



[스파르타코딩클럽] 웹개발 종합반 - 4주차



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

▼ 단축키 모음

▼ 새로고침

- `F5`

▼ 저장

- Windows: `Ctrl` + `S`
- macOS: `command` + `S`

▼ 전체선택

- Windows: `Ctrl` + `A`
- macOS: `command` + `A`

▼ 잘라내기

- Windows: `Ctrl` + `X`
- macOS: `command` + `X`

▼ 콘솔창 줄바꿈

- `shift` + `enter`

▼ 코드정렬

- Windows: `Shift` + `Alt` + `F`
- macOS: `Shift` + `Option` + `F`

▼ 들여쓰기

- `Tab`
- 들여쓰기 취소 : `Shift` + `Tab`

▼ 주석

- Windows: `Ctrl` + `/`
- macOS: `command` + `/`

[수업 목표]

1. Flask 프레임워크를 활용해 API를 만들 수 있다
2. [화성땅 공동구매] 프로젝트를 생성해 API를 만들고 클라이언트에 연결한다
3. [스파르타피디아] 프로젝트를 생성해 API를 만들고 클라이언트에 연결한다

[목차]

01. 4주차 오늘 배울 것
02. Flask 시작하기 - 서버만들기 (1)
03. Flaks 시작하기 - 서버만들기 (2)
04. Flask 시작하기 - HTML 파일 주기

05. Flask 시작하기 - 본격 API 만들기
 06. [화성땅 공동구매] - 프로젝트 세팅
 07. [화성땅 공동구매] - 뼈대 준비하기
 08. [화성땅 공동구매] - POST연습(주문 저장하기)
 09. [화성땅 공동구매] - GET 연습(주문 보여주기)
 10. [스파르타피디아] - 프로젝트 세팅
 11. [스파르타피디아] - 조각 기능 구현해보기
 12. [스파르타피디아] - 뼈대 준비하기
 13. [스파르타피디아] - POST 연습(포스팅하기)
 14. [스파르타피디아] - GET 연습(보여주기)
 15. 4주차 끝 & 숙제 설명
 HW. 4주차 숙제 해설



모든 토글을 열고 닫는 단축키

Windows : **Ctrl** + **alt** + **t**

Mac : **⌘** + **⌥** + **t**

01. 4주차 오늘 배울 것

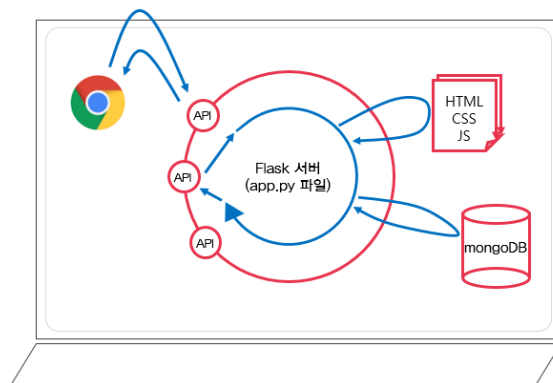
▼ 1) 4주차: Flask, 미니프로젝트1, 미니프로젝트2

이번 주 완성본 1. 화성땅공동구매 → [결과물 링크](#)

이번 주 완성본 2. 스파르타피디아 → [결과물 링크](#)



오늘은 HTML과 mongoDB까지 연동해서 서버를 만들어봅시다!



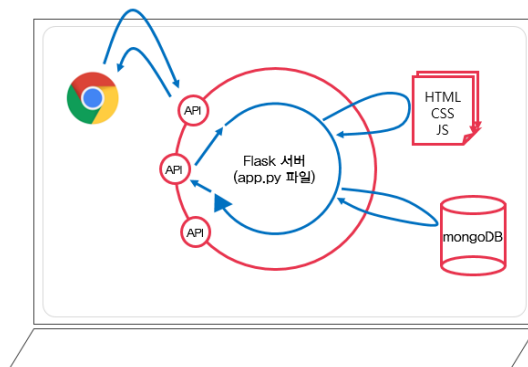
x 2개 더!



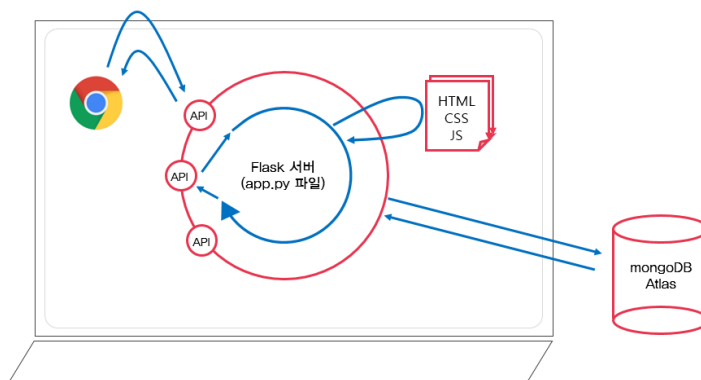
나중에 또 이야기하겠지만 헷갈리면 안되는 것!

우리는 컴퓨터가 한 대 잡아요... 그래서 같은 컴퓨터에다 서버도 만들고, 요청도 할 거예요. 즉, 클라이언트 = 서버가 되는 것이죠.

이것을 바로 "로컬 개발환경"이라고 한답니다! 그림으로 보면, 대략 이렇습니다.



😎 하지만 우리는 **mongoDB Atlas** 라는 클라우드 서비스를 이용하니, 아래와 같겠죠!



▼ 2) **sparta** 폴더 → **projects** 폴더 아래에, 다섯개 만들고 시작하기

😎 웹개발의 꽃, 백엔드-프론트엔드를 연결하는 일이 익숙해지도록,
연습 → 화성땅공동구매 → 스파르타피디아 → (숙제: 팬명록) → 버킷리스트

총 4번에 걸쳐 반복 실습을 진행 할 예정입니다. 다 해내고 나면 아-주 익숙해질거예요!

😎 코드 관리를 위해 미리 아래와 같이 폴더구조를 만들고 시작합니다!

- **01.prac** : "flask 연습 코드를 작성합니다." (오늘)
- **02.mars** : "화성땅공동구매" 관련 코드를 작성합니다. (오늘)
- **03.pedia** : "스파르타피디아" 관련 코드를 작성합니다. (오늘)
- **04.bucket** : "버킷리스트" 관련 코드를 작성합니다. (5주차)
- **05.fan** : "팬명록" 관련 코드를 작성합니다. (5주차)

02. Flask 시작하기 - 서버만들기 (1)

🔥 **sparta** → **projects** → **prac** 폴더에서 시작!

▼ 1) 폴더 생성과 가상환경 설치

▼ 1) 폴더와 파일 생성

👁️ prac 폴더 구조

prac
|— venv
|— app.py (서버)
|— templates
 |— index.html (클라이언트 파일)

😎 Flask는 만들 프로젝트의 폴더 구조가 정해져 있어요! 규칙을 지켜주세요!

- 폴더 안에 `app.py` 파일을 생성합니다!
- 폴더 안에 `templates` 폴더를 생성합니다!
- `templates` 폴더 안에 `index.html` 파일을 생성합니다!

🤔 아무 이름이나 쓰면 안되나요? → 몇 가지 규칙이 있습니다!

- `templates` 폴더는 반드시 고정해야 합니다! Flask의 규칙이에요!
- `app.py` 는 변경해도 좋습니다만, 라이브러리 이름과 같은 것을 이름으로 사용하면 안돼요!
- `index.html` 은 변경해도 좋습니다만, 첫 페이지는 일반적으로 `index.html` 을 사용해요!

🔥 준비가 끝났다면 화면 상단 `File > Open Folder` 로 prac 폴더로 이동합니다!

▼ 2) 가상환경 venv

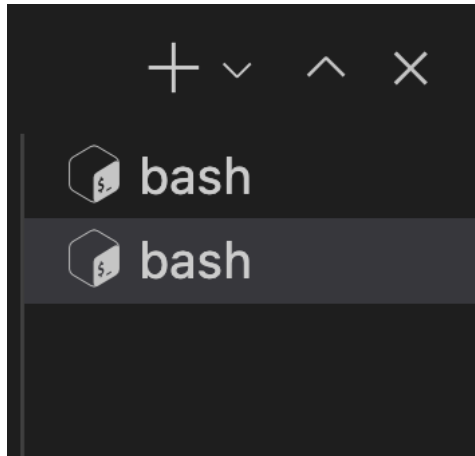
😎 우리는 웹을 만들기 위해 사용할 편리한 도구모음, 라이브러리를 사용할 거예요!

- 라이브러리를 담아둘 공구함, 가상환경 기억나시죠?
- 우리가 원하는 라이브러리만 설치해서 관리할 수 있도록 폴더에 하나씩 설치해줄거예요!

▼ 3) venv 생성하기

- 화면 상단 `Terminal > New Terminal` 을 클릭!

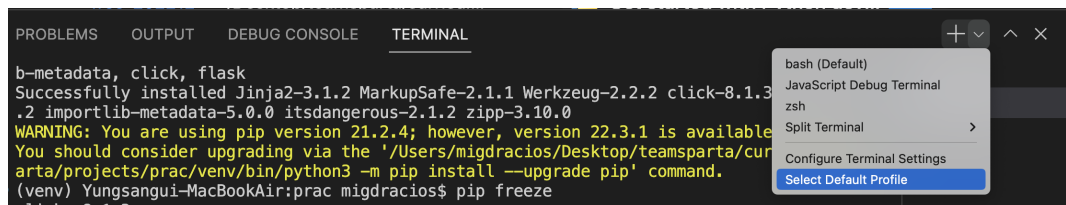
터미널이 bash로 생성된 것을 볼 수 있어요!



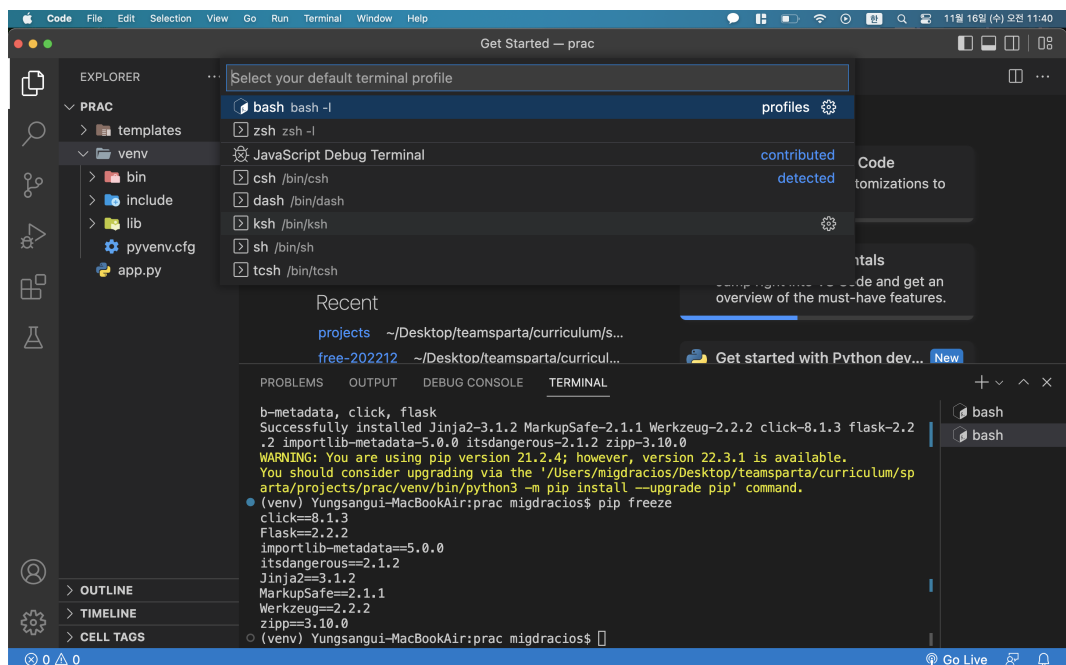
▼ 터미널에 bash가 아닌 것이 뜬다면?

- 터미널에서 오른쪽의 십자버튼(+) 옆의 더보기버튼(▼)을 클릭해주세요!

Select Default Profile 을 클릭!



화면 위쪽에 목록이 뜨게되면 bash를 골라주시고 엔터를 눌러주세요!

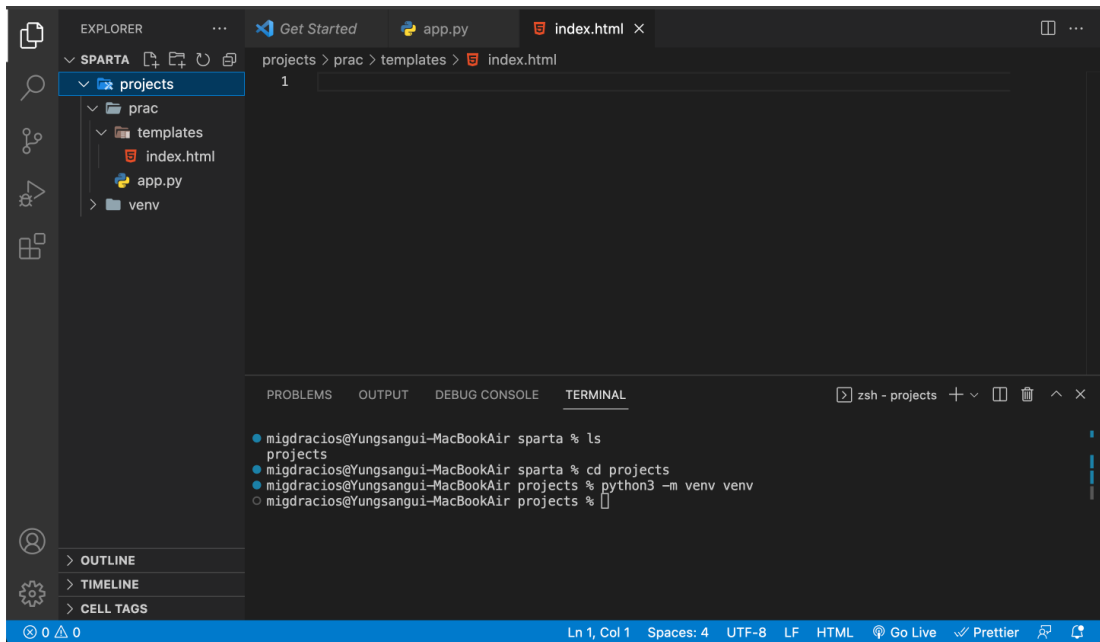


- python(백은 python3) -m venv venv 를 입력한 뒤 엔터!

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
● Yungsangui-MacBookAir:prac migdracios$ python3 -m venv venv
○ Yungsangui-MacBookAir:prac migdracios$
```

그러면 venv 폴더가 projects 폴더 내부에 생성되었습니다!



▼ 4) venv 활성화하기

😎 가상환경을 만든 것이 끝이 아닙니다! 활성화 시켜줘야 라이브러리를 컴퓨터가 읽어줘요!

- 화면 상단 `Terminal > New Terminal` 을 클릭! (열려있다면 그대로 진행해주세요!)
- 터미널에서 오른쪽의 십자버튼(+)을 클릭해주세요!

터미널에서 `(venv)` 라고 뜨게 된다면 가상환경이 활성화 된거예요!

```
● Yungsangui-MacBookAir:prac migdracios$ source /Use
  ulum/sparta/projects/prac/venv/bin/activate
○ (venv) Yungsangui-MacBookAir:prac migdracios$
```

▼ 🚩 십자버튼을 눌러도 활성화되지 않는다면?

03. Flaks 시작하기 - 서버만들기 (2)

▼ 2) Flask 패키지 설치하기



가상환경에 Flask 라이브러리를 설치 해 봅시다!

- 화면 상단 `Terminal > New Terminal` 을 클릭! (열려 있다면 그대로 진행해줍니다)
- 터미널에 `pip install flask` 라고 입력해주고 엔터를 눌러주면!
- 해커가 된 것 처럼 코드가 자동으로 막 내려와요! 설치 과정이니까 걱정마세요!

터미널 아래 부분의 `Successfully installed ...` 라고 적혀 있다면 설치 완료!

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Using cached itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Jinja2>=3.0
Using cached Jinja2-3.1.2-py3-none-any.whl (133 kB)
Collecting MarkupSafe>=2.0
Using cached MarkupSafe-2.1.1-cp310-cp310-macosx_10_9_universal2.whl (17 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, Werkzeug, Jinja2, flask
Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.2.2 click-8.1.3 flask-2.2.2 itsdangerous-2.1.2

[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: pip install --upgrade pip
(venv) migdracios@Yungsangui-MacBookAir projects %
```

▼ 3) Flask 기초 : 기본 실행

- **Flask 프레임워크**: 서버를 구동시켜주는 편한 코드 모음.
- 웹 서버를 구동하는데 필요한 복잡한 코드들을 쉽게 가져다 쓸 수 있습니다.



프레임워크를 쓰지 않으면 태양초를 뺏아서 고추장을 만드는 격!
프레임워크는 3분 요리/소스 세트라고 생각하면 되겠습니다!

- `app.py` 파일에 아래 코드를 붙여넣어줍니다.

▼ [코드스니펫] flask 시작 코드

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return 'This is Home!'

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

- 화면 상단 `Terminal > New Terminal` 을 클릭!
- (venv)가 보인다면 가상환경 활성화 상태!
- 터미널에서 `python(맥은 python3) app.py` 라고 적고 엔터를 눌러보세요!

터미널에서 아래 처럼 보이면 잘 작동한거예요!

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python - prac + - [ ] [ ] ^ X

* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://192.168.0.27:5001
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 201-664-330
```

▼ 👁 맥 환경에서 실행이 안된다면?

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python - prac + - [ ] [ ] ^ X

[notice] To update, run: pip install --upgrade pip
• (venv) migdracios@Yungsangui-MacBookAir projects % ls
  prac      venv
• (venv) migdracios@Yungsangui-MacBookAir projects % cd prac
• (venv) migdracios@Yungsangui-MacBookAir prac % ls
  app.py    templates
⊙ (venv) migdracios@Yungsangui-MacBookAir prac % python3 app.py
  * Serving Flask app 'app'
  * Debug mode: on
  Address already in use
  Port 5000 is in use by another program. Either identify and stop that program, or start the server with a
  different port.
```

- 네트워크 5000번 주소(포트)가 이미 사용 중 이라고 하네요!

app.py 파일로 들어와서 마지막 줄에 port=5000을 5001로 바꿔보세요!

```
Get Started app.py X index.html

projects > prac > app.py
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route('/')
5  def home():
6      return 'This is Home!'
7
8  if __name__ == '__main__':
9      app.run('0.0.0.0',port=5001,debug=True)
```

다시 실행! 짬!


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python - prac + v [] [X] ^ X

* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://192.168.0.27:5001
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 201-664-330
```

- 터미널에 나오는 <http://127.0.0.1:5000> 링크 부분에 마우스를 대로 **Ctrl(맥은 Command) 클릭!**

😎 화면에 `This is Home!` 이라는 메시지가 보이시나요? 그렇다면 성공한 것! 🌟

- 종료하는 방법

👁️ 터미널 창을 클릭하시고, `ctrl + c` 을 누르시면 서버를 종료할 수 있습니다.

▼ 4) Flask 기초 : URL 나눠보기

😎 `@app.route('/')` 부분을 수정해서 URL을 나눌 수 있습니다! 간단하죠?

- 주의할점!
 - url 별로 함수명이 같거나,
 - `route('/')` 내의 주소가 같으면 안됩니다!

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return 'This is Home!'

@app.route('/mypage')
def mypage():
    return 'This is My Page!'

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

04. Flask 시작하기 - HTML 파일 주기

▼ 1) Flask 기초 : Flask의 기본 폴더 구조



Flask 서버를 만들 때, 항상,

프로젝트 폴더 안에,

└ **templates** 폴더 (html파일을 넣어둡니다)

└ **app.py** 파일

이렇게 두 개를 만들어두고 시작하세요. 이제 각 폴더의 역할을 알아보시다!

- 참고!! venv는 실제로는 보이지만, 안보인다~라고 생각하세요! 기억하시죠?

▼ 2) Flask 기초: HTML 파일 불러오기



templates 폴더의 역할을 알아보겠습니다.

HTML 파일을 담아두고, 불러오는 역할을 하죠!

1. 간단한 index.html 파일을 templates 안에 만들기

▼ [코드스니펫] index.html 예제코드

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <title>Document</title>

  <script>
    function hey(){
      alert('안녕!')
    }
  </script>
</head>
<body>
  <button onclick="hey()">나는 버튼!</button>
</body>
</html>
```

2. html 파일 불러오기



flask 내장함수 `render_template`를 이용합니다. 바로 이게 프레임워크의 위력!

```
from flask import Flask, render_template
app = Flask(__name__)

## URL 별로 함수명이 같거나,
## route('/') 등의 주소가 같으면 안됩니다.

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

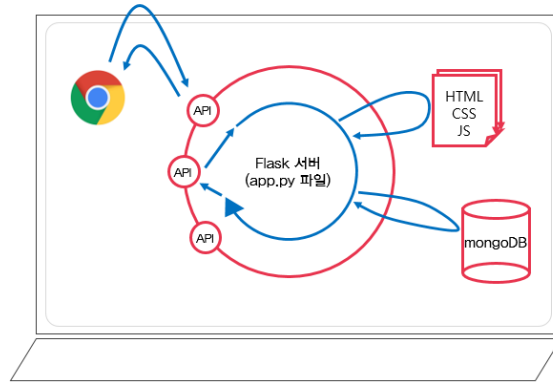
if __name__ == '__main__':
    app.run('0.0.0.0', port=5000, debug=True)
```

05. Flask 시작하기 - 본격 API 만들기

▼ 1) 들어가기 전에 : GET, POST 요청 타입 리마인드



은행 창구에서 은행원의 역할을 하는 서버! 만들어 봅시다!



x 2개 더!

- 은행의 창구가 API와 같다는 것을 기억하시나요?
- 같은 예금 창구에서도 개인 고객이나 기업 고객이나에 따라 처리하는 것이 다른 것처럼,
- 클라이언트가 요청 할 때에도, "방식"이 존재합니다.
- HTTP 라는 통신 규약을 따른다는 거 잊지 않으셨죠? 클라이언트는 요청할 때 HTTP request method(요청 메소드)를 통해, 어떤 요청 종류인지 응답하는 서버 쪽에 정보를 알려주는 거예요.
- 여러 방식이 존재하지만, 가장 많이 쓰이는 **GET**, **POST** 방식에 대해서 다루겠습니다!

▼ 1) GET 요청

- 통상적으로 데이터 조회(Read)를 요청할 때, 사용합니다!

예) 영화 목록 조회

→ 데이터 전달 : URL 뒤에 물음표를 붙여 key=value로 전달

▼ 2) POST 요청

- 통상적으로 데이터 생성(Create), 변경(Update), 삭제(Delete) 요청 할 때 사용합니다!

예) 회원가입, 회원탈퇴, 비밀번호 수정

→ 데이터 전달 : 바로 보이지 않는 HTML

▼ 2) GET, POST 요청에서 클라이언트의 데이터를 받는 방법

😎 고객이 명세서를 작성하면 은행원이 내용을 봐야겠죠?

- 예를 들어, 클라이언트에서 서버에 **title_give** 란 키 값으로 데이터를 들고왔다고 생각합니다.
(주민등록번호 라는 키 값으로 900120- .. 을 가져온 것과 같은 의미)
- **print()** 코드를 사용해서 들어온 데이터를 터미널에 눈으로 볼 수 있게 찍어봅시다!

▼ [코드스니펫] JQuery 임포트

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

▼ [코드스니펫] GET 요청 API코드

```
@app.route('/test', methods=['GET'])
def test_get():
    title_receive = request.args.get('title_give')
    print(title_receive)
    return jsonify({'result':'success', 'msg': '이 요청은 GET!'})
```

▼ [코드스니펫] GET 요청 확인 Fetch 코드

```
fetch("/test").then(res => res.json()).then(data => {
  console.log(data)
})
```

▼ [코드스니펫] POST 요청 API코드


```
@app.route('/test', methods=['POST'])
def test_post():
    title_receive = request.form['title_give']
    print(title_receive)
    return jsonify({'result':'success', 'msg': '이 요청은 POST!'})
```

▼ [코드스니펫] POST 요청 확인 Fetch코드

```
let formData = new FormData();
formData.append("title_give", "블랙팬서");

fetch("/test", { method: "POST", body: formData }).then(res => res.json()).then(data => {
  console.log(data)
})
```


06. [화성땅 공동구매] - 프로젝트 세팅

 sparta → projects → mars 폴더에서 시작!

▼ 1) 프로젝트 분석 - 완성작부터 보기!

- **mars** : "화성땅공동구매" 관련 코드를 작성합니다.

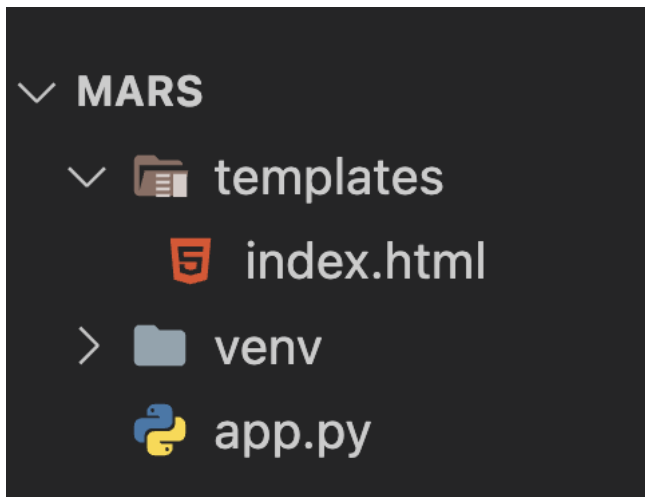
▼ 2) 프로젝트 설정 - flask 폴더 구조 만들기

 templates 폴더 + app.py 만들기! 기억나시죠?

 mars 폴더 구조

```

---
prac
|— venv
|— app.py (서버)
|— templates
    |— index.html (클라이언트 파일)
```



😎 Flask는 만들 프로젝트의 폴더 구조가 정해져 있어요! 규칙을 지켜주세요!

- 폴더 안에 `app.py` 파일을 생성합니다!
- 폴더 안에 `templates` 폴더를 생성합니다!
- `templates` 폴더 안에 `index.html` 파일을 생성합니다!

😞 아무 이름이나 쓰면 안되나요? → 몇 가지 규칙이 있습니다!

- `templates` 폴더는 반드시 고정해야 합니다! Flask의 규칙이에요!
- `app.py` 는 변경해도 좋습니다만, 라이브러리 이름과 같은 것을 이름으로 사용하면 안돼요!
- `index.html` 은 변경해도 좋습니다만, 첫 페이지는 일반적으로 `index.html` 을 사용해요!

🔥 준비가 끝났다면 화면 상단 `File > Open Folder` 로 mars 폴더로 이동합니다!

▼ 3) 가상환경 생성, 패키지 설치하기

▼ 1) 가상환경 생성, 활성화

- 화면 상단 `Terminal > New Terminal` 을 클릭!
- `python(맥 python3) -m venv venv` 입력 후 엔터!
- 터미널 오른쪽의 십자버튼을 클릭!
- `(venv)` 라고 뜨면 활성화까지 완료!

▼ 2) 패키지 설치

🔥 이번에 필요한 패키지는 3개 : `flask`, `pymongo`, `dnspython`

- 화면 상단 `Terminal > New Terminal` 을 클릭!(열려있다면 그대로 진행!)

▼ [코드스니펫] 패키지 설치코드

```
pip install flask pymongo dnspython
```

- 여러 개를 설치할 때는 띄어쓰기로 구분해요!

▼ 📌 원하는 라이브러리를 확인해보려면?

- 터미널에서 `pip freeze` 를 입력하고 엔터!
- 터미널에서 flask, pymongo, dnspython을 찾아보죠!

😎 제가 찾던 라이브러리 여기 있네요!

내용이 뭐가 많да구요?

우리가 사용하는 라이브러리가 다른 라이브러리를 필요로 해서 그것도 설치해주는 거예요!

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
zsh - projects + - [ ] [ ] ^ x

[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: pip install --upgrade pip
• (venv) migdracios@Yungsangui-MacBookAir projects % pip freeze
click==8.1.3
dnspython==2.2.1
Flask==2.2.2
itsdangerous==2.1.2
Jinja2==3.1.2
MarkupSafe==2.1.1
pymongo==4.3.2
Werkzeug==2.2.2
○ (venv) migdracios@Yungsangui-MacBookAir projects %
```

07. [화성땅 공동구매] - 뼈대 준비하기

▼ 1) 프로젝트 준비 - `app.py` 준비하기

▼ [코드스니펫] 화성땅 공동구매-app.py

```
from flask import Flask, render_template, request, jsonify
app = Flask(__name__)

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

@app.route("/movie", methods=["POST"])
def movie_post():
    sample_receive = request.form['sample_give']
    print(sample_receive)
    return jsonify({'msg': 'POST 연결 완료!'})

@app.route("/movie", methods=["GET"])
def movie_get():
    return jsonify({'msg': 'GET 연결 완료!'})

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

▼ 2) 프로젝트 준비 - `index.html` 준비하기

▼ [코드스니펫] 화성땅 공동구매-index.html

```
<!DOCTYPE html>
<html lang="en">

<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" />

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztqcQTWfSpd3yD65VohhpuuCOMLASjC" crossorigin="anonymous" />
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
        crossorigin="anonymous"></script>

  <link href="https://fonts.googleapis.com/css2?family=Gowun+Batang:wght@400;700&display=swap" rel="stylesheet" />

  <title>선착순 공동구매</title>
```

```

<style>
* {
  font-family: "Gowun Batang", serif;
  color: white;
}

body {
  background-image: linear-gradient(0deg,
    rgba(0, 0, 0, 0.5),
    rgba(0, 0, 0, 0.5)),
    url("https://cdn.aitimes.com/news/photo/202010/132592_129694_3139.jpg");
  background-position: center;
  background-size: cover;
}

h1 {
  font-weight: bold;
}

.order {
  width: 500px;
  margin: 60px auto 0px auto;
  padding-bottom: 60px;
}

.mybtn {
  width: 100%;
}

.order>table {
  margin: 40px 0;
  font-size: 18px;
}

option {
  color: black;
}
</style>
<script>
$(document).ready(function () {
  show_order();
});
function show_order() {
  fetch('/mars').then((res) => res.json()).then((data) => {
    console.log(data)
    alert(data['msg'])
  })
}
function save_order() {
  let formData = new FormData();
  formData.append("sample_give", "샘플데이터");

  fetch('/mars', { method: "POST", body: formData }).then((res) => res.json()).then((data) => {
    console.log(data);
    alert(data["msg"]);
  });
}
</script>
</head>

<body>
<div class="mask"></div>
<div class="order">
<h1>화성에 땅 사내기!</h1>
<h3>가격: 평 당 500원</h3>
<p>
  화성에 땅을 사들 수 있다고?<br />
  앞으로 백년 간 오지 않을 기회. 화성에서 즐기는 노후!
</p>
<div class="order-info">
<div class="input-group mb-3">
<span class="input-group-text">이름</span>
<input id="name" type="text" class="form-control" />
</div>
<div class="input-group mb-3">
<span class="input-group-text">주소</span>
<input id="address" type="text" class="form-control" />
</div>
<div class="input-group mb-3">
<label class="input-group-text" for="size">평수</label>
<select class="form-select" id="size">
<option selected>-- 주문 평수 --</option>
<option value="10평">10평</option>
<option value="20평">20평</option>
<option value="30평">30평</option>
<option value="40평">40평</option>

```

```

        <option value="50평">50평</option>
      </select>
    </div>
    <button onclick="save_order()" type="button" class="btn btn-warning mybtn">
      주문하기
    </button>
  </div>
  <table class="table">
    <thead>
      <tr>
        <th scope="col">이름</th>
        <th scope="col">주소</th>
        <th scope="col">평수</th>
      </tr>
    </thead>
    <tbody id="order-box">
      <tr>
        <td>홍길동</td>
        <td>서울시 용산구</td>
        <td>20평</td>
      </tr>
      <tr>
        <td>임꺽정</td>
        <td>부산시 동구</td>
        <td>10평</td>
      </tr>
      <tr>
        <td>세종대왕</td>
        <td>세종시 대왕구</td>
        <td>30평</td>
      </tr>
    </tbody>
  </table>
</div>
</body>
</html>

```

▼ 3) 프로젝트 준비 - mongoDB Atlas 창 띄워두기

▼ [코드스니펫] mongoDB Atlas 주소

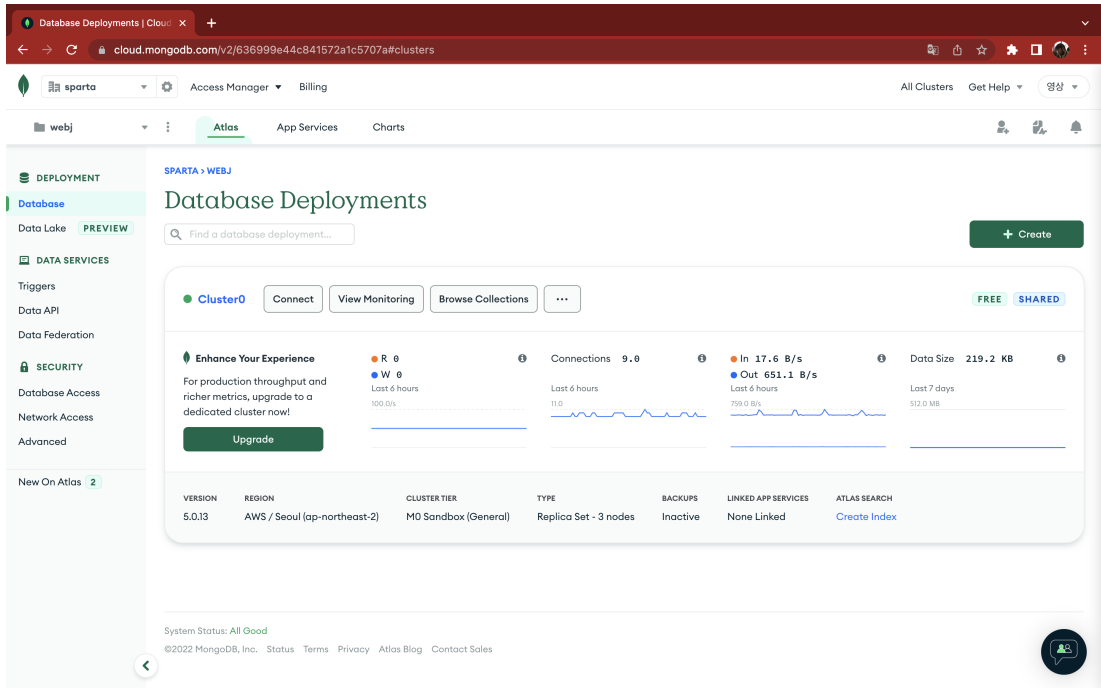
```
https://cloud.mongodb.com/
```

▼ mongoDB 데이터베이스 들어가는 법

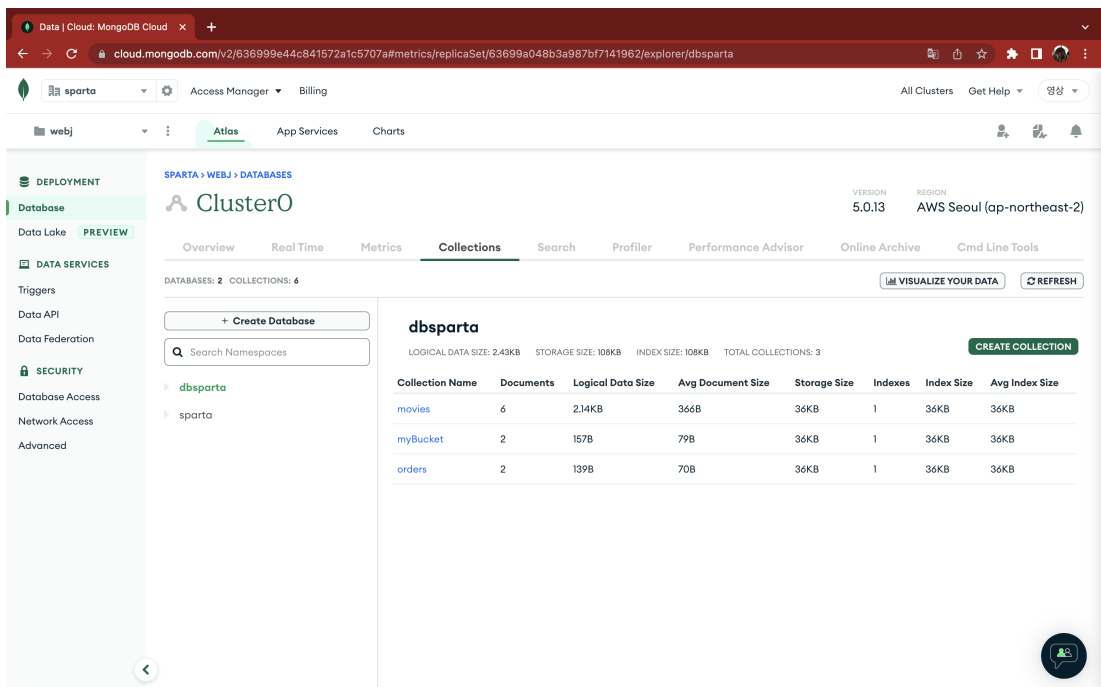
😎 지난 주차에서 mongoDB Atlas! 기억나시죠?

- 아직 연결한 것은 없으니 보이는 것은 없지만, 이따 확인하려면 미리 띄워둬야죠!

화면 왼쪽의 Database를 클릭한 뒤 **Cluster 0** 클릭해주세요!



화면 가운데의 여러 탭 중 **Collection** 탭을 클릭해주세요!



08. [화성땅 공동구매] - POST연습(주문 저장하기)

▼ 1) API 만들고 사용하기 - 이름, 주소, 평수 저장하기(Create → POST)



데이터 생성은 POST 방식! 기억나시죠?

▼ 1) 데이터 명세

- 1. 요청 정보 : URL= `/mars` , 요청 방식 = `POST`
- 2. 클라(fetch) → 서버(flask) : `name` , `address` , `size`
- 3. 서버(flask) → 클라(fetch) : 메시지를 보냄 (주문 완료!)

▼ 2) 클라이언트와 서버 연결 확인하기

[서버 코드 - `app.py`]

```
@app.route("/mars", methods=["POST"])
def mars_post():
    sample_receive = request.form['sample_give']
    print(sample_receive)
    return jsonify({'msg': 'POST 연결 완료!'})
```

[클라이언트 코드 - `index.html`]

```
function save_order() {
    let formData = new FormData();
    formData.append("sample_give", "샘플데이터");

    fetch('/mars', {method: "POST", body: formData}).then((res) => res.json()).then((data) => {
        console.log(data);
        alert(data["msg"]);
    });
}

<button onclick="save_order()" type="button" class="btn btn-warning mybtn">주문하기</button>
```

▼ 3) 서버부터 만들기

데이터베이스에 연결해줍시다!

```
from pymongo import MongoClient

client = MongoClient('내 URL')
db = client.dbsparta
```

- `name` , `address` , `size` 정보를 받아서, 저장하면 되겠죠?
- 우리가 일전에 만들어둔 `dbtest.py` 파일을 불러와봅시다!

```
@app.route("/mars", methods=["POST"])
def mars_post():
    name_receive = request.form['name_give']
    address_receive = request.form['address_give']
    size_receive = request.form['size_give']

    doc = {
        'name': name_receive,
        'address': address_receive,
        'size': size_receive
    }

    db.orders.insert_one(doc)

    return jsonify({'msg': '주문 완료!'})
```

▼ 4) 클라이언트 만들기

- 이번엔 `name` , `address` , `size` 정보를 보내주면 되겠죠?
- `formData`에 데이터를 넣고, 보내줍니다!

```
function save_order() {
  let name = $("#name").val();
  let address = $("#address").val();
  let size = $("#size").val();

  let formData = new FormData();
  formData.append("name_give", name);
  formData.append("address_give", address);
  formData.append("size_give", size);

  fetch('/mars', {method: "POST", body: formData}).then((res) => res.json()).then((data) => {
    console.log(data);
    alert(data["msg"]);
    window.location.reload();
  });
}
```

▼ 5) 완성 확인하기

- 데이터를 입력하고
- DB에 잘 들어갔는지 확인해보면 되겠죠?



09. [화성땅 공동구매] - GET 연습(주문 보여주기)

▼ 1) API 만들고 사용하기 - 저장된 주문을 화면에 보여주기(Read → GET)

🕶 데이터 조회는 GET 방식! 기억나시죠?

▼ 1) 데이터 명세

- 1. 요청 정보 : URL= `/mars`, 요청 방식 = `GET`
- 2. 클라(fetch) → 서버(flask) : 없음
- 3. 서버(flask) → 클라(fetch) : 전체 주문을 보내주기

▼ 2) 클라이언트와 서버 연결 확인하기

[서버 코드 - `app.py`]

```
@app.route("/mars", methods=["GET"])
def mars_get():
    return jsonify({'msg': 'GET 연결 완료!'})
```

[클라이언트 코드 - `index.html`]

```
$(document).ready(function() {
  show_order()
})
function show_order() {
  fetch('/mars').then((res) => res.json()).then((data) => {
    console.log(data);
  });
}
```

```

        alert(data["msg"]);
    });
}

```

▼ 3) 서버부터 만들기

받을 것 없이 **orders** 에 주문정보를 담아서 내려주기만 하면 됩니다!

```

@app.route("/mars", methods=["GET"])
def mars_get():
    orders_list = list(db.orders.find({}, {'_id': False}))
    return jsonify({'orders': orders_list})

```

▼ 4) 클라이언트 만들기

- 주문정보는 리스트 형식이겠죠? **forEach** 문으로 반복하면서 데이터를 뽑아냅시다!
- 뽑아낸 데이터는 **temp_html** 에 담아줘야겠죠?
- 담아준 **temp_html**을 넣을 자리를 찾아 제이쿼리로 **append** !


```

function show_order() {
    $("#order-box").empty();
    fetch('/mars').then((res) => res.json()).then((data) => {
        // console.log(data)
        data["orders"].forEach((row) => {
            let name = row["name"];
            let address = row["address"];
            let size = row["size"];

            let temp_html = `<tr>
                <td>${name}</td>
                <td>${address}</td>
                <td>${size}</td>
            </tr>`;
            $("#order-box").append(temp_html);
        });
    });
}

```

▼ 5) 완성 확인하기

 **동작 테스트:** 화면을 새로고침 했을 때, DB에 저장된 리뷰가 화면에 올바르게 나타나는지 확인합니다.

▼ 2) 전체 완성 코드

▼ [📄 코드] app.py - 화성땅 공동구매 서버

```

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

app = Flask(__name__)
client = MongoClient('내 URL')
db = client.dbsparta

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

@app.route("/mars", methods=["POST"])
def mars_post():
    name_receive = request.form['name_give']
    address_receive = request.form['address_give']
    size_receive = request.form['size_give']

    doc = {
        'name': name_receive,

```

```

        'address': address_receive,
        'size': size_receive
    }

    db.orders.insert_one(doc)

    return jsonify({'msg': '주문 완료!'})

@app.route("/mars", methods=["GET"])
def mars_get():
    orders_list = list(db.orders.find({}, {'_id': False}))
    return jsonify({'orders': orders_list})

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

```

▼ [코드] index.html - 화성땅 공동구매 클라이언트

```

<!DOCTYPE html>
<html lang="en">
<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" />

  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztCQTWfsdpd3yD65VohhpuuCOMLASjC"
    crossorigin="anonymous"
  />
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script
    src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-Mrcw6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/twtIaxVXM"
    crossorigin="anonymous"
  ></script>

  <link
    href="https://fonts.googleapis.com/css2?family=Gowun+Batang:wght@400;700&display=swap"
    rel="stylesheet"
  />

  <title>선착순 공동구매</title>

  <style>
    * {
      font-family: "Gowun Batang", serif;
      color: white;
    }

    body {
      background-image: linear-gradient(
        0deg,
        rgba(0, 0, 0, 0.5),
        rgba(0, 0, 0, 0.5)
      ),
      url("https://cdn.aitimes.com/news/photo/202010/132592_129694_3139.jpg");
      background-position: center;
      background-size: cover;
    }

    h1 {
      font-weight: bold;
    }

    .order {
      width: 500px;
      margin: 60px auto 0px auto;
      padding-bottom: 60px;
    }

    .mybtn {
      width: 100%;
    }

    .order > table {
      margin: 40px 0;
      font-size: 18px;
    }

    option {
      color: black;
    }
  </style>

```

```

</style>
<script>
$(document).ready(function () {
  show_order();
});
function show_order() {
  $("#order-box").empty();
  fetch('/mars').then((res) => res.json()).then((data) => {
    // console.log(data)
    data["orders"].forEach((row) => {
      let name = row["name"];
      let address = row["address"];
      let size = row["size"];

      let temp_html = `<tr>
        <td>${name}</td>
        <td>${address}</td>
        <td>${size}</td>
      </tr>`;
      $("#order-box").append(temp_html);
    });
  });
}
function save_order() {
  let name = $("#name").val();
  let address = $("#address").val();
  let size = $("#size").val();

  let formData = new FormData();
  formData.append("name_give", name);
  formData.append("address_give", address);
  formData.append("size_give", size);

  fetch('/mars', {method: "POST", body: formData}).then((res) => res.json()).then((data) => {
    console.log(data);
    alert(data["msg"]);
    window.location.reload();
  });
}
</script>
</head>
<body>
<div class="mask"></div>
<div class="order">
  <h1>화성에 땅 사놓기!</h1>
  <h3>가격: 평 당 500원</h3>
  <p>
    화성에 땅을 사들 수 있다고?<br />
    앞으로 백년 간 오지 않을 기회. 화성에서 즐기는 노후!
  </p>
  <div class="order-info">
    <div class="input-group mb-3">
      <span class="input-group-text">이름</span>
      <input id="name" type="text" class="form-control" />
    </div>
    <div class="input-group mb-3">
      <span class="input-group-text">주소</span>
      <input id="address" type="text" class="form-control" />
    </div>
    <div class="input-group mb-3">
      <label class="input-group-text" for="size">평수</label>
      <select class="form-select" id="size">
        <option selected>-- 주문 평수 --</option>
        <option value="10평">10평</option>
        <option value="20평">20평</option>
        <option value="30평">30평</option>
        <option value="40평">40평</option>
        <option value="50평">50평</option>
      </select>
    </div>
    <button onclick="save_order()" type="button" class="btn btn-warning mybtn">주문하기</button>
  </div>
  <table class="table">
    <thead>
      <tr>
        <th scope="col">이름</th>
        <th scope="col">주소</th>
        <th scope="col">평수</th>
      </tr>
    </thead>
    <tbody id="order-box">
      <tr>
        <td>홍길동</td>
        <td>서울시 용산구</td>
        <td>20평</td>
      </tr>
    </tbody>
  </table>

```

```

        <td>임꺽정</td>
        <td>부산시 동구</td>
        <td>10평</td>
    </tr>
    <tr>
        <td>세종대왕</td>
        <td>세종시 대왕구</td>
        <td>30평</td>
    </tr>
</tbody>
</table>
</div>
</body>
</html>

```


10. [스파르타피디아] - 프로젝트 세팅


 sparta → projects → movie 폴더에서 시작!

▼ 1) 문제 분석 - 완성작부터 보기!

- `movie` : "스파르타피디아" 관련 코드를 작성합니다. (오늘)

▼ 2) 프로젝트 설정 - flask 폴더 구조 만들기

 templates 폴더 + `app.py` 만들기! 기억나시죠?

 movie 폴더 구조

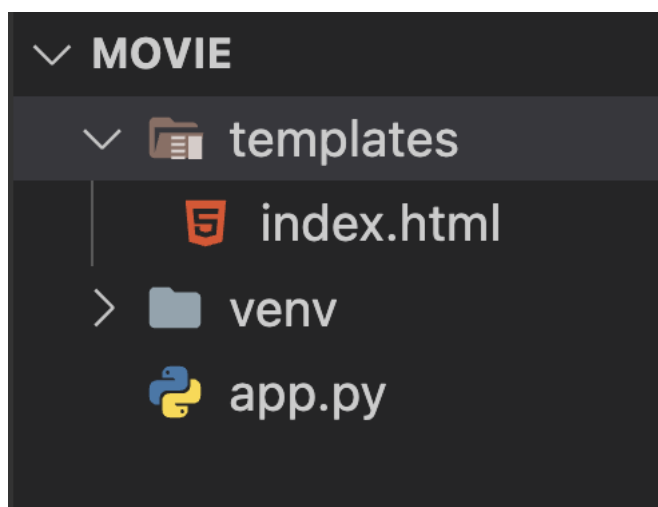
prac


|— venv

|— `app.py` (서버)

|— templates

|— index.html (클라이언트 파일)



 Flask는 만들 프로젝트의 폴더 구조가 정해져 있어요! 규칙을 지켜주세요!

- 폴더 안에 `app.py` 파일을 생성합니다!
- 폴더 안에 `templates` 폴더를 생성합니다!
- `templates` 폴더 안에 `index.html` 파일을 생성합니다!

😞 아무 이름이나 쓰면 안되나요? → **몇 가지 규칙이 있습니다!**

- `templates` 폴더는 반드시 고정해야 합니다! Flask의 규칙이에요!
- `app.py` 는 변경해도 좋습니다만, **라이브러리 이름과 같은 것을 이름으로 사용하면 안돼요!**
- `index.html` 은 변경해도 좋습니다만, 첫 페이지는 일반적으로 `index.html` 을 사용해요!

🔥 준비가 끝났다면 화면 상단 `File > Open Folder` 로 `movie` 폴더로 이동합니다!

▼ 3) 가상환경 생성, 패키지 설치하기

▼ 1) 가상환경 생성, 활성화

- 화면 상단 `Terminal > New Terminal` 을 클릭!
- `python(맥 python3) -m venv venv` 입력 후 엔터!
- 터미널 오른쪽의 십자버튼을 클릭!
- `(venv)` 라고 뜨면 활성화까지 완료!

▼ 2) 패키지 설치

🔥 이번에 필요한 패키지는 3개 : `flask` , `pymongo` , `dnspython` , `bs4` , `requests`

- 화면 상단 `Terminal > New Terminal` 을 클릭!(열려있다면 그대로 진행!)

▼ [코드스니펫] 패키지 설치코드

```
pip install flask pymongo dnspython
```

- 여러 개를 설치할 때는 띄어쓰기로 구분해요!

▼ 📌 원하는 라이브러리를 확인해보려면?

- 터미널에서 `pip freeze` 를 입력하고 엔터!
- 터미널에서 `flask`, `pymongo`, `dnspython`, `bs4`, `requests`을 찾아보죠!

😎 제가 찾던 라이브러리 여기 있네요!

내용이 뭐가 많대구요?

우리가 사용하는 라이브러리가 다른 라이브러리를 필요로 해서 그것도 설치해주는 거예요!


```
(venv) Yungsangui-MacBookAir:movie migdracios$ pip freeze
beautifulsoup4==4.11.1
bs4==0.0.1
certifi==2022.9.24
charset-normalizer==2.1.1
click==8.1.3
dnspython==2.2.1
Flask==2.2.2
idna==3.4
importlib-metadata==5.0.0
itsdangerous==2.1.2
Jinja2==3.1.2
MarkupSafe==2.1.1
pymongo==4.3.2
requests==2.28.1
soupsieve==2.3.2.post1
```

11. [스파르타피디아] - 조각 기능 구현해보기

▼ 1) 조각기능?

😎 프로젝트 내부에서 꼭 만들 필요는 없습니다!

- flask 라이브러리를 사용하지 않는 기능이 있다면?
 - → `app.py` 밖에서 **조각으로 구현해 붙여보세요!**
- 그렇게 하는 이유가 있나요?
 - → 일일이 테스트를 flask에서 브라우저를 열고 진행하지 않아서 **더 빨라요!**
- 어떻게 하는데요?
 - → `app.py` 말고 다른 파일을 만들어봐요! 예) `dbprac.py`
 - 새로 만든 파일에서 코드를 작업해서 성공하면 복사해서 `app.py`로 가져와요!

▼ 2) 웹스크래핑을 사용해 URL에서 페이지 정보 가져오기

▼ 1) meta 태그를 활용하기!

😎 meta태그는 눈에 보이는 것 이외의 웹의 속성을 설명해주는 태그예요!

- 우리가 공유하고 싶은 웹이 있다면 링크로 담아서 카톡에 공유하죠?
- 링크만 보냈는데 달려오는 정보들이 meta태그의 한 예시입니다!

아래 예시에서 보이는 세 가지의 정보! 이게 meta태그의 역할 중 하나예요!

1. 썸네일 사진 - `og:image`
2. 썸네일 제목 - `og:title`
3. 썸네일 설명 - `og:description`



▼ 2) meta 태그 정보 가져오는 조각기능 만들기

- 이 정보를 어디에 쓰냐구요? 우리가 만들 스파르타피디아에서 **포스팅 정보에 들어갈 녀석들이죠!**
- 왜 이걸로 크롤링 하나구요? 우리가 원하는 정보를 **더 쉽게 가져올 수 있기 때문이죠!**
- 기본 준비부터 합시다! 연습을 위해 `meta_prac.py` 파일을 만들어봅시다.

▼ [코드스니펫] meta_prac.py - 크롤링 기본 코드

```
import requests
from bs4 import BeautifulSoup

url = 'https://movie.naver.com/movie/bi/mi/basic.naver?code=191597'

headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.36
data = requests.get(url, headers=headers)

soup = BeautifulSoup(data.text, 'html.parser')

# 여기에 코딩을 해서 meta tag를 먼저 가져와보겠습니다.
```

- 이제 meta 태그를 크롤링 해봅시다 → 구글링: `meta 태그 크롤링`

select_one을 이용해 meta 태그를 가져와 봅시다

```
og_image = soup.select_one('meta[property="og:image"]')
og_title = soup.select_one('meta[property="og:title"]')
og_description = soup.select_one('meta[property="og:description"]')

print(og_image)
print(og_title)
print(og_description)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

//nClick 초기화 영역 끝
</script>
</body>
</html>
• (venv) migdracios@Yungsangui-MacBookAir movie % python3 meta_prac.py
<meta content="https://movie-phinf.pstatic.net/20210622_174/1624324910624JhEq2_JPEG/movie_image.jpg?type=m665_443_2" property="og:image"/>
<meta content="보스 베이비 2" property="og:title"/>
<meta content="베이비 주식회사의 레전드 보스 베이비에서 인생 만렙 CEO가 된 '태드', 베이비인 줄 알았던 조카 '티...' property="og:description"/>
```

가져온 meta 태그의 content를 가져와봅시다

```
image = og_image['content']
title = og_title['content']
description = og_description['content']

print(image)
```

```
print(title)
print(description)
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
<meta content="https://movie-phinf.pstatic.net/20210622_174/1624324910624JhEq2_JPEG/movie_image.jpg?type=
<meta content="보스 베이비 2" property="og:title"/>
<meta content="베이비 주식회사의 레전드 보스 베이비에서 인생 만렙 CEO가 된 '테드'.베이비인 줄 알았던 조카
(venv) migdracios@Yungsangui-MacBookAir movie % python3 meta_prac.py
https://movie-phinf.pstatic.net/20210622_174/1624324910624JhEq2_JPEG/movie_image.jpg?type=m665_443_2
보스 베이비 2
베이비 주식회사의 레전드 보스 베이비에서 인생 만렙 CEO가 된 '테드'.베이비인 줄 알았던 조카 '티...
(venv) migdracios@Yungsangui-MacBookAir movie %
```



완성 되었다면 프로젝트로 가져다 쓸 준비 완료!

12. [스파르타피디아] - 뼈대 준비하기

▼ 1) 프로젝트 준비 - [app.py](#) 준비하기

▼ [코드스니펫] [app.py](#) - 스파르타피디아

```
from flask import Flask, render_template, request, jsonify
app = Flask(__name__)

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

@app.route("/movie", methods=["POST"])
def movie_post():
    sample_receive = request.form['sample_give']
    print(sample_receive)
    return jsonify({'msg': 'POST 연결 완료!'})

@app.route("/movie", methods=["GET"])
def movie_get():
    return jsonify({'msg': 'GET 연결 완료!'})

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

▼ 2) 프로젝트 준비 - [index.html](#) 준비하기

▼ [코드스니펫] [index.html](#) - 스파르타피디아

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
        crossorigin="anonymous"></script>

  <title>스파르타 피디아</title>

  <link href="https://fonts.googleapis.com/css2?family=Gowun+Dodum&display=swap" rel="stylesheet">

  <style>
    * {
      font-family: 'Gowun Dodum', sans-serif;
    }
  </style>
</head>

<body>
  <div class="container">
    <div class="row">
      <div class="col">
        <h1>스파르타 피디아</h1>
      </div>
    </div>
  </div>
</body>
</html>
```

```
.mytitle {
    width: 100%;
    height: 250px;

    background-image: linear-gradient(0deg, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url('https://movie-phinf.pstatic.com/movie-phinf-pstatic-com.nhn/image/upload_thumbnail/movie-phinf-pstatic-com.nhn');
    background-position: center;
    background-size: cover;

    color: white;

    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}

.mytitle > button {
    width: 200px;
    height: 50px;

    background-color: transparent;
    color: white;

    border-radius: 50px;
    border: 1px solid white;

    margin-top: 10px;
}

.mytitle > button:hover {
    border: 2px solid white;
}

.mycomment {
    color: gray;
}

.mycards {
    margin: 20px auto 0px auto;
    width: 95%;
    max-width: 1200px;
}

.mypost {
    width: 95%;
    max-width: 500px;
    margin: 20px auto 0px auto;
    padding: 20px;
    box-shadow: 0px 0px 3px 0px gray;

    display: none;
}

.mybtns {
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: center;

    margin-top: 20px;
}

.mybtns > button {
    margin-right: 10px;
}
</style>
<script>
$(document).ready(function(){
    listing();
});

function listing() {
    fetch('/movie').then((res) => res.json()).then((data) => {
        console.log(data)
        alert(data['msg'])
    })
}

function posting() {
    let formData = new FormData();
    formData.append("sample_give", "샘플데이터");

    fetch('/movie', {method : "POST", body : formData}).then((res) => res.json()).then((data) => {
        console.log(data)
        alert(data['msg'])
    })
}
```

```

        function open_box(){
            $('#post-box').show()
        }
        function close_box(){
            $('#post-box').hide()
        }
    }
</script>
</head>

<body>
<div class="mytitle">
    <h1>내 생애 최고의 영화들</h1>
    <button onclick="open_box()">영화 기록하기</button>
</div>
<div class="mypost" id="post-box">
    <div class="form-floating mb-3">
        <input id="url" type="email" class="form-control" placeholder="name@example.com">
        <label>영화URL</label>
    </div>
    <div class="form-floating">
        <textarea id="comment" class="form-control" placeholder="Leave a comment here"></textarea>
        <label for="floatingTextarea2">코멘트</label>
    </div>
    <div class="mybtns">
        <button onclick="posting()" type="button" class="btn btn-dark">기록하기</button>
        <button onclick="close_box()" type="button" class="btn btn-outline-dark">닫기</button>
    </div>
</div>
<div class="mycards">
    <div class="row row-cols-1 row-cols-md-4 g-4" id="cards-box">
        <div class="col">
            <div class="card h-100">
                
                <div class="card-body">
                    <h5 class="card-title">영화 제목이 들어갑니다</h5>
                    <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                    <p>***</p>
                    <p class="mycomment">나의 한줄 평을 씁니다</p>
                </div>
            </div>
        </div>
        <div class="col">
            <div class="card h-100">
                
                <div class="card-body">
                    <h5 class="card-title">영화 제목이 들어갑니다</h5>
                    <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                    <p>***</p>
                    <p class="mycomment">나의 한줄 평을 씁니다</p>
                </div>
            </div>
        </div>
        <div class="col">
            <div class="card h-100">
                
                <div class="card-body">
                    <h5 class="card-title">영화 제목이 들어갑니다</h5>
                    <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                    <p>***</p>
                    <p class="mycomment">나의 한줄 평을 씁니다</p>
                </div>
            </div>
        </div>
        <div class="col">
            <div class="card h-100">
                
                <div class="card-body">
                    <h5 class="card-title">영화 제목이 들어갑니다</h5>
                    <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                    <p>***</p>
                    <p class="mycomment">나의 한줄 평을 씁니다</p>
                </div>
            </div>
        </div>
    </div>
</div>
</body>
</html>

```

▼ 3) 프로젝트 준비 - mongoDB Atlas 창 띄워두기

▼ [코드스니펫] MongoDB Atlas 주소

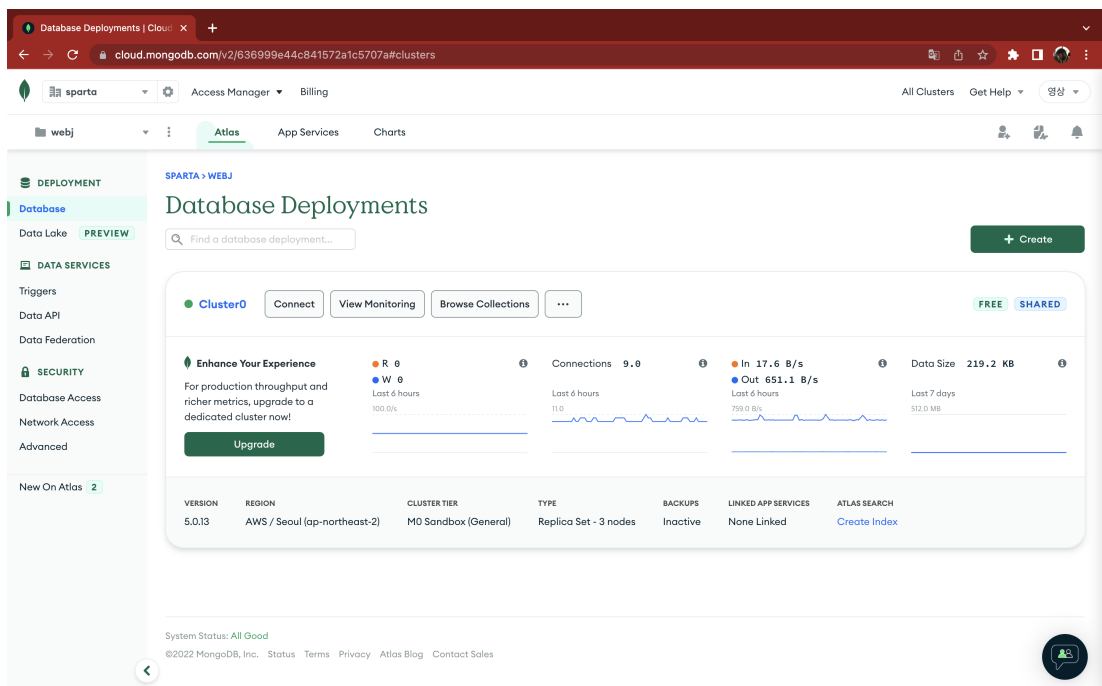
<https://cloud.mongodb.com/>

▼ MongoDB 데이터베이스 들어가는 법

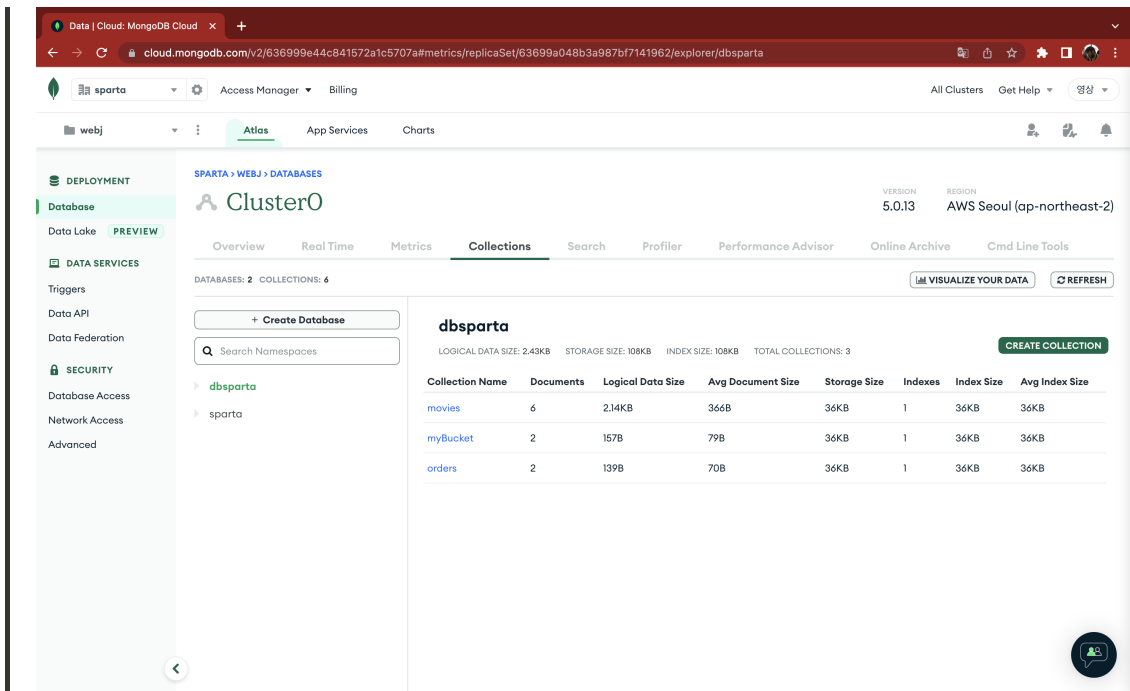
😎 지난 주차에서 MongoDB Atlas! 기억나시죠?

- 아직 연결한 것은 없으니까 보이는 것은 없지만, 이따 확인하려면 미리 띄워둬야죠!

화면 왼쪽의 Database를 클릭한 뒤 **Cluster 0** 클릭해주세요!



화면 가운데의 여러 탭 중 **Collection** 탭을 클릭해주세요!



13. [스파르타피디아] - POST 연습(포스팅하기)

▼ 1) API 만들고 사용하기 - 포스팅 API(Create → POST)

☎ 데이터 생성은 POST 방식! 기억나시죠?

▼ 1) 데이터 명세

- 1. 요청 정보 : URL = `/movie`, 요청 방식 = `POST`
- 2. 클라(fetch) → 서버(flask) : `url`, `comment`
- 3. 서버(flask) → 클라(fetch) : 메시지를 보냄 (`포스팅 완료!`)

▼ 2) 클라이언트와 서버 연결 확인하기

[서버 코드 - `app.py`]

```
@app.route("/movie", methods=["POST"])
def movie_post():
    sample_receive = request.form['sample_give']
    print(sample_receive)
    return jsonify({'msg': 'POST 연결 완료!'})
```

[클라이언트 코드 - `index.html`]

```
function posting() {
    fetch('/movie', {method: "POST", body: formData,}).then((res) => res.json()).then((data) => {
        console.log(data);
        alert(data["msg"]);
    });
}

<button onclick="posting()" type="button" class="btn btn-dark">기록하기</button>
```

▼ 3) 서버부터 만들기

- `url`, `comment` 정보를 받아서, 저장하면 되겠죠?
- 그리고, 우리가 미리 만든 `meta_prac.py` 도 참고해서 붙여봅시다!
- 우리가 일전에 만들어둔 `dbtest.py` 파일을 불러와봅시다!

```
@app.route("/movie", methods=["POST"])
def movie_post():
    url_receive = request.form['url_give']
    comment_receive = request.form['comment_give']

    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Sa
    data = requests.get(url_receive, headers=headers)

    soup = BeautifulSoup(data.text, 'html.parser')

    og_image = soup.select_one('meta[property="og:image"]')
    og_title = soup.select_one('meta[property="og:title"]')
    og_description = soup.select_one('meta[property="og:description"]')

    image = og_image['content']
    title = og_title['content']
    description = og_description['content']

    doc = {
        'image':image,
        'title':title,
        'desc':description,
        'comment':comment_receive
    }

    db.movies.insert_one(doc)

    return jsonify({'msg':'포스팅 완료!'})
```

▼ 4) 클라이언트 만들기

- 이번엔 `url`, `comment` 정보를 보내주면 되겠죠?
- `formData`에 데이터를 넣고 보내줍시다!
- 요청을 보내고 **포스팅 완료!** 라고 받은 메시지를 `alert()` 로 띄워줍니다
- 그 다음은 자연스럽게 보이기 위해 `window.location.reload()` 새로고침을 해줍시다!

```
function posting() {
    let url = $('#url').val()
    let comment = $('#comment').val()

    let formData = new FormData()
    formData.append('url_give', url)
    formData.append('comment_give', comment)

    fetch('/movie', {method: "POST", body: formData}).then(res => res.json()).then(data => {
        console.log(data)
        alert(data['msg'])
        window.location.reload()
    })
}
```

▼ 5) 완성 확인하기

- DB에 잘 들어갔는지 확인해보면 되겠죠?
- 그 전에, 연습했던 `movies` 컬렉션을 삭제해줄게요!

```
_id: ObjectId('637238dd14861e986f5e22f6')
image: "https://movie-phinf.pstatic.net/20221108_228/1667874041329YcSHk_JPEG/m..."
title: "데시벨"
desc: "물이 끓는 주전자 소리, 창문 여는 소리, 놀이터 아이들의 웃음 소리...잠시 후, 거대한 굉음과 함께 단..."
comment: "야호 재밌어요!"
```


14. [스파르타피디아] - GET 연습(보여주기)

▼ 1) API 만들고 사용하기 - 포스트 보여주기 API(Read → GET)

😎 데이터 조회는 GET 방식! 기억나시죠?

▼ 1) 데이터 명세

- 1. 요청 정보 : URL= `/movie`, 요청 방식 = `GET`
- 2. 클라(fetch) → 서버(flask) : 없음
- 3. 서버(flask) → 클라(fetch) : `전체 주문을 보내주기`

▼ 2) 클라이언트와 서버 확인하기

[서버 코드 - `app.py`]

```
@app.route("/movie", methods=["GET"])
def movie_get():
    return jsonify({'msg': 'GET 연결 완료!'})
```

[클라이언트 코드 - `index.html`]

```
$(document).ready(function(){
    listing();
});

function listing() {
    fetch('/movie').then((res => res.json())).then((data) => {
        console.log(data)
        alert(data['msg'])
    })
}
```

▼ 3) 서버부터 만들기

받을 것 없이 `orders` 에 주문정보를 담아서 내려주기만 하면 됩니다!

```
@app.route("/movie", methods=["GET"])
def movie_get():
    movie_list = list(db.movies.find({}, {'_id': False}))
    return jsonify({'movies': movie_list})
```

▼ 4) 클라이언트 만들기

- 주문정보는 리스트 형식이겠죠? `forEach` 문으로 반복하면서 데이터를 뽑아냅니다!
- 뽑아낸 데이터는 `temp_html` 에 담아줘야겠죠?
- 담아준 `temp_html` 을 넣을 자리를 찾아 제이쿼리로 `append` !

```
function listing() {
    $('#cards-box').empty()
    fetch('/movie').then(res => res.json()).then(data => {
        // console.log(data)
        let movies = data['movies']
        movies.forEach(movie => {
            //console.log(movie)
            let title = movie['title']
            let image = movie['image']
```

```

        let desc = movie['desc']
        let comment = movie['comment']

        let temp_html = `<div class="col">
            <div class="card h-100">
                
                <div class="card-body">
                    <h5 class="card-title">{title}</h5>
                    <p class="card-text">{desc}</p>
                    <p class="mycomment">{comment}</p>
                </div>
            </div>
        </div>`

        $('#cards-box').append(temp_html)
    })
}

```

▼ 5) 완성 확인하기



동작 테스트: 화면을 새로고침 했을 때, DB에 저장된 리뷰가 화면에 올바르게 나타나는지 확인합니다.

▼ 2) 전체 완성 코드

▼ [코드] app.py - 스파르타피디아 서버

```

from flask import Flask, render_template, request, jsonify
app = Flask(__name__)

from pymongo import MongoClient
client = MongoClient('내 mongoDB URL')
db = client.dbsparta

import requests
from bs4 import BeautifulSoup

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

@app.route("/movie", methods=["POST"])
def movie_post():
    url_receive = request.form['url_give']
    comment_receive = request.form['comment_give']

    headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.36'}
    data = requests.get(url_receive, headers=headers)

    soup = BeautifulSoup(data.text, 'html.parser')

    ogtitle = soup.select_one('meta[property="og:title"]')['content']
    ogdesc = soup.select_one('meta[property="og:description"]')['content']
    ogimage = soup.select_one('meta[property="og:image"]')['content']

    doc = {
        'title':ogtitle,
        'desc':ogdesc,
        'image':ogimage,
        'comment':comment_receive
    }
    db.movies.insert_one(doc)

    return jsonify({'msg':'저장완료!'})

@app.route("/movie", methods=["GET"])
def movie_get():
    all_movies = list(db.movies.find({}, {'_id':False}))
    return jsonify({'result':all_movies})

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

```

▼ [코드] index.html - 스파르타피디아 클라이언트

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztQCtwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-Mrcw6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxvXM"
    crossorigin="anonymous"></script>

  <title>스파르타 피디아</title>

  <link href="https://fonts.googleapis.com/css2?family=Gowun+Dodum&display=swap" rel="stylesheet">

  <style>
    * {
      font-family: 'Gowun Dodum', sans-serif;
    }

    .mytitle {
      width: 100%;
      height: 250px;

      background-image: linear-gradient(0deg, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url('https://movie-phinf.pstatic');
      background-position: center;
      background-size: cover;

      color: white;

      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
    }

    .mytitle>button {
      width: 200px;
      height: 50px;

      background-color: transparent;
      color: white;

      border-radius: 50px;
      border: 1px solid white;

      margin-top: 10px;
    }

    .mytitle>button:hover {
      border: 2px solid white;
    }

    .mycomment {
      color: gray;
    }

    .mycards {
      margin: 20px auto 0px auto;
      width: 95%;
      max-width: 1200px;
    }

    .mypost {
      width: 95%;
      max-width: 500px;
      margin: 20px auto 0px auto;
      padding: 20px;
      box-shadow: 0px 0px 3px 0px gray;

      display: none;
    }

    .mybtns {
      display: flex;
      flex-direction: row;
      align-items: center;
      justify-content: center;

      margin-top: 20px;
    }
  </style>
</html>
```

```

        .mybtns>button {
            margin-right: 10px;
        }
    </style>
    <script>
        $(document).ready(function () {
            listing();
        });

        function listing() {
            fetch('/movie').then((res) => res.json()).then((data) => {
                let rows = data['result']
                $('#cards-box').empty()
                rows.forEach((a)=>{
                    let comment = a['comment']
                    let title = a['title']
                    let desc = a['desc']
                    let image = a['image']

                    let temp_html = `<div class="col">
                        <div class="card h-100">
                            
                            <div class="card-body">
                                <h5 class="card-title">${title}</h5>
                                <p class="card-text">${desc}</p>
                                <p>***</p>
                                <p class="mycomment">${comment}</p>
                            </div>
                        </div>
                    `
                    $('#cards-box').append(temp_html)
                })
            })
        }

        function posting() {
            let url = $('#url').val()
            let comment = $('#comment').val()

            let formData = new FormData();
            formData.append("url_give", url);
            formData.append("comment_give", comment);

            fetch('/movie', { method: "POST", body: formData }).then((res) => res.json()).then((data) => {
                alert(data['msg'])
                window.location.reload()
            })
        }

        function open_box() {
            $('#post-box').show()
        }
        function close_box() {
            $('#post-box').hide()
        }
    </script>
</head>

<body>
    <div class="mytitle">
        <h1>내 생애 최고의 영화들</h1>
        <button onclick="open_box()">영화 기록하기</button>
    </div>
    <div class="mypost" id="post-box">
        <div class="form-floating mb-3">
            <input id="url" type="email" class="form-control" placeholder="name@example.com">
            <label>영화URL</label>
        </div>
        <div class="form-floating">
            <textarea id="comment" class="form-control" placeholder="Leave a comment here"></textarea>
            <label for="floatingTextarea2">코멘트</label>
        </div>
        <div class="mybtns">
            <button onclick="posting()" type="button" class="btn btn-dark">기록하기</button>
            <button onclick="close_box()" type="button" class="btn btn-outline-dark">닫기</button>
        </div>
    </div>
    <div class="mycards">
        <div class="row row-cols-1 row-cols-md-4 g-4" id="cards-box">
            <div class="col">
                <div class="card h-100">
                    
                    <div class="card-body">
                        <h5 class="card-title">영화 제목이 들어갑니다</h5>
                        <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```
<p>***</p>
<p class="mycomment">나의 한줄 평을 씁니다</p>
</div>
</div>
<div class="col">
  <div class="card h-100">
    
    <div class="card-body">
      <h5 class="card-title">영화 제목이 들어갑니다</h5>
      <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
      <p>***</p>
      <p class="mycomment">나의 한줄 평을 씁니다</p>
    </div>
  </div>
</div>
<div class="col">
  <div class="card h-100">
    
    <div class="card-body">
      <h5 class="card-title">영화 제목이 들어갑니다</h5>
      <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
      <p>***</p>
      <p class="mycomment">나의 한줄 평을 씁니다</p>
    </div>
  </div>
</div>
<div class="col">
  <div class="card h-100">
    
    <div class="card-body">
      <h5 class="card-title">영화 제목이 들어갑니다</h5>
      <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
      <p>***</p>
      <p class="mycomment">나의 한줄 평을 씁니다</p>
    </div>
  </div>
</div>
</div>
</body>
</html>
```

15. 4주차 끝 & 숙제 설명

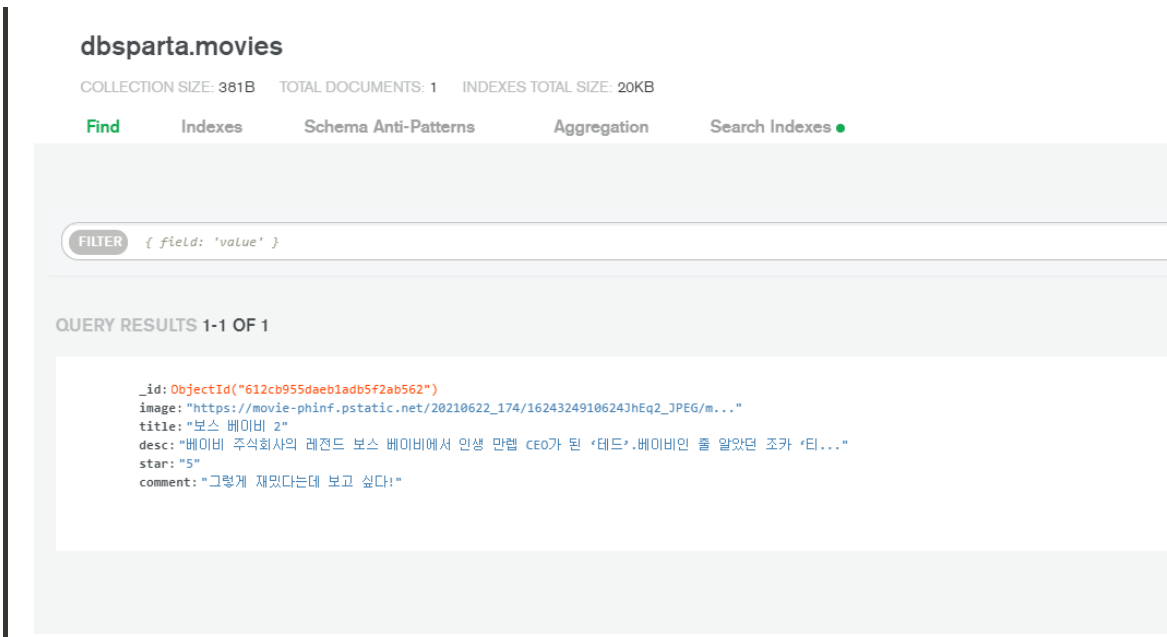
스파르타피디아에 **별점**을 넣는 기능을 추가해 봅시다!
기존 함수에 추가해봐요!

▼ [코드스니펫] index.html 별점 추가하기

```
<div class="input-group mb-3">
  <label class="input-group-text" for="inputGroupSelect01">별점</label>
  <select class="form-select" id="star">
    <option selected>- 선택하기 -</option>
    <option value="1">★</option>
    <option value="2">★★</option>
    <option value="3">★★★</option>
    <option value="4">★★★★</option>
    <option value="5">★★★★★</option>
  </select>
</div>
```

▼ 👁 이렇게 생겼어요!

별점을 주면 값을 숫자로 넣고 데이터베이스에 저장합니다! (POST)



포스트를 보여주면 별점의 숫자만큼 보여주세요!

▼ 💡 힌트!

▼ 1) 별점 값 가져오기

```
<smovielect class="form-smovielect" id="star">
  <option smovielect>-- 선택하기 --</option>
  <option value="1">★</option>
  <option value="2">★★</option>
  <option value="3">★★★</option>
  <option value="4">★★★★</option>
  <option value="5">★★★★★</option>
</smovielect>
```

- 별점마다 `value` 속성 값이 들어있네요!

▼ 2) 숫자를 별로 표시하기

- 별점이 ★ 모양으로 나와야해요! → 숫자만큼 ★ 을 반복하려면?
- 구글링 → 자바스크립트 문자 반복 → `repeat()` 를 확인해보세요..!

HW. 4주차 숙제 해설

▼ 1) 숙제 해설영상

▼ 1) POST - 별점 등록하기

▼ 클라이언트

- 기존 POST 요청에 별의 값을 가져오기
 - `let star = $('#star').val()`
- 값을 formData에 넣고 append
 - `formData.append('star_give', star)`

▼ 서버

- 값 받기
 - `star_receive = request.form['star_give']`
- 데이터베이스에 저장할 값에 star_receive 넣기

```

doc = {
    'image':image,
    'title':title,
    'desc':description,
    'star':star_receive,
    'comment':comment_receive
}

```

▼ 2) GET - 별점 보여주기

▼ 서버

- 변동사항 없음 그대로 불러오기

▼ 클라이언트

- data에서 star 값 찾기
 - `console.log(data)`
 - `let star = movie['star']`
- star 값을 별 모양으로 표시
 - `let star_image = '*'.repeat(star)`
 - `<p>${star_image}</p>`

▼ 2) 속제 답안코드

▼ [코드스니펫] app.py - 4주차 속제 답안 코드

```

from flask import Flask, render_template, request, jsonify
app = Flask(__name__)

from pymongo import MongoClient
client = MongoClient('내 mongoDB URL')
db = client.dbsparta

import requests
from bs4 import BeautifulSoup

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

@app.route("/movie", methods=["POST"])
def movie_post():
    url_receive = request.form['url_give']
    comment_receive = request.form['comment_give']
    star_receive = request.form['star_give']

    headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3661.18 Safari/537.36'}
    data = requests.get(url_receive, headers=headers)

    soup = BeautifulSoup(data.text, 'html.parser')

    ogtitle = soup.select_one('meta[property="og:title"]')['content']
    ogdesc = soup.select_one('meta[property="og:description"]')['content']
    ogimage = soup.select_one('meta[property="og:image"]')['content']

    doc = {
        'title':ogtitle,
        'desc':ogdesc,
        'image':ogimage,
        'comment':comment_receive,
        'star':star_receive
    }
    db.movies.insert_one(doc)

    return jsonify({'msg':'저장완료!'})

@app.route("/movie", methods=["GET"])
def movie_get():
    all_movies = list(db.movies.find({}, {'_id':False}))
    return jsonify({'result':all_movies})

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

```

▼ [코드스니펫] index.html - 4주차 숙제 답안 코드

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztQCwFsp3yD65VohhpuuC0mLASjC" crossorigin="anonymous">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxvXM"
    crossorigin="anonymous"></script>

<title>스파르타 피디아</title>

<link href="https://fonts.googleapis.com/css2?family=Gowun+Dodum&display=swap" rel="stylesheet">

<style>
  * {
    font-family: 'Gowun Dodum', sans-serif;
  }

  .mytitle {
    width: 100%;
    height: 250px;

    background-image: linear-gradient(0deg, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url('https://movie-phinf.pstatic');
    background-position: center;
    background-size: cover;

    color: white;

    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
  }

  .mytitle>button {
    width: 200px;
    height: 50px;

    background-color: transparent;
    color: white;

    border-radius: 50px;
    border: 1px solid white;

    margin-top: 10px;
  }

  .mytitle>button:hover {
    border: 2px solid white;
  }

  .mycomment {
    color: gray;
  }

  .mycards {
    margin: 20px auto 0px auto;
    width: 95%;
    max-width: 1200px;
  }

  .mypost {
    width: 95%;
    max-width: 500px;
    margin: 20px auto 0px auto;
    padding: 20px;
    box-shadow: 0px 0px 3px 0px gray;

    display: none;
  }

  .mybtns {
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: center;
  }

```



```

        margin-top: 20px;
    }

    .mybtns>button {
        margin-right: 10px;
    }
</style>
<script>
    $(document).ready(function () {
        listing();
    });

    function listing() {
        fetch('/movie').then((res) => res.json()).then((data) => {
            let rows = data['result']
            $('#cards-box').empty()
            rows.forEach((a) => {
                let comment = a['comment']
                let title = a['title']
                let desc = a['desc']
                let image = a['image']
                let star = a['star']

                let star_repeat = '*'.repeat(star)

                let temp_html = `<div class="col">
                    <div class="card h-100">
                        
                        <div class="card-body">
                            <h5 class="card-title">${title}</h5>
                            <p class="card-text">${desc}</p>
                            <p>${star_repeat}</p>
                            <p class="mycomment">${comment}</p>
                        </div>
                    </div>
                `
                $('#cards-box').append(temp_html)
            })
        })
    }

    function posting() {
        let url = $('#url').val()
        let comment = $('#comment').val()
        let star = $('#star').val()

        let formData = new FormData();
        formData.append("url_give", url);
        formData.append("comment_give", comment);
        formData.append("star_give", star);

        fetch('/movie', { method: "POST", body: formData }).then((res) => res.json()).then((data) => {
            alert(data['msg'])
            window.location.reload()
        })
    }

    function open_box() {
        $('#post-box').show()
    }
    function close_box() {
        $('#post-box').hide()
    }
</script>
</head>

<body>
    <div class="mytitle">
        <h1>내 생애 최고의 영화들</h1>
        <button onclick="open_box()">영화 기록하기</button>
    </div>
    <div class="mypost" id="post-box">
        <div class="form-floating mb-3">
            <input id="url" type="email" class="form-control" placeholder="name@example.com">
            <label>영화URL</label>
        </div>
        <div class="input-group mb-3">
            <label class="input-group-text" for="inputGroupSelect01">별점</label>
            <select class="form-select" id="star">
                <option selected>-- 선택하기 --</option>
                <option value="1">★</option>
                <option value="2">★★</option>
                <option value="3">★★★</option>
                <option value="4">★★★★</option>
                <option value="5">★★★★★</option>
            </select>
        </div>
    </div>

```

```

        </div>
        <div class="form-floating">
            <textarea id="comment" class="form-control" placeholder="Leave a comment here"></textarea>
            <label for="floatingTextarea2">코멘트</label>
        </div>
        <div class="mybtns">
            <button onclick="posting()" type="button" class="btn btn-dark">기록하기</button>
            <button onclick="close_box()" type="button" class="btn btn-outline-dark">닫기</button>
        </div>
    </div>
    <div class="mycards">
        <div class="row row-cols-1 row-cols-md-4 g-4" id="cards-box">
            <div class="col">
                <div class="card h-100">
                    
                    <div class="card-body">
                        <h5 class="card-title">영화 제목이 들어갑니다</h5>
                        <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                        <p>***</p>
                        <p class="mycomment">나의 한줄 평을 씁니다</p>
                    </div>
                </div>
            </div>
            <div class="col">
                <div class="card h-100">
                    
                    <div class="card-body">
                        <h5 class="card-title">영화 제목이 들어갑니다</h5>
                        <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                        <p>***</p>
                        <p class="mycomment">나의 한줄 평을 씁니다</p>
                    </div>
                </div>
            </div>
            <div class="col">
                <div class="card h-100">
                    
                    <div class="card-body">
                        <h5 class="card-title">영화 제목이 들어갑니다</h5>
                        <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                        <p>***</p>
                        <p class="mycomment">나의 한줄 평을 씁니다</p>
                    </div>
                </div>
            </div>
            <div class="col">
                <div class="card h-100">
                    
                    <div class="card-body">
                        <h5 class="card-title">영화 제목이 들어갑니다</h5>
                        <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                        <p>***</p>
                        <p class="mycomment">나의 한줄 평을 씁니다</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
</body>
</html>

```

▼ [코드스니펫] app.py - 4주차 숙제 뼈대 코드

```

from flask import Flask, render_template, request, jsonify
app = Flask(__name__)

from pymongo import MongoClient
client = MongoClient('내 URL')
db = client.dbsparta

import requests
from bs4 import BeautifulSoup

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

@app.route("/movie", methods=["POST"])
def movie_post():
    url_receive = request.form['url_give']
    comment_receive = request.form['comment_give']

    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/'
    }
    data = requests.get(url_receive, headers=headers)

    soup = BeautifulSoup(data.text, 'html.parser')

    og_image = soup.select_one('meta[property="og:image"]')
    og_title = soup.select_one('meta[property="og:title"]')
    og_description = soup.select_one('meta[property="og:description"]')

    image = og_image['content']
    title = og_title['content']
    description = og_description['content']

    doc = {
        'image':image,
        'title':title,
        'desc':description,
        'comment':comment_receive
    }

    db.movies.insert_one(doc)

    return jsonify({'msg':'리뷰 등록 완료!'})

@app.route("/movie", methods=["GET"])
def movie_get():
    movies_list = list(db.movies.find({}, {'_id':False}))
    return jsonify({'movies':movies_list})

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

```

▼ [코드스니펫] index.html - 4주차 숙제 뼈대 코드

```

<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-Mrcw6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/twTxxVtX"
        crossorigin="anonymous"></script>

    <title>스파르타 피디아</title>

    <link href="https://fonts.googleapis.com/css2?family=Gowun+Dodum&display=swap" rel="stylesheet">

    <style>
        * {
            font-family: 'Gowun Dodum', sans-serif;
        }

        .mytitle {
            width: 100%;
            height: 250px;

            background-image: linear-gradient(0deg, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url('https://movie-phinf.pstatic.net/
            background-position: center;

```

```

        background-size: cover;

        color: white;

        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
    }

    .mytitle > button {
        width: 200px;
        height: 50px;

        background-color: transparent;
        color: white;

        border-radius: 50px;
        border: 1px solid white;

        margin-top: 10px;
    }

    .mytitle > button:hover {
        border: 2px solid white;
    }

    .mycomment {
        color: gray;
    }

    .mycards {
        margin: 20px auto 0px auto;
        width: 95%;
        max-width: 1200px;
    }

    .mypost {
        width: 95%;
        max-width: 500px;
        margin: 20px auto 0px auto;
        padding: 20px;
        box-shadow: 0px 0px 3px 0px gray;

        display: none;
    }

    .mybtns {
        display: flex;
        flex-direction: row;
        align-items: center;
        justify-content: center;

        margin-top: 20px;
    }

    .mybtns > button {
        margin-right: 10px;
    }
</style>
<script>
    $(document).ready(function(){
        listing();
    });

    function listing() {
        $('#cards-box').empty()
        fetch('/movie').then(res => res.json()).then(data => {
            let movies = data['movies']
            movies.forEach(movie => {
                let title = movie['title']
                let image = movie['image']
                let desc = movie['desc']
                let comment = movie['comment']

                let temp_html = `<div class="col">
                    <div class="card h-100">
                        
                        <div class="card-body">
                            <h5 class="card-title">${title}</h5>
                            <p class="card-text">${desc}</p>
                            <p>${star_image}</p>
                            <p class="mycomment">${comment}</p>
                        </div>
                    </div>
                </div>`
            })
        })
    }
</script>

```

```

        $('#cards-box').append(temp_html)
    })
})

function posting() {
    let url = $('#url').val()
    let comment = $('#comment').val()

    let formData = new FormData()
    formData.append('url_give', url)
    formData.append('comment_give', comment)

    fetch('/movie', {method: "POST", body: formData}).then(res => res.json()).then(data => {
        console.log(data)
        alert(data['msg'])
        window.location.reload()
    })
}

function open_box(){
    $('#post-box').show()
}
function close_box(){
    $('#post-box').hide()
}
</script>
</head>

<body>
<div class="mytitle">
    <h1>내 생애 최고의 영화들</h1>
    <button onclick="open_box()">영화 기록하기</button>
</div>
<div class="mypost" id="post-box">
    <div class="form-floating mb-3">
        <input id="url" type="email" class="form-control" placeholder="name@example.com">
        <label>영화URL</label>
    </div>
    <div class="input-group mb-3">
        <label class="input-group-text" for="inputGroupSmoviect01">별점</label>
        <smoviect class="form-smoviect" id="star">
            <option smoviected="-- 선택하기 --</option>
            <option value="1">*</option>
            <option value="2">***</option>
            <option value="3">****</option>
            <option value="4">*****</option>
            <option value="5">*****</option>
        </smoviect>
    </div>
    <div class="form-floating">
        <textarea id="comment" class="form-control" placeholder="Leave a comment here"></textarea>
        <label for="floatingTextarea2">코멘트</label>
    </div>
    <div class="mybtns">
        <button onclick="posting()" type="button" class="btn btn-dark">기록하기</button>
        <button onclick="close_box()" type="button" class="btn btn-outline-dark">닫기</button>
    </div>
</div>
<div class="mycards">
    <div class="row row-cols-1 row-cols-md-4 g-4" id="cards-box">
        <div class="col">
            <div class="card h-100">
                
                <div class="card-body">
                    <h5 class="card-title">영화 제목이 들어갑니다</h5>
                    <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                    <p>***</p>
                    <p class="mycomment">나의 한줄 평을 씁니다</p>
                </div>
            </div>
        </div>
        <div class="col">
            <div class="card h-100">
                
                <div class="card-body">
                    <h5 class="card-title">영화 제목이 들어갑니다</h5>
                    <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
                    <p>***</p>
                    <p class="mycomment">나의 한줄 평을 씁니다</p>
                </div>
            </div>
        </div>
        <div class="col">
            <div class="card h-100">

```

```

        
        <div class="card-body">
            <h5 class="card-title">영화 제목이 들어갑니다</h5>
            <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
            <p>***</p>
            <p class="mycomment">나의 한줄 평을 씁니다</p>
        </div>
    </div>
</div>
<div class="col">
    <div class="card h-100">
        
        <div class="card-body">
            <h5 class="card-title">영화 제목이 들어갑니다</h5>
            <p class="card-text">여기에 영화에 대한 설명이 들어갑니다.</p>
            <p>***</p>
            <p class="mycomment">나의 한줄 평을 씁니다</p>
        </div>
    </div>
</div>
</div>
</body>

</html>

```