우리에게 보여지는 네이버 블로그 URL은 사실 가짜 주소인 것을 알고 계신가요? 이번 포스팅에서는 네이버 블로그의 도서 리뷰를 크롤링하는 과정에서 학습한 내용을 공유합니다.

네이버 추천도서 정보 웹 크롤링

추천모델을 생성하는 연습용 프로젝트를 진행하고 있습니다. 네이버 책의 소설 > 나라별 소설 카테고리의 추천도서 969권의 정보를 수집, Word2Vec 모델 기반으로 추천 모델을 생성하고자 합니다.

수집하고자 하는 책 정보의 항목은 다음과 같습니다.

- 책 제목 텍스트
- 책 상세 웹페이지 URL 텍스트
- 상세 웹페이지의 '책소개' 항목 텍스트
- 상세 웹페이지의 '책 속으로' 항목 텍스트
- 상세 웹페이지의 '네티즌 리뷰' 항목 텍스트
    - 개인들의 네이버 블로그에 작성된 책 리뷰
    - 리뷰 총합이 100건을 초과할 경우 100건만 수집

문제는 리뷰 텍스트의 수집에서 발생했습니다. Selenium(chromedriver)와 BeatifulSoup 등 다양한 방법으로 접근해 봐도, 본문 텍스트를 가져오지 못했는데요.

네이버 API를 사용하여 urllib.request.Request 객체에 API 사용자 인증 정보를 header 추가하여도 마찬가지였습니다. 무엇이 문제였을까요?

가짜 URL, 진짜 URL

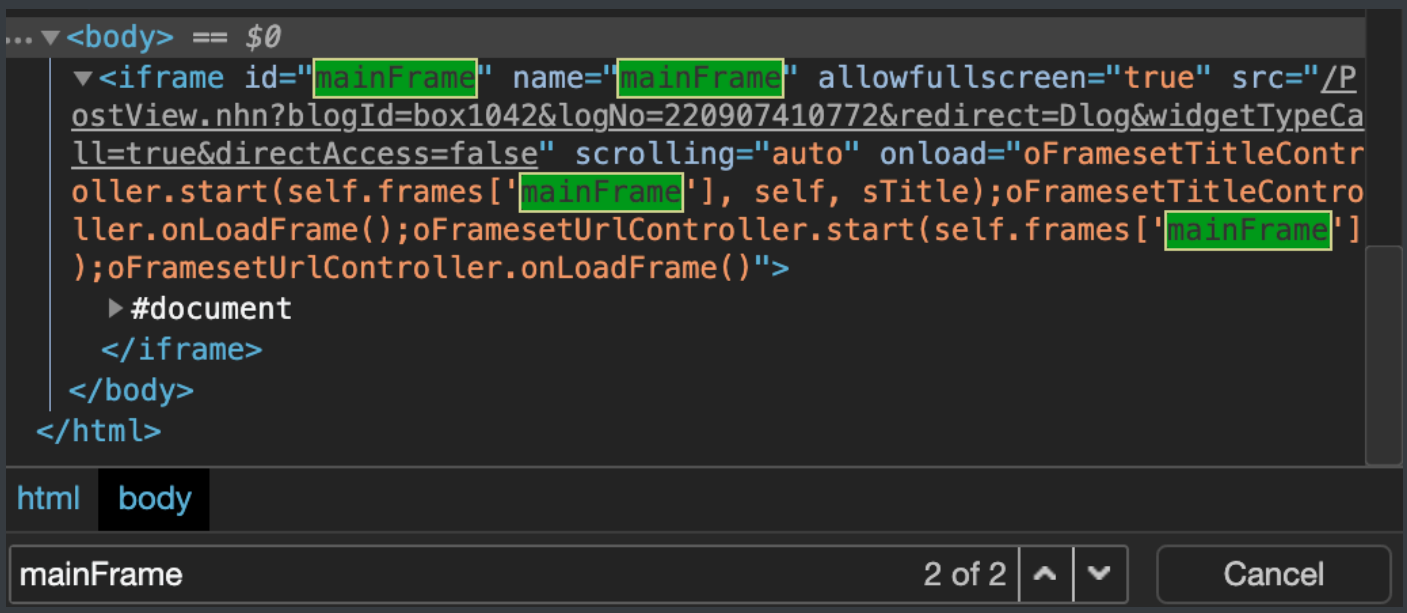
네이버 블로그에 업로드된 포스트는 각각 가짜 주소와 진짜 주소를 가지고 있습니다. 가짜 주소는 우리가 일반적으로 네이버 블로그 포스트에 접근할 때 보여지는 주소이며, 웹 크롤링이 불가능하도록 본문 등 콘텐츠 텍스트가 숨겨져 있습니다.

도저히 콘텐츠 텍스트를 얻을 수가 없었던 이유입니다. 가짜 URL에서 정보를 가져오려고 시도했던 것이니까요. 이와 반대로, 진짜 URL은 크롤링이 가능한 주소이며, 숨겨져 있습니다.

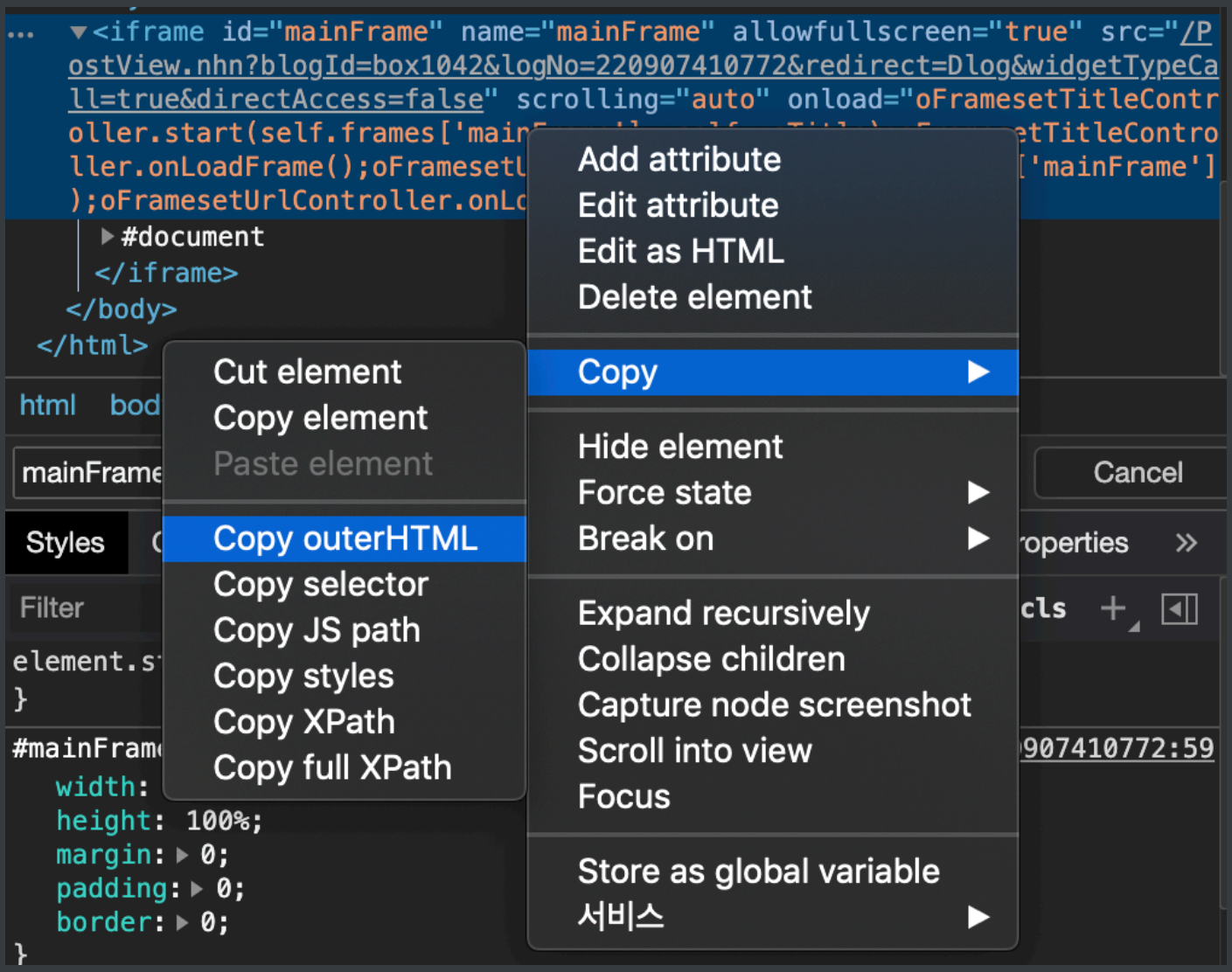
어디에 숨겨져 있을까요? 아래 도서 리뷰 포스트를 예시로 들어보겠습니다.

- https://blog.naver.com/box1042/220907410772

크롬 브라우저로 위 주소에 접속하여 마우스 우클릭 > 검사(investigation)를 찍어 보면 해당 페이지의 코드를 확인할 수 있습니다. Ctrl+F로 'mainFrame'을 검색하면 다음과 같은 결과를 확인할 수 있는데요.



위 코드를 마우스 우클릭하여 Copy > Copy outerHTML하면, id와 name 속성을 mainFrame으로 가지는 iframe에 대한 코드를 확인할 수 있습니다.


<iframe id="mainFrame" name="mainFrame" allowfullscreen="true" src="/PostView.nhn?blogId=box1042&logNo=220907410772&redirect=Dlog&widgetTypeCall=true&directAccess=false" scrolling="auto" onload="oFramesetTitleController.start(self.frames['mainFrame']
#document
</iframe>
</body>
</html>
Add attribute
Edit attribute
Edit as HTML
Delete element
Copy
Hide element
Force state
Break on
Expand recursively
Collapse children
Capture node screenshot
Scroll into view
Focus
Store as global variable
서비스
Cut element
Copy element
Paste element
Copy outerHTML
Copy selector
Copy JS path
Copy styles
Copy XPath
Copy full XPath
Cancel
Styles
Filter
width:
height: 100%;
margin: 0;
padding: 0;
border: 0;

```
<iframe id="mainFrame" name="mainFrame" allowfullscreen="true"
src="/PostView.nhn?
blogId=box1042&amp;logNo=220907410772&amp;from=search&amp;redirect=Log&a
mp;widgetTypeCall=true&amp;topReferer=https%3A%2F%2Fsearch.naver.com%2Fs
earch.naver%3Fquery%3D%255B%25EC%25A0%2595%25EC%2583%2581%25EC%2596%25B4
%25ED%2595%2599%25EC%259B%2590%2520%25EC%259D%25B8%25EC%25B2%259C%25EB%2
582%25A8%25EB%258F%2599%25EB%25B6%2584%25EC%259B%2590%28%25EA%25B5%25AC%
25EC%259B%2594%25EB%258F%2599%29%255D%25EC%2584%259C%25EC%259A%25B8%25EB
%258C%2580%2520%25EA%25B6%258C%25EC%259E%25A5%2520%25EB%258F%2584%25EC%2
584%259C%2520-
%2520%25EC%2595%2588%25EB%2582%2598%2520%25EC%25B9%25B4%25EB%25A0%2588%2
5EB%25A6%25AC%25EB%2582%2598%2522%26nso%3D%26where%3Dblog%26sm%3Dtab_viw
.all&amp;directAccess=false" scrolling="auto"
onload="oFramesetTitleController.start(self.frames['mainFrame'], self,
sTitle);oFramesetTitleController.onLoadFrame();oFramesetUrlController.st
art(self.frames['mainFrame']);oFramesetUrlController.onLoadFrame()">
</iframe>
```

위 코드에서 src라는 속성에 부여된 텍스트가 있습니다. "/PostView.nhn?blogId=..."로 시작하는 텍스트인데요. 이 텍스트가 바로 진짜 주소값의 일부를 가리킵니다. 정확히 말하면, https://blog.naver.com 다음에 오는 주소값입니다.

정리하면, https://blog.naver.com/box1042/220907410772 라는 네이버 블로그 포스트의 웹크롤링 가능한 진짜 주소는 다음과 같게 됩니다.

```
https://blog.naver.com/PostView.nhn?
blogId=box1042&amp;logNo=220907410772&amp;from=search&amp;redirect=Log&a
mp;widgetTypeCall=true&amp;topReferer=https%3A%2F%2Fsearch.naver.com%2Fs
earch.naver%3Fquery%3D%255B%25EC%25A0%2595%25EC%2583%2581%25EC%2596%25B4
%25ED%2595%2599%25EC%259B%2590%2520%25EC%259D%25B8%25EC%25B2%259C%25EB%2
582%25A8%25EB%258F%2599%25EB%25B6%2584%25EC%259B%2590%28%25EA%25B5%25AC%
25EC%259B%2594%25EB%258F%2599%29%255D%25EC%2584%259C%25EC%259A%25B8%25EB
%258C%2580%2520%25EA%25B6%258C%25EC%259E%25A5%2520%25EB%258F%2584%25EC%2
584%259C%2520-
%2520%25EC%2595%2588%25EB%2582%2598%2520%25EC%25B9%25B4%25EB%25A0%2588%2
5EB%25A6%25AC%25EB%2582%2598%2522%26nso%3D%26where%3Dblog%26sm%3Dtab_viw
.all&amp;directAccess=false
```

이제 가짜 주소로부터 진짜 주소를 획득하는 방법을 알았습니다. 아래와 같은 코드를 작성하여, 네이버 블로그 포스트의 본문 내용을 가져와 봅니다.

```
from bs4 import BeautifulSoup
import requests
import urllib.request
import urllib.error
import urllib.parse
import re
import json

naver_client_id = "##############"
naver_client_secret = "#########"

def get_naver_blog_text(client_id_, client_secret_, keyword):
  """
  client_id_ : 네이버 검색 API ID
  client_secret_ : 네이버 검색 API password
  keyword : 검색어 (네이버 블로그 포스트 제목)
  """
```

```
    # naver API info
    client_id = client_id_
    client_secret = client_secret_

    encText = urllib.parse.quote(keyword)

    # 검색 쿼리를 전송할 url address
    url = "https://openapi.naver.com/v1/search/blog?query=" + encText #
json 결과
    # url = "https://openapi.naver.com/v1/search/blog.xml?query=" +
encText # xml 결과

    # 쿼리 url 전송
    request = urllib.request.Request(url)
    # Naver API 인증
    request.add_header("X-Naver-Client-Id",client_id)
    request.add_header("X-Naver-Client-Secret",client_secret)

    # 서버 응답 수신
    response = urllib.request.urlopen(request)
    rescode = response.getcode()

    # 서버 응답 코드 200 (정상)
    if(rescode==200):
        # 서버가 전송한 내용으로부터 블로그 포스트 url 추출
        response_body = response.read()
        response_body.decode('utf-8')
        response_body_dict = json.loads(response_body.decode('utf-8'))

        # 블로그 포스트 제목으로 검색한 결과의 첫번째 아이템의 url
        blog_post_url = response_body_dict['items'][0]
['link'].replace("amp;", "")

        # 추출한 블로그 포스트 url로부터 BeautifulSoup 생성
        get_blog_post_content_code = requests.get(blog_post_url)
```

```
            get_blog_post_content_text = get_blog_post_content_code.text
            get_blog_post_content_soup =
    BeautifulSoup(get_blog_post_content_text, 'lxml')

            # BeatifulSoup으로부터 웹크롤링 가능한 진짜 url 획득
            real_blog_post_url = "http://blog.naver.com" +
    get_blog_post_content_soup.select('#mainFrame')[0].get('src')

            # 진짜 url로부터 BeautifulSoup 생성
            get_real_blog_post_content_code =
    requests.get(real_blog_post_url)
            get_real_blog_post_content_text =
    get_real_blog_post_content_code.text
            get_real_blog_post_content_soup =
    BeautifulSoup(get_real_blog_post_content_text, 'lxml')

            # 포스트의 본문 텍스트를 저장할 리스트
            contents = []

            # 진짜 url 웹페이지의 본문 태그 내용에 대하여 for loop
            for content in
    get_real_blog_post_content_soup.select('div#postViewArea'):
                # 본문 텍스트 추출
                blog_post_content_text = content.get_text()

                # HTML 태그 제거
                remove_html_tag = re.compile('<.*?>')
                blog_post_full_contents = str(blog_post_content_text)

                # 리스트에 HTML 태그 제거한 본문 텍스트 저장
                contents.append(blog_post_full_contents)

                # 리스트에 존재하는 텍스트를 string으로 합쳐서 반환
                return ' '.join(contents)

        # 서버 응답 코드가 200이 아님 (비정상)
```

```
        else:
            # 에러 코드를 출력하고 None 반환
            print("Error Code:" + rescode)
            return None
```

위 코드의 동작을 확인합니다.

```
get_naver_blog_text(naver_client_id, naver_client_secret,
                    "[정상어학원 인천남동분원(구월동)]서울대 권장 도서 - 안나 카레리나")
```

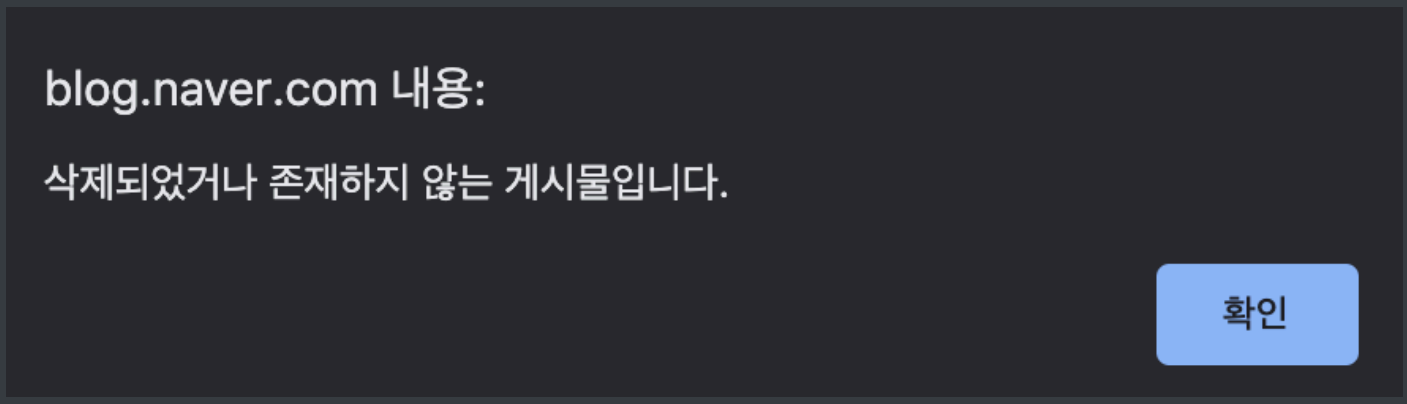
```
'\n\n\n\n\n\n\n안나카레니나 하\n\n작가\n레프 톨스토이\n출판\n범우사\n발매\n2015.04.10.\n\n리뷰보기\n\n\n\u200b\n\xa0\n\xa0\n★ 저자 및 작품소개\xa0\n작가 소개레프 톨스토이러시아의 소설가 · 사상가. 도스토예프스키, 투르게네프와 더불어 '러시아 3대 문호'로 일컬어지고 있다. 1828년 남러시아 툴라 근처에 있는 영지 야스나야 폴랴나에서 명문 백작가의 4남으로 태어났으며 어려서 부모를 잃고 고모을 후견인으로 성장했다. 카잔대학에 입학했으나 중도에 자퇴했다. 1847년에 고향으로 돌아와 농장일에 전념했으나 실패하고 1851년에 카프카즈의 군대에 들어갔다. 1852년 처녀작 『유년시대』를 발표하여 투르게니에프로부터 문학성을 인정받기도 하였다. 그 후 러시아 농민의 비참한 현실에 눈을 뜬 그는 농민계몽을 위해 야스나야 폴랴나 학교를 세우고 농노해방운동에도 활발히 참여하였다. 그후 1869년에 완성한 『전쟁과 평화』로 세계적인 작가로서의 명성을 얻었으며, 러시아의 현실과 고통받는 러시아 민중의 삶을 여러 각도에서 포착하여 생동감 있게 그려내 오늘날까지도 19세기 러시아 문학을 대표하는 세계적 문호로 인정받고 있다. 1870년대 후반기에 수많은 정신적 갈등과 고뇌를 겪고 난 뒤 홀연히 농부로 변신하였으며 1885년에는 뽀스레드니끄(중개인이라는 뜻) 출판사를 만들어 러시아 민화와 복음서의 진리를 대중에 쉽게 이해할 수 있도록 꾸민 책들을 펴내기 시작했다.1870년대 후반 『안나 카레리나』의 마지막 몇 장을 쓸 무렵 그는 모든 것을 무의미한 것으로 만들어버리는 죽음에의 공포에 사로잡혀 인생의 의미에 대한 고민을 계속하게 된다. 결국 삶의 의의는 과학이나 철학도 설명할 수 없고, 이성의 힘에 의지해서도 해결되지 않으며, 죽음을 두려워하지 않고 자연스러운 것으로 받아들이는 민중의 태도에서 배우지 않으면 안 된다는 결론에 도달한다. 그녀의 비극은 사회 가치에 대한 대안을 추구하는 레빈의 행위와 평행을 이루는데, 레빈은 자신의 영지에 있는 농부들 사이에서 해답을 찾는다. 『안나 카레리나』에서 정신적 위기와 극복이 이른바 톨스토이의 회심(回心)이며 『참회록』 속에 서술된 고백의 내용이다. 여기서부터 톨스토이는 현대의 타락한 그리스도를 배제하고 원시 그리스도에 복귀하여 근로, 채식, 금주, 금연의 생활을 영위했다. 원시 기독교의 소박성을 지닌 포괄적인 비전에 부합된 삶을 살려고 노력함으로써 예언적인 현자가 되었다. 톨스토이는 그렇지 않으면 뒤얽혀버렸을 인생에서 자기 책의 핵심을 형성해 주는 인생의 의미를 끊임없이 추구했다. 도덕적 필연성과 합리적 기독교 윤리에 바탕해 농민적 무정부주의, 악에 대한 무저항 정신으로 대변되는 그의 사상은 한때 전 서계로 퍼져 톨스토이즘을 불러일으키기도 했다. 이후 수많은 평론과 소책자, 교훈적인 단편소설 등을 통해 사랑과 믿음으로 가득 찬 삶에 대한 자신의 신념을 주장하고 인간이 만들어낸 정부, 교회 등의 제도와 사유재산을 부정하는 자신의 견해를 전파했다.톨스토이의 걸작 『전쟁과 평화』는 나폴레옹 침략 사건을 러시아의 여러 가정 문제를 통해 그려낸 거대한 서사시다. 이 작품에서는 특히 자신들의 삶 속에서 중요성과 의미를 찾고자 하는 두 사람, 즉 안드레이 볼콘스키 왕과 피에르 베주호프에게 초점이 맞춰져 있다. 톨스토이는 남은 생애를 자신의 원칙에 따라 살려고 노력하는 한편, 자신의 철학을 책으로 구현하면서 보냈다. 종교적 전향 이후에는 비록 도덕주의자 톨스토이가 인생과 인간 경험의 활력 및 다양성을 뛰어나게 포착해 낸 예술가 톨스토이보다 우세할지라도, 그 시기에 나온 작품들 가운데 일부는 그의 가장 훌륭한 작품에 속한다. 특히 『이반 일리치의 죽음』(1886)과 『크로우처 소나타』(1891)가 그렇다. 톨스토이의 신비주의와 금욕주의는 헌신적인 추종자들을 매혹시킨 반면, 아내와 가족으로부터는 그를 소외시켰다. 82살 되던 해 그는 그의 가르침과 그의 개인적 부유함의 부등으로 괴로워하던 중, 그의 아내와 말다툼 한 후 집을 나왔다. 3일 후, 1910년 11월 20일 빈촌의 정거장에서 폐렴으로 죽었다.러시아 민화에 기반을 둔 『바보 이반』, 『사람은 무엇으로 사는가』, 『인간에게는 얼마만큼의 땅이 필요한가』 등과 같은 짧지만 진정한 교훈을 주며 삶의 의미를 반추하게 하는 작품들을 써내기도 했다. 그 외의 작품으로 『교의신학비판』『참회록』『나의 신앙』『부활』『유년시대』『소년시대』『청년시대』『세바스토폴 이야기』『카자흐 사람들』등이 있다.[예스24 제공]작품 소개\n미모의 귀족부인 안나의 부정한 사랑을 중심으로 1870년대 러시아 귀족사회의 양상을 묘사한 러시아 대문호 톨스토이의 장편. 근대 소설의 한 전형으로서, 나폴레옹의 모스크바 침략을 중심으로 러시아 사회를 그린 <전쟁과 평화>, 자신의 사상, 종교, 예술의 일체를 구현한 <부활>과 더불어 3대 걸작의 하나로 일컬어진다.톨스토이의 수많은 소설 중에서도 예술적 완성도가 가장 높아 토마스 만, 도스토예프스키 등 훌륭한 작가들로부터 극찬을 받은 작품으로 일종의 가정 소설이자 사회 소설이다.그는 여기서 안나와 브론스키의 구원받을 수 없는 관능적인 사랑에 레빈과 키치의 진정한 기독교적 사랑을 대치시키고 있다.  전자가 단순한 육체적이며 이기적인 사랑인 데 비해, 후자는 형이상학적 개념의 자기 희생적인 사랑이다. 바로 여기에 작자가 전하려는 참된 교훈의 핵심이 있다.이 핵심을 가능한 한 예술적으로 명확하게 제시하기 위해 그는 놀라운 형상의 흐름 속에서 두 가지의 사랑을 대비, 묘사하고 있다. 이 이질적인 두 정열의 흐름은 아내 돌리를 "임신기계"로 밖에 생각지 않는 향락주의자 오블론스키 부부에 의해 서로 관련지어지면서 하나의 통일된 세계를 형성하고 있다.- 범우사[인터넷 교보문고 제공]\xa0\n\xa0\n\xa0\n\xa0\n\xa0\n\xa0\n\xa0\n\n\xa0\n\n\n\n'
```

위 코드로 본문을 가져오지 못하는 경우?

정상적인 작동을 확인했지만, 작동하지 않는 경우도 발생했습니다. 예를 들면 다음과 같은 블로그 포스트였는데요.

- https://blog.naver.com/here-now-lee/222215012401

동일한 방식으로 위에서 작성한 함수를 실행했지만 None이 반환되었습니다. 위 블로그 포스트의 진짜 URL을 찾아서, 웹브라우저를 통해 직접 접근을 시도해 보았습니다. 그 결과, 다음과 같은 팝업을 확인했습니다.



이 경우, 아래 2가지에 대하여 실패했다고 생각할 수 있습니다.

1. 네이버 검색 API를 통해 접근했으나 실패
2. 진짜 URL 주소로 직접 접근했으나 실패

이러한 실패 경우에 대해서도 크롤링을 수행할 수 있는 방법은 추가적인 고민이 필요해 보입니다. 이상으로 오늘의 포스팅을 마칩니다.

Reference

- 네이버 검색 API 사용하기
- 네이버 블로그의 특정 키워드 크롤링하기
- 파이선 크롤링 체험기
- jy617lee/naver_blog_crawler