키지의 WebDriver를 импорт

```
>>> from bs4 import BeautifulSoup
>>> from selenium import webdriver
```

2. 크롬 WebDriver 객체를 생성

```
>>> wd = webdriver.Chrome('C:/Users/kmj/My_Python/WebDriver/chromedriver.exe')
```

3. 웹 페이지를 연결

```
>>> wd.get("https://www.coffeebeankorea.com/store/store.asp")
```

– Selenium/WebDriver를 импорт하여 크롬 WebDriver 객체를 생성

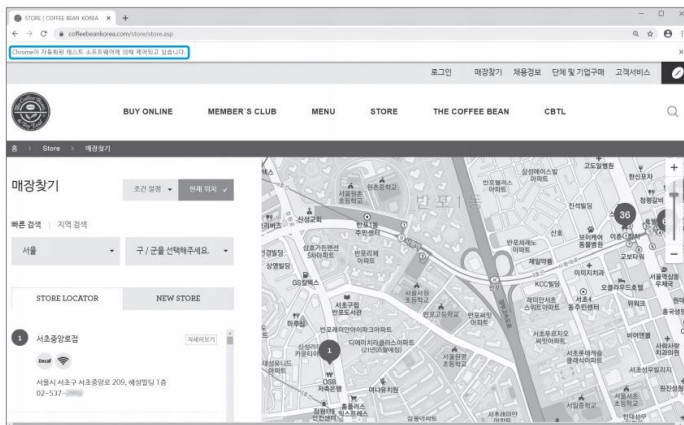


그림 6-20 Selenium이 제어하는 크롬 창에 표시된 웹 페이지

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

■ 파이썬 셀 창에서 크롤링하기

4. 자바스크립트 함수 호출해 매장 정보 페이지 열기

```
>>> wd.execute_script("storePop2(1)")
```

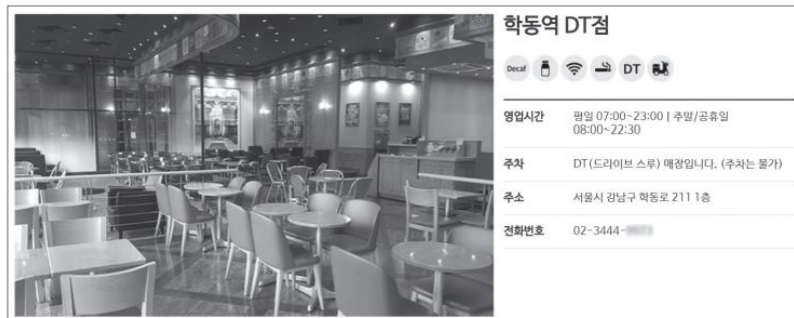


그림 6-21 자바스크립트 함수 호출에 의해 팝업 창으로 나타난 매장 정보

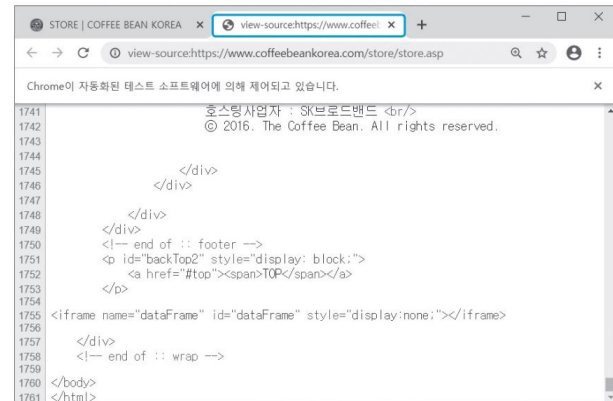


그림 6-22 팝업 창으로 나타난 매장 정보의 소스 보기

5. 자바스크립트 함수가 수행된 페이지의 소스 코드를 저장

```
>>> html = wd.page_source
```

6. BeautifulSoup 객체를 생성

```
>>> soupCB1 = BeautifulSoup(html, 'html.parser')
```


02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

■ 파이썬 셸 창에서 크롤링하기

7. HTML 소스 코드 형태로 출력하여 확인

```
>>> print(soupCB1.prettify())
...
</div>
<!-- end of :: wrap -->
<div class="matizCoverLayer" id="matizCoverLayer0" style="position: fixed;
z-index: 99999; left: 0px; top: 0px; width: 1028px; height: 620px; display:
block;">
<div class="matizCoverLayerBg" id="matizCoverLayer0Bg" style="position: fixed;
background: rgb(0, 0, 0); opacity: 0.75; z-index: 99999; left: 0px; top: 0px;
width: 100%; height: 100%;">
</div>
<div class="matizCoverLayerContent" id="matizCoverLayer0Content"
style="position: fixed; z-index: 99999; left: 34px; top: 50px;">
<div class="store_popup">
...
<div class="store_txt">
  <h2>
    학동역 DT점
  </h2>
  <p class="tag">
    <span class="decaf" title="디카페인">
      디카페인
    </span>
  ...
```

```
<table class="store_table">
  <tbody>
    ...
    <th>
      주소
    </th>
    <td>
      서울시 강남구 학동로 211 1층
    ...
    <th>
      전화번호
    </th>
    <td>
      02-3444-0000
    </td>
    ...
  </tbody>
</table>
```

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

- 파이썬 셀 창에서 크롤링하기
 - 매장 정보가 있는 테이블 부분의 HTML 소스
 - 매장 이름
 - `<div class="store_txt">` 태그 내부의 `<h2>` 태그
 - 매장 주소, 전화번호
 - `<table class="store_table">` 태그 내부의 `<td>` 태그

```
*Python 3.7.7 Shell*
File Edit Shell Debug Options Window Help

<div class="store_txt">
  <h2>
    학동역 DT점 _____ 매장 이름
  </h2>
  <p class="tag">

  <tbody>
    <tr>
      <th>
        영업시간
      </th>
      <td>
        평일 07:00~23:00 | 주말/공휴일 08:00~22:30
      </td>
    </tr>
    <tr>
      <th>
        주차
      </th>
      <td>
        DT(드라이브 스루) 매장입니다. (주차는 불가)
      </td>
    </tr>
    <tr>
      <th>
        주소
      </th>
      <td>
        서울시 강남구 학동로 211 1층 _____ 매장 주소
        <!--span class="lot">(서울시 강남구 학동로 211 1층)</span-->
      </td>
    </tr>
    <tr>
      <th>
        전화번호
      </th>
      <td>
        02-3444-_____ 전화번호
      </td>
    </tr>
  </tbody>
</div>
```

그림 6-23 팝업 창에 대한 HTML 소스 코드 확인

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

■ 파이썬 셸 창에서 크롤링하기

8. 매장 정보 추출하기 (매장 이름)

```
>>> store_name_h2 = soupCB1.select("div.store_txt > h2")
>>> store_name_h2
[<h2>학동역 DT점</h2>]
>>> store_name = store_name_h2[0].string
>>> store_name
'학동역 DT점'
```

9. 매장 정보 추출하기 (매장 주소)

```
>>> store_info = soupCB1.select("div.store_txt > table.store_table > tbody > tr
> td")
>>> store_info
[<td>평일 : 07:00~23:00 | 주말 : 08:00~22:00</td>, <td>DT(드라이브 스루) 매장입니
다.</td>, <td>서울시 강남구 학동로 211 1층 <!--span class="lot">(서울시 강남구 학동로
211 1층)</span--></td>, <td>02-3444-0000</td>]
>>> store_address_list = list(store_info[2])
>>> store_address_list
['서울시 강남구 학동로 211 1층', 'span class="lot">(서울시 강남구 학동로 211 1층)</
span']
>>> store_address = store_address_list[0]
>>> store_address
'서울시 강남구 학동로 211 1층'
```

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

- 파이썬 셸 창에서 크롤링하기

10. 매장 정보 추출하기(매장 전화번호)

```
>>> store_phone = store_info[3].string  
>>> store_phone  
'02-3444-0000'
```

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

■ 파이썬 파일을 작성하여 크롤링하기

- 매장 한 개에 대한 크롤링 작업을 매장의 갯수만큼 반복해서 처리하면서 result 리스트에 저장

[프로그램 6-2] CoffeeBeanCrawler.py

```
from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
import datetime
from selenium import webdriver
import time

#[CODE 1]
def CoffeeBean_store(result):
    CoffeeBean_URL = "https://www.coffeebeankorea.com/store/store.asp"
    wd = webdriver.Chrome("WebDriver설치경로 /chromedriver.exe")

for i in range(1, 300): #매장 수만큼 반복
    wd.get(CoffeeBean_URL)
    time.sleep(1) #웹페이지 연결할 동안 1초 대기
    try:
        wd.execute_script("storePop2(%d)" %i)
        time.sleep(1) #스크립트 실행할 동안 1초 대기
        html = wd.page_source
        soupCB = BeautifulSoup(html, 'html.parser')
        store_name_h2 = soupCB.select("div.store_txt > h2")
        store_name = store_name_h2[0].string
        print(store_name) #매장 이름 출력하기
        store_info = soupCB.select("div.store_txt > table.store_table > tbody > tr > td")
```

```
store_address_list = list(store_info[2])
store_address = store_address_list[0]
store_phone = store_info[3].string
result.append([store_name]+[store_address]+[store_phone])
except:
    continue
return

#[CODE 0]
def main():
    result = []
    print('CoffeeBean store crawling >>>>>>>>>>>>>>>>>>>>>>')
    CoffeeBean_store(result) #[CODE 1]

    CB_tbl = pd.DataFrame(result, columns = ('store', 'address','phone'))
    CB_tbl.to_csv('./CoffeeBean.csv', encoding = 'cp949', mode = 'w',
        index = True)

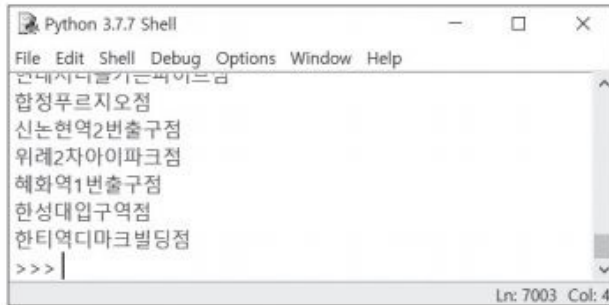
if __name__ == '__main__':
    main()
```

02. 동적 웹 페이지 크롤링

■ 동적 웹 페이지 크롤링 실습

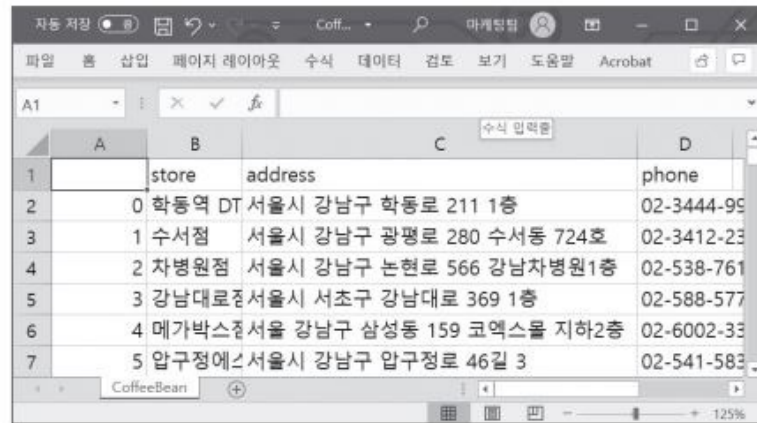
■ 파이썬 파일을 작성하여 크롤링하기

- 작성한 파이썬 파일을 My_Python 폴더에 CoffeeBeanCrawler.py로 저장
- [F5]를 눌러 실행하면 파이썬 셸 창에 print(store_name)문 실행 결과가 출력되고 CSV 파일이 생성



```
Python 3.7.7 Shell
File Edit Shell Debug Options Window Help
합정푸르지오점
신논현역2번출구점
위례2차아이파크점
혜화역1번출구점
한성대입구역점
한티역디마크빌딩점
>>>
```

(a) 파이썬 셸 창에 출력된 실행 결과



	A	B	C	D
1		store	address	phone
2	0	학동역 DT	서울시 강남구 학동로 211 1층	02-3444-99
3	1	수서점	서울시 강남구 광평로 280 수서동 724호	02-3412-23
4	2	차병원점	서울시 강남구 논현로 566 강남차병원1층	02-538-761
5	3	강남대로점	서울시 서초구 강남대로 369 1층	02-588-577
6	4	메가박스점	서울 강남구 삼성동 159 코엑스몰 지하2층	02-6002-33
7	5	압구정점	서울시 강남구 압구정로 46길 3	02-541-583

(b) CSV 파일 내용

그림 6-24 CoffeeBeanCrawler.py 파일의 실행 결과

Section 03 웹 크롤링 활용 (출처: 파이썬 데이터분석 for Beginner)

■ 시간 간격 크롤러

- 시간 간격을 두고 작동하는 크롤러를 작성
- 시간에 따라 변경되는 웹 데이터: 실시간 검색어, 현재 날씨, 인기 뉴스 목록 등
- 네이트에서 제공하는 날씨 중 온도, 습도, 강수량, 풍향을 1시간마다 CSV 파일로 저장함

Section 03 웹 크롤링 활용

■ 시간 간격 크롤러

- 1시간마다 CSV 파일을 열어서 날짜와 시간을 기록하는 코드

Code09-12.py

```
01 import csv
02 import time
03 import datetime
04
05 csvName = 'C:/CookAnalysis/CSV/datetime.csv'
06 with open(csvName, 'w', newline='') as csvFp:
07     csvWriter = csv.writer(csvFp)
08     csvWriter.writerow(['연월일', '시분초'])
09
10 count = 10
11 while count > 0 :
12     count -= 1
13
14     now = datetime.datetime.now()
15     yymmdd = now.strftime('%Y-%m-%d')
16     hhmmss = now.strftime('%H:%M:%S')
17     time_list = [ymmdd, hhmmss]
18     print(time_list)
```

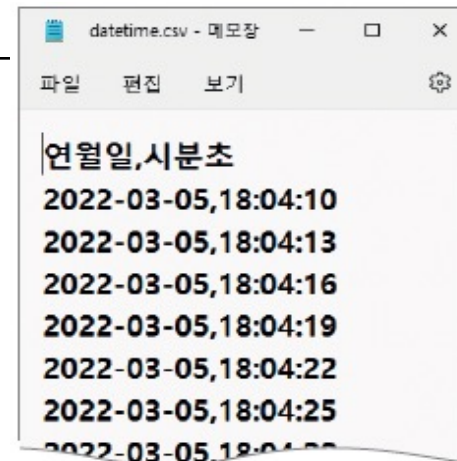

Section 03 웹 크롤링 활용

■ 시간 간격 크롤러

- 1시간마다 CSV 파일을 열어서 날짜와 시간을 기록하는 코드

Code09-12.py

```
19
20     with open(csvName, 'a', newline='') as csvFp:
21         csvWriter = csv.writer(csvFp)
22         csvWriter.writerow(time_list)
23
24     time.sleep(3)
```



Section 03 웹 크롤링 활용

■ 속초 날씨 크롤러

- 엣지 브라우저에서 네이트 날씨(<https://news.nate.com/weather>)에 접속
- 지역을 [강원영동]-[속초]로 변경한 후 [이동]을 클릭
- 주소창에 변경된 URL을 복사



그림 9-12 네이트 날씨에서 속초 지역의 날씨 확인

Section 03 웹 크롤링 활용

■ 속초 날씨 크롤러

- 개발자 도구로 날씨 부분을 클릭해서 코드를 확인

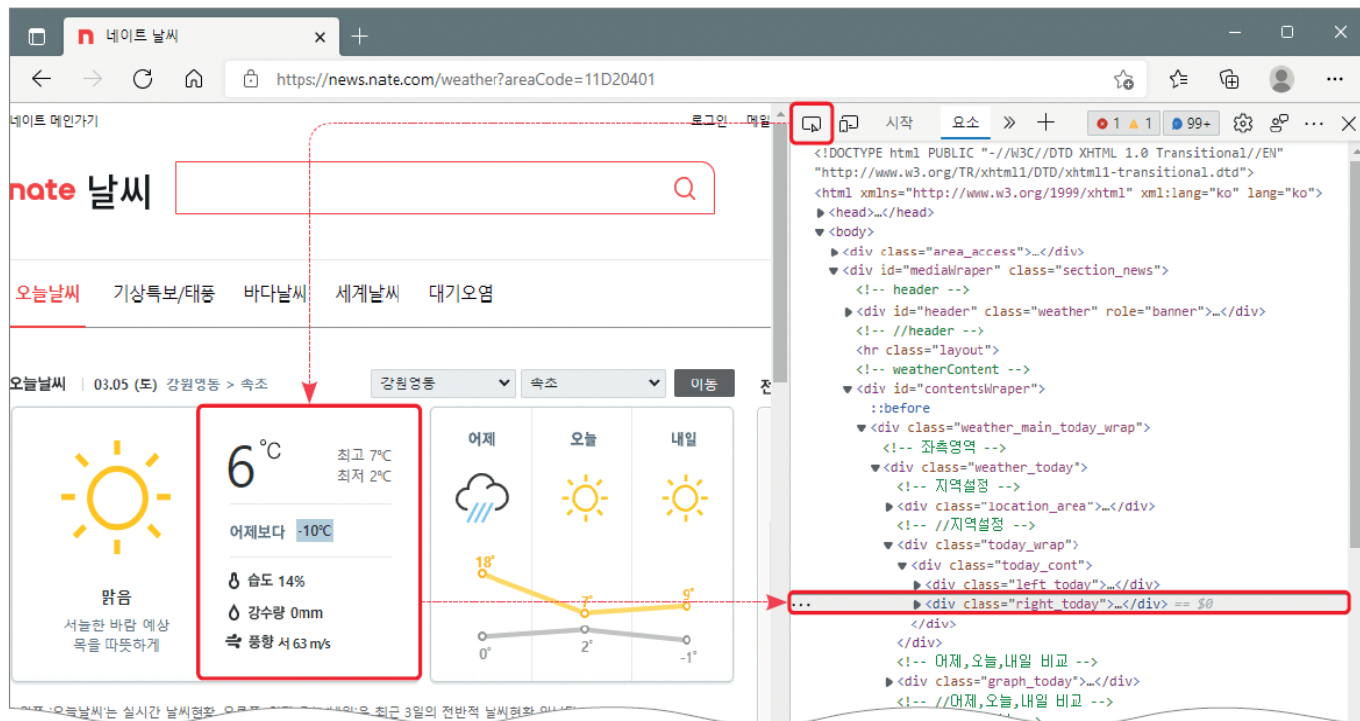


그림 9-13 속초 지역의 날씨 키워드 추출 1

Section 03 웹 크롤링 활용

■ 속초 날씨 크롤러

- 오른쪽의 <div class=""right_today"> 부분을 확장하여 관련 정보의 태그 및 클래스를 확인

표 9-1 확인할 정보와 클래스

정보	태그 및 클래스
기온	<p class="celsius">
습도	<p class="humidity">
강수량	<p class="rainfall">
풍속	<p class="wind">

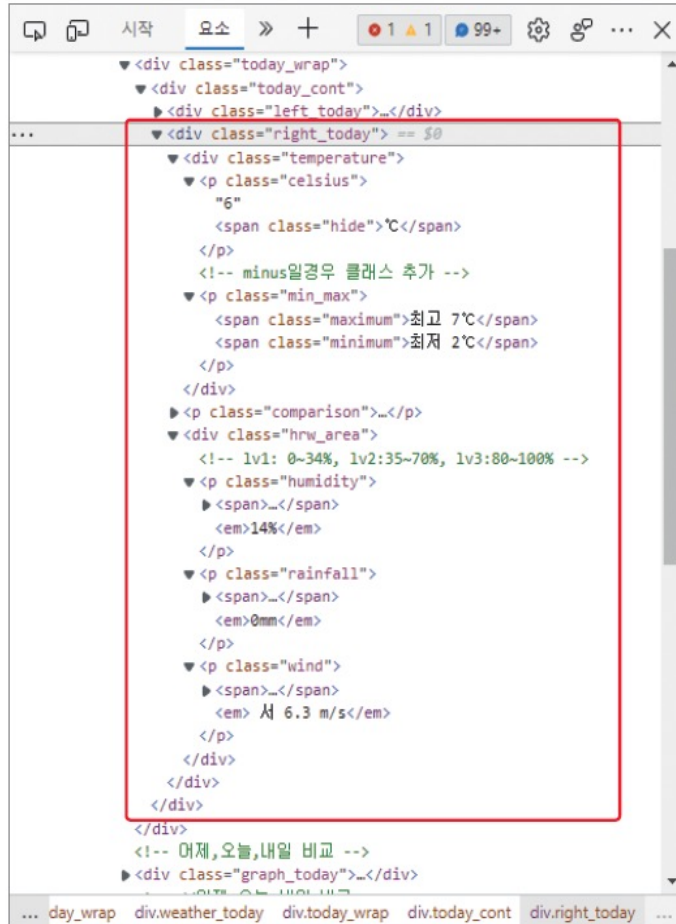


그림 9-14 속초 지역의 날씨 키워드 추출 2

Section 03 웹 크롤링 활용

■ 속초 날씨 크롤러

- 속초의 날씨를 주기적으로 추출해서 CSV 파일로 저장하는 완전한 버전의 크롤러를 제작

Code09-13.py

```
01 import csv
02 import time
03 import datetime
04 import bs4
05 import urllib.request
06
07 csvName = 'C:/CookAnalysis/CSV/sokcho_weather.csv'
08 with open(csvName, 'w', newline='') as csvFp:
09     csvWriter = csv.writer(csvFp)
10     csvWriter.writerow(['연월일', '시분초', '온도', '습도', '강수량', '풍향'])
11
12 nateUrl = "https://news.nate.com/weather?areaCode=11D20401"
13 while True :
14     htmlObject = urllib.request.urlopen(nateUrl)
15     webPage = htmlObject.read()
16     bsObject = bs4.BeautifulSoup(webPage, 'html.parser')
17     tag = bsObject.find('div', {'class': 'right_today'})
18     temper = tag.find('p', {'class': 'celsius'}).text
19     humi = tag.find('p', {'class': 'humidity'}).text
```

Section 03 웹 크롤링 활용

■ 속초 날씨 크롤러

- 속초의 날씨를 주기적으로 추출해서 CSV 파일로 저장하는 완전한 버전의 크롤러를 제작

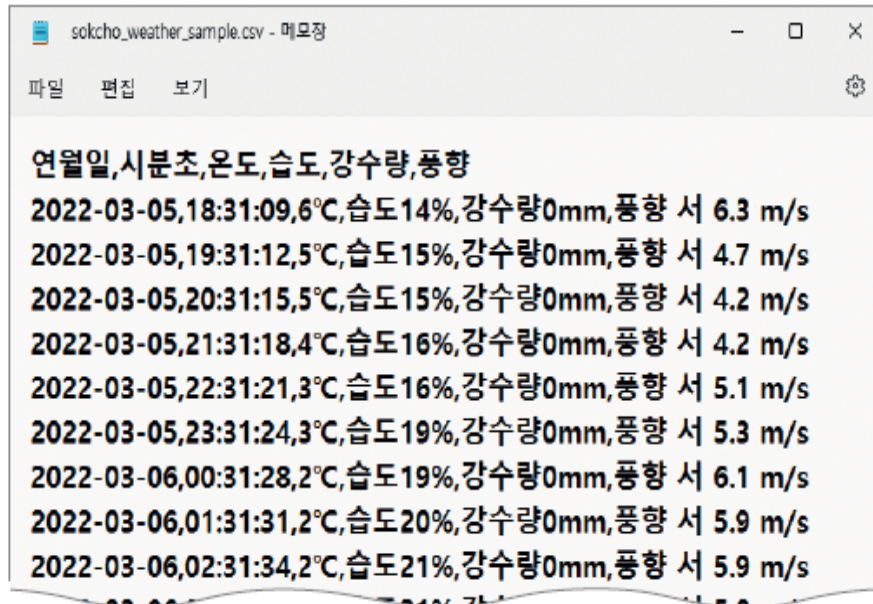
Code09-13.py

```
20     rain = tag.find('p', {'class': 'rainfall'}).text
21     wind = tag.find('p', {'class': 'wind'}).text
22
23     now = datetime.datetime.now()
24     yymmdd = now.strftime('%Y-%m-%d')
25     hhmmss = now.strftime('%H:%M:%S')
26
27     weather_list = [ymmdd, hhmmss, temper, humi, rain, wind]
28     with open(csvName, 'a', newline='') as csvFp:
29         csvWriter = csv.writer(csvFp)
30         csvWriter.writerow(weather_list)
31
32     time.sleep(3600)
```

Section 03 웹 크롤링 활용

■ 속초 날씨 크롤러

- 속초의 날씨를 주기적으로 추출해서 CSV 파일로 저장하는 완전한 버전의 크롤러를 제작





데이터 과학 기반의 파이썬 빅데이터 분석

감사합니다.
