



박소희 <parksohee63@gmail.com>

[프로젝트] 2. 소개팅 어디서 하지?

1 개의 메일

박소희 <parksohee63@gmail.com>

2020년 8월 13일 오전 2:54

받는사람: 박소희 <parksohee63@gmail.com>

서울시 맛집 검색 서비스입니다. 이번 시간을 통해 다음 내용을 배울 수 있습니다.

1. 네이버 검색 API 를 이용하여 서울 시내 맛집 정보 크롤링 & DB 저장
2. 네이버 지도 API 를 이용하여 지도 내 맛집 마커 표시
3. 스크롤 DIV를 만들고 맛집 이름을 클릭하면 지도 위에 정보 띄우기
4. OG 태그 사용하기

(코로나맵도 비슷하게 만들었겠죠?)

[\[소개팅어디서하지\] 강의영상 모음\(클릭\)](#)

1. [소개팅어디서하지] - 프로젝트 세팅

pycharm의 new project를 클릭해서, **sparta** → **project** → **matjip** 폴더를 만들고 시작하기!


- 1. 완성작보기
 - [링크](#)를 통해 살펴보기
- 2. 패키지 설치하기 & 폴더 세팅하기

설치할 패키지: flask pymongo requests

static, templates 폴더 + [app.py](#) 이제 너무 익숙하죠? 거기에, crawl 폴더를 하나만 더 만들어주세요!


빠대 코드([app.py](#), [index.html](#))들은, 같이 만들어가면서 붙일게요!

요렇게!

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4a07da6c-3277-48bc-8701-7a2e788a0a1c/Untitled.png>


2. [소개팅어디서하지] - 네이버 검색 API 신청하기

- 3. 네이버 검색 API 이용 신청하기
 - 맛집 데이터는 [네이버 검색 API](#)(클릭)를 활용하겠습니다.
- 1. '오픈 API 이용 신청'을 클릭해주세요
- 2. 환경추가 > WEB 설정 클릭 후 다음과 같이 내용을 입력해주세요. 그리고 등록하기 버튼을 클릭합니다.

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/54ca6e1d-d6dc-4a05-9a8f-b39fb3f60819/Untitled.png>

3. 아래와 같이 Client ID, Client Secret을 확인할 수 있습니다.

- Client Secret의 경우 '보기' 버튼을 누르면 확인하실 수 있습니다. API 인증 시 활용하는데 말 그대로 시크릿 키이기 때문에 다른 사람이 알면 안되니 조심스레 보관해주세요!
- 타인에게 유출되었을 경우 '보기'를 누른 다음에 나타나는 '재발급' 버튼을 통해 새롭게 발급받을 수 있습니다.

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/7ee3c6d9-5f61-4932-a965-1f4a68c8caab/Untitled.png>

3. [소개팅어디서하지] - 맛집 정보 크롤링하기

- 4. 맛집 정보 크롤링하기

◦ (1) crawl 폴더 > matjip_to_db.py 파일 시작 코드

```
import requests
import pprint
from pymongo import MongoClient
```


```
# 맛집 데이터는 seoul_matjip 이라는 데이터베이스에 저장하겠습니다.
client = MongoClient('localhost', 27017)
db = client.seoul_matjip
```

```
# 서울시 구마다 맛집을 검색해보겠습니다.
seoul_gu = ["종로구", "중구", "용산구", "성동구", "광진구", "동대문구", "중랑구", "성북구", "강북구", "도봉구", "노원구", "은평구", "서대문구", "마포구", "양천구", "강서구", "구로구", "금천구", "영등포구", "동작구", "관악구", "서초구", "강남구", "송파구", "강동구"]
```

```
# 네이버 검색 API 신청을 통해 발급받은 아이디와 시크릿 키를 입력합니다.
client_id = "나의 클라이언트 아이디"
client_secret = "나의 시크릿 키"
```


◦ (2) 구별 맛집 검색해보기

- 네이버 검색 가이드 > 지역 (링크) 을 살펴봅니다
- 데이터를 받는 양식으로 JSON을 선호하니 아래 주소를 사용하겠습니다.

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/43df7c13-fdd6-4f05-9fcc-9bd02ba9c796/Untitled.png>


<<https://openapi.naver.com/v1/search/local.json>>

- 필수 요청 변수는 query, 나머지는 선택 변수입니다. 구별 맛집 10개를 가져오고 유사도 순으로 가져올 예정이니 다음과 같이 주소를 확정하겠습니다. UTF-8 방식으로 인코딩해야 한다는군요!

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/7a7637c6-18c5-458a-8c60-ffc5938f2847/Untitled.png>

```
# GET 방식이므로 "?" 와 "&"를 활용합니다!
# '강남구 맛집'을 검색한다면 다음과 같은 주소가 되겠네요.
keyword = urllib.parse.quote('강남구 맛집') # 한글 인코딩
<https://openapi.naver.com/v1/search/local.json?query=인코딩결과&
display=10&start=1&sort=random>
```

- 출력 결과를 보면 우리는 JSON 방식을 요청하므로 items에 검색 결과가 담겨온다고 합니다.

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/99a0bd29-a8c8-4446-8375-af940bc630a3/Untitled.png>

- 네이버 검색 API 사용해보기

```
import requests
import pprint
from pymongo import MongoClient
```

```
# 맛집 데이터는 seoul_matjip 이라는 데이터베이스에 저장하겠습니다.
client = MongoClient('localhost', 27017)
db = client.seoul_matjip
```

```
# 서울시 구마다 맛집을 검색해보겠습니다.
seoul_gu = ["종로구", "중구", "용산구", "성동구", "광진구", "동대문구", "중랑구", "성북구", "강북구", "도봉구", "노원구", "은평구", "서대문구", "마포구", "양천구", "강서구", "구로구", "금천구", "영등포구", "동작구", "관악구", "서초구", "강남구", "송파구", "강동구"]
```

```
# 네이버 검색 API 신청을 통해 발급받은 아이디와 시크릿 키를 입력합니다.
client_id = "나의 클라이언트 아이디"
client_secret = "나의 시크릿 키"
```

```
# 검색어를 '강남구 맛집'으로 하겠습니다.
keyword = '강남구 맛집'
# url에 전달받은 검색어를 삽입합니다.
api_url = f"<https://openapi.naver.com/v1/search/local.json?query={
keyword}&display=10&start=1&sort=random>"
print(api_url)
# 아이디와 시크릿 키를 부가 정보로 같이 보냅니다.
```

```

headers = {'X-Naver-Client-Id': client_id, 'X-Naver-Client-Secret': client_secret }
# 검색 결과를 data에 저장합니다.
resp = requests.get(api_url, headers=headers)
# 받아온 JSON 결과를 딕셔너리로 변환합니다.
data = resp.json()
# 검색 결과 중 items를 꺼내어 반환합니다.
pprint.pprint(data)

```

- 구별 맛집을 검색하고 그 결과를 보는 코드는 다음과 같습니다.

```

import requests
import pprint
import time
from pymongo import MongoClient

# 맛집 데이터는 seoul_matjip 이라는 데이터베이스에 저장하겠습니다.
client = MongoClient('localhost', 27017)
db = client.seoul_matjip

# 서울시 구마다 맛집을 검색해보겠습니다.
seoul_gu = ["종로구", "중구", "용산구", "성동구", "광진구", "동대문구", "종랑구", "성북구",
"강북구", "도봉구", "노원구", "은평구", "서대문구", "마포구", "양천구", "강서구", "구로구",
"금천구", "영등포구", "동작구", "관악구", "서초구", "강남구", "송파구", "강동구"]

# 네이버 검색 API 신청을 통해 발급받은 아이디와 시크릿 키를 입력합니다.
client_id = "나의 클라이언트 아이디"
client_secret = "나의 시크릿 키"

# 검색어를 전달하면 결과를 반환하는 함수
def get_naver_result(keyword):
    time.sleep(0.1)
    # url에 전달받은 검색어를 삽입합니다.
    api_url = f"<a href='\"https://openapi.naver.com/v1/search/local.json?query={keyword}&display=10&start=1&sort=random\"'>https://openapi.naver.com/v1/search/local.json?query={
keyword}&display=10&start=1&sort=random</a>"
    # 아이디와 시크릿 키를 부가 정보로 같이 보냅니다.
    headers = {'X-Naver-Client-Id': client_id, 'X-Naver-Client-Secret': client_secret }
    # 검색 결과를 data에 저장합니다.
    data = requests.get(api_url, headers=headers)
    # 받아온 JSON 결과를 딕셔너리로 변환합니다.
    data = data.json()
    return data['items']


# 구별로 검색을 실행합니다.
for gu in seoul_gu:
    # '강남구 맛집', '종로구 맛집', '용산구 맛집' .. 을 반복해서 인코딩합니다.
    keyword = f'{gu} 맛집'
    # 맛집 리스트를 받아옵니다.
    matjip_list = get_naver_result(keyword)

    # 구별 맛집 구분선입니다.
    print("*"*80 + gu)

    for matjip in matjip_list:
        # 맛집을 인쇄합니다.
        pprint.pprint(matjip)

```

다음과 같은 결과가 뜨면 성공!

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/d5c8da1f-c07c-4d70-80ab-03fdc4e97b3e/Untitled.png>

◦ (3) DB에 저장하기

- DB에는 insert_many로 한 번에 저장합니다.

```

import requests
import pprint
import urllib.parse
import time
from pymongo import MongoClient

# 맛집 데이터는 seoul_matjip 이라는 데이터베이스에 저장하겠습니다.
client = MongoClient('localhost', 27017)
db = client.seoul_matjip

```

```

# 서울시 구마다 맛집을 검색해보겠습니다.
seoul_gu = ["종로구", "중구", "용산구", "성동구", "광진구", "동대문구", "중랑구", "성북구",
"강북구", "도봉구", "노원구", "은평구", "서대문구", "마포구", "양천구", "강서구", "구로구",
"금천구", "영등포구", "동작구", "관악구", "서초구", "강남구", "송파구", "강동구"]

# 네이버 검색 API 신청을 통해 발급받은 아이디와 시크릿 키를 입력합니다.
client_id = "나의 클라이언트 아이디"
client_secret = "나의 시크릿 키"

# 검색어를 전달하면 결과를 반환하는 함수
def get_naver_result(keyword):
    time.sleep(0.1)
    # url에 전달받은 검색어를 삽입합니다.
    api_url = f"<https://openapi.naver.com/v1/search/local.json?query={
keyword}&display=10&start=1&sort=random>"
    # 아이디와 시크릿 키를 부가 정보로 같이 보냅니다.
    headers = {'X-Naver-Client-Id': client_id, 'X-Naver-Client-Secret': client_secret }
    # 검색 결과를 data에 저장합니다.
    data = requests.get(api_url, headers=headers)
    # 받아온 JSON 결과를 딕셔너리로 변환합니다.
    data = data.json()
    return data['items']

# 저장할 전체 맛집 목록입니다.
docs = []
# 구별로 검색을 실행합니다.
for gu in seoul_gu:
    # '강남구 맛집', '종로구 맛집', '용산구 맛집' .. 을 반복해서 인코딩합니다.
    keyword = f'{gu} 맛집'
    # 맛집 리스트를 받아옵니다.
    matjip_list = get_naver_result(keyword)

    # 구별 맛집 구분선입니다.
    print("*"*80 + gu)

    for matjip in matjip_list:
        # 구 정보를 추가합니다.
        matjip['gu'] = gu
        # 맛집을 인쇄합니다.
        pprint.pprint(matjip)
        # docs에 맛집을 추가합니다.
        docs.append(matjip)

# 맛집 정보를 저장합니다.
db.matjip.insert_many(docs)

```

- 실행 후 Robo3T에서 제대로 저장된 것을 확인합니다.

◦ (4) matjip_to_db.py 완성코드

```

import requests
import pprint
import urllib.parse
import time
from pymongo import MongoClient

# 맛집 데이터는 seoul_matjip 이라는 데이터베이스에 저장하겠습니다.
client = MongoClient('localhost', 27017)
db = client.seoul_matjip

# 서울시 구마다 맛집을 검색해보겠습니다.
seoul_gu = ["종로구", "중구", "용산구", "성동구", "광진구", "동대문구", "중랑구", "성북구", "강북구",
"도봉구", "노원구", "은평구", "서대문구", "마포구", "양천구", "강서구", "구로구", "금천구",
"영등포구", "동작구", "관악구", "서초구", "강남구", "송파구", "강동구"]

# 네이버 검색 API 신청을 통해 발급받은 아이디와 시크릿 키를 입력합니다.
client_id = "나의 클라이언트 아이디"
client_secret = "나의 시크릿 키"

# 검색어를 전달하면 결과를 반환하는 함수
def get_naver_result(keyword):
    time.sleep(0.1)
    # url에 전달받은 검색어를 삽입합니다.

```

```

api_url = f"<a href='https://openapi.naver.com/v1/search/local.json?query={keyword}&display=10&start=1&sort=random'>https://openapi.naver.com/v1/search/local.json?query={keyword}&display=10&start=1&sort=random</a>"
# 아이디와 시크릿 키를 부가 정보로 같이 보냅니다.
headers = {'X-Naver-Client-Id': client_id, 'X-Naver-Client-Secret': client_secret }
# 검색 결과를 data에 저장합니다.
data = requests.get(api_url, headers=headers)
# 받아온 JSON 결과를 딕셔너리로 변환합니다.
data = data.json()
return data['items']

# 저장할 전체 맛집 목록입니다.
docs = []
# 구별로 검색을 실행합니다.
for gu in seoul_gu:
    # '강남구 맛집', '종로구 맛집', '용산구 맛집' .. 을 반복해서 인코딩합니다.
    keyword = f'{gu} 맛집'
    # 맛집 리스트를 받아옵니다.
    matjip_list = get_naver_result(keyword)

    # 구별 맛집 구분선입니다.
    print("*"*80 + gu)

    for matjip in matjip_list:
        # 구 정보를 추가합니다.
        matjip['gu'] = gu
        # 맛집을 인쇄합니다.
        pprint.pprint(matjip)
        # docs에 맛집을 추가합니다.
        docs.append(matjip)

# 맛집 정보를 저장합니다.
db.matjip.insert_many(docs)


```

4. [소개팅어디서하지] - 네이버 지도 API 신청하고 간단히 사용해보기


• 5. 네이버 지도 API 이용해보기

- 맛집 정보를 지도에 보여주기 위해서 네이버 지도 API 이용을 신청해보겠습니다.
- 지도 API는 검색 API와 다르게 클라우드 플랫폼을 통해 신청해야 합니다.


- (1) [링크](#)를 통해 회원가입을 진행해주세요.

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b1a0b204-b48e-41af-8772-32a118e0ac0c/Untitled.png>

- (2) [지도 API 링크](#)에서 의 '이용 신청하기' 버튼을 클릭하세요.


 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/8662d129-c45d-4095-8b52-0435b227b143/Untitled.png>

- (3) 'Application 등록' 버튼을 클릭합니다.

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a0f82083-677d-4350-a7de-b8f9dde292e/Untitled.png>


- (4) 설정 정보를 입력합니다.

- Application 이름 : matjip-sample 입력
- Maps : Web Dynamic Map 체크
- Web 서비스 URL : <http://localhost:5000> 입력 후 '+' 추가' 버튼 클릭
- 입력 완료 후 '등록' 클릭

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/19da29d3-6e05-4827-90ed-9f860f366ad6/Untitled.png>


- (5) 지도 API 인증 아이디 확인

[링크](#)를 통해 '인증 정보' 버튼을 클릭하여 클라이언트 ID를 확인합니다. 이는 나중에 지도 API에 사용하게 됩니다.

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ccbbe017-f259-468f-9381-7710215f9dc1/Untitled.png>

○ (6) 지도 띄워보기

- test.html 파일을 만들고, [링크](#)의 Hello World 예제를 복사합니다.

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e904c9a7-98aa-4be7-ada2-1c9e1a196cfd/Untitled.png>


- 본인의 지도 API 클라이언트 아이디를 YOUR_CLIENT_ID 부분에 삽입합니다. (아래 코드 8번째 줄 끝을 보세요!)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no">
  <title>간단한 지도 표시하기</title>
  <script type="text/javascript" src="<a href="https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=YOUR_CLIENT_ID">https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=YOUR_CLIENT_ID</a>"></script>
</head>
<body>
<div id="map" style="width:100%;height:400px;"></div>


<script>
var mapOptions = {
  center: new naver.maps.LatLng(37.3595704, 127.105399),
  zoom: 10
};

var map = new naver.maps.Map('map', mapOptions);
</script>
</body>
</html>
```

- 위 코드를 저장하고 브라우저에서 띄우면 '네이버 지도 Open API 인증이 실패했습니다.'라는 메시지를 보게 됩니다. 인증에 이 페이지를 추가하기 위해 주소창의 주소를 전체 복사합니다.

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/23cb1789-cb2b-45ae-97b5-db9101869651/Untitled.png>

- 어플리케이션 관리 페이지([링크](#))에서 '변경'을 클릭한 후, 위에서 복사한 주소를 Web 서비스 URL에 붙여넣고 '추가'를 클릭한 뒤 '저장' 버튼을 클릭합니다.
- 기존 창을 새로고침하면 아래와 같은 지도를 볼 수 있습니다.

 <https://s3-us-west-2.amazonaws.com/secure.notion-static.com/7e27ec94-141a-412a-82b6-caadc1aee02f/Untitled.png>

5. [소개팅어디서하지] - 네이버 지도 API 신청하고 서버 완성하기

- 6. 검색 기능 - index.html, [app.py](#) 준비하기

○ index.html

이번에는 <script></script> 코드를 body 태그 제일 아래에 넣는 이유

네이버 지도 API를 작동시키기 위해서는 id="map"인 div가 존재해야 하기 때문입니다.

아래 코드를 복사붙여넣기 한 이후, 11번째 줄 YOUR_CLIENT_ID 를 본인의 지도 API 키로 바꿉니다.

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport"
    content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-
```

```

scalable=no">
<title>스파르타코딩클럽 | 맛집 검색</title>
<script type="text/javascript"
src="<a href="https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=YOUR_CLIENT_ID&submodules=geocoder">https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=YOUR_CLIENT_ID&submodules=geocoder</a>"></script>

<link rel="stylesheet" href="<a href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css</a>"
integrity="sha384-Vkoo8x4CGs03+Hhvx8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">

<script src="<a href="https://code.jquery.com/jquery-3.4.1.slim.min.js">https://code.jquery.com/jquery-3.4.1.slim.min.js</a>"
integrity="sha384-J6qa4849bIE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>
<script src="<a href="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js">https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js</a>"
integrity="sha384-Q6E9RHvblyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTml3UkxdQRVvoxMfooAo"
crossorigin="anonymous"></script>
<script src="<a href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js">https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js</a>"
integrity="sha384-wfSDF2E50Y2D1uUdj003uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl30g8i fwb6"
crossorigin="anonymous"></script>

<script src=" <a href="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js</a>"></script>
<style>
#map {
width: 700px;
height: 500px;
margin: 50px auto 50px auto;
}

.wrap {
width: 700px;
margin: 10px auto;
}

.matjip-list {
overflow: scroll;
width: 700px;
height: 800px;
}
</style>

</head>

<body>
<div class="wrap">
<h1>오늘 소개팅 어디서 할까?</h1>
<div class="input-group mb-3">
<input id="gu-name" type="text" class="form-control" placeholder="예시) 강남구">
<div class="input-group-append">
<button type="button" class="btn btn-success" onclick="find_my_best_place()">만남
성공하자!!</button>
</div>
</div>
<div id="map"></div>

<div class="matjip-list" id="matjip-box">
</div>
</div>

<script>
let seoulGu = ["종로구", "중구", "용산구", "성동구", "광진구", "동대문구", "중랑구", "성북
구", "강북구", "도봉구", "노원구", "은평구", "서대문구", "마포구", "양천구", "강서구", "구로구",
"금천구", "영등포구", "동작구", "관악구", "서초구", "강남구", "송파구", "강동구"];
</script>
</body>

</html>

```

- o **app.py**

```

from flask import Flask, render_template, request, jsonify
from pymongo import MongoClient

app = Flask(__name__)

```

```

## URL 별로 함수명이 같거나,
## route('/') 등의 주소가 같으면 안됩니다.
client = MongoClient('localhost', 27017)
db = client.seoul_matjip

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/matjip', methods=["GET"])
def get_matjip():
    # gu_receive 라는 변수에 전달받은 구 이름을 저장합니다.
    # 구 이름에 해당하는 모든 맛집 목록을 불러옵니다.
    # matjip_list 라는 키 값에 맛집 목록을 담아 클라이언트에게 반환합니다.
    return jsonify({'result': 'success'})

if __name__ == '__main__':
    app.run('0.0.0.0', port=5000, debug=True)

```

- 7. 검색 기능 - 서버 완성하기

```

from flask import Flask, render_template, request, jsonify
from pymongo import MongoClient

app = Flask(__name__)

## URL 별로 함수명이 같거나,
## route('/') 등의 주소가 같으면 안됩니다.
client = MongoClient('localhost', 27017)
db = client.seoul_matjip

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/matjip', methods=["GET"])
def get_matjip():
    # gu_receive 라는 변수에 전달받은 구 이름을 저장합니다.
    gu_receive = request.args.get('gu_give')
    # 구 이름에 해당하는 모든 맛집 목록을 불러옵니다.
    matjip_list = list(db.matjip.find({'gu': gu_receive}, {'_id': False}))
    # matjip_list 라는 키 값에 맛집 목록을 담아 클라이언트에게 반환합니다.
    return jsonify({'result': 'success', 'matjip_list': matjip_list})

if __name__ == '__main__':
    app.run('0.0.0.0', port=5000, debug=True)

```

6. [소개팅어디서하지] - 지도에 마커 띄우기

- 8. 검색 기능 - 클라이언트 완성하기

- (1) 맛집 검색 요청 검사하기

```

// 서울시 전체 구 이름 목록
let seoulGu = ["종로구", "중구", "용산구", "성동구", "광진구", "동대문구", "중랑구", "성북구", "강북구", "도봉구", "노원구", "은평구", "서대문구", "마포구", "양천구", "강서구", "구로구", "금천구", "영등포구", "동작구", "관악구", "서초구", "강남구", "송파구", "강동구"];

// 전달받은 구 이름이 seoulGu 목록에 있는지 확인
function isValidGuName(guName) {
    for (let i = 0; i < seoulGu.length; i++) {
        if (guName == seoulGu[i]) {
            return true;
        }
    }
    return false;
}

// 맛집 검색 요청
function find_my_best_place() {
    let guName = $('#gu-name').val();
    if (guName == '') {
        alert('구 이름을 입력하세요');
    }
}

```



```

        return;
    }
    if (isValidGuName(guName) == false) {
        alert('올바른 구 이름을 입력하세요');
        return;
    }
}

```

◦ (2) 맛집 정보 서버에 요청해서 받아오기

```

function find_my_best_place() {

    let guName = $('#gu-name').val();
    if (guName == '') {
        alert('구 이름을 입력하세요');
        return;
    }
    if (isValidGuName(guName) == false) {
        alert('올바른 구 이름을 입력하세요');
        return;
    }

    // 기존 맛집 목록이 있으면 지우기
    $('#matjip-box').empty();
    // 맛집 정보 요청
    $.ajax({
        type: "GET",
        url: `/matjip?gu_give=${guName}`,
        data: {},
        success: function (response) {
            // 맛집 요청 성공 여부 검사
            if (response['result'] == 'success') {
                // 전달받은 맛집 리스트를
                matjipList에 저장한다.
                let matjipList = response['matjip_list'];
                console.log(matjipList);
            } else {
                alert('검색이 실패하였습니다.');
```

◦ (3) TM128 좌표를 위도/경도로 변환하기

TM128 좌표는 무엇이고 위도/경도로 변환하는 이유는 무엇인가요?

지도 상의 좌표를 나타내는 방법은 TM128, 위도/경도 등 여러 가지가 있습니다. 네이버 검색 API는 좌표를 TM128로 표현하고, 네이버 지도 API는 좌표를 위도/경도로 표현합니다. 따라서 검색에서 얻은 데이터를 지도에서 표현하기 위해 좌표 간의 변환이 필요합니다. (더 자세한 내용은 [링크](#)를 참조하세요)

```

function find_my_best_place() {

    let guName = $('#gu-name').val();
    if (guName == '') {
        alert('구 이름을 입력하세요');
        return;
    }
    if (isValidGuName(guName) == false) {
        alert('올바른 구 이름을 입력하세요');
        return;
    }

    // 기존 맛집 목록이 있으면 지우기
    $('#matjip-box').empty();
    // 맛집 정보 요청
    $.ajax({
        type: "GET",
        url: `/matjip?gu_give=${guName}`,
        data: {},
        success: function (response) {
            // 맛집 요청 성공 여부 검사
            if (response['result'] == 'success') {
```

```

matjipList에 저장한다.
    let matjipList = response['matjip_list'];
    // 전달받은 맛집 리스트를

    // TM128 좌표를 위도/경도 좌표로 변환하기

    matjipList =

    getMatjipListWithGeoData(matjipList);
    } else {
        alert('검색이 실패하였습니다.');
```

```

    }
    });
}

// matjipList 좌표 정보를 바꾸기
function getMatjipListWithGeoData(matjipList) {
    // 반환할 맛집 목록
    let result = [];

    for (let i = 0; i < matjipList.length; i++) {
        // 개별 맛집 데이터를 matjip에 저장
        let matjip = matjipList[i];
        let mapx = matjip['mapx'];
        let mapy = matjip['mapy'];
        // TM128 좌표를 위도(lat), 경도(lng) 디크셔너리로 반환
        let geoData = getLatLng(mapx, mapy);
        // geoData 라는 이름으로 맛집 데이터에 추가
        matjip['geoData'] = geoData;
        // 반환할 맛집 목록에 추가
        result.push(matjip);
    }
    // 맛집 목록 반환
    return result;
}

// TM128 좌표를 위도(lat), 경도(lng) 디크셔너리로 반환
function getLatLng(mapx, mapy) {
    // 문자열 -> 숫자로 변환
    let x = parseInt(mapx);
    let y = parseInt(mapy);

    // 네이버가 제공하는 변환 함수 사용
    let geoInfo = naver.maps.TransCoord.fromTM128ToLatLng(new naver.maps.Point(x, y));
    // 변환 디크셔너리 반환
    return { 'lat': geoInfo._lat, 'lng': geoInfo._lng }
}

```

◦ (4) HTML 태그 만들기

```

function find_my_best_place() {

    let guName = $('#gu-name').val();
    if (guName == '') {
        alert('구 이름을 입력하세요');
        return;
    }
    if (isValidGuName(guName) == false) {
        alert('올바른 구 이름을 입력하세요');
        return;
    }

    // 기존 맛집 목록이 있으면 지우기
    $('#matjip-box').empty();
    // 맛집 정보 요청
    $.ajax({
        type: "GET",
        url: `/matjip?gu_give=${guName}`,
        data: {},
        success: function (response) {
            // 맛집 요청 성공 여부 검사
            if (response['result'] == 'success') {
                // 전달받은 맛집 리스트를

```

```

matjipList에 저장한다.
    let matjipList = response['matjip_list'];

// TM128 좌표를 위도/경도 좌표로 변
환하기

matjipList =

getMatjipListWithGeoData(matjipList);

// 맛집을 HTML로 추가하기
addHTML(matjipList);

    } else {
        alert('검색이 실패하였습니다.');
```

◦ (5) 지도에 마커 띄우기

```

function find_my_best_place() {

    let guName = $('#gu-name').val();
    if (guName == '') {
        alert('구 이름을 입력하세요');
        return;
    }
    if (isValidGuName(guName) == false) {
        alert('올바른 구 이름을 입력하세요');
        return;
    }

    // 기존 맛집 목록이 있으면 지우기
    $('#matjip-box').empty();
    // 맛집 정보 요청
    $.ajax({
        type: "GET",
        url: `/matjip?gu_give=${guName}`,
        data: {},
        success: function (response) {

            // 맛집 요청 성공 여부 검사
            if (response['result'] == 'success') {

                // 전달받은 맛집 리스트를
                matjipList에 저장한다.
                let matjipList = response['matjip_list'];

                // TM128 좌표를 위도/경도 좌표로 변
                환하기

                matjipList =

                getMatjipListWithGeoData(matjipList);

                // 맛집을 HTML로 추가하기
                addHTML(matjipList);
                // 지도 그리기
                drawMap(matjipList);

            } else {
                alert('검색이 실패하였습니다.');
```

```

    }

    function drawMap(matjipList) {
        // 1등 맛집의 위치 정보를 geoData에 저장합니다.
        let geoData = matjipList[0]['geoData'];
        // 마커 목록 만들기
        let markerList = [];
        // 1등 맛집을 지도의 중심에 놓습니다.
        let numberOne = new naver.maps.LatLng(geoData['lat'], geoData['lng']),
            map = new naver.maps.Map('map', {
                center: numberOne,
                zoom: 12
            }),
            marker = new naver.maps.Marker({
                position: numberOne,
                map: map
            });

        // 2등부터 마지막까지 맛집 데이터를 지도에 표시합니다.
        for (let i = 1; i < matjipList.length; i++) {
            let matjip = matjipList[i];
            let position = new naver.maps.LatLng(matjip['geoData']['lat'], matjip['geoData']['lng'])
            marker = new naver.maps.Marker({
                position: position,
                map: map
            });
        }
    }
}

```

7. [소개팅어디서하지] - 마무리

- 9. 마커 클릭 기능 & OG 태그 붙이기
 - (1) 마커 클릭하면 지도 위에 정보 띄우기

```

function drawMap(matjipList) {
    let geoData = matjipList[0]['geoData'];
    let markerList = [];
    let contents = [];
    let numberOne = new naver.maps.LatLng(geoData['lat'], geoData['lng']),
        map = new naver.maps.Map('map', {
            center: numberOne,
            zoom: 12
        }),
        marker = new naver.maps.Marker({
            position: numberOne,
            map: map
        });

    for (let i = 0; i < matjipList.length; i++) {
        let matjip = matjipList[i];
        let position = new naver.maps.LatLng(matjip['geoData']['lat'], matjip['geoData']['lng'])
        marker = new naver.maps.Marker({
            position: position,
            map: map
        });

        // 마커를 클릭했을 때 보여줄 창을 HTML 태그로 만들기
        let contentString = `<div class="iw_inner">
            <h3>${matjip['title']}</h3>
            <p>${matjip['address']}<br />
                ${matjip['category']}<br />
                <a href="${matjip['link']}" target="_blank">링크</a>
            </p>
        </div>`;
        // contents에 태그를 저장
        contents.push(contentString);
        // 마커 목록에 markerList 저장하기
        markerList.push(marker);
    }

    // 클래스명이 matjip-title 카드 정보를 추출
    const matjipElement = $(''.matjip-title');
}

```

```

for (let i = 0; i < markerList.length; i++) {
  let marker = markerList[i];
  let infowindow = new naver.maps.InfoWindow({
    content: contents[i],
    maxWidth: 140,
    backgroundColor: "#eee",
    borderColor: "#2db400",
    borderWidth: 5,
    anchorSize: new naver.maps.Size(30, 30),
    anchorSkew: true,
    anchorColor: "#eee",
    pixelOffset: new naver.maps.Point(20, -20)
  });

  // 마커를 클릭했을 때 지도 위에 정보 띄우기
  naver.maps.Event.addListener(marker, "click", function (e) {
    if (infowindow.getMap()) {
      infowindow.close();
    } else {
      infowindow.open(map, marker);
    }
  });
}
}

```

◦ (2) 맛집 이름 클릭하면 지도 위에 정보 띄우기

```

function drawMap(matjipList) {
  let geoData = matjipList[0]['geoData'];
  let markerList = [];
  let contents = [];
  let numberOne = new naver.maps.LatLng(geoData['lat'], geoData['lng']),
  map = new naver.maps.Map('map', {
    center: numberOne,
    zoom: 12
  }),
  marker = new naver.maps.Marker({
    position: numberOne,
    map: map
  });

  for (let i = 0; i < matjipList.length; i++) {
    let matjip = matjipList[i];
    let position = new naver.maps.LatLng(matjip['geoData']['lat'], matjip['geoData']['lng'])
    marker = new naver.maps.Marker({
      position: position,
      map: map
    });

    // 마커를 클릭했을 때 보여줄 창을 HTML 태그로 만들기
    let contentString = `<div class="iw_inner">
      <h3>${matjip['title']}</h3>
      <p>${matjip['address']}<br />
        ${matjip['category']}<br />
        <a href="${matjip['link']}" target="_blank">링크</a>
      </p>
    </div>`;
    // contents에 태그를 저장
    contents.push(contentString);
    // 마커 목록에 markerList 저장하기
    markerList.push(marker);
  }

  // 클래스명이 matjip-title 카드 정보를 추출
  const matjipElement = $('matjip-title');

  for (let i = 0; i < markerList.length; i++) {
    let marker = markerList[i];
    let infowindow = new naver.maps.InfoWindow({
      content: contents[i],
      maxWidth: 140,
      backgroundColor: "#eee",
      borderColor: "#2db400",
      borderWidth: 5,

```

```

        anchorSize: new naver.maps.Size(30, 30),
        anchorSkew: true,
        anchorColor: "#eee",
        pixelOffset: new naver.maps.Point(20, -20)
    });
    naver.maps.Event.addListener(marker, "click", function (e) {
        if (infowindow.getMap()) {
            infowindow.close();
        } else {
            infowindow.open(map, marker);
        }
    });

    // 맛집 HTML의 상호명을 클릭했을 때 지도 위에 정보 띄우기
    matjipElement[i].addEventListener('click', function (e) {
        e.preventDefault();
        if (infowindow.getMap()) {
            infowindow.close();
        } else {
            infowindow.open(map, marker);
        }
    })
}
}
}

```

◦ (3) OG 태그 삽입하기

- static > images 폴더에 OG 태그로 사용할 이미지 파일을 넣어주세요.

[ogimage_matjip.png](#)

- OG 태그 코드입니다. 3번째 줄의 파일명과 확장자를 이미지 파일에 맞게 반드시 변경한 이후 <head> ~ </head> 태그 사이에 삽입해주세요!

```

<meta property="og:title" content="내 사이트의 제목" />
<meta property="og:description" content="보고 있는 페이지의 내용 요약" />
<meta property="og:image" content="{ url_for('static', filename='images/파일명.확장자') }" />

```

- 확인하기

[링크](#)를 통해 실제로 작동하는지 확인해볼 수 있습니다. (주소를 붙여넣고 '디버그' 클릭)

• 10. 완성 코드

◦ index.html

```

<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-
scalable=no">
        <meta property="og:title" content="스파르타코딩클럽 | 맛집 검색" />
        <meta property="og:description" content="중요한 소개팅, 맛집에서 성공하세요!" />
        <meta property="og:image" content="{ url_for('static', filename='images/파일명.확
장자') }" />
    <title>스파르타코딩클럽 | 맛집 검색</title>
    <script type="text/javascript"
        src="<https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=YOUR_CLIENT_ID&
submodules=geocoder>"></script>

    <link rel="stylesheet" href="<https://stackpath.bootstrapcdn.com/bootstrap/4.
4.1/css/bootstrap.min.css>"
        integrity="sha384-Vkoo8x4CGs03+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">

    <script src="<https://code.jquery.com/jquery-3.4.1.slim.min.js>"
        integrity="sha384-J6qa4849bIE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>

```

```

<script src="<https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js>"
  integrity="sha384-Q6E9RHvblyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
  crossorigin="anonymous"></script>
<script src="<https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js>"
  integrity="sha384-wfSDF2E50Y2D1uUdj003uMBJnjuUD4Ih7YwaYd1iQfktj0Uod8GCExl30g8i fwB6"
  crossorigin="anonymous"></script>

<script src=" <https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js>"></script>
<style>
  #map {
    width: 700px;
    height: 500px;
    margin: 50px auto 50px auto;
  }

  .wrap {
    width: 700px;
    margin: 10px auto;
  }

  .matjip-list {
    overflow: scroll;
    width: 700px;
    height: 800px;
  }
</style>

</head>

<body>
  <div class="wrap">
    <h1>오늘 소개팅 어디서 할까?</h1>
    <div class="input-group mb-3">
      <input id="gu-name" type="text" class="form-control" placeholder="예시) 강남구">
      <div class="input-group-append">
        <button type="button" class="btn btn-success" onclick="find_my_best_place()">만남
        성공하자!!</button>
      </div>
    </div>
    <div id="map"></div>

    <div class="matjip-list" id="matjip-box">
    </div>
  </div>

  <script>
    let seoulGu = ["종로구", "중구", "용산구", "성동구", "광진구", "동대문구", "중랑구", "성북
구", "강북구", "도봉구", "노원구", "은평구", "서대문구", "마포구", "양천구", "강서구", "구로구",
"금천구", "영등포구", "동작구", "관악구", "서초구", "강남구", "송파구", "강동구"];

    function isValidGuName(guName) {
      for (let i = 0; i < seoulGu.length; i++) {
        if (guName == seoulGu[i]) {
          return true;
        }
      }
      return false;
    }

    function getLatLng(mapx, mapy) {
      let x = parseInt(mapx);
      let y = parseInt(mapy);

      let geoInfo = naver.maps.TransCoord.fromTM128ToLatLng(new naver.maps.Point(x, y));
      return { 'lat': geoInfo._lat, 'lng': geoInfo._lng }
    }

    function getMatjipListWithGeoData(matjipList) {

      let result = [];

      for (let i = 0; i < matjipList.length; i++) {
        let matjip = matjipList[i];
        let mapx = matjip['mapx'];

```

```

        let mapy = matjip['mapy'];
        let geoData = getLatLng(mapx, mapy);
        matjip['geoData'] = geoData;
        result.push(matjip);
    }

    return result;
}

function find_my_best_place() {

    let guName = $('#gu-name').val();
    if (guName == '') {
        alert('구 이름을 입력하세요');
        return;
    }
    if (isValidGuName(guName) == false) {
        alert('올바른 구 이름을 입력하세요');
        return;
    }

    $('#matjip-box').empty();
    $.ajax({
        type: "GET",
        url: `/matjip?gu_give=${guName}`,
        data: {},
        success: function (response) {
            if (response['result'] == 'success') {
                let matjipList = response['matjip_list'];
                matjipList = getMatjipListWithGeoData(matjipList);
                addHTML(matjipList);
                drawMap(matjipList);
            } else {
                alert('검색이 실패하였습니다.');
```

</h5>

```

    }

    function makeCard(matjip) {
        return `<div class="card">
            <div class="card-body">
                <h5 class="card-title"><a href="#" class="matjip-title">${matjip['title']}

                <h6 class="card-subtitle mb-2 text-muted">${matjip['category']}</h6>
                <p class="card-text">${matjip['roadAddress']}</p>
                <a href="${matjip['link']}" target="_blank" class="card-link">링크</a>
                <a href="#" class="card-link">${matjip['telephone']}</a>
            </div>
        </div>`;
    }

    function drawMap(matjipList) {
        let geoData = matjipList[0]['geoData'];
        let markerList = [];
        let contents = [];
        let numberOne = new naver.maps.LatLng(geoData['lat'], geoData['lng']),
            map = new naver.maps.Map('map', {
                center: numberOne,
                zoom: 12
            }),
            marker = new naver.maps.Marker({
                position: numberOne,
                map: map
            });
    }

```



```

    for (let i = 0; i < matjipList.length; i++) {
      let matjip = matjipList[i];
      let position = new naver.maps.LatLng(matjip['geoData']['lat'], matjip['geoData']
['lng'])

      marker = new naver.maps.Marker({
        position: position,
        map: map
      });

      let contentString = `<div class="iw_inner">
        <h3>${matjip['title']}</h3>
        <p>${matjip['address']}<br />
          ${matjip['category']}<br />
          <a href="${matjip['link']}" target="_blank">링크</a>
        </p>
      </div>`;

      contents.push(contentString);
      markerList.push(marker);
    }

    const matjipElement = $('matjip-title');

    for (let i = 0; i < markerList.length; i++) {
      let marker = markerList[i];
      let infowindow = new naver.maps.InfoWindow({
        content: contents[i],
        maxWidth: 140,
        backgroundColor: "#eee",
        borderColor: "#2db400",
        borderWidth: 5,
        anchorSize: new naver.maps.Size(30, 30),
        anchorSkew: true,
        anchorColor: "#eee",
        pixelOffset: new naver.maps.Point(20, -20)
      });
      naver.maps.Event.addListener(marker, "click", function (e) {
        if (infowindow.getMap()) {
          infowindow.close();
        } else {
          infowindow.open(map, marker);
        }
      });

      matjipElement[i].addEventListener('click', function (e) {
        e.preventDefault();
        console.log('clicked');
        if (infowindow.getMap()) {
          infowindow.close();
        } else {
          infowindow.open(map, marker);
        }
      })
    }
  }
</script>
</body>

</html>

```

o app.py

```

from flask import Flask, render_template, request, jsonify
from pymongo import MongoClient

app = Flask(__name__)

## URL 별로 함수명이 같거나,
## route('/') 등의 주소가 같으면 안됩니다.
client = MongoClient('localhost', 27017)
db = client.seoul_matjip

@app.route('/')

```

```
def home():
    return render_template('index.html')

@app.route('/matjip', methods=["GET"])
def get_matjip():
    # gu_receive 라는 변수에 전달받은 구 이름을 저장합니다.
    gu_receive = request.args.get('gu_give')
    # 구 이름에 해당하는 모든 맛집 목록을 불러옵니다.
    matjip_list = list(db.matjip.find({'gu': gu_receive}, {'_id': False}))
    # matjip_list 라는 키 값에 맛집 목록을 담아 클라이언트에게 반환합니다.
    return jsonify({'result': 'success', 'matjip_list': matjip_list})

if __name__ == '__main__':
    app.run('0.0.0.0', port=5000, debug=True)
```