

Web Crawling

우제혁
2023.05.15

목 차

01

- **Crawling이란?**

02

- **Instagram Crawling**

03

- **검색엔진 크롤링**
- **크롤링 과 보안**

Crawling이란?

크롤링 : 웹사이트에서 데이터를 수집하는 과정을 의미

↳ 특정 사이트에서 일정 부분의 값을 가져오고 이를 다룰수 있으면 매우 유용하다

3가지 방법

Beautiful Soup

HTML, XML 등과 같은 마크업 언어를 파싱하기 위한 라이브러리.
Beautiful Soup을 사용하면 HTML 코드를 더 쉽게 분석하고, 원하는 데이터를 추출할 수 있다.
Beautiful Soup은 매우 유연하고 강력한 라이브러리이며, 웹 크롤링에서 가장 많이 사용되는 라이브러리 중 하나.

JavaScript 또는 기타 동적 요소를 처리할 수 없다.

Selenium WebDriver

웹 어플리케이션을 테스트하기 위한 도구로 개발되었으며, 웹사이트를 자동으로 조작하는 데 사용.
Selenium WebDriver는 웹 브라우저의 동작을 모방하여, 실제 웹 브라우저처럼 웹사이트를 조작할 수 있습니다.
셀레니움 웹 드라이버는 테스트를 위해 개발된 도구지만, 웹 크롤링에서도 자주 사용.

JavaScript 또는 기타 동적 요소를 처리할 수 있다.

Requests

Requests는 HTTP 요청을 보내고, HTTP 응답을 받기 위한 Python 라이브러리.
Requests를 사용하면 쉽고 빠르게 HTTP 요청을 처리하고, HTTP 응답을 처리할 수 있다.
Requests는 매우 직관적인 API를 제공하며, 간단한 웹 크롤링 작업에서 가장 많이 사용되는 라이브러리 중 하나.

활용 예시 Requests를 사용하여 HTML 코드를 가져온 후,
Beautiful Soup을 사용하여 필요한 데이터를 추출하고,
Selenium WebDriver를 사용하여 웹사이트를 자동으로 조작하여 데이터를 수집

목 차

02

Instagram Crawling

Instagram Crawling 코드 분석

초기 설정

```
scraper = SNSCrawler('D:\chromedriver_win32\chromedriver.exe')  
  
instagram_profile = scraper.get_instagram_profile('joongsint')
```

```
def get_instagram_profile(self, username):  
    login_name = ''  
    login_pw = ''  
    try:  
        url = 'https://www.instagram.com/' + username  
        options = webdriver.ChromeOptions()  
        options.add_argument('headless')  
        options.add_argument('--disable-extensions')  
        options.add_argument('--disable-gpu')  
        options.add_argument('--no-sandbox')  
        options.add_argument('--lang=ko_KR.UTF-8')  
        driver = webdriver.Chrome(executable_path=self.driver_path, options=options)  
        driver.get(url)  
        time.sleep(5)  
        self.login_instagram(driver, url, login_name, login_pw)  
        time.sleep(5)
```

Web driver 경로를 지정해주고 검색할 프로필 id를 인자로 전달.

인스타그램은 로그인 이 되어야 원할한 프로필 검색이 가능

따라서 로그인 할수있는 정보를 입력해줘야하고 크롤링을 위한 driver의 옵션들을 설정.

이후 로그인해주는 함수를 실행해 주는데 이때 self를 앞에 붙여줘야 class안에 있는 함수로 제대로 인식.

추가로 sleep을 해주는 이유는 크롤링할때 계정이 차단당하지 않기위해서 이다.

Instagram Crawling

인스타 로그인 함수

```
def login_instagram(self, driver, target_url, login_name, login_pw):
    insta_url = 'https://www.instagram.com'
    driver.implicitly_wait(10)
    driver.get(insta_url)
    time.sleep(3)

    username_input = driver.find_element(By.CSS_SELECTOR, "input[name='username']")
    password_input = driver.find_element(By.CSS_SELECTOR, "input[name='password']")

    username_input.send_keys(login_name)
    password_input.send_keys(login_pw)

    login_button = driver.find_element(By.XPATH, "//button[@type='submit']")
    login_button.click()

    print('인스타그램 로그인')
    time.sleep(3)
    driver.get(target_url)
    time.sleep(3)
    print('인스타그램 진입성공')
```

인스타 로그인 함수에서는 원래 url로 돌아가기위한 url을 인자로 받고 각 아이디와 비밀번호를 인자로 받는다.

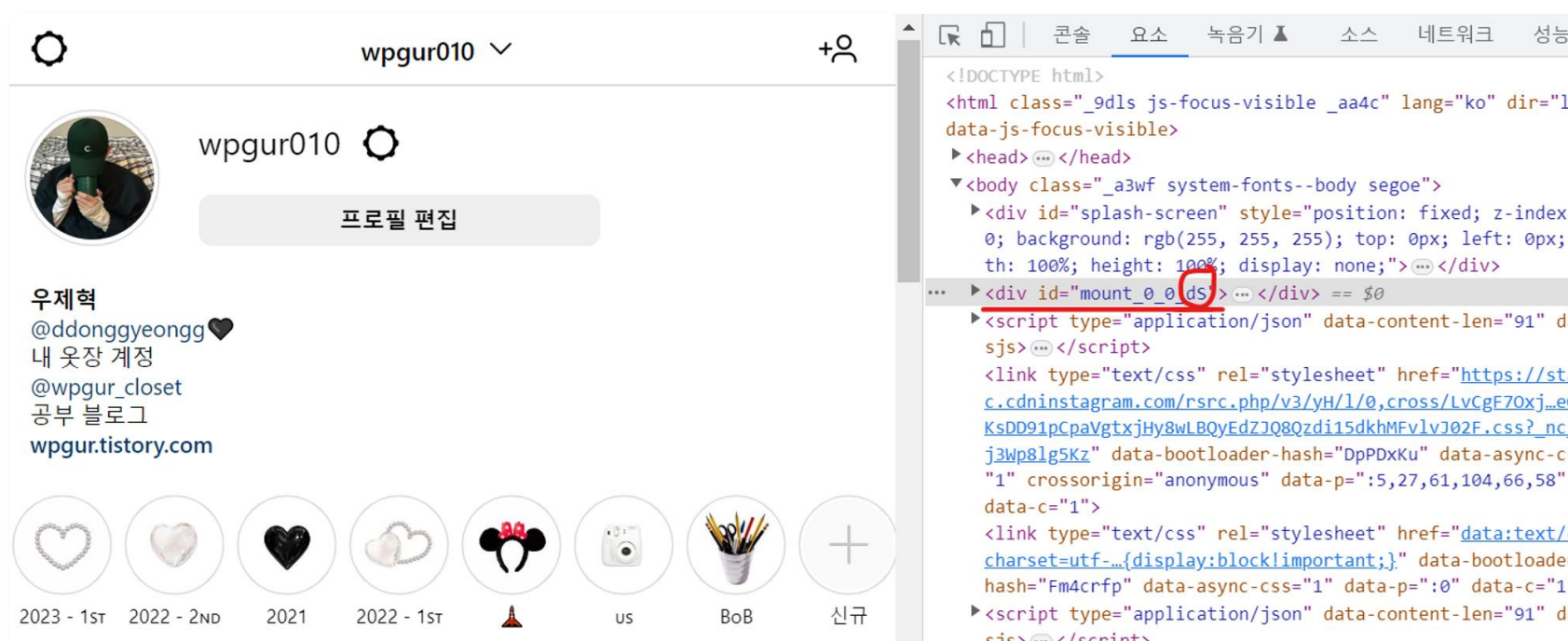
함수가 실행 되면 id 와 pw가 입력 되는 위치를 By.CSS_SELECTOR 로 찾아 준뒤,

값을 .send_keys 로 넣어 주고 로그인 페이지의 submit 타입의 버튼은 로그인 버튼 하나기에 버튼을 찾아 click해줘서 로그인 을 진행한다.

이후 로그인을 하기 위해 잠시 driver의 url을 'https://www.instagram.com' 로 바꿔준걸 다시 크롤링 하기위한 target_url 로 바꿔준다.

Instagram Crawling

크롤링 제한 장치 우회



인스타그램은 최상위 div의 id값을 랜덤으로 바꿔준다..
원래 크롤링은 저 id값을 통해 크롤링을 해오는데 마지막 두글자를 계속 랜덤으로 바꾸며 크롤링을 방지하고 있었다.

따라서 생각해 낸 방법은 정규표현식으로 마지막 글자 두글자를 매번 새롭게 가져오는 것이다.

mount_0_0_ 부분은 같으니 모든 코드를 가지고 있는 driver.page_source 변수를 사용해서 mount_0_0_ 뒤에 두글자를 함께 가져올수 있게 코드를 짜서 bio_text에 저장했다.

```
try:
    bio_text = re.findall(r'mount_0_0_[a-zA-Z0-9_]{2}', str(driver.page_source))
    bio_text = bio_text[0]
except:
    print('bio_text error')
```

bio_text[0] 으로 한 이유는 javascript에도 mount_0_0_ 이 단어가 들어있어 2개가 저장되는데 그중 하나만 가져오려고 했다.

Instagram Crawling

본격적인 크롤링(데이터 수집)

```
try:
    name = driver.find_element(By.TAG_NAME, 'title').get_attribute('textContent')
    name = name.split('•')[0]
except:
    name = 'name'

try:
    bio = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.CSS_SELECTOR, f"#{bio_text} > div > div > div.x9f619.x1n2onr6.x1ja2u2z > div > div > div >
    bio = bio.text
except:
    bio = bio_text

try:
    post = driver.find_element(By.CSS_SELECTOR, 'meta[name="description"]')
    post = post.get_attribute('content').split('-')[0]
    # post = post.text
except:
    post = None

try:
    profile_img = driver.find_element(By.CSS_SELECTOR, 'meta[property="og:image"]')
    profile_img = profile_img.get_attribute('content')
except:
    profile_img = None
```

```
try:
    profile_data = {
        'sns' : 'instagram',
        'name': name,
        'bio': bio,
        'post': post,
        'profile_img': profile_img,
    }
```

최상위 div의 id값도 제대로 가져올수 있게 되어서 가져오고 싶은 부분을 css값으로 찾는다.

이름, 자기소개, 게시물 정보, 프로필 이미지의 url 총 4가지의 정보를 가져온다.

bio(자기소개) 코드 부분을 보면 최상위 div의 id값(bio_text)을 이용하는 걸 볼 수 있다.

크롤링 한 내용들을 profile_data 리스트에 담아 return 해준다.


Instagram Crawling

← → ↺ ⓘ 127.0.0.1:8085/insta_result

앱 해커들의 놀이터, D... NAVER Papago 개인 공부 송터디 캡스톤 디자인 - O... 중부대학교 LMS New chat wpgur 블로그관리 | TISTO... 받은편지함 (133) -... Hacktivity Google 문서 https://los.rubiya.kr... 새 탭

Social Media Info

Instagram



Name: JungSINT(@joongsint)

Bio: JungSINT OSINT 회사입니다 문의: 010-1234-9997 긴급:010-1827-1828 이메일: wpgur@joongsint.co.kr jungsint.64bit.kr taehyeon.01.09, seungceolyeo, god_jhh님 외 3명이 팔로우합니다

Post: 팔로워 10명, 게시물 1개

돌아가기

목 차

03

- 검색엔진 크롤링
- 크롤링 과 보안

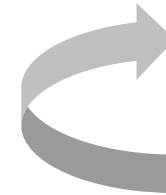
검색엔진 크롤링

```
if __name__ == '__main__':  
    gd = GoogleDetector()  
    gd.search(' ')  
    results = gd.get_results()  
    print(results)
```

1. 객체 선언
2. 검색단어 입력
3. 결과 받아오기

결과 값

```
soup = BeautifulSoup(html)  
['https://www.joongbu.ac.kr/', 'https://namu.wiki/w/%EC%A4%91%EB%B6%80%EB%8C%80%ED%95%99%EA%B5%90', 'http://addon.jinhakapply.com/RatioV1/RatioH/Ratio11310411.html', 'https://www.iacf.co.kr/', 'https://ko.wikipedia.org/wiki/%EC%A4%91%EB%B6%80%EB%8C%80%ED%95%99%EA%B5%90', 'https://www.univ100.kr/review/116', 'https://www.instagram.com/explore/tags/%EC%A4%91%EB%B6%80%EB%8C%80/', 'https://www.facebook.com/joongbuuniversity/?locale=ko_KR']
```



```
from urllib.parse import quote_plus  
from bs4 import BeautifulSoup  
from selenium import webdriver  
import requests  
  
class GoogleDetector():  
    def __init__(self):  
        self.driver = webdriver.Chrome('C:\wpgur\python 스크립트\chromedriver')  
        self.google_baseurl = 'https://www.google.com/search?q='  
  
    def get_google_url(self, query):  
        return f'{self.google_baseurl}{quote_plus(query)}'  
  
    def search(self, query):  
        url = self.get_google_url(query)  
        self.driver.get(url)  
        self.driver.implicitly_wait(10)  
  
    def get_soup_by_source(self):  
        html = self.driver.page_source  
        soup = BeautifulSoup(html)  
        return soup  
  
    def get_results(self):  
        results = []  
        soup = self.get_soup_by_source()  
        for i in soup.select('.yuRUbf'):  
            results.append(i.a.attrs['href'])  
        return results
```

크롤링 과 보안

Directory Listing 취약점

취약한 서버설정으로 발생하는 취약점으로 브라우징하는 모든 파일들을 보여주는 취약점

대부분의 경우 디렉토리 목록을 보여준다 해도 보안과 연결되지는 않지만 파일 저장 및 열람이 가능하다면 이야기가 달라집니다. 백업파일 및 소스코드, 스크립트 파일의 유출로 인한 계정정보 유출과 다양한 정보를 공격자에게 제공하게 됨으로 제2, 제3의 공격에 이용될 수도 있다.

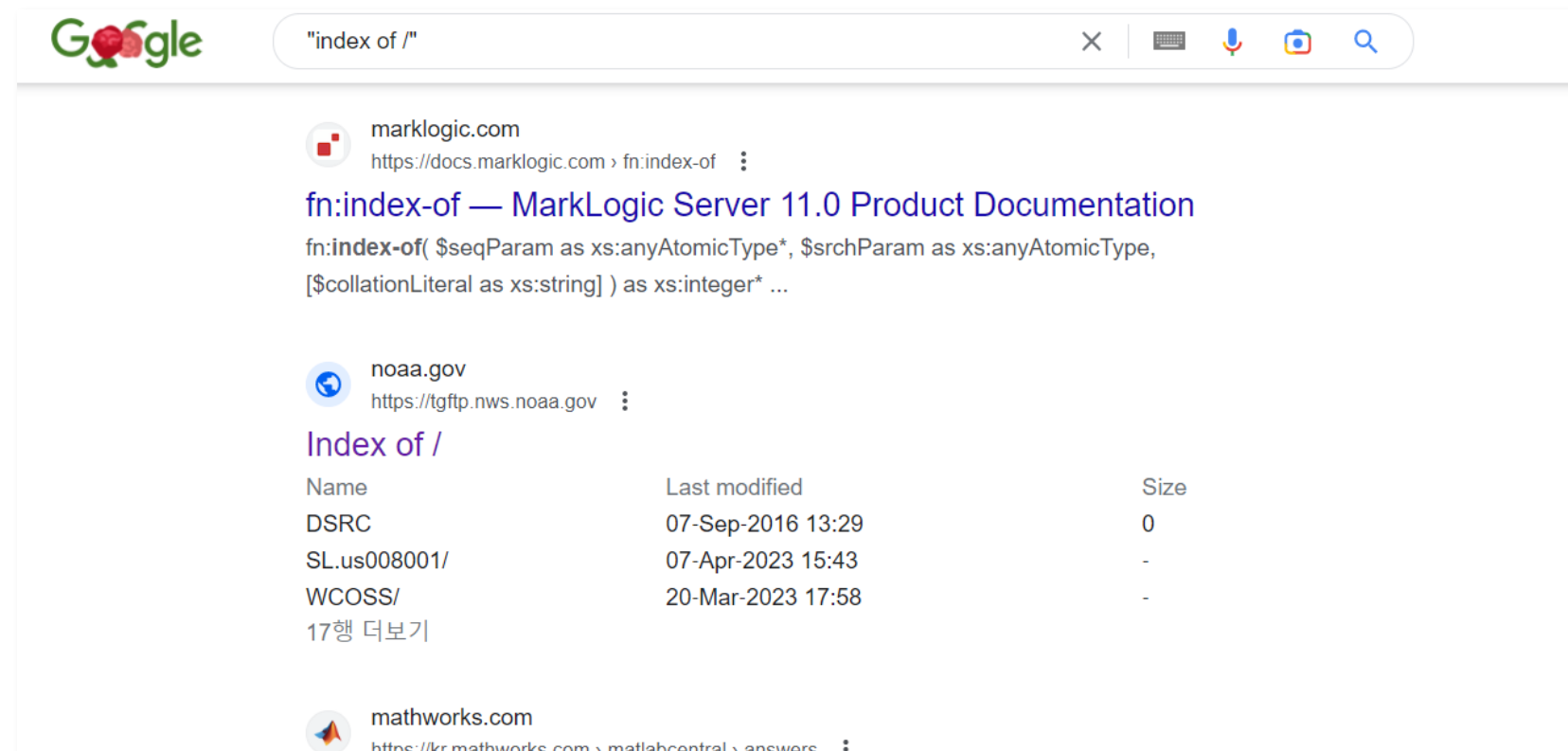
<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 bbs/	11-Jul-2010 02:19	-	
 cart/	26-Oct-2010 15:53	-	
 contents/	31-Oct-2010 16:33	-	
 cyholic.php	25-Oct-2010 23:46	1.8K	
 cyholic/	19-Oct-2010 00:45	-	
 editor/	04-Jul-2010 14:17	-	
 fileUpload/	04-Jul-2010 17:35	-	
 n-ctrlv.php	15-Oct-2010 07:46	914	
 pops/	26-Oct-2010 03:42	-	
 user/	17-Jul-2010 15:18	-	

Apache/2.2.3 (CentOS) Server at cyrang.com Port 80

크롤링 과 보안

크롤링과 보안

구글에 검색되는 디렉토리 리스팅을 수집하여 보안 대안책을 마련할 수 있으며 필터링 작업을 통해 특정 사이트의 디렉토리 리스팅을 발견 할 수 있다.



- index of
- parent directory
- last modified
- directory listing
- listing of
- name, size, description
- modified
- filetype
- contents
- inurl:ftp (intext|inbody):"index of"
- "index of /" "type"
- "index of /" +.htaccess
- "index of /" +passwd
- "index of /" +password.txt
- "index of /" +.passwd
- "index of /" +.password
- "index of /" +.htpasswd
- "index of /" +.htaccess
- "index of /" +.git
- "index of /" +.svn

크롤링 과 보안

```
73  if __name__ == '__main__':
74
75
76      gd = GoogleDetector()
77      results = gd.get_results('"index of /"')
78      print('find_log_file')
79      pprint(results)
```

객체 선언 및 검색어 전달

```
class GoogleDetector():
    #class 선언시 driver 및 기본 url 설정
    def __init__(self):
        chrome_options = webdriver.ChromeOptions()
        chrome_options.add_argument("--headless")
        chrome_options.add_argument("--disable-gpu")
        chrome_options.add_argument("--no-sandbox")
        self.driver = webdriver.Chrome('C:\wpgur\chromedriver',chrome_options=chrome_options)
        self.google_baseurl = 'https://www.google.com/search?q='
```

Class 선언 및 옵션 설정

--headless
브라우저 창을 표시하지 않고 백그라운드에서 실행하도록 지시

--disable-gpu
브라우저에서 GPU 가속을 사용하지 않도록 설정.

--no-sandbox
브라우저의 보안 기능 중 하나인 샌드박싱(Sandboxing)을 비활성화

크롤링 과 보안

```
#results 리스트에 찾은 url 넣기
def get_results(self,query):
    results = []
    url = self.get_google_url(query)

    #다음 페이지 이동
    for i in range(2):
        time.sleep(5) #차단 될까봐 5초나 쉰음

        #다음 페이지 url 설정
        if i != 0:
            url = self.next_page_set(url)

    soup = self.search(url)
    for i in soup.select('.yuRUbf'):
        results.append(i.a.attrs['href'])
        #검색결과중 index가 들어가있는것만 추가
        # if 'index of ' in str(i.a.attrs['href']):
        #     results.append(i.a.attrs['href'])

    self.driver.close()
    return results
```

```
30     #url 설정
31     def get_google_url(self, query):
32         return f'{self.google_baseurl}{quote_plus(query)}'
33
```

`self.google_baseurl = 'https://www.google.com/search?q='`

```
def next_page_set(self,url):
    if 'start' not in url:
        url += '&start=10'
        return url
    else:
        print("past"+url)
        url = url[:-2]+str(int(url[-2:])+10)
        print("new:"+url)
        return url
```


크롤링 과 보안

크롤링과 보안

```
soup = self.search(url)
for i in soup.select('.yuRUbf'):
    results.append(i.a.attrs['href'])
    #검색결과중 index가 들어가있는것만 추가
    # if 'index of ' in str(i.a.attrs['href']):
    #     results.append(i.a.attrs['href'])

self.driver.close()
return results
```

설정된 url을 통해 search 함수를 실행 하게 되는데



```
34 #soup 설정
35 def get_soup_by_source(self):
36     html = self.driver.page_source
37     soup = BeautifulSoup(html)
38     return soup
39
40 #search
41 def search(self, url):
42     self.driver.get(url)
43     self.driver.implicitly_wait(10)
44     return self.get_soup_by_source()
45
```

Url의 값으로 web driver을 통해 전체적인 html을 가져오고 이를 이용해서 beautiful soup으로 필요한 부분을 찾아 result에 넣어준다.

크롤링 과 보안

크롤링과 보안

```
53 soup = self.search(url)
54 for i in soup.select('.yuRUbf'):
55     results.append(i.a.attrs['href'])
56     #검색결과 중 index가 들어가있는것만 추가
57     # if 'index of ' in str(i.a.attrs['href']):
58     #     results.append(i.a.attrs['href'])
```

저기 주석 되어있는 부분을 통해 검색 결과를 더 세분화 하여 필터링 할 수도 있다.

크롤링 과 보안

검색된 디렉토리 리스팅

```
find_log_file
['https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/indexOf',
 'https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/String/indexOf',
 'http://hkdb.co.kr/down/',
 'http://hanarental.ipdisk.co.kr:8080/publist/HDD1/%EB%93%9C%EB%9D%BC%EC%9D%B4%EB%B2%84%EC%9E%90%EB%A3%8C%EC%8B%A4/(%EA%B3%84%EC%95%B
D%EC%84%9C%EC%99%B8%EC%9E%90%EB%A3%8C)/',
 'https://www.w3schools.com/jsref/jsref_indexof_array.asp',
 'https://docs.marklogic.com/fn:index-of',
 'https://tgftp.nws.noaa.gov/',
 'https://www.inveniacorp.com/chi/',
 'https://kr.mathworks.com/matlabcentral/answers/194618-how-to-find-the-index-of-the-closest-value-to-some-number-in-1d-array',
 'https://kr.mathworks.com/matlabcentral/answers/381398-how-can-i-find-the-index-of-a-the-characters-within-a-string',
 'https://download.blender.org/release/',
 'https://archive.ics.uci.edu/ml/machine-learning-databases/',
 'https://archive.apache.org/dist/tomcat/',
 'https://archive.apache.org/dist/hadoop/core/',
 'https://dictionary.cambridge.org/ko/%EC%82%AC%EC%A0%84/%EC%98%81%EC%96%B4/index-of-coincident-indicators',
 'https://dictionary.cambridge.org/ko/%EB%B0%9C%EC%9D%8C/%EC%98%81%EC%96%B4/index-of-refraction',
 'https://kostat.go.kr/menu.es?mid=a20202020000',
 'https://recruit.chamc.co.kr/images/download/RG20201970-001/',
 'https://www.heritage.org/index/']
```


질문 및 피드백