Web Scraping 응용편

이상엽

포함 내용

● BeautifulSoup 에서 정규식 사용하기

● Network inspection 을 통해서 특정한 파일에 접근하기

- POST request 방식 사용하기

● Selenium 사용하여 source codes 실시간으로 다운로드 받기

1. BeautifulSoup 에서 정규식[1] 사용하기

정규식은 문자열의 패턴을 만들어 특정한 텍스트안에 해당 패턴과 매치되는 문자열이 있는지를 알고자 할 때 사용합니다. BeautifulSoup 에서 제공되는 find()나 find_all()함수를 이러한 정규식과 함께 사용할 수 있습니다. 특히 find_all()함수와 유용하게 사용될 수 있습니다. find_all()함수가 입력받는 파라미터의 값으로 정규식의 패턴을 사용할 수 있는 것입니다.예를 들어, 아래와 같은 코드 같은 경우에는 html 에 저장되어 있는 source codes 에서 tag 의 이름이 'sp'로 시작하는 모든 tag 를 찾습니다.

```
import re
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
soup.find_all(re.compile(r'sp'))
```

파이썬에서 제공된는 re 모듈을 사용해서 문자열 패턴을 만드는 경우에는 compile 이라는 함수를 사용합니다. 예를 들어, 아래와 같은 source codes 가 있다고 하는 경우에

Comment [S11]: 정규식 패턴을 만들기 위해서는 re 에서 제공되는 compile()이라는 함수를 사용합니다. 그리고 우리가 만들고자 하는 패턴을 compile()함수의 파라미터로 제공합니다.

Comment [S12]: r 은 raw string 을 의미합니다.

[1]정규식에 대한 내용은 정규식 관련 파일을 참조하시기 바랍니다.

```
<html lang="ko">
	<head>
		<title> Online Data Scraping Test</title>
	</head>

	<span class="red_price">35.6</span>
	<span class="blue_price">37.5</span>
	<spang class="blue_price">43.2</spang>
	<spark method="manual">24.3</spark>
</html>
```

soup.find_all(re.compile(r'sp'))

가 반환하는 결과는 아래와 같습니다. 즉, sp 로 시작하는 모든 tag 를 list 형태로 반환하게 됩니다.

```
[<span class="red_price">35.6</span>,
 <span class="blue_price">37.5</span>,
 <spang class="blue_price">43.2</spang>,
 <spark method="manual">24.3</spark>]
```

이번에는 위의 source codes 에서 tag 의 속성 값을 이용해서 tag 를 찾을 때 정규식을 이용해 보도록 하겠습니다.

만약, 아래와 같은 코드가 있다고 한다면,

soup.find_all('span', attrs={'class':re.compile(r'price')})

이 코드가 반환하는 결과값은 아래와 같습니다. 즉, 이름이 span 인 tag 들 중에서 class 속성을 갖고 그 값에 price 라는 값을 문자열의 일부로 포함하고 있는모든 span tag 들을 찾습니다.

```
[<span class="red_price">35.6</span>, <span class="blue_price">37.5</span>]
```

아래의 코드는 어떠한 결과를 반환을 할까요?

soup.find_all(attrs={'class':re.compile(r'price')})

위의 코드는 class 라는 속성을 갖는 tag 들 중에서 그 값에 price 라는 문자열이 포함되어 있는 모든 tag 를 찾습니다. 따라서 위의 source codes 에서는 아래와 같은 tag 들이 찾아지게 됩니다.

```
[<span class="red_price">35.6</span>,
 <span class="blue_price">37.5</span>,
 <spang class="blue_price">43.2</spang>]
```

Comment [S13]: 'span'이라는 이름을 갖는 tag 들만 찾습니다.

Comment [S14]: 이번에는 class 라는 속성의 값에 대해서 패턴을 만듭니다.

Comment [S15]: 찾고자 하는 tag 의 이름을 제공하지 않았습니다.

● 우리가 추출하고자 하는 정보가 원본 source codes 에 들어있지 않는 경우

기본편에서 설명한 것 처럼 우리가 원하는 정보를 특정 웹페이지에서 파이썬을 이용해서 수집하는 경우에는 다음과 같은 과정을 일반적으로 거칩니다.

① 원하는 정보를 담고 있는 웹 페이지의 source codes 다운로드 하기
② 다운로드 받은 source codes 에서 우리가 원하는 정보를 담고 있는 tag 찾기
③ 해당 tag 에서 우리가 원하는 정보 추출하기

보통의 경우에는①의 목적으로는 requests 라는 모듈을 그리고②, ③을 위해서는 BeatifulSoup 을 사용합니다. 하지만, 이러한 방법으로 모든 정보를 추출할 수 있는 것은 아닙니다. 우리는 원하는 정보가 브라우저 상에서는 보이더라도 requests 를 통해서 다운로드 받는 원본 source codes 에는 그러한 내용이 저장되어 있지 않는 경우가 있습니다. 이러한 경우에는 requests 만을 사용해서는 우리가 원하는 정보를 Python 을 이용해서 추출할 수가 없습니다. requests.get()을 통해서 얻는 source codes 에는 그러한 정보가 들어있지 않기 때문입니다.

requests 를 가지고 다운로드 받는 source codes 는 우리가 원하는 정보를 담고 있는 웹 페이지의 source codes 를 웹 브라우저상에서 보는 것과 비슷합니다. 즉, 웹 브라우저 상에서 source codes 를 확인했는데 그 안에 우리가 원하는 정보가 저장되어 있지 않다면 이는 requests 만을 사용해서는 해당 정보를 파이썬에서 추출할 수 없다는 것을 의미합니다.

원본 source codes 에 우리가 원하는 정보가 들어있지 않는 경우에는 크게 아래 두 가지 경우가 해당합니다.

i) 해당 정보가 원본 source codes 안에 저장되어서 서버에서 저장되는 것이 아니라, 별도의 파일을 통해서 전달되는 경우

ii) 해당 정보가 원본 source codes 가 서버로 부터 다운로드 받아진 이후 추가적인 서버와의 실시간 통신을 통해서 받아지는 경우

위와 같은 경우에는 requests 만을 통해서는 우리가 원하는 정보를 추출할 수가 없습니다.

이를 위해서 사용하는 추가적인 방법이 브라우저에서 제공되는 network inspection 기능을 사용하거나 아니면 selenium 이라는 프로그램을 사용하는 것입니다. Network inspection 기능은 보통 i)의 경우에 사용되고 selenium 은 ii)경우에 사용됩니다.

Comment [S16]: 원본 source codes 라고 표현하는 이유는 requests.get()을 통해서 다운로드 받는 source codes 는 브라우저가 해당 페이지를 화면에 display 하기 위해서 서버로부터 받는 최초의 source codes 와 같기 때문입니다. 브라우저는 특정한 하나의 웹페이지를 완변하게 display 하기 위해서 서버와의 통신을 한번만 하는 것이 아니라, 원본 source codes 를 다운로드 받은 이후에도 지속적으로 통신합니다.

Comment [S17]: 별도의 파일은 고유의 url 주소를 갖고, 이러한 url 주소는 보통 원본 source codes 안에 저장되어 있습니다.

2. Network inspection 기능을 통해서 특정한 파일에 접근하기

우리가 특정한 웹페이지를 브라우저 상에서 보기 위해서 브라우저는 서버로부터 굉장히 많은 수의 파일을 다운로드 받습니다. 각 파일은 별도의 데이터를 저장하고 있습니다. 그러한 데이터는 텍스트 데이터 일 수도 있고, 사운드 혹은 이미지 데이터일 수도 있습니다. 많은 경우, 우리가 원하는 정보가 원본 source codes 에 담겨 있지 않는 경우에는 별도의 파일에 그 내용이 저장되어 있습니다. 원본 source codes 에는 해당 파일의 주소가 저장되어 있고 브라우저는 일차적으로 원본 source codes 를 다운로드 받고 추가적으로 각 파일의 주소를 이용해서 각 파일의 내용을 다운로드 받습니다.

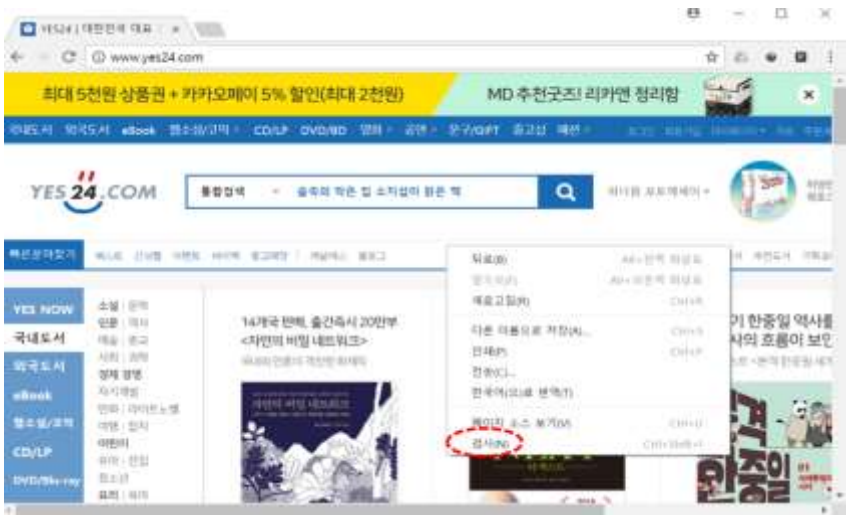
우리가 원하는 정보가 원본 source codes 에 저장되어 있지 않고, 브라우저가 서버로부터 다운로드 받는 별도의 파일에 저장되어 있는 경우에, 해당 정보를 추출하기 위해서 우리가 해야 되는 것은 브라우저를 이용해서 해당 정보를 담고있는 파일이 무엇인지를 확인하고, 그 파일의 고유 경로를 사용해서 해당 파일에 직접적으로 파이썬을 이용해서 접근하여 해당 파일이 저장하고 있는 내용을 추출하는 것입니다.

Comment [S18]: 하나의 파일의 경로도 보통은 url 주소입니다.

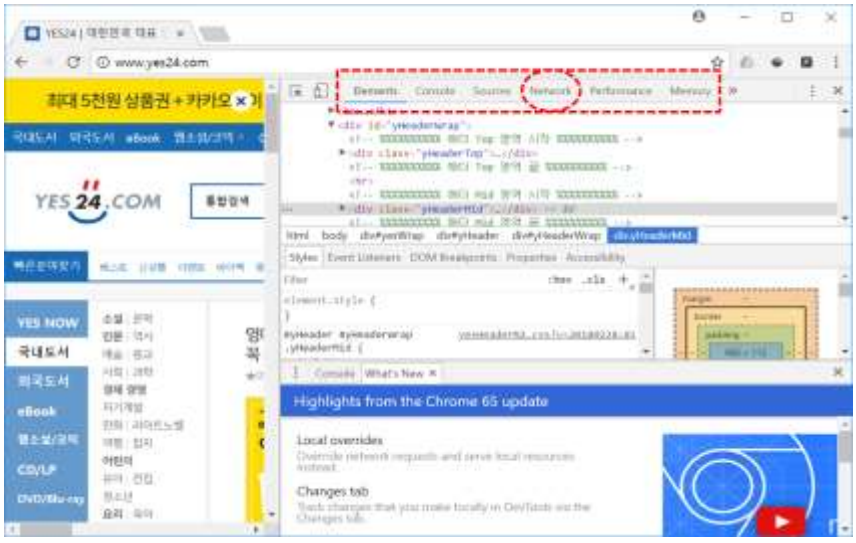
이를 위해서는 먼저 브라우저가 우리가 추출하고자 하는 정보를 담고 있는 웹 페이지를 display 하기 위해서 서버로 부터 어떠한 파일들을 다운로드 받는지를 확인해야 합니다. 이는 브라우저에서 제공하는 network inspection 기능을 사용해서 확인할 수 있습니다. 여기서는 Chrome 브라우저를 기본으로 설명하도록 하겠습니다.

Comment [S19]: Internet explorer 는 coding 에는 별로 적합하지 않으므로 되도록이면 Chrome 이나 Firefox 등의 다른 브라우저를 사용하시기 바랍니다.

특정한 웹페이지를 브라우저 상에 display 하기 위해서 브라우저가 서버로부터 다운로드 받는 파일의 목록을 확인하기 위해서는 해당 웹페이지 상에서 마우스의 오른쪽 버튼을 클릭합니다. 아래 그림은 www.yes24.com 이라는 사이트 상에서 해당 페이지를 display 하기 위해서 브라우저가 어떠한 파일을 주고 받는지 확인하고자 마우스의 오른쪽 버튼을 누른 경우입니다. 마우스의 오른쪽 버튼을 클릭하면 아래 그림에서 처럼 여러가지 메뉴가 나오는데, 그 중에서 제일 아래있는 '검사(N)' 메뉴를 통해서 서버와의 자세한 통신 기록을 확인할 수가 있습니다.



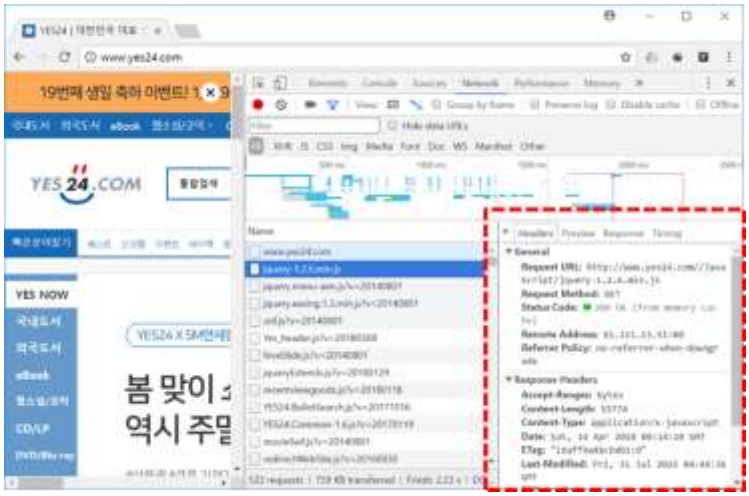
'검사(N)'을 클릭하면 해당 웹페이지를 검사할 수 있는 여러가지 옵션이 나옵니다. 검사 창의 상단을 보면 Element, Console, Source 등의 메뉴가 있습니다. 이중에서 Network 메뉴를 클릭하면 서버와의 자세한 통신 기록을 볼 수가 있습니다.



Network 메뉴를 클릭하면 처음에는 아마도 아무런 내용이 나오지 않을 것입니다. 왜냐하면, 이미 해당 페이지를 보여주기 위해서 필요한 대부분의 파일을 다운로드 받은 상태이기 때문입니다. 해당 페이지를 display 하기 위해서 서버로부터 다운로드 받는 파일들의 목록을 확인하기 위해서는 새로고침(F5) 버튼을

클릭하여 해당 페이지를 다시 로딩해야 합니다. 키보드에서 F5 버튼을 누르면 아래와 같이 여러개의 파일이 나오는 것을 알 수가 있습니다. 이러한 파일들이 브라우저가 해당 웹페이지를 display 하기 위해서 서버로부터 다운로드 받는 파일들입니다.

www.yes24.com 이라는 파일을 처음 다운로드 받는 것으로 나오는데 이 파일에 우리가 브라우저를 통해서 볼 수 있는 해당 페이지의 원본 source codes 가 들어 있습니다. 그리고 두번째 파일은 jquery-1.2.5.min.js 라는 자바스크립트 파일입니다. 특정한 파일의 구체적인 내용을 살펴보기 위해서는 해당 파일의 이름을 클릭하면 됩니다. 두번째 파일은 jquery-1.2.5.min.js 을 클릭해 보겠습니다. 그 결과는 아래와 같습니다.



특정한 파일의 세부 정보로는 Headers, Preview, Response, Timing 등이 있습니다.

Headers 정보에는 위에 나오는 것 처럼 해당 파일 자체의 URL 주소 (Request URL) 그리고 Request mode (GET)등의 정보가 나옵니다. 우리는 해당 파일의 URL 주소를 가지고 해당 파일의 내용을 직접적으로 서버로부터 다운로드 받을 수가 있습니다. Preview 는 해당 파일의 브라우저 상에서 어떻게 display 되는지를 미리볼 수 있는 메뉴입니다. Response 는 해당 파일이 담고 있는 source codes 라고 생각하면 됩니다. Timing 은 해당 파일을 서버로부터 다운로드 받는데 걸리는 시간과 관련된 정보를 보여줍니다. 이러한 여러 메뉴들 중에서 우리가 중요하게 사용하게 되는 정보는 Header 에 있는 정보입니다.

Network inspection 기능을 가지고 데이터 추출하는 방법을 아래 웹페이지의 내용을 가지고 좀 더 구체적으로 설명하도록 하겠습니다.

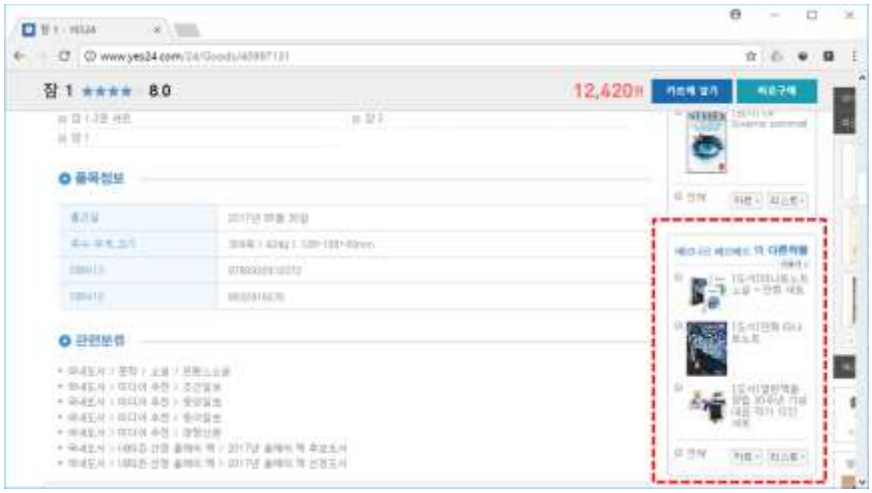
우리가 수집 하고자 하는 정보가 '잠 1'이라는 책과 관련된 정보라고 가정을 하겠습니다. 이러한 정보를 yes24 에서 제공하는 웹페이지에서 얻고자 한다라고 가정을 합니다. 해당 책의 정보를 담고 있는 웹페이지의 URL 주소는 아래와 같습니다.

URL: http://www.yes24.com/24/Goods/40997131

해당 웹페이지를 보면 아래와 같은 정보들이 포함되어 있습니다.

이러한 정보들 중에서 책의 제목, 정가, 판매가 등의 정보는 브라우저에서 제공되는 페이지 소스 보기 메뉴를 통해서 볼 수 있는 source codes 안에 저장되어 있는 것을 확인할 수 가 있습니다. 즉, 이러한 정보는 requests 모듈을 통해서 수집할 수가 있는 것입니다. 왜냐하면, requests.get()함수를 가지고 다운로드 받는 해당 웹페이지의 source codes 는 브라우저 상에서 보이는 source codes 와 거의 동일하기 때문입니다.

이러한 책의 기본적인 정보 이외에 추가적으로 해당 웹페이지에서 제공되는 정보들 중에서 '작가의 다른 작품들'에 대한 정보를 수집하고자 한다고 가정해 보겠습니다.



이러한 경우에 우리가 일차적으로 확인해야 하는 것은 해당 정보가 원본 source codes 에 저장이 되어 있는지입니다. 브라우저 상에서 마우스 오른쪽 버튼을 클릭해서 나오는 메뉴들 중에서 '페이지 소스 보기 (Ctrl+U)' 메뉴를 클릭하고, 찾기 (Ctrl+F) 메뉴를 통해서 '작가의 다른 작품들' 중 하나인 "[도서]타나토노트"를 source codes 에서 찾아봅니다. 해당 정보가 있는 경우에는 requests.get()를 사용해서 원본 source codes 를 다운로드 받고 그 안에서 우리가 원하는 정보를 BeautifulSoup 을 이용하여 추출하면 됩니다. 하지만, 우리가 원하는 정보인 '작가의 다른 작품들'에 대한 정보가 원본 source codes 에 저장되어 있지 않은 것을 확인할 수가 있습니다.

이러한 경우에 첫번째로 사용할 수 있는 방법이 브라우저에서 제공되는 network inspection 기능입니다. 즉, 이 기능을 통해서 해당 웹페이지를 보여주기 위해서 브라우저가 서버로부터 다운로드 받는 파일들 중에서 우리가 원하는 정보인 '작가의 다른 작품들'에 대한 정보를 담고 있는 별도의 파일이 있는지를 확인하는 것입니다. 확인한 후에 해당 파일의 고유 URL 주소를 가지고 해당 파일에 직접 접근하여 해당 파일을 구성하고 있는 source codes 를 다운로드 받아서 그 안에서 우리가 원하는 정보를 추출하면 됩니다.

순서는 아래와 같습니다.

① 해당 웹페이지에서 마우스 오른쪽 버튼을 클릭하고 검사 (N) 메뉴 선택하기

② 검사 메뉴로 나오는 윈도우에서 '네트워크' 탭 클릭하기

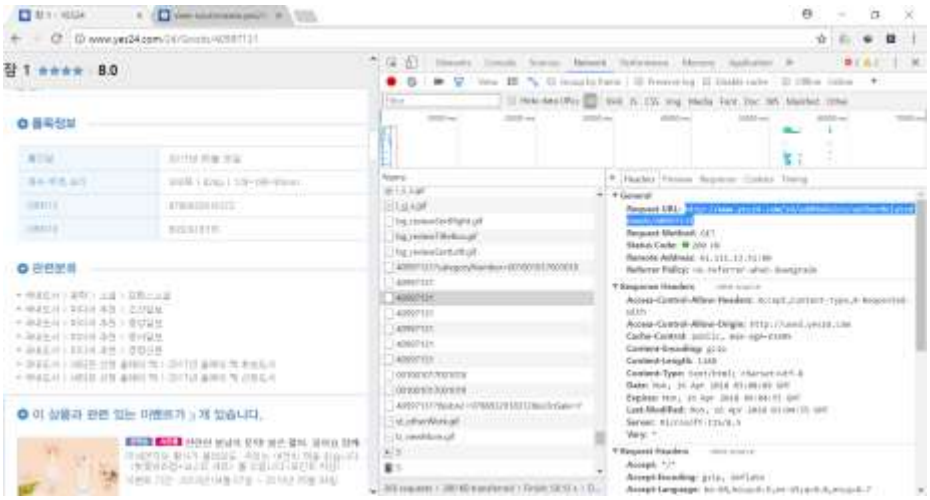
③ 네트워크 탭에서 새로고침 메뉴 실행하기 (키보드에서 F5 버튼 누르기)

④ 브라우저가 다운로드 받는 파일들 중에서 '작가의 다른 작품들' 정보를 담고 있는 파일 확인하기

⑤ 해당 파일을 클릭하여 나오는 정보들 중에서 해당 파일의 URL 주소 복사하기

⑥ 해당 URL 주소를 가지고 requests.get()을 이용해서 해당 파일의 source codes 를 다운로드 받기

⑦ BeautifulSoup 모듈을 사용해서 다운로드 받은 source codes 에서 원하는 정보를 담고 있는 tag 에 접근후 해당 정보를 추출하기



해당 파일의 내용은 아래 그림 처럼 브라우저의 새창에 해당 파일의 주소를 붙여넣기를 해서 확인할 수 있습니다.

Comment [S110]: 해당 주소를 브라우저의 새 창에 붙여넣어 해당 파일의 내용 확인하기

Comment [S111]: 아래 그림에 나오는 것 처럼 파일이름이 40997131 인 여러개의 파일들 중에서 URL 주소가 http://www.yes24.com/24/addModules/authorRelatedGoods/40997131 라는 파일에 우리가 원하는 정보인 '작가의 다른 작품들'이 담겨 있는 것을 확인할 수 있습니다.

아래와 같은 코드로 해당 파일에 대한 source codes 를 서버로부터 직접적으로 다운로드 받을 수 있습니다.

그리고, BeautifulSoup 의 find_all()함수를 사용해서 우리가 원하는 정보를 추출할 수 있습니다.

```
import requests
from bs4 import BeautifulSoup

url = 'http://www.yes24.com/24/addModules/authorRelatedGoods/40997131'
r = requests.get(url)
soup = BeautifulSoup(r.text, 'lxml')
results = soup.find_all('a')
print(results[:4])
```

Comment [S112]: 우리가 원하는 정보를 담고있는 파일의 URL 주소

Comment [S113]: requests.get()함수를 가지고 해당 파일의 source codes 를 다운로드 받습니다.

위 코드의 결과는 아래와 같습니다.

```
[<a href="/24/AuthorFile/Author/543?Pcode=009"><img alt="더보기" class="more"
src="http://image.yes24.com/sysimage/category/b_viewMore.gif"/></a>, <a
href="/24/goods/58397337?Pcode=009"><img alt="[도서]타나토노트 소설 + 만화 세트"
src="http://image.yes24.com/goods/58397337/S"/>[도서]타나토노트 소설 + 만화 세트</a>, <a
href="/24/goods/57950537?Pcode=009"><img alt="[도서]만화 타나토노트"
src="http://image.yes24.com/goods/57950537/S"/>[도서]만화 타나토노트</a>, <a
href="/24/goods/31950778?Pcode=009"><img alt="[도서]열린책들 창립 30 주년 기념 대표 작가 12 인 세트"
src="http://image.yes24.com/goods/31950778/S"/>[도서]열린책들 창립 30 주년 기념 대표 작가 12 인
세트</a>]
```

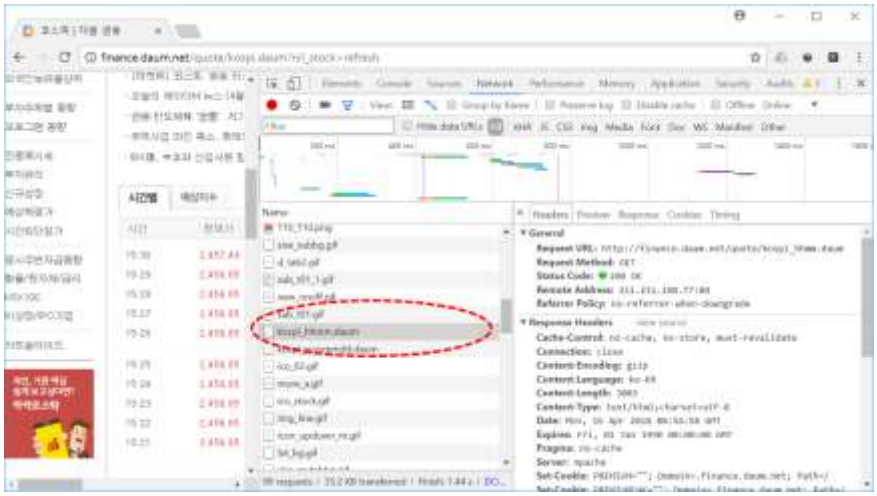
하나의 예를 더 들어보도록 하겠습니다.

이번에는 다음에서 제공되는 KOSPI 정보를 보여주는 아래 웹페이지에서 특정한 정보를 추출하고자 한다라고 가정을 하겠습니다.

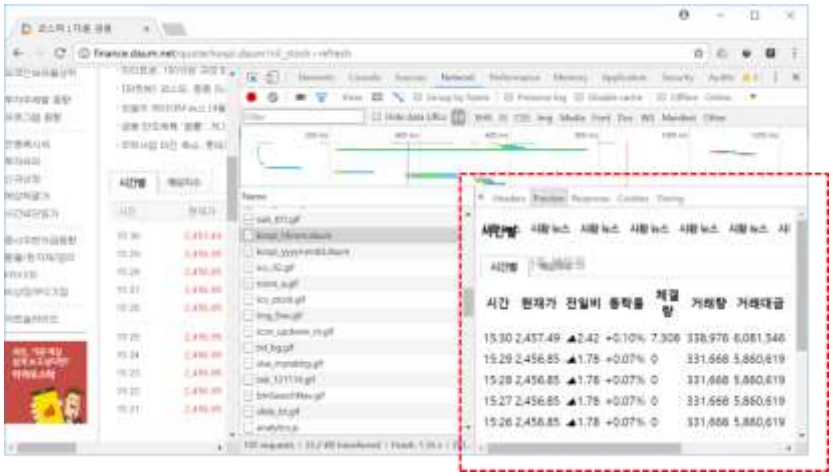
URL: http://finance.daum.net/quote/kospi.daum

이 정도들 중에서 대부분은 원본 source codes 안에 저장되어 있는 것을 확인할 수 있습니다. 이러한 정보는 requests.get()함수를 사용해서 추출할 수 있습니다. 그런데, 만약 우리가 해당 페이지에서 추출하고자 하는 정보가 아래와 같은 시간별 KOSPI 지수 정보라면 어떻게 할까요?

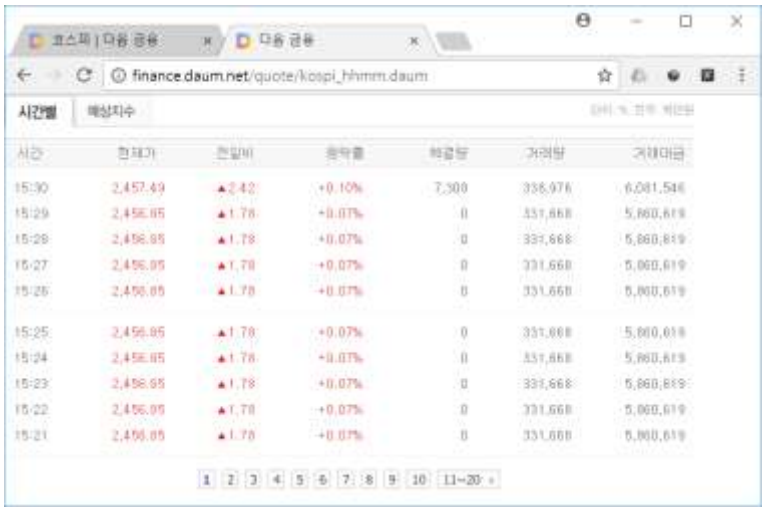
안타깝게도 원본 source codes 에는 해당 정보가 저장되어 있지 않습니다. 이 때 사용할 수 있는 방법이 브라우저에서 제공되는 network inspection 기능입니다. 이번에도 앞에서 했던 것과 마찬가지로 위의 페이지에서 마우스 오른쪽 버튼을 클릭해서 '검사(N)' 메뉴를 선택하고 나오는 결과에서 'Network' 탭을 선택을 하고, 키보드에서 F5 를 눌러 해당 페이지를 다시 로딩 합니다. 그럼 해당 페이지를 보여주기 위해서 브라우저가 서버로부터 다운로드 받는 파일들의 목록이 나옵니다. 그중에서 우리가 원하는 정보는 아래와 같은 파일에 저장이 되어 있음을 확인할 수 있습니다.



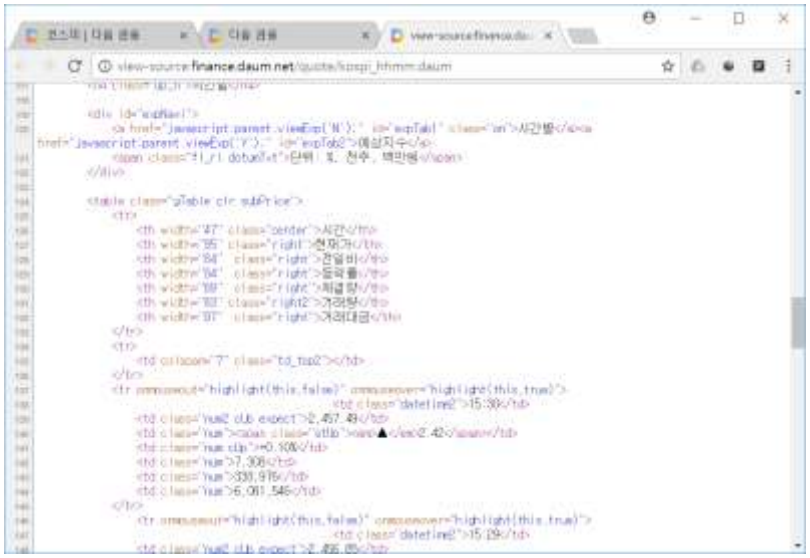
해당 파일이 담고 있는 내용은 'Preview' 메뉴에서 미리보기를 할 수 있습니다 (아래 그림 참조).



해당 파일의 URL 주소를 복사하여 새창에 붙여넣기를 하면 아래와 같은 결과가 나옵니다.



그리고 해당 URL 주소를 가지고 requests.get()을 사용해서 해당 페이지의 source codes 를 다운로드 받으면 우리가 원하는 정보가 다음과 같이 source codes 에 저장되어 있음을 확인할 수 있습니다.



위의 source codes 를 다운로드 받은 후 BeautifulSoup 을 통해서 우리가 원하는 정보를 추출할 수 있습니다.

POST request 방식 사용하기

브라우저(예, Chrome, IE, Firefox 등)는 사용자가 접근하고자 하는 웹페이지 (예, www.daum.net)의 내용을 브라우저 상에 표시하기 위해서 먼저 url 주소를 가지고 해당 서버에 request message 를 보냅니다. 그러면 해당 서버는 request message 를 보낸 브라우저의 validation 을 체크하고 아무런 문제가 없으면 response message 를 브라우저에게 보내게 됩니다. 그러면 브라우저는 response message 에 저장이 되어있는 사용자가 접근하려고 하는 웹페이지의 소스 코드를 정해진 규칙에 의해서 user friendly 한 방식으로 화면에 display 합니다.

이것이 웹 커뮤니케이션의 기본입니다.

그런데, 브라우저가 서버에 request message 를 보내는 방식에는 여러가지 방식이 존재합니다. 가장 많이 일반적으로 사용되는 방법은 GET 방식입니다. GET 방식은 request message 를 보낼 때, 요청의 구체적 내용 (즉, 어떠한 페이지에 대한 어떠한 자료를 원하는지에 대한 정보 (이를 파라미터 정보라고 합니다))를 url 주소의 일부로 표현해서 서버에 request message 를 보냅니다.

GET 방식을 사용하는 대표적인 서비스가 네이버의 검색 서비스입니다. 여러분이 검색창에 '세종시 아파트'라고 입력하고 검색 버튼을 클릭하게 되면, 브라우저의 주소창의 주소는 'https://search.naver.com/search.naver?query=세종시+아파트' 라고 변경됩니다. 즉, 사용자가 원하는 구체적인 정보를 URL 주소의 일부로 포함해서 서버에 request message 를 보내게 되고, 이러한 방식을 GET 방식이라고 합니다.

GET 방식 다음으로 많이 사용되는 방식이 POST 방식입니다. POST 방식도 브라우저가 서버에 request message 를 보낼 때 사용하는데, GET 과의 가장 큰 차이는 사용자가 원하는 구체적인 정보에 대한 파라미터 값을 URL 주소의 일부로 전달하는 것이 아니라, 메시지에 포함시켜서 보내는 방식입니다. 위의 예에서, GET 방식의 경우는 query 라는 파라미터의 값인 '세종시+아파트'를 URL 주소에 포함시켰지만, POST 방식에서는 {query: '세종시+아파트'} 라는 사전 형태의 데이터를 request message 에 저장해서 보냅니다. URL 주소에는 포함되지 않습니다.

Python 에서는 서버와의 커뮤니케이션을 위해서 requests 모듈을 사용하는데, request message 전송 방식에 따라 사용되는 함수가 다릅니다. GET 방식의 request message 를 보낼 때는 get() 함수를 사용하고, POST 방식의 request message 를 보낼 때는 post()함수를 사용합니다. get() 함수의 경우는 인자 값으로 접근하고자 하는 웹페이지의 URL 주소만을 제공하면 되지만, post()함수는 그 보다 조금 더 많은 정보를 제공해야 합니다. 기본적으로 다음과 같이 사용합니다.

r = requests.post(url, data = payload, headers = headers)

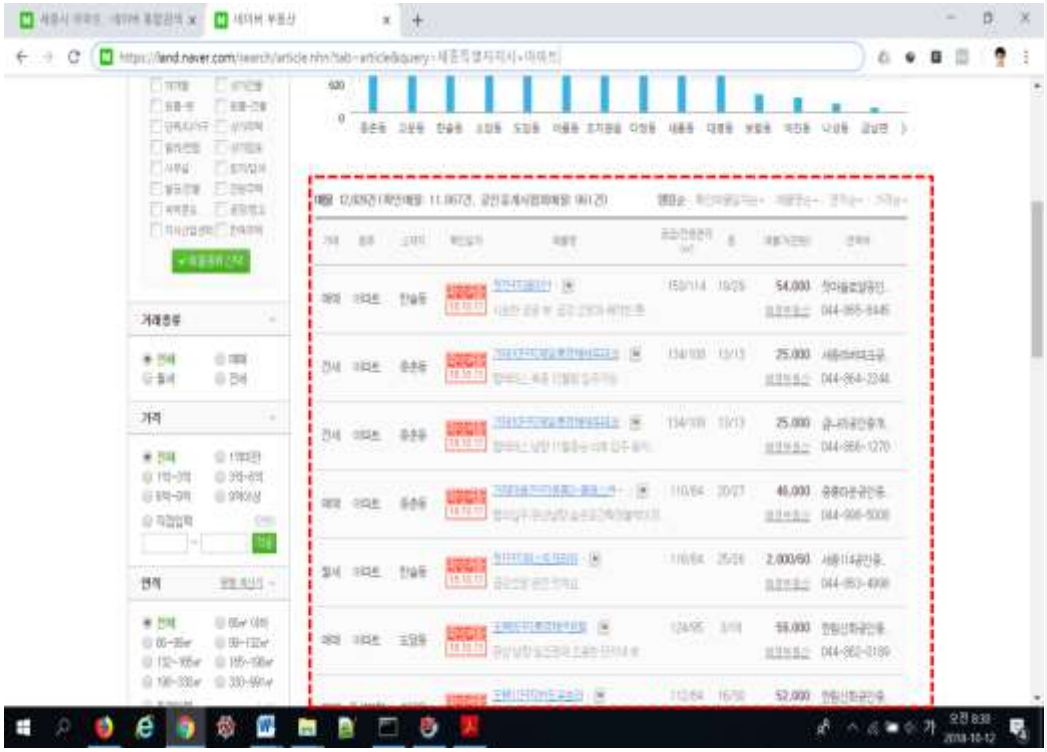
즉, 세 가지 파라미터에 대한 인자 값을 제공해야 하는 것입니다. 첫번째는 접근하고자 하는 페이지의 url 주소가 됩니다. 그리고 두 번째는 사용자가 서버로부터 원하는 구체적인 정보의 값을 가지고 있는 data 가 됩니다. 그리고 headers 에 대한 정보를 제공합니다. headers 는 request message 의 header 정보들을 담고 있습니다. header 정보에는 request message 를 서버로 전송하는데 필요한 커뮤니케이션 정보들이 포함됩니다 (예, 브라우저 정보, IP 주소, referer 등).

post()함수에서 중요한 부분이 data 부분입니다. data 부분에 여러분이 원하는 정보에 따른 parameter 의 값들을 사전 (dictionary) 데이터 형태로 만들어서 제공해야 하는 것입니다. 많은 경우에 여러분이 원하는 정보는 여러분이 브라우저를 통해서 보고 있는 정보가 됩니다. 즉, 브라우저 상에 있는 정보와 같은 정보를 Python 을 이용해서 얻기 위해서는 브라우저가 어떠한 파라미터 값을 사용해서 POST request message 를 보냈는지를 확인하고 같은 정보를 Python 에서 제공되는 requests.post() 함수에 인자값으로 제공을 해야하는 것입니다.

브라우저가 어떠한 파라미터 값을 사용해서 POST request message 를 보냈는지는 브라우저에서 제공하는 검사 (Network inspection) 기능을 사용해서 확인할 수 있습니다. 이를 위해서 여러분이 원하는 정보를 담고 있는 페이지에서 마우스의 오른쪽 버튼을 클릭하면 됩니다.

예) https://land.naver.com/search/article.nhn?tab=article&query=세종특별자치시+아파트

네이버에서 아파트 시세를 보여주는 웹페이지의 정보를 추출하고자 한다라고 가정하겠습니다. 해당 페이지의 url 은 위와 같습니다. 해당 페이지는 아래와 같은 정보를 제공합니다. 여기에서 빨간색 상자 부분에 해당하는 정보를 추출해 보겠습니다.

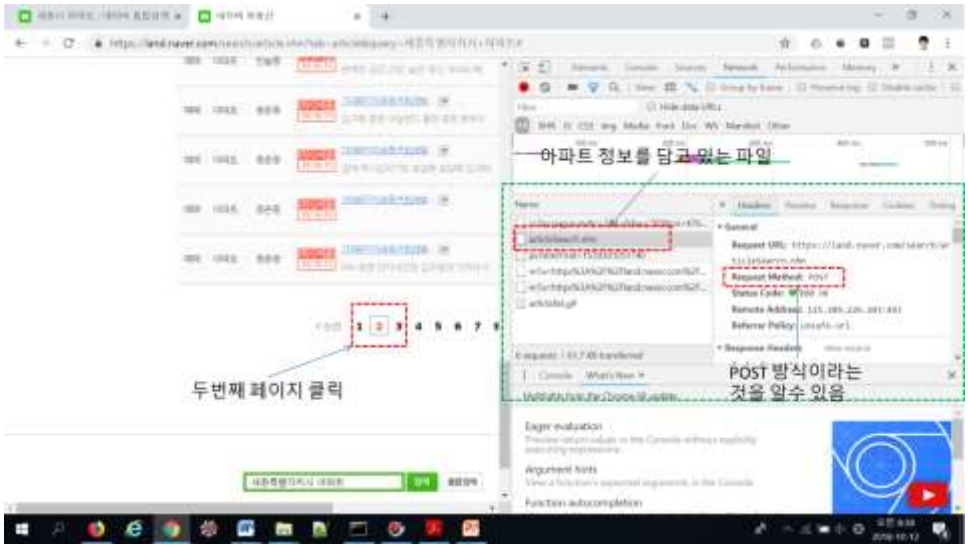


빨간색 부분의 정보는 아쉽게도 페이지 소스코드를 보기를 했을 때 나오는 소스코드에는 포함되어 있지 않습니다. 해당 정보를 담고 있는 별도의 파일이 있는지를 network inspection 기능을 사용해서 살펴보겠습니다. Chrome 에서 마우스 오른쪽 버튼을 클릭했을 때 나오는 메뉴 중, '검사(N)' 메뉴를 선택하면 나오는 탭 중에서 'Network'를 클릭하여 해당 페이지를 보여주기 위해서 브라우저가 서버로부터 어떠한 파일들을 다운로드 받는지를 확인할 수 있습니다. Network 탭을 선택하고, 네이버에서 두번째 페이지를 클릭해 보겠습니다 (아래 그림 참조).

두번째 페이지를 클릭하면, 오른쪽 패널에 새로운 파일들의 목록이 보이게 됩니다. 그 중에서 'articleSearch.nhn'이라는 파일이 우리가 원하는 정보를 담고 있는 파일입니다. 이는 해당 파일을 클릭했을 때 나오는 메뉴 중에서 'Preview'나 'Response' 부분을 확인해 보면 (대략적으로) 알 수 있습니다.

Comment [S114]: 이러한 스킬은 어느정도의 경험이 필요합니다.

아래 그림에 나오는 것 처럼 해당 파일을 클릭했을 때 나오는 정보 중 'Header' 탭에 있는 정보를 살펴보면, 해당 파일의 url 주소와 request method 정보를 확인할 수 있습니다. request method 가 POST 인 것을 알 수 있습니다. 이러한 방식으로 데이터를 수집하는 경우에는 페이지의 번호가 바뀌어도 주소창의 url 주소는 변하지 않습니다 (즉, https://land.naver.com/search/article.nhn?tab=article&query=세종특별자치시+아파트) .



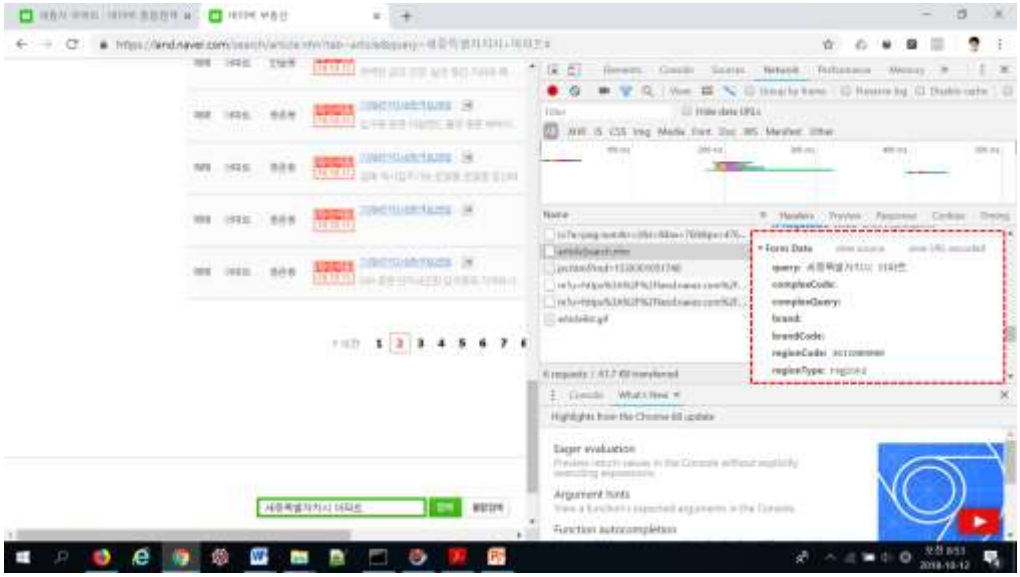
이러한 파일의 정보를 Python 을 이용해서 수집하기 위해서는 requests 모듈에서 제공하는 post() 함수를 사용해야 합니다.

r = requests.post(url, data = payload, headers = headers)

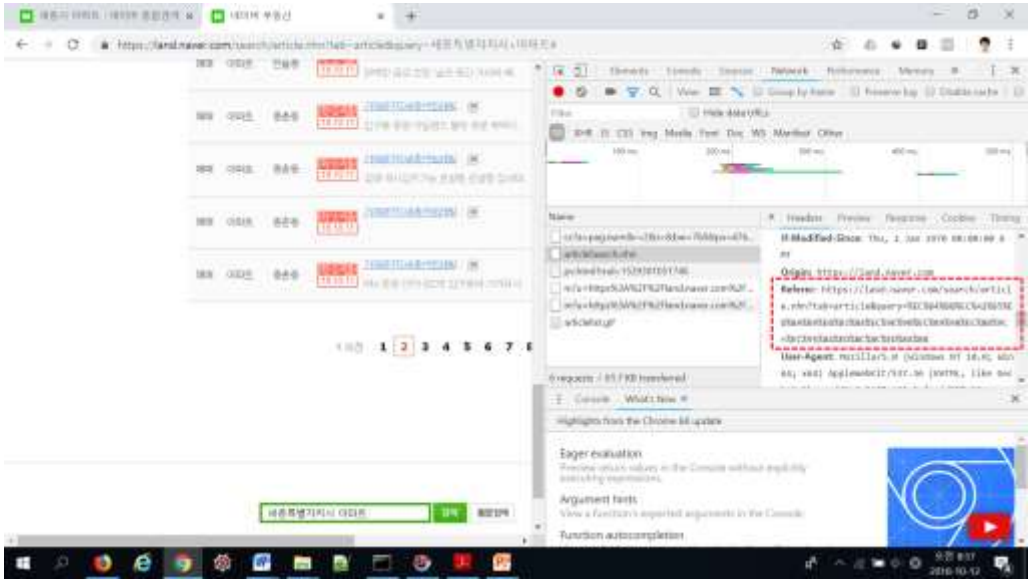
위의 코드에서 첫번째 파라미터인 url 은 해당 파일의 url 이 됩니다. 여기서는 위의 그림에서 확인할 수 있는 것 처럼 https://land.naver.com/search/articleSearch.nhn 이 됩니다.

data 부분에는 여러분들이 네이버 페이지에서 볼 수 있는 정보들을 Python 을 이용해서 수집하기 위해 제공해야 하는 파라미터 정보들이 들어갑니다. 이러한 정보도 브라우저에서 제공하는 network inspection 기능을 사용하여 얻을 수 있습니다. 위의 그림에서와 같이 articleSearch.nhn 이라는 파일을 클릭했을 때 나오는 탭 중에서 headers 탭을 클릭하고 아래로 계속 내려가 보면 아래 그림에서 보이는 'Form Data'라는 부분이 보입니다. 여기에 나오는 정보들이 네이버 페이지에서 보는 정보를 다운로드 받을 때 필요한 추가적인 정보 (파라미터 정보라고 함)에 해당합니다.

주요 정보로는 query, page 등의 정보들이 있습니다. 이러한 추가적인 정보를 payload 라는 이름의 dictionary 변수로 생성을 해서 post() 함수의 두번째 파라미터인 data 의 인자로 전달하면 됩니다.



그리고, 마지막으로 post 함수의 세번째 파라미터인 headers 정보를 제공해야 합니다. 이는 해당 메시지를 서버에 보낼 때 수행되는 컴퓨터간 커뮤니케이션에 필요한 정보를 의미합니다. 그 중에서 반드시 필요한 정보가 Referer 정보입니다 이 정보는 Headers 탭에서 확인할 수 있습니다. 아래 그림의 빨간색 상자 부분입니다.



이러한 정보를 이용해서 실제로 데이터를 수집해 보겠습니다.

```
import requests

payload = {
   'page':'1',
   'query':'세종특별자치시 아파트 매매',
   'regionCode':'3611000000',
   'regionType':'region2',
   'rletType':'A01:A03:A04',
   'tradeType':'A1'
   }

Referer=
'https://land.naver.com/search/article.nhn?tab=article&query=%EC%84%B8%EC%A2%85%ED%8A%B9%EB%B3%84%EC%9E%90%EC%B9%98%EC%8B%9C+%EC%95%84%ED%8C%8C%ED%8A%B8%20%EB%A7%A4%EB%A7%A4'

headers = {'Referer':Referer }

url = 'https://land.naver.com/search/articleSearch.nhn'
r = requests.post(url, data = payload, headers = headers)
```

Comment [S115]: for 문을 이용해서 여러 페이지의 정보를 자동으로 수집할 수 있습니다. 이러한 경우에는 'page'의 값을 순차적으로 변경해줘야 합니다. 그리고, time.sleep(1)을 사용하는 것을 잊지마세요!!

위와 같은 코드를 실행하고 r.text 의 내용을 확인해 보면 여러분이 원하는 정보가 아래 그림과 같이 사전의 형태로 저장되어 있는 것을 알 수 있습니다.

```
print(r.text)
```

{"articleInfo":{"ctmArticleCount":6673,"ctmArticleList":[{"articleNumber":"1818285015","articleName":"롯7단지래미안","articleTypeCode":"A01","articleTypeCodeName":"아파트","articleTypeCodeNameBr":"아파트","tradeTypeCode":"A1","tradeTypeCodeName":"매매","addressCodeDisplay":"3611010600","cityName":"세종시","dvsnName":"","secName":"한솔동","complexCode":"103449","baseSize":"150","sizeTitle1":"공급면적","sizeTitle2":"전용면적","size1":"150","size2":"114","basePrice":"54000","rentDepositPrice":"","price":"54,000","building":"712동","floor":"19/29","fieldArticleYn":"0","imageExistYn":"0","tradeCompletion":"0","tradeCompletionDate":"","baseYmdt":"201810
1207","registYmd":"18.10.11","registPathType":"","cpId":"bizmk","cpName":"매경부동산","realterName":"초마을로알공인중개사사무소","realterTelNo":"044-865-8445","realterId":"cb658445","confirmDate":"20181011000000","articleDescription":"시원한 공원 뷰, 금강 조망과

사실 위의 데이터 타입은 사전 형태가 아닙니다. json 이라는 또 다른 데이터 타입입니다. json 데이터 타입은 웹 커뮤니케이션에서 사용되는 데이터 타입의 하나입니다. Python 의 사전 데이터 형태와 비슷한 형태입니다. 하지만, json 은 Python 에서 제공되는 기본 데이터 타입이 아닙니다. 따라서, json 타입의 데이터를 Python 에서 사용하기 위해서는 사전 데이터 타입으로 변경을 해줘야 합니다. 이는 json 모듈을 통해서 아래와 같이 할 수 있습니다.

```
import json

data_dict = json.loads(r.text)
```

data_dict 는 위의 그림에 나오는 정보와 동일한 정보를 담고 있는 사전 형태의 변수입니다. 사전에 들어있는 키값을 사용해서 여러분들이 원하는 정보를 추출할 수 있습니다.

3. Selenium 사용하기

우리가 추출하고자 하는 정보가 requests.get()을 통해서 다운로드 받는 원본 source codes 에 포함되어 있지 않을 때 사용할 수 있는 또 다른 방법이 selenium 을 사용하는 것입니다. requests.get()을 통해서 다운로드 받는 원본 source codes 에 우리가 원하는 정보가 들어있지 않은 가장 큰 이유 중 하나는 해당 정보가 브라우저가 해당 웹페이지를 display 하기 위해서 최초에 다운로드 받는 source codes 가 아니라 이후에 실시간으로 통신을 하면서 다운로드 받는 데이터에 포함되어 있게 때문에 그렇습니다. 즉, 브라우저는 하나의 웹페이지를 display 하기 위해서 서버와 한번만 통신하는 것이 아니라 연결된 connection 을 통해서 이후에 지속적으로 실시간으로 통신을 하면서 필요한 내용을 실시간으로 다운로드 받습니다. 하지만, 원본 source codes (즉, 브라우저가 처음 서버로부터 다운로드 받는 source codes)에는 이러한 실시간으로 다운로드 받는 내용이 포함되어 있지 않습니다. 따라서 우리가 최종적으로 추출하고자 하는 데이터가 이렇게 실시간으로 서버로부터 다운로드 받아지는 데이터라면, requests.get()을 가지고는 추출할 수가 없습니다. 다행히 network inspection 기능을 통해서 찾을 수 있는 특정 파일에 해당 내용이 담겨 있다면 network inspection 기능을 통해서 해당 데이터를 수집할 수 있지만, 항상 그러한 파일을 찾을 수 있는 것은 아닙니다.

이러한 경우에 사용할 수 있는 방법이 selenium 을 사용하는 것입니다. Selenium 은 컴퓨터 프로그래밍 언어를 통해서 특정한 웹브라우저를 통제하고자 할 때 사용할 수 있는 프로그램입니다. 우리는 selenium 을 이용해서 특정한 웹페이지의 source codes 를 서버로부터 실시간으로 다운로드 받을 수 있습니다. 이렇게 실시간으로 다운로드 받는 source codes 에는 우리가 브라우저에서 보는 내용이 담겨 있습니다. 즉, 우리가 추출하고자 하는 정보를 포함하고 있는 것입니다.

사용자가 특정한 웹페이지를 방문하기 위해서는 일련의 행동들을 합니다. 예를 들어, 브라우저를 실행시키고 접속하고자 하는 웹 페이지의 URL 을 주소창에 입력하고, 해당 페이지에서 특정한 링크나 아이콘을 클릭해서 원하는 페이지로 이동할 수가 있습니다. 그리고 최종 페이지에서 사용자가 원하는 정보를 브라우저를 통해서 볼 수가 있는 것입니다.

Selenium 은 이러한 사용자의 행동을 흉내내어 특정한 웹페이지를 display 하는데 필요한 데이터를 서버로부터 실시간으로 다운로드 받습니다. Selenium 이 실시간으로 다운로드 받는 이러한 데이터에는 우리가 추출하고자 하는 정보가 포함되어 있습니다.

Selenium 설치

Selenium 을 파이썬에서 사용하기 위해서는 selenium 이라는 모듈을 먼저 설치해야 합니다. Anaconda 를 통해서는 기본적으로 설치가 되지 않기 때문에 selenium 은 사용자가 추가적으로 설치를 해줘야 합니다. 이는 아래와 같은 명령 프롬프트 창에서 다음과 같은 명령어를 실행시켜 할 수가 있습니다.

pip install selenium

Selenium 을 설치한 이후에는 해당 모듈을 Python 에서 import 해서 사용하면 됩니다. 아래와 같이 import 를 할 수가 있습니다.

from selenium import webdriver

Selenium 모듈에 포함되어 있는 모든 하위 모듈을 사용하는 것이 아니라 기본적으로는 webdriver 라는 모듈만을 사용합니다. 즉, webdriver 모듈을 사용해서 특정한 웹 브라우저를 통제할 수가 있습니다. webdriver 를 이용해서 특정한 브라우저를 통제하기 위해서는 해당 브라우저의 driver 를 다운로드 받아야 합니다. 여기서는 Chome 의 경우를 설명하도록 하겠습니다. 즉, selenium 의 webdriver 모듈을 통해서 Chrome 브라우저를 통제하기 위해서는 먼저 Chrome driver 를 다운로드 받아야 합니다. 이를 위해서 Google 에서 chrome driver 를 검색합니다. 나오는 검색 결과중에서 Chrome driver downloads 결과를 클릭하면 다음과 같이 chrome driver 를 다운로드 받을 수 있는 사이트로 이동합니다. 여기서 최신 driver 를 다운로드 받습니다. 여러분이 사용하는 OS 에 맞는 version 의 파일을 다운로드 받으면 됩니다. 윈도우 같은 경우에는 xxx 파일을 다운로드 받습니다. zip 으로 압축이된 파일인데 압축을 풀어서 .exe 파일을 특정한 폴더에 저장합니다. 폴더의 이름을 기억해야 합니다.

해당 driver 를 가지고 Chrome 브라우저를 통제할 것이라고 파이썬에서 명시를 해줘야지만 우리가 selenium 을 통해서 Chrome 브라우저를 통제할 수가 있습니다. 이는 아래와 같이 할 수가 있습니다.

```
driver = webdriver.Chrome(executable_path=r'C:\Users\Downloads\chromedriver_win32\chromedriver.exe')
```

Comment [S116]: 경로의 string 값에 r 을 붙여 raw string 형태로 경로값을 입력하게 됩니다.

Comment [S117]: chromedriver.exe 가 저장되어 있는 폴더의 전체 경로입니다.

이렇게 하면 새로운 Chrome 브라우저가 실행이 되고, driver 라는 변수 (object)를 통해서 해당 Chrome 브라우저를 통제할 수가 있습니다. 즉, driver 변수를 통해서 브라우저상에서 사람이 하는 행동을 흉내낼 수 있는 것입니다.

브라우저를 위와 같이 실행한 이후에 특정한 웹페이지로 이동하기 위해서는 selenium 에서 제공되는 get()함수를 사용합니다. requests 의 get()함수와 비슷하게 접속하고자 하는 웹페이지의 URL 주소를 selenium get()함수의 파라미터로 입력합니다.

예를 들어서 우리가 최종적으로 접속하고자 하는 웹 페이지의 URL 주소가 다음과 같다라고 가정하겠습니다.

URL: http://www.yes24.com/24/goods/59673203

그런데 이 페이지를 해당 URL 주소를 가지고 직접 접속하는 것이 아니라 yes24.com 의 첫 페이지에서 이동한다고 가정해 보겠습니다. 이를 위해서는

www.yes24.com 을 입력해서 yes24.com 의 main page 로 이동

Main page 에서 '베스트' 탭 클릭

베스트셀러 목록에서워너원 공식 포토 에세이! 내 책장에 워너원을 저장~ 링크 클릭

이러한 과정을 통해서 우리는 http://www.yes24.com/24/goods/59673203 에 해당하는 페이지로 이동할 수가 있습니다.

이러한 과정을 selenium 을 통해서 해보도록 하겠습니다. 즉, selenium 을 통해서 사용자가 한 행동을 똑같이 흉내내어 우리가 원하는 페이지로 이동하는 것입니다. 이를 위해서 우리가 selenium 을 통해서 해야하는 작업들은 아래와 같습니다.

Chrome 브라우저 실행하기

www.yes24.com main page 에 접속하기

'베스트' 탭 클릭하기

워너원 공식 포토 에세이! 내 책장에 워너원을 저장~ 링크 클릭하기

이러한 과정을 거쳐서 selenium 을 통해 해당 페이지로 이동한 후에 우리가 해야하는 것은 해당 페이지의 source codes 를 실시간으로 서버로 부터 다운로드 받는 것입니다. 그리고 그 source codes 안에 담겨져 있는 우리가 최종적으로 다운로드 받고자 하는 내용을 BeautifulSoup 을 통해서 추출하면 됩니다.

아래 코드와 같이 selenium 을 통해서 우리가 원하는 페이지로 이동할 수 있습니다.

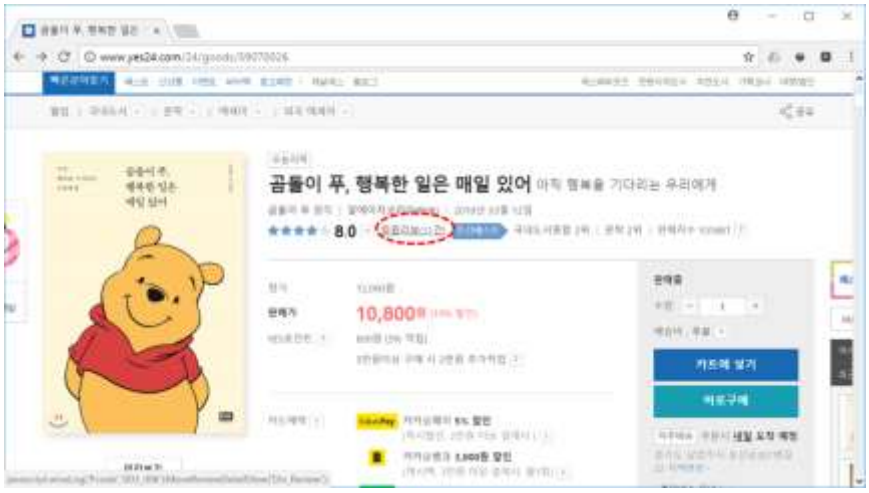
```
import requests
```

```
from bs4 import BeautifulSoup
from selenium import webdriver

driver =
webdriver.Chrome(executable_path=r'C:\Users\Sang\Downloads\chromedriver_win32_1\chromedriver.exe')

url = 'http://www.yes24.com'
driver.get(url)
element1 = driver.find_element_by_xpath('//*[@id="yesFixCorner"]/dl/dd/ul[1]/li[1]/a')
element1.click()
element2 = driver.find_element_by_xpath('//*[@id="bestList"]/ol/li[1]/p[1]/a')
element2.click()
html = driver.page_source
driver.close()
```

원본 source codes 에 담겨있지 않은 정보 추출하기

이번에는 원본 source codes (즉, requests.get()을 통해서 다운로드 되는 source codes)에 포함되지 않은 정보를 slenium 을 통해서 추출해 보도록 하겠습니다.

yes24.com 에 있는 '곰돌이 푸, 행복한 일은 매일 있어' 라는 책에 대한 독자들의 리뷰 정보를 추출해 보도록 하겠습니다. 이러한 정보는 책의 정보를 담고 있는 페이지 ()에서 '회원리뷰' 링크를 클릭하여 볼 수가 있습니다.



Comment [S118]: chromedriver 파일을 이용해서 selenium 을 통해서 Chrome 브라우저를 실행시키고 그 통제권한을 driver 라는 변수에 할당합니다. 그러면, 우리는 driver 라는 변수를 통해서 실행된 Chrome 브라우저를 통제할 수가 있습니다.

Comment [S119]: 우리가 브라우저를 실행시킨 이후 최초로 접속하는 페이지입니다.

Comment [S120]: 해당 URL 주소를 가지고 selenium 에서 제공되는 get()함수를 통해서 해당 페이지로 이동합니다.

Comment [S121]: Main page 로 이동한 다음 우리가 해야하는 것은 '베스트'라는 탭을 클릭하는 것입니다 '베스트' 탭에 해당하는 element 를 find_element_by_xpath()라는 함수를 사용해서 찾습니다.이 함수는 특정 element 를 찾는데 해당 element 의 xpath 를 사용합니다. 특정 element 와 xpath 대한 자세한 내용은 Appendix 를 참고하시기 바랍니다.

Comment [S122]: '베스트' 탭의 xpath

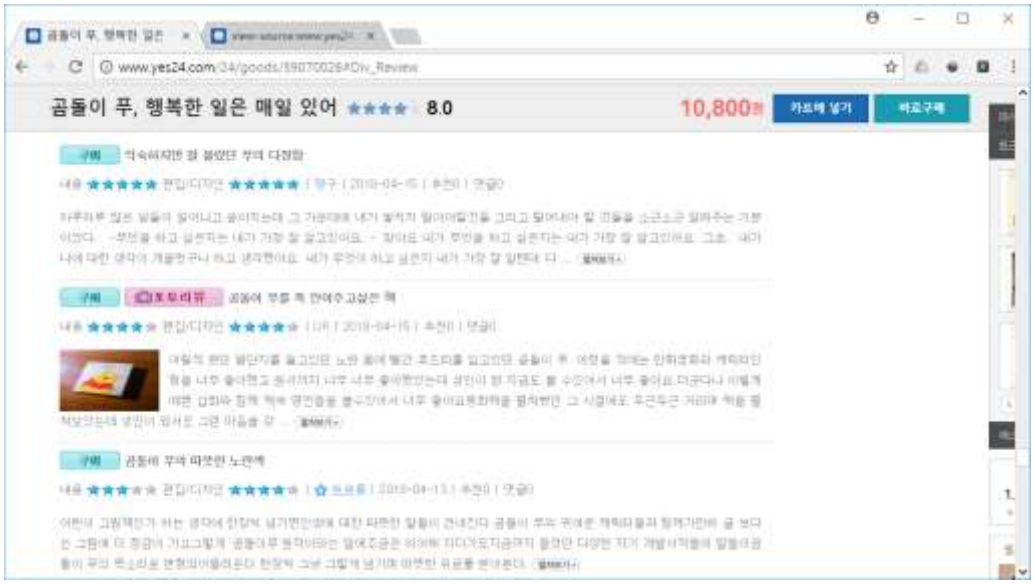
Comment [S123]: '베스트' 탭에 해당하는 element 를 찾은 후 클릭하기 위해서 click() 함수를 사용합니다. 그러면, 베스트셀러 목록을 보여주는 페이지로 이동합니다.

Comment [S124]: 그 다음에 우리가 찾아야하는 element 는 우리가 보고자 하는 페이지의 링크입니다. 역시나 마찬가지로 find_element_by_xpath() 함수를 통해서 해당 링크를 찾은 후, click()함수를 사용해서 클릭을 합니다.

Comment [S125]: 최종적으로 이동한 웹페이지의 source codes 를 실시간으로 서버로부터 다운로드 받습니다.

Comment [S126]: 해당 브라우저를 종료합니다.

'회원리뷰' 링크를 클릭하면 브라우저를 통해서 아래와 같은 독자의 리뷰 정보를 볼 수 있습니다.



만약 이러한 리뷰 정보가 우리가 추출하고자 하는 정보라고 한다면 1 차적으로 생각해 볼수 있는 방법이 requests.get() 함수를 사용하는 것입니다. 이러한 방법을 통해서 해당 정보를 추출하기 위해서는 원본 source codes 안에 해당 정보가 저장되어 있어야 합니다. 하지만, 아쉽게도 브라우저의 페이지 소스 보기 메뉴를 통해서 확인해 본 결과 독자의 리뷰 정보는 원본 source codes 에는 저장되어 있지 않은 것을 확인할 수 있습니다. 이때 사용할 수 있는 방법이 selenium 을 사용하는 것입니다. 즉, 리뷰 정보가 원본 source codes 에 저장이 되어 있지 않다라는 뜻은 브라우저가 해당 정보를 화면에 display 하기 위해서는 해당 정보를 실시간으로 서버로부터 다운로드 받는다는 것을 의미합니다. 우리는 이러한 데이터를 selenium 을 통해서 실시간으로 다운로드 받을 수 있습니다.

이를 위해서 우리는 selenium 이 해당 정보를 보기위해 사용자가 하는 행동을 흉내내도록 만들어야 합니다. 해당 정보를 보기 위해서 사용자는 다음과 같은 행동을 합니다.

해당 책의 페이지로 이동하기 (http://www.yes24.com/24/goods/59070026)

Comment [S127]: yes24 의 main page 에서 관련 링크를 클릭해서 이동할 수도 있지만, 여기서는 해당 페이지의 URL 주소를 사용해서 직접 접속한다고 가정하겠습니다.

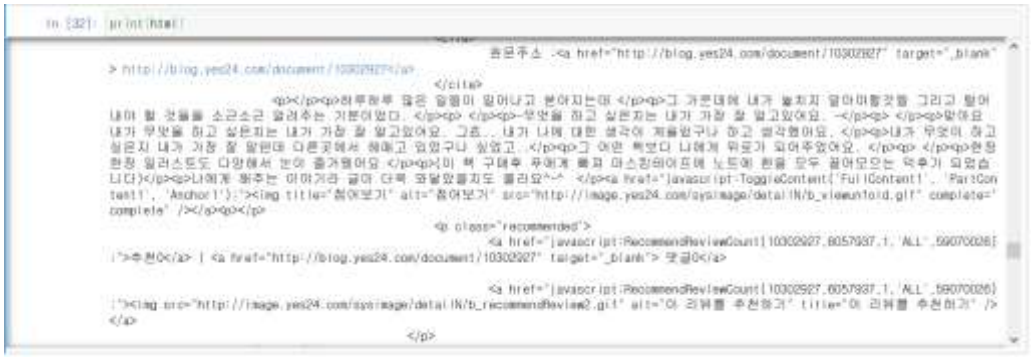
해당 웹페이지에서 '회원리뷰' 링크 클릭하기

아래와 같은 코드를 사용해서 selenium 을 통해 리뷰 정보가 나오는 페이지로 이동할 수 있습니다.

```
import requests
```

```
from bs4 import BeautifulSoup
from selenium import webdriver

driver =
webdriver.Chrome(executable_path=r'C:\Users\Sang\Downloads\chromedriver_win32_1\chromedriver.exe')
url = 'http://www.yes24.com/24/goods/59070026'
driver.get(url)

element1 = driver.find_element_by_xpath('//*[@id="yDetailTopWrap"]/div[2]/div[1]/span[3]/span[2]/a')
element1.click()

html = driver.page_source
driver.close()
```

위의 코드를 실행시키면 html 안에는 아래와 같이 화면에서 볼 수 있는 독자 리뷰 정보가 저장되어 있는 것을 확인할 수 있습니다.



우리가 원하는 정보를 담고 있는 source codes 를 selenium 을 통해 다운로드 받은 후 BeautifulSoup 을 사용해서 해당 정보를 추출할 수 있습니다.

Comment [S128]: '회원리뷰' 링크 (하나의 element)의 xpath 입니다.

Comment [S129]: click() 함수를 사용해서 '회원리뷰' 링크를 클릭합니다. 그러면 회원리뷰 정보를 보여주는 페이지로 이동하게 됩니다.

Comment [S130]: 독자들의 리뷰 정보에 대한 source codes 를 실시간으로 서버로부터 다운로드 받습니다. 이 source codes 안에는 독자들의 리뷰 정보가 담겨 있습니다.

Appendix

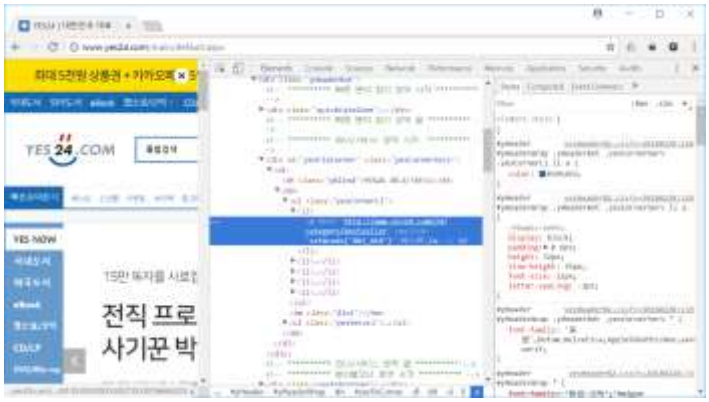
1. 웹 페이지를 구성하는 elements 와 xpath

하나의 웹페이지를 구성하고 있는 여러가지 요소 (예, 링크, 이미지, 텍스트 등)을 elements 라고 부릅니다. 즉, 하나의 웹페이지는 여러 개의 elements 로 구성되어 있는 것입니다. 그리고, 각 element 는 고유한 정보를 갖습니다. xpath, name, id 등이 그러한 것입니다. 이중에서 xpath 는 특정 element 의 고유 경로라고 생각할 수 있습니다. 즉 하나의 웹페이지를 구성하는 element 마다 고유한 xpath 가 있는 것입니다. 이러한 element 의 경로는 컴퓨터에 존재하는 특정 파일의 경로와 비슷하다고 생각할 수 있습니다. 이러한 xpath 를 사용해서 우리는 특정한 element 를 찾을 수가 있습니다.

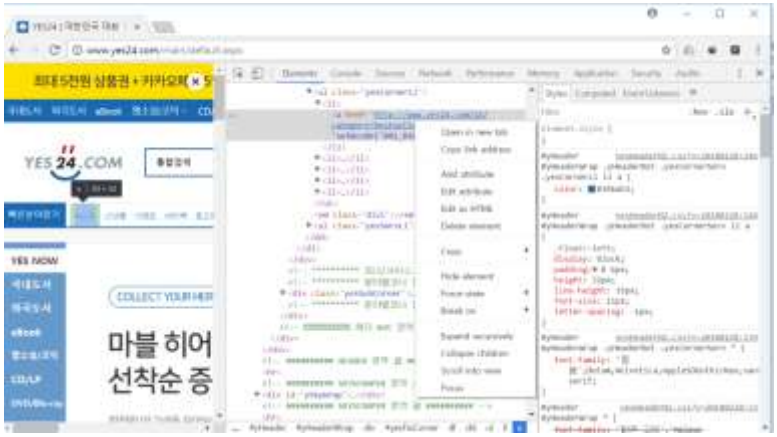
특정 element 의 xpath 찾기

웹페이지를 구성하는 특정 element (image, text, link 등)의 고유경로인 템소를 찾기 위해서 우리는 웹브라우저를 사용할 수 있습니다. 예를 들어서, www.yes24.com 페이지에서 '베스트'탭에 해당하는 element 의 xpath 를 찾고자 한다라고 가정을 하겠습니다.

이를 위해서 '베스트' 탭 위해 마우스 커서를 위치시킨 후 오른쪽 버튼을 클릭합니다. 나오는 메뉴 중에서 '검사 (N)' 을 클릭합니다. 그러면 아래와 같은 화면을 볼 수 있습니다.

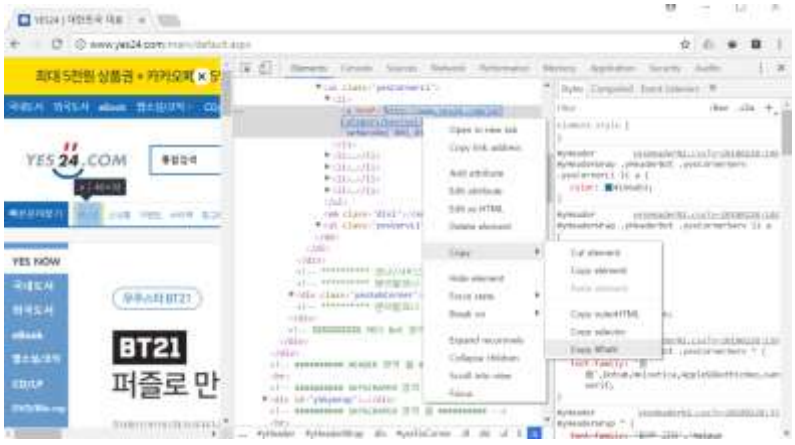


각 element 는 고유의 tag 정보를 가지고 있습니다. 특정 element 에 대해서 '검사 (N)' 기능을 실행하게 되면 해당 tag 의 내용이 파란색으로 하이라이트 됩니다. 해당 element 의 xpath 는 해당 tag 의 xpath 와 같습니다. 즉, 우리는 highlighted 된 tag 의 xpath 를 알아야 합니다. 이를 위해서 파란색으로 highlighted 된 부분에 마우스 커서를 다시 위치시키고 오른쪽 버튼을 다시 한번 클릭합니다. 그러면 아래와 같은 여러개의 메뉴들이 나옵니다. 이 메뉴들 중에서 중간 부분에 있는 'Copy' 메뉴 위에 마우스 커서를 위치합니다.



그러면 아래와 같은 세부 메뉴들이 나옵니다. 그 메뉴들 중에서 'Copy XPath' 를 클릭하면, 해당 tag (즉, 해당 element)의 xpath 를 복사할 수 있습니다. 우리가 해야하는 것은 이렇게 복사된 내용을 find_element_by_xpath() 함수의 파라미터로 붙여 넣기를 하는 것입니다. 그럼 아래와 같은 값이 입력 됩니다.

```
driver.find_element_by_xpath('//*[@id="yesFixCorner"]/dl/dd/ul[1]/li[1]/a')
```


YES24.COM
YES NOW
BT21
퍼즐로 만