# 웹

## 크롤링

# 목차

# 1. 크롤링(crawling) 이란?

•Web상에 존재하는 Contents를 수집하는 작업 (프로그래밍으로 자동화 가능)

- HTML 페이지를 **가져와서**, HTML/CSS등을 **파싱**하고, 필요한 데이터만 추출하는 기법
- **Open API(Rest API)**를 제공하는 서비스에 Open API를 호출해서, 받은 데이터 중 필요한 데이터만 추출하는 기법
- **Selenium**등 브라우저를 프로그래밍으로 조작해서, 필요한 데이터만 추출하는 기법

# 2. urllib

- URL을 통해 여러 작업을 수행하는 모듈들의 패키지

urllib 패키지에는 4개의 모듈이 있습니다.

1. urllib.request : URL 요청을 위한 클래스나 함수들이 정의

2. urllib.parse : URL의 구문을 분석하기 위한 클래스나 함수들이 정의

3. urllib.error : urllib.request 모듈에 의해 발생된 예외 처리를 위한 예외 클래스들이 정의

4. urllib.robotparse : 웹사이트의 robots.txt 파일에 대한 분석을 위한 단일 클래스가 정의

# 2. [urllib.request 모듈]

urllib.request.urlopen(url, data=None, imeout, cafile=None, capath=None, cadefault=False, context=None)

: URL을 열기 위한 함수입니다.

**url** 인자에는 문자열로 된 url이나 Request 객체를 넣어줍니다.

(Request 객체는 urllib.request.Request 클래스로 인스턴트를 받아옵니다. )

**data** 인자에는 POST 방식으로 요청 시의 데이터를 넣어줍니다.

**timeout** 인자는 선택적으로 사용합니다. 연결 시도에 대한 시간 초과 '초(seconds)'를 지정합니다.

# 2. [urllib.request 모듈]

urlopen() 함수는 몇 개의 함수를 지원합니다.

1. geturl() : 받아온 리소스의 URL를 반환 받습니다.

2. info() : 패킷의 메타 데이터(헤더 등)를 반환 받습니다.

3. getcode() : 응답 패킷의 HTTP 상태 코드를 반환 받습니다.

4. read() : 받아온 데이터를 바이트형으로 반환 받습니다.

5. readline() : 받아온 데이터를 바이트형으로 한 줄씩 반환 받습니다.

6. close() : 연결된 요청을 닫습니다.

# 2. [urllib.request 모듈]

```
import urllib.request

response=urllib.request.urlopen("http://www.naver.com")

print(response.geturl())

print(response.getcode())

#헤더의 정보
print(response.info())

#print( response.read())
print( response.read().decode())
```

# 2. [urllib.request.Request 모듈]

urllib.request.Request(url, data=None, headers = {}, origin_req_host=None, unverifiable=False, method=None)
: urllib.request.urlopen()함수의 인자로 넘겨주기 위해, URL 요청을 인스턴트화하기 위해 사용.

**url** 인자에는 유효한 URL의 문자열을 넣습니다.

**data** 인자에는 요청에 대한 추가 데이터의 객체를 지정하거나 None을 지정합니다.

**headers** 인자에는 요청에 필요한 헤더를 넣어줍니다. 인자는 Dictionary형.

**origin_req_host** 인자에는 원본 트랙잭션의 요청 호스트를 지정합니다. 기본 값은 URL을 요청한 호스트입니다.

**unverifiable** 인자에는 요청에 대해 검증할 수 없는지의 여부(True / False)를 지정합니다. 기본 값은 False 입니다.

**method** 인자에는 HTTP 요청 메소드가 들어갑니다. 기본 값은 GET이고 HEAD, POST 등을 지정할 수 있습니다.

Request 클래스를 통해 해당 요청에 대한 인스턴트를 받아와 urlopen의 인자로 사용합니다

# 2. [urllib.request.Request 모듈]

1. Request.full_url : 전달된 원본 URL을 담고 있는 변수입니다.

2. Request.type : URL Scheme을 담고 있는 변수입니다. (ex. http, https, ftp, ...)

3. Request.host : 요청된 URL의 호스트 부분을 담고 있는 변수입니다.
   콜론으로 구분된 포트 넘버를 포함할 수 있습니다.

4. Request.origin_req_host : 포트를 제외한 원래의 호스트 부분을 담고 있는 변수입니다.

5. Request.method : 요청에 사용되는 HTTP 메소드를 설정하는 변수입니다.

6. Request.get_method() : 요청에 사용될 HTTP 메소드를 반환 받습니다.

# 2. [urllib.request.Request 모듈]

```
import urllib.request
request=urllib.request.Request("http://www.naver.com")
response=urllib.request.urlopen(request)
print(response.read().decode())
print(request.get_method())
request.method='POST'
print(request.get_method())
print(request.type)
print(request.host)
```

# 2. [urllib.parse 모듈]

[Function] **urllib.parse.urlparse**(*urlstring, scheme='', allow_fragments=True*)
: URL의 구문을 분석하여 6개의 구성 요소를 6 tuple로 반환합니다.

**urlstring** 인자에는 분석할 URL을 문자열로 넣습니다.

**scheme** 인자에는 URL의 Scheme을 지정합니다. URL에 Scheme이 포함되지 않은 경우에 지정합니다.

**allow_fragments** 인자에는 fragment 식별자의 정의 여부를 지정(True/False) 합니다. False로 지정한 경우 fragment는 빈 값으로 설정됩니다.

# 2. [urllib.parse 모듈]

[Function] urllib.parse.urlunparse(parts)

: urllib.parse.urlparse()로부터 반환된 6 tuple로부터 URL을 생성합니다.

# 2. [urllib.parse 모듈]

```
import urllib.parse

parse = urllib.parse.urlparse('192.168.147.128/bWAPP/portal.php?id=bee&pw=bug#example',
scheme='http')
print(parse)

parse = urllib.parse.urlparse('192.168.147.128/bWAPP/portal.php?id=bee&pw=bug#example',
scheme='http', allow_fragments=False)
print(parse)

print(parse.path)

url=urllib.parse.urlunparse(parse)
print(url)
```

# 2. [urllib.parse 모듈]

[Function] urllib.parse.urlencode(query, doseq=False, safe=' ', encoding=None, errors=None, quote_via=quote_plus)
: 인자로 받은 매핑 객체 또는 2개의 요소로 이루어진 tuple을 POST 방식으로 전송하기 위한 데이터로 변환해줍니다.
(query 구성 : { key : value })

**query** 인자에는 변환할 데이터 문자열을 넣어줍니다.

**doseq** 인자에는 하나의 key에 여러 개의 value가 존재할 때, 이를 개별의 쌍으로 분리할지 결정합니다.(True/False)
기본 값은 False로 지정되어 있으며, 만약 doseq=False이고 하나의 key에 여러 개의 value가 존재할 경우 이 value를 문자열로 인식합니다.

**safe** 인자에는 변환하지 않을 값을 지정합니다. urllib.parse.quote()의 인자 safe와 같은 의미입니다.

**errors** 인자 역시 urllib.parse.quote()의 인자 errors와 같은 의미입니다. 인코딩 에러를 처리할 방식을 지정합니다.

**quote_via** 인자에는 특수문자를 문자열로 변환하는 함수를 지정해줍니다. (기본값 = quote_plus)
urllib.parse.quote(), urllib.parse.quote_plus(), urllib.parse.quote_from_bytes()를 지정할 수 있습니다

# 2. [urllib.parse 모듈]

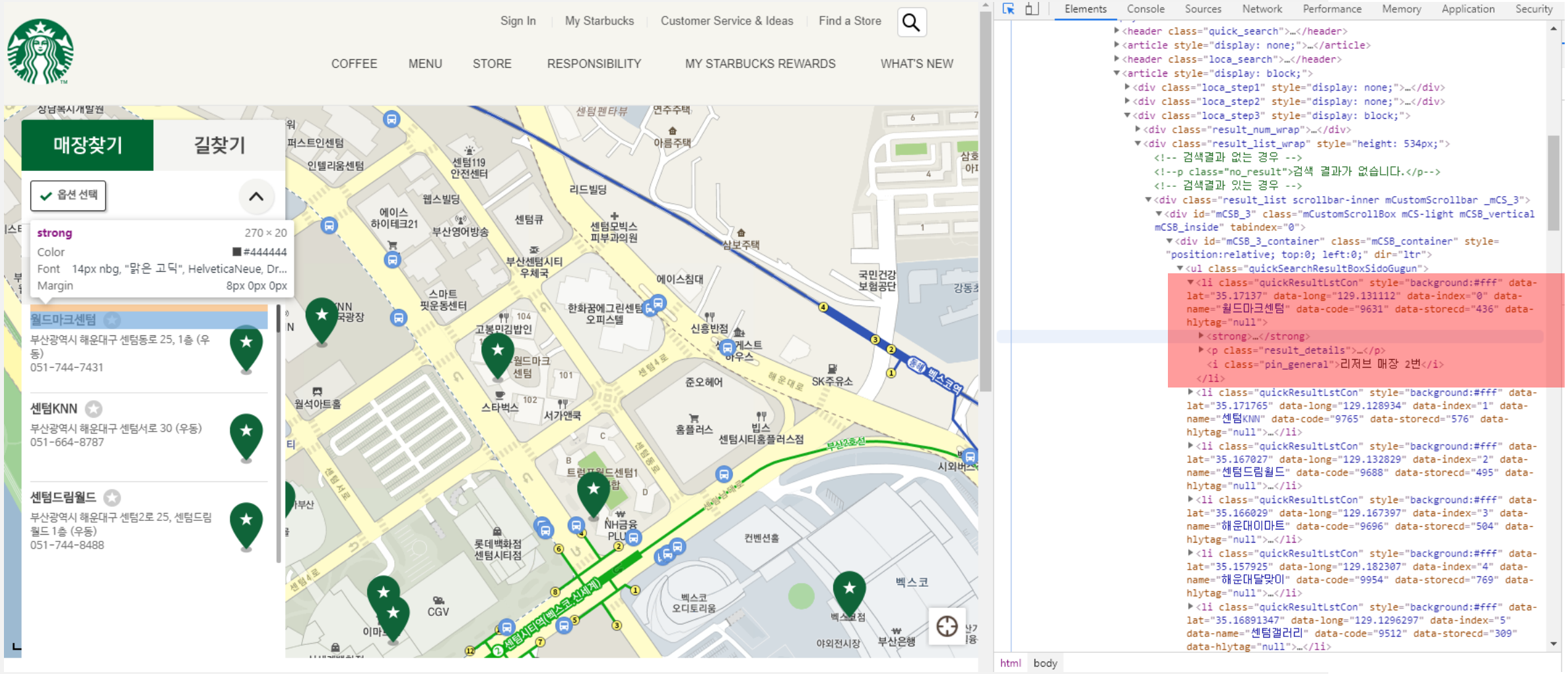
```
import urllib.parse

parameter = {'key1' : 'value1', 'key2' : 'value2 test'}

data1 = urllib.parse.urlencode(parameter, quote_via=urllib.parse.quote)
data2 = urllib.parse.urlencode(parameter)

print(data1)
print(data2)


parameter = {'key1' : 'value1', 'key2' : ['value2', 'value3']}

data1 = urllib.parse.urlencode(parameter)
data2 = urllib.parse.urlencode(parameter, doseq=True)

print(data1)
print(data2)
```

# 3. BeautifulSoup

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

# 4. 셀레니움(selenium)

https://www.istarbucks.co.kr/store/store_map.do?disp=quick



# 4. 셀레니움(selenium)

```
from bs4 import BeautifulSoup
import requests

starbugs = requests.get('https://www.istarbucks.co.kr/store/store_map.do')
st_bs=BeautifulSoup(starbugs.text,'lxml')
print(st_bs.select('li.quickResultLstCon'))
```

select('li.quickResultLstCon')으로 선택해서 정보를 가져올 수 없다.
이유는 우리가 받은 정보는 지역을 검색하기 전 정보이기 때문이다.
검색을 하고 버튼을 누르면 브라우저는 필요한 정보만 가지고 와서
코드를 부분적으로 변경하기 때문에 requests.get(url)나 urllib.request.urlopen(url)로
불러온 정보는 변경 후의 정보는 가지고 있지 않다.

셀레니움은 원래 브라우저를 조정해서 웹을 테스트하는 용도 사용된다.
브라우저의 화면을 조정하여 브라우저 화면에 나타난 버튼을 눌러보거나 아이디, 암호 등을
입력하고 로그인을 하거나 하는 등의 행위를 할 수 있다.

# 4. 셀레니움(selenium)

1. 셀레니움 설치

   ```
   pip install selenium
   ```

2. 웹드라이브 다운로드(크롬드라이브를 사용할 경우,본인 크롬 드라이브와 같은 버전으로 다운로드)

   https://sites.google.com/a/chromium.org/chromedriver/downloads Chrome

   https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/ Edge

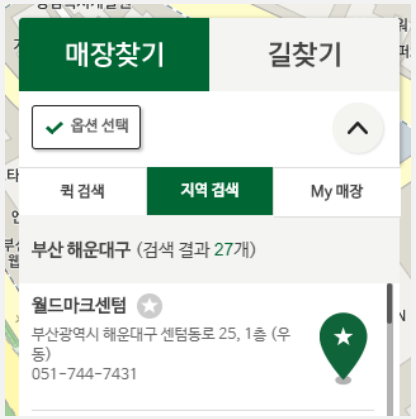
   https://github.com/mozilla/geckodriver/releases Safari

3. 셀레니움을 통한 브라우저 제어를 시도

   ```
   driver = webdriver.Chrome('./crawling/data/chromedriver')
   driver.get('https://www.istarbucks.co.kr/store/store_map.do')
   ```

# 4. 셀레니움(selenium)

4. 브라우저에서 버튼을 찾아 누름



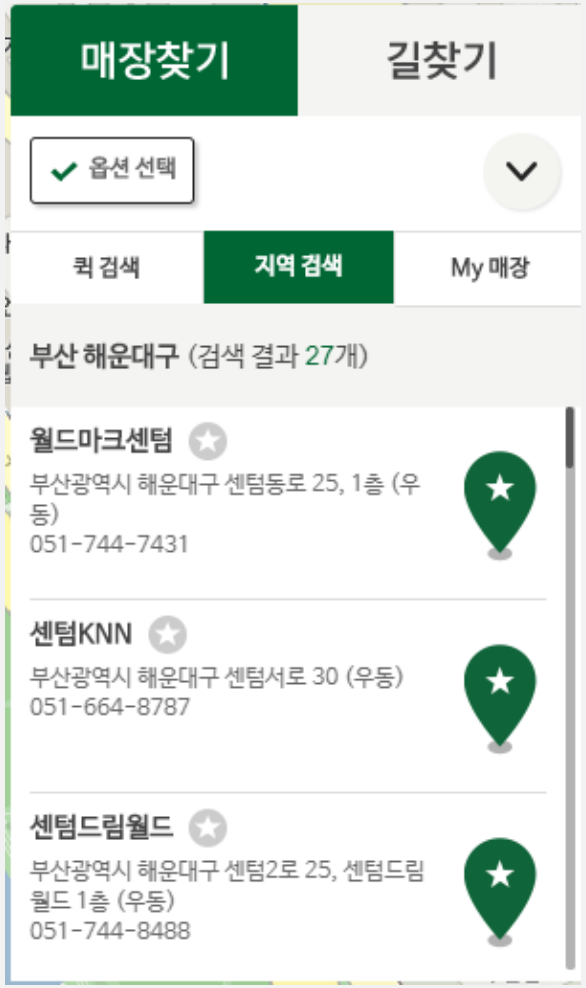

```
<header class="quick_search">…</header>
<article style="display: none;">…</article>
<header class="loca_search"> == $0
  <h3 class="on">
    <a href="javascript:void(0);">지역 검색</a>
  </h3>
</header>
<article style="display: block;">…</article>
<header class="my_store">…</header>
<article style="display: none;">…</article>
```

```
loca = driver.find_element_by_class_name('loca_search')
loca.click()

sido = driver.find_element_by_class_name('sido_arae_box')
li=sido.find_elements_by_tag_name('li')
li[5].click()
```


```
<div class="loca_step1" style="display: block;">
  <div class="loca_step1_ttl">STEP 1 : 시/도를 선택해 주세요.</div>
  <div class="loca_step1_cont">
    <ul class="sido_arae_box">
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>
        <a href="javascript:void(0);" class="set_sido_cd_btn" data-sidocd="05">부산</a>
      </li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
      <li>…</li>
    </ul>
  </div>
</div>
```

# 4. 셀레니움(selenium)

5. 동적으로 변한 브라우저에서 소스 정보를 가져옴

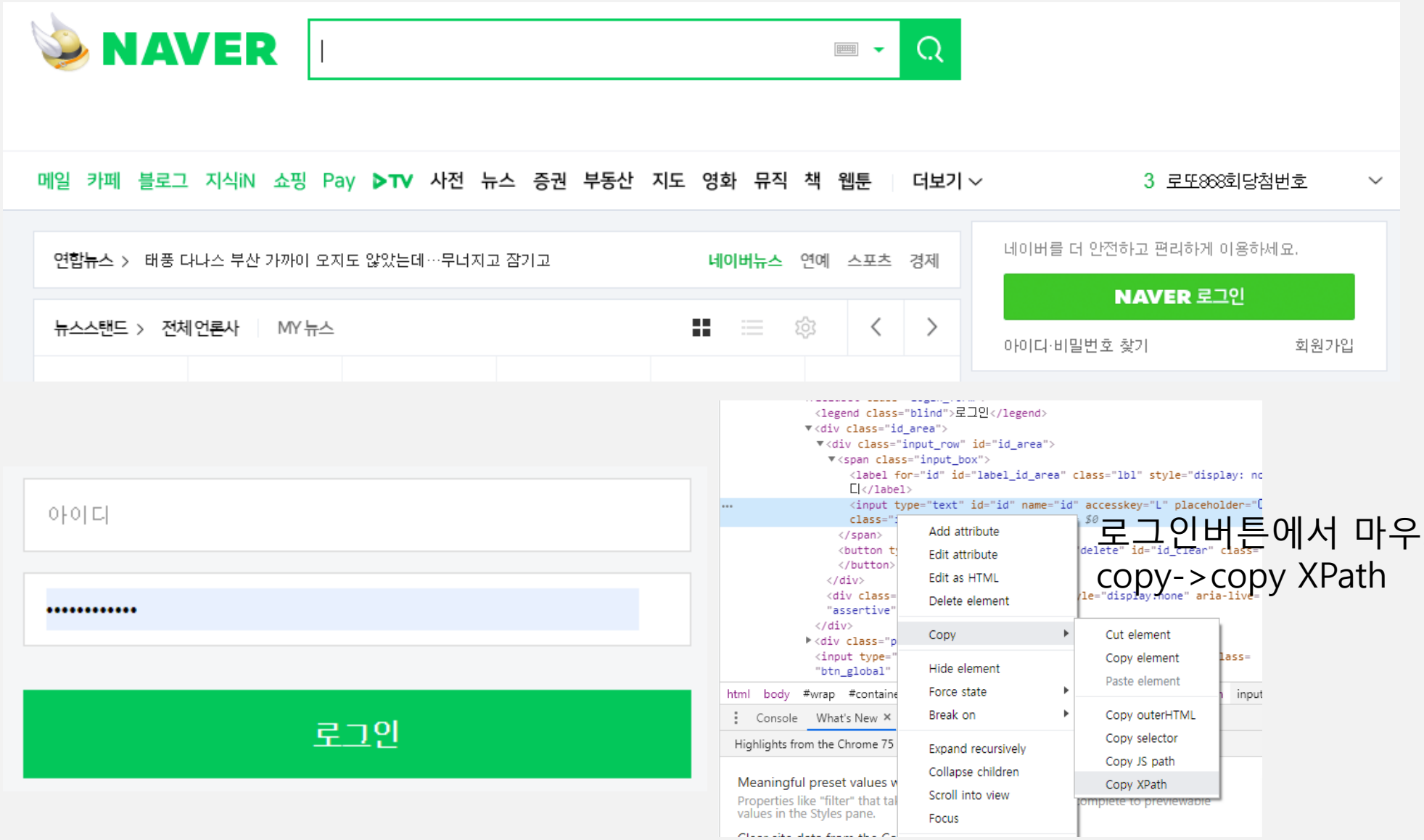



```
▼<ul class="quickSearchResultBoxSidoGugun">
  ▼<li class="quickResultLstCon" style="background:#fff" data-lat="35.17137" data-long="129.131112" data-index="0" data-name="월드마크센텀" data-code="9631" data-storecd="436" data-hlytag="null">
    ▶<strong>…</strong>
    ▶<p class="result_details">…</p>
     <i class="pin_general">리저브 매장 2번</i>
    </li>
  ▶<li class="quickResultLstCon" style="background:#fff" data-lat="35.171765" data-long="129.128934" data-index="1" data-name="센텀KNN" data-code="9765" data-storecd="576" data-hlytag="null">…</li>
```

```
source=driver.page_source

bs=BeautifulSoup(source,'lxml')
entire=bs.find('ul',class_='quickSearchResultBoxSidoGugun')
li_list=entire.find_all('li')

for li in li_list:
    print(li.find('p').text)
```

```
from bs4 import BeautifulSoup
from selenium import webdriver
import requests,time

driver = webdriver.Chrome('./crawling/data/chromedriver')
driver.get('https://www.istarbucks.co.kr/store/store_map.do')
time.sleep(10)

loca = driver.find_element_by_class_name('loca_search')
loca.click()
time.sleep(10)

sido = driver.find_element_by_class_name('sido_arae_box')
li=sido.find_elements_by_tag_name('li')
li[5].click()
time.sleep(10)

sido = driver.find_element_by_class_name('gugun_arae_box')
li=sido.find_elements_by_tag_name('li')
li[16].click()
time.sleep(10)

source=driver.page_source

bs=BeautifulSoup(source,'lxml')
entire=bs.find('ul',class_='quickSearchResultBoxSidoGugun')
li_list=entire.find_all('li')

for li in li_list:
print(li.find('p').text)
```

# 4. 셀레니움(selenium)

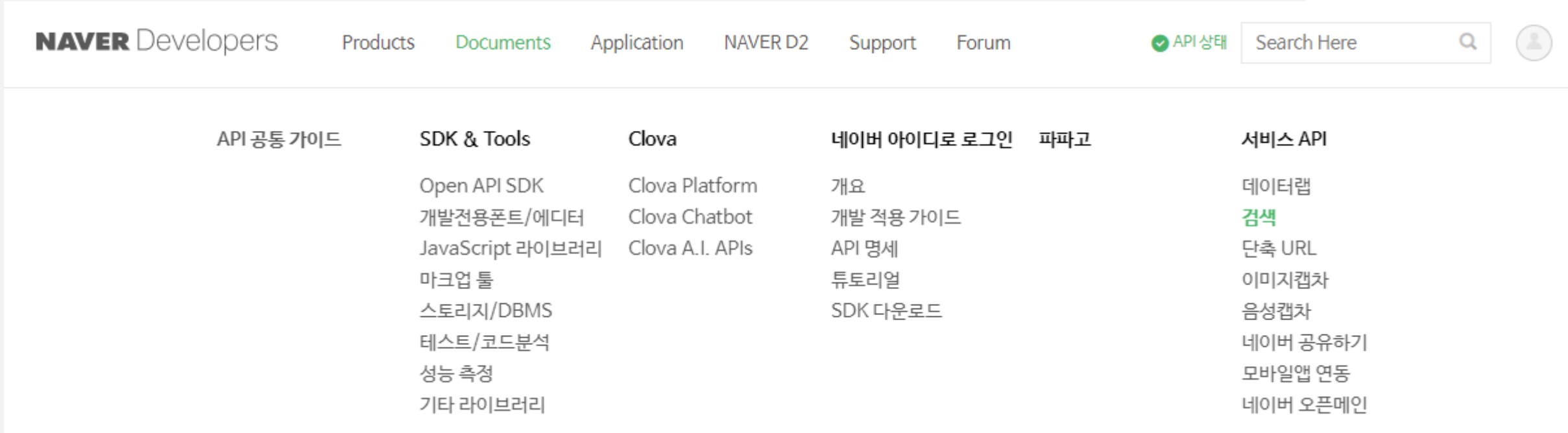



로그인버튼에서 마우스 오른쪽 버튼 클릭
copy->copy XPath

```
from selenium import webdriver
from bs4 import BeautifulSoup
import time

driver = webdriver.Chrome('./crawling/data/chromedriver')
driver.get('https://nid.naver.com/nidlogin.login')
driver.implicitly_wait(3)

id='네이버아이디'
pw= ' 네이버비밀번호'

#driver.find_element_by_name('id').send_keys(id)
#driver.find_element_by_name('pw').send_keys(pw)

driver.execute_script("document.getElementsByName('id')[0].value=₩'"+id+"₩'")
driver.execute_script("document.getElementsByName('pw')[0].value=₩'"+pw+"₩'")
time.sleep(0.5)

#element창 로그인버튼 소스 선택된 상태에서 마우스 오른쪽 버튼 클릭 copy->copy XPath
driver.find_element_by_xpath('//*[@id="frmNIDLogin"]/fieldset/input').click()
time.sleep(1)
driver.find_element_by_xpath('//*[@id="frmNIDLogin"]/fieldset/span[1]/a').click()
time.sleep(1)
driver.find_element_by_xpath('//*[@id="login_maintain"]/span[1]').click()
time.sleep(1)

# Naver 페이 들어가기
driver.get('https://order.pay.naver.com/home')
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')
notices = soup.select('div.p_inr > div.p_info > a > span')
point = soup.select_one('.my_npoint strong')
print(point.string)

time.sleep(15)
driver.close()
```

# 네이버 API

https://developers.naver.com/main/ Documents - 서비스 api - 검색



# 오픈 API 신청

Documents > 서비스 API > 검색 > 블로그

API 공통 가이드
SDK & Tools
Clova
네이버 아이디로 로그인
지도
파파고
**서비스 API**

데이터랩
**검색**
- 블로그
- 뉴스

## 검색 > 블로그

네이버 블로그 검색 결과를 출력해주는 REST API입니다. 비로그인 오픈 API이므로 GET으로 호출할 때 HTTP Header에 애플리케이션 등록 시 발급받은 Client ID와 Client Secret 값을 같이 전송해 주시면 활용 가능합니다.

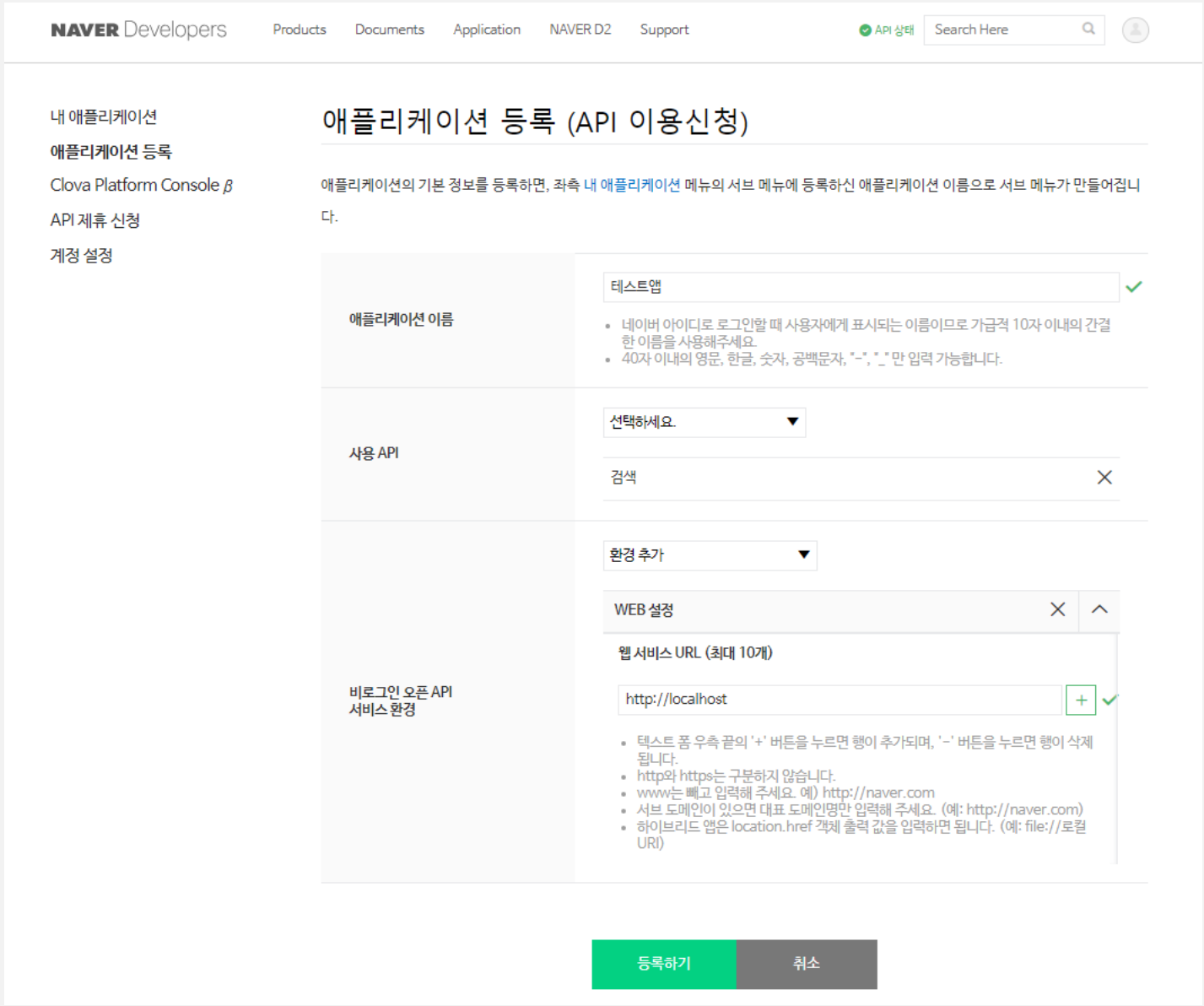
오픈 API 이용 신청 〉

### 0.API 호출 예제

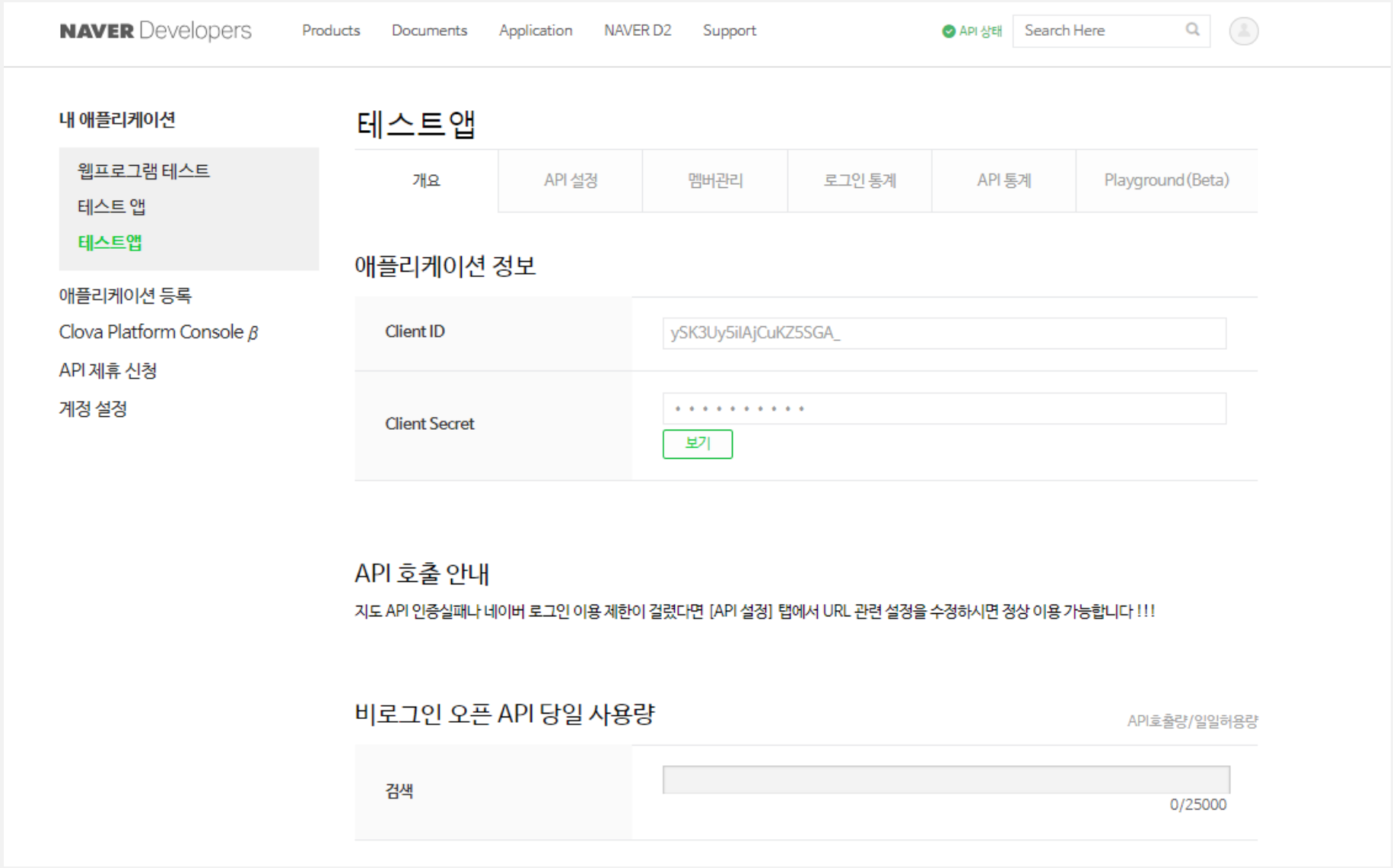
예제 실행 전에 아래 1.준비사항 항목들을 꼭 체크하시길 바랍니다.

| Java | PHP | Node.js | Python | C# |
|---|---|---|---|---|

# 오픈 API 신청


NAVER Developers
Products
Documents
Application
NAVER D2
Support
API 상태
Search Here
내 애플리케이션
애플리케이션 등록
Clova Platform Console β
API 제휴 신청
계정 설정
애플리케이션 등록 (API 이용신청)
애플리케이션의 기본 정보를 등록하면, 좌측 내 애플리케이션 메뉴의 서브 메뉴에 등록하신 애플리케이션 이름으로 서브 메뉴가 만들어집니다.
애플리케이션 이름
테스트앱
네이버 아이디로 로그인할 때 사용자에게 표시되는 이름이므로 가급적 10자 이내의 간결한 이름을 사용해주세요.
40자 이내의 영문, 한글, 숫자, 공백문자, "-", "_"만 입력 가능합니다.
사용 API
선택하세요.
검색
비로그인 오픈 API 서비스 환경
환경 추가
WEB 설정
웹 서비스 URL (최대 10개)
http://localhost
텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '−' 버튼을 누르면 행이 삭제됩니다.
http와 https는 구분하지 않습니다.
www는 빼고 입력해 주세요. 예) http://naver.com
서브 도메인이 있으면 대표 도메인명만 입력해 주세요. (예: http://naver.com)
하이브리드 앱은 location.href 객체 출력 값을 입력하면 됩니다. (예: file://로컬URI)
등록하기
취소

# 오픈 API 신청


NAVER Developers
Products
Documents
Application
NAVER D2
Support
API 상태
Search Here
내 애플리케이션
웹프로그램 테스트
테스트 앱
테스트앱
애플리케이션 등록
Clova Platform Console β
API 제휴 신청
계정 설정
테스트앱
개요
API 설정
멤버관리
로그인 통계
API 통계
Playground (Beta)
애플리케이션 정보
Client ID
ySK3Uy5ilAjCuKZ5SGA_
Client Secret
보기
API 호출 안내
지도 API 인증실패나 네이버 로그인 이용 제한이 걸렸다면 [API 설정] 탭에서 URL 관련 설정을 수정하시면 정상 이용 가능합니다 !!!
비로그인 오픈 API 당일 사용량
API호출량/일일허용량
검색
0/25000

# 0.API 호출 예제

예제 실행 전에 아래 1.준비사항 항목들을 꼭 체크하시길 바랍니다.

| Java | PHP | Node.js | Python | C# |
|---|---|---|---|---|

```
# 네이버 검색 API예제는 블로그를 비롯 전문자료까지 호출방법이 동일하므로 blog검색만 대표로 예제를 올렸습니다.
# 네이버 검색 Open API 예제 - 블로그 검색
import os
import sys
import urllib.request
client_id = "YOUR_CLIENT_ID"
client_secret = "YOUR_CLIENT_SECRET"
encText = urllib.parse.quote("검색할 단어")
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
request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request)
rescode = response.getcode()
if(rescode==200):
    response_body = response.read()
    print(response_body.decode('utf-8'))
else:
    print("Error Code:" + rescode)
```

# 네이버 검색 API예제는 블로그를 비롯 전문자료까지 호출 방법이 동일하므로 blog검색만 대표로 예제를 올렸습니다.
# 네이버 검색 Open API 예제 - 블로그 검색

```
import os, sys, urllib.request

# naver개발자 센터에서 발급받은 client_id,client_secret
client_id = "client_id"
client_secret = "client_secret“

encText = urllib.parse.quote("봄나물")
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

request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request)
rescode = response.getcode()
if(rescode==200):
    response_body = response.read()
    print(response_body.decode('utf-8'))
else:
    print("Error Code:" + rescode)
```

공공데이터 포털 https://www.data.go.kr/

영화진흥위원회 http://www.kobis.or.kr/

기상자료개방포털 https://data.kma.go.kr

카카오지도api http://apis.map.kakao.com/

네이버api https://developers.naver.com/docs/common/openapiguide/

농산물유통정보 https://www.kamis.or.kr/customer/reference/openapi_list.do

국가대중교통정보센터 https://tago.go.kr/v5/use/openapi.jsp