



스파르타코딩클럽 5주차



매 주차 강의자료 시작에 PDF파일과 영상 링크를 올려두었어요!

▼ PDF 강의자료 다운받기

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2e3658b8-eea1-44c3-b5f2-784010272bd/e/week05.pdf>

▼ 영상강의 참고하기

- 곧 업로드 될 예정입니다!



모든 토플을 열고 닫는 단축키

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

Mac : **⌘** + **option** + **t**

목차



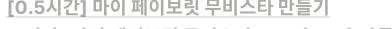
수업 목표



체크인



배우고 적용하기 - 전반부



5분꿀팁 - 사람이 이해하는 코드(코딩 컨벤션)



웹 기초 동작 원리 - 5주차

[0.5시간] 마이 페이보릿 무비스타 만들기

[1시간] 마이 페이보릿 무비스타 - API(GET) 만들어 사용하기

[1.5시간] 마이 페이보릿 무비스타 - API(POST) 만들어 사용하기



배우고 적용하기 - 후반부



5분꿀팁 - 기록하며 배우기

[1.5시간]: 프로젝트 개요 & 목표 설정

[1.5시간]: 퓨터 개별 피드백



소화 타입

숙제 설명

복습 도우미



체크아웃

[가입/설치] - 다음 시간을 위해 미리 가입/설치해와야 할 것들



수업 목표

- API 만들고 사용하기

- 프로젝트 완성시키기 - '마이 페이보릿 무비스타'(링크)

- 내 프로젝트 기획안 완성하기

전반 3시간



체크인



튜터님은 체크인과 함께 출석 체크(링크)를 진행해주세요!

스파르타코딩클럽 출석체크

<http://spartacodingclub.shop/attendance>

▼ "(오늘은) 1분 체크인"을 진행합니다 - 프로젝트 생김새(레이아웃) 확인!

- 퓨터는 타이머를 띄워주세요! ([링크](#)).
- 각자 30초 씩 돌아가며 프로젝트의 개요를 설명합니다.
- 퓨터는 발표가 끝날 때마다 30초 내외로 피드백을 전달합니다.
(범위, 참고해보면 좋은 서비스)

💡 배우고 적용하기 - 전반부

📚 5분 꿀팁 - 사람이 이해하는 코드(코딩 컨벤션)

▼ 코딩 컨벤션 - 사람이 이해할 수 있는 코드를 위한

컴퓨터가 이해할 수 있는 코드는 어느 바보나 다 짤 수 있다.

좋은 프로그래머는 사람이 이해할 수 있는 코드를 짠다.

Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

— Martin Fowler, [『Refactoring』](#), Addison-Wesley Professional June(1999), p35

❓ 코딩 컨벤션(Coding Convention) 이 무엇인가요?

❗ 코딩 스타일을 포함한 규약(조직체 안에서, 서로 지키도록 협의하여 정하여 놓은 규칙)입니다. 코딩 컨벤션은 사람을 위한 약속이라고 할 수 있어요. 사람의 이해를 돋고 읽고 관리하기 쉬운 코드를 만들기 위해서 존재합니다.

맞춤법에 맞는 글은 훨씬 보기 좋죠? 코드도 마찬가지입니다. **코드가 일관성 있는 스타일을 가지고 있다면, 읽기도 좋고 코드의 의도를 파악하기도 편합니다.** 가독성이 좋아져 버그를 발생시킬 수 있는 코드를 발견할 확률도 높아지죠. 더불어 다른 사람들과 협업하기에도 좋습니다.

프로그래밍 언어마다, 회사마다 코딩 컨벤션을 가지고 있어요. 우리가 JS를 배울 때에 사용한 camelCase(단어가 연결되는 첫 글자를 대문자로 함)하고, Python에서는 snake_case(_ 로 단어 연결)를 사용한 것도 코딩 컨벤션이죠.

▼ Javascript 코딩 컨벤션들

- 코딩 스타일 소개

코딩 스타일

개발자는 가능한 한 간결하고 읽기 쉽게 코드를 작성해야 합니다. 복잡한 문제를 간결하고 사람이 읽기 쉬운 코드로 작성해 해결하는 것이야말로 진정한 프로그래밍 기술입니다. 좋은 코드 스타일은 이런 기술을 연마하는 데 큰 도움을 줍니다. 몇 가지 추천할만한 규칙을 아래 치트 시트에 표시해보았습니다(자세한 설명 [여기](#)).

👉 <https://ko.javascript.info/coding-style>



JAVASCRIPT.INFO
The Modern JavaScript Tutorial

- AirBnB

airbnb/javascript

JavaScript Style Guide. Contribute to airbnb/javascript development by creating an account on GitHub.

👉 <https://github.com/airbnb/javascript>



- NHN

코딩컨벤션

코딩 컨벤션은 읽고, 관리하기 쉬운 코드를 작성하기 위한 일종의 코딩 스타일 규약이다. 특히 자바스크립트는 다른 언어에 비해 유연한 문법구조(동적 타입, this 바인딩, 네이티브 객체 조작 가능)를 가지기 때문에 개발자 간 통일된 규약이 없다면 코드의 의도를 파악하거나 오류를 찾기 어렵다. 코딩 컨벤션을 준수하는 경우 더 좋은 코드를 작성할 수 있다.

👉 https://ui.toast.com/fe-guide/ko_CODING-CONVENTION/

Use TOAST UI to Make Your Web Delicious!



- 이 외에도 `JS coding convention`이라는 키워드로 검색하면 많은 자료를 보실 수 있어요.

▼ Python 코딩 컨벤션

- PEP8 - Python 공식 스타일 가이드

PEP 8 -- Style Guide for Python Code

The official home of the Python Programming Language

👉 <https://www.python.org/dev/peps/pep-0008/>



- Google

google/styleguide

Table of Contents Python is the main dynamic language used at Google. This style guide is a list of dos and don'ts for Python programs. To help you format code correctly, we've created a settings file for Vim. For Emacs, the default settings should be fine.

👉 <https://github.com/google/styleguide/blob/gh-pages/pyguide.md>



- `Python coding convention`이라는 키워드로 검색하면 더 많은 자료를 볼 수 있습니다.
- 😊 지금 단계에서는 처음부터 코딩 컨벤션을 다 내 코드에 적용하려는 부담을 내려놓으세요. 이런 것들이 있구나~ 하고 키워드만 기억해놓으셔도 충분합니다!

【 한 걸음 더】

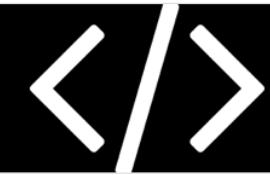
- 앞으로 할 내 프로젝트에 수업시간에 배운 이름 짓기(naming) 컨벤션만 적용해보기
- ▼ 앞으로 개발자 커리어에 관심이 있는 사람은 아래 키워드도 한 번 확인해보세요.

- 키워드 : `linter`, `클린 코드(Clean Code)`

linter를 이용한 코딩스타일과 에러 체크하기

홀쇼핑처럼 6번째 상품인 까르보돈까스는 부여과 짹먹을 선택해야 합니다. 코딩도 둘중에 하나를 선택해야 하는 경우가 많은데 협업을 위해 코딩 스타일을 설정하고 규칙에 어긋난 코드를 한땀한땀 수정했던 순간이 떠올라 linter에 대해 이야기합니다. 코딩스타일, 코딩 표준이라고도 불리는 코딩 컨벤션은 코드를 작성하는 경우에 매우 중요한 역할을 합니다.

👉 <https://subicura.com/2016/07/11/coding-convention.html>



Clean Code: Naming

Names are the smallest building block of code. Names are everywhere: Function, class, file, component, and container all have names. Names are also the basic building blocks of clean code. There are some hard and fast rules to a good name. A good name should answer only

👉 <https://medium.com/better-programming/clean-code-naming-b90740cbea12>



Clean code is simple and direct. Clean code reads like well-written prose.

— Grady Booch, author of *Object-Oriented Analysis and Design with Applications*

웹 기초 동작 원리 - 5주차

▼ 지난주 복습 - 4주차 - API, Flask, GET / POST 방식 요청

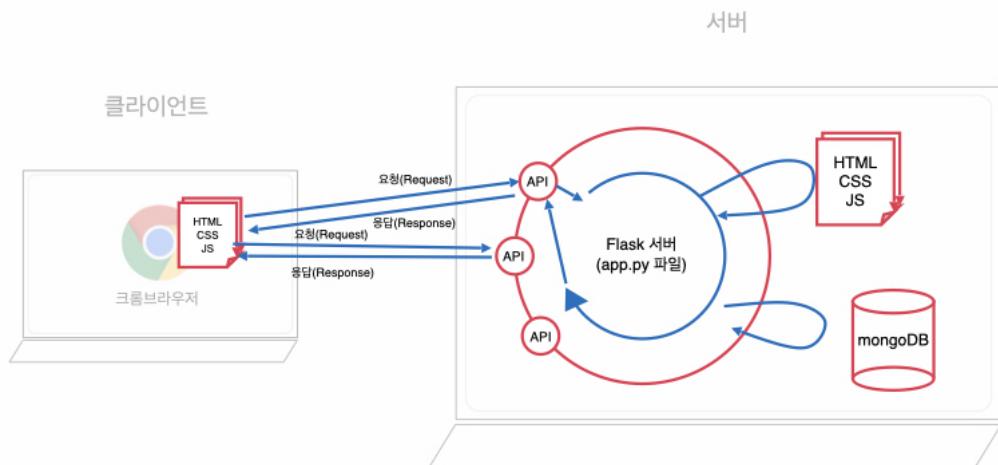
- API(Application Programming Interface): 응용 프로그램(application, 애플리케이션)에서 기능을 사용하거나 데이터를 주고 받기 위한 기능

- 우리가 사용하는 API는 아래 적힌 모든 것을 미리 약속해두고 그대로 동작합니다.
 - 1. 요청 정보 :** 요청 URL, 요청 방식 (GET / POST ...)
 - 2. 서버가 제공할 기능 :** 데이터 조회(Read), 데이터 생성(Create) 등
 - 3. 응답 데이터 :** 응답 데이터 형식 (어떤 key로 어떤 데이터를 줄지, 예. response['img'])
- Flask : 웹을 만들고 서버를 구동시키기 편하게 하는 프레임워크(Flask 공식 문서 / 비공식 한글 번역문서)
 - `@app.route('/')` 부분을 수정해서 URL을 부여할 수 있습니다
 - **templates 폴더** : HTML 파일을 담아둡니다. 실행할 때에는 이 폴더에서 화면을 불러오죠.
 - **static 폴더** : 이미지나 css파일과 같은 정적 파일을 담아두는 역할을 하지요!
- 클라이언트 요청 방식 - GET, POST
 - GET → 통상적으로! 데이터 조회(Read)를 요청할 때
예) 영화 목록 조회
→ **데이터 전달** : URL 뒤에 물음표를 붙여 key=value로 전달
→ 예: google.com?q=북극곰
 - POST → 통상적으로! 데이터 생성(Create), 변경(Update), 삭제(Delete) 요청 할 때
예) 회원가입, 회원탈퇴, 비밀번호 수정
→ **데이터 전달** : 바로 보이지 않는 HTML body에 key:value 형태로 전달



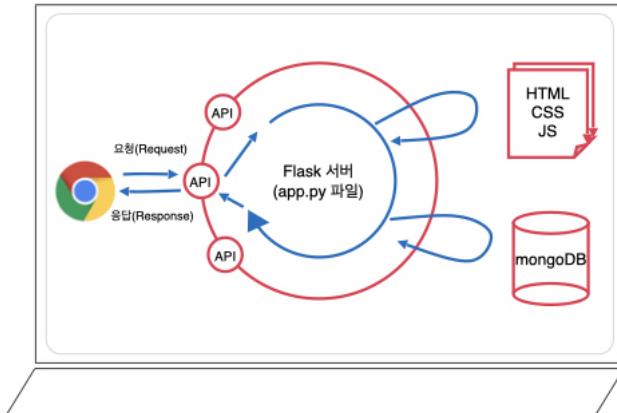
오늘은 머리 속 퍼즐을 맞추기 위해, 저번주와 비슷한 난이도의 프로젝트를 실습합니다.

여기까지 배웠다면, 이제 6~8주차 프로젝트 준비 완료!!



우리 프로젝트는 클라이언트(요청하는 쪽)과 서버(응답하는 쪽)이 한 컴퓨터에 있는 **로컬 개발환경**으로 구성이 되어있습니다!

다음주에 서버와 클라이언트가 다른 컴퓨터에 있는 실제 서비스 환경처럼 프로젝트를 진행해볼 거에요.



[0.5시간] 마이 페이보릿 무비스타 만들기



API에 익숙해지기 위해서 다시 한 번 차근차근 만들어 나가볼거에요.
sparta > projects > moviestar 가 프로젝트 폴더!

▼ 1) 문제 분석 - 완성작 → 만들 API 분석

👉 완성작 보러가기

- 자유롭게 사용하며 어떤 기능이 있는지 관찰해봅시다.
- 기능
 1. 영화인 정보를 카드로 보여주기(Read)
 2. 좋아요 기능(Update)
 - 영화인 카드의 '위로!' 버튼을 누르면, 좋아요 옆 숫자가 추가되고,
 - 좋아요가 많은 순서대로 영화인 카드가 위에서부터 정렬되기
 3. 삭제하기(Delete)
 - 영화인 카드의 '삭제'버튼을 누르면 카드가 삭제됨



우리가 만들 API를 정리하면 아래와 같습니다.

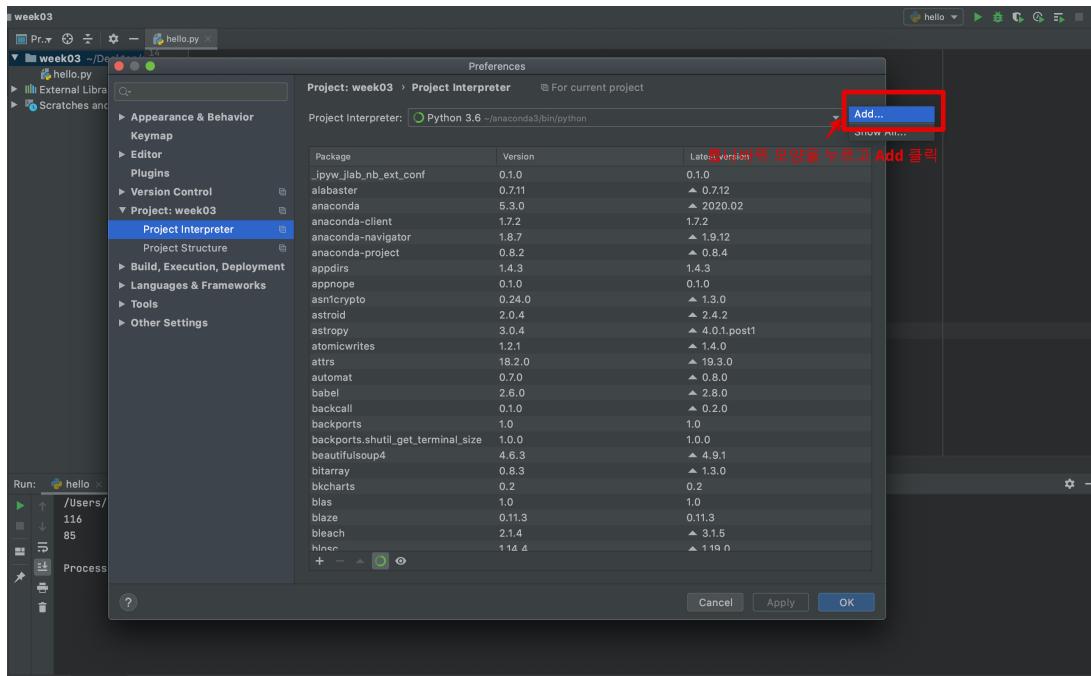
1. 조회(Read) 기능: 영화인 정보 전체를 조회
2. 좋아요(Update) 기능: 클라이언트에서 받은 이름(name_give)으로 찾아서 좋아요(like)를 증가
3. 삭제(Delete) 기능: 클라이언트에서 받은 이름(name_give)으로 영화인을 찾고, 해당 영화인을 삭제

▼ 2) 프로젝트 설정 - 새 프로젝트니까, 가상환경 설정하기

- Pycharm에서 File - Open에서 **sparta/projects/moviestar** 폴더를 선택

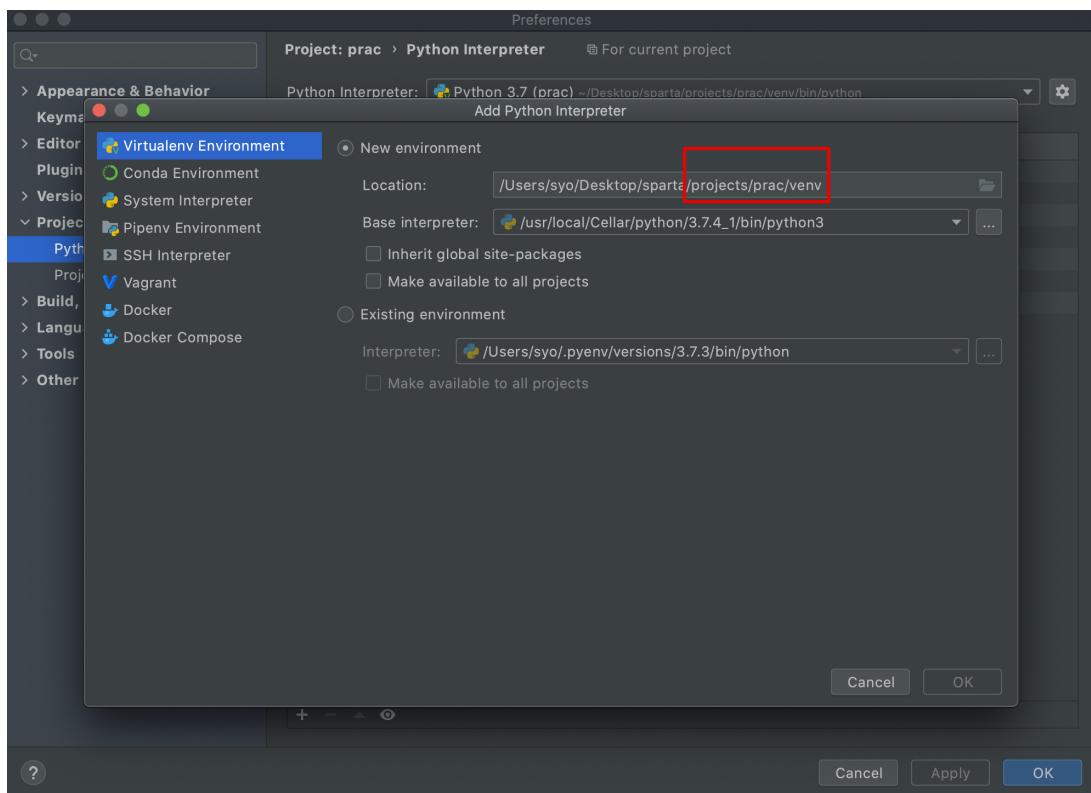
▼ 1. 가상 환경 설치 및 확인

- Windows : File → setting → project interpreter
- Mac : Preference → Project Interpreter로 접근
 - 텁니바퀴 add 버튼 클릭



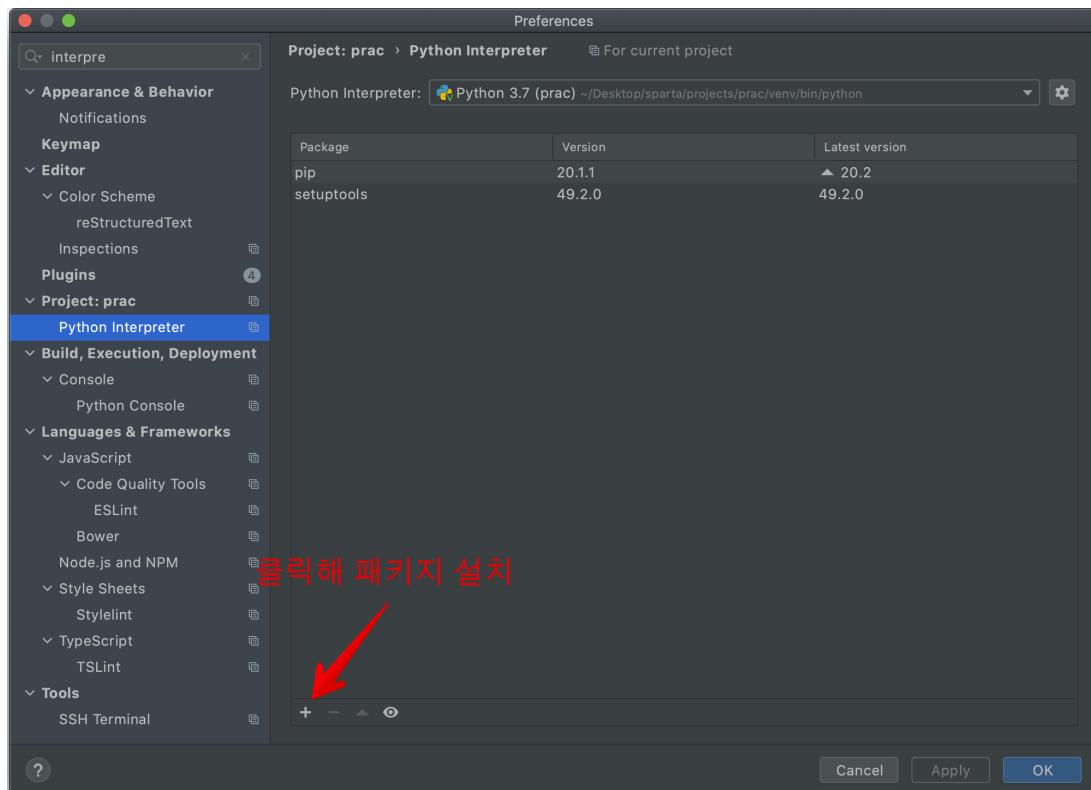
- virtual environment - New environment 선택

- Location을 현재 폴더/venv로 설정해주세요. 현재 프로젝트 폴더(prac) 아래에 가상환경 폴더(venv)가 생성이 됩니다. 우리는 projects/moviestar/venv로 보이겠죠?

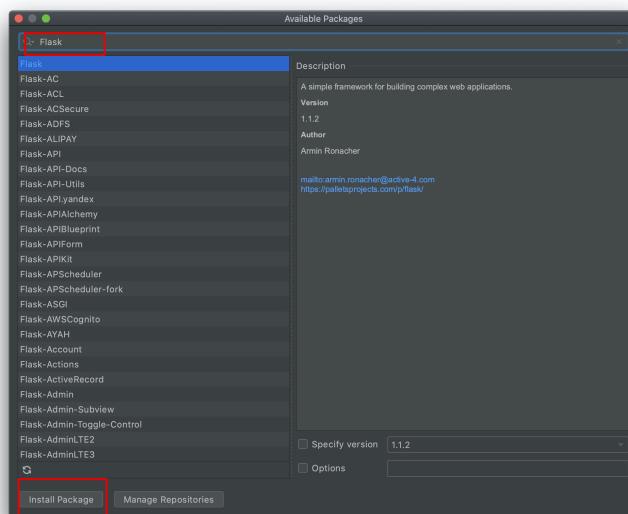


▼ 2. pip(python install package) 사용 - Flask 패키지 설치해보기

- Flask를 사용하기 위해서 패키지를 설치해보겠습니다.
- project interpreter 화면에서 + 버튼을 누르면 아래 창이 뜹니다!



- `flask` 로 검색한 후, Install package 클릭



- 위 방법처럼 `pymongo` 와 `requests`, `beautifulsoup4` 도 패키지 설치 해주세요!

▼ 3. Flask 기본 폴더 구조 만들기



리마인드!

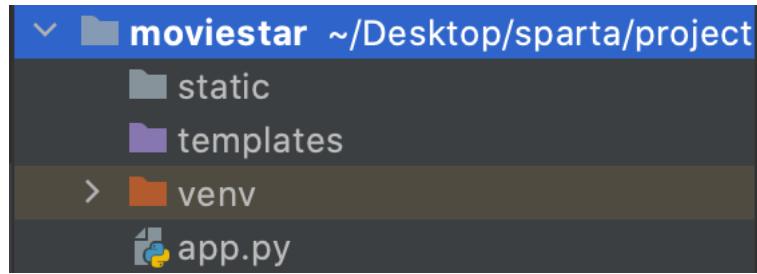
Flask 를 사용할 때는 기본 설정을 지키기! 프로젝트 폴더 안에, 폴더를 만들어주세요!

static 폴더 : 이미지, css파일을 넣어둡니다.

templates 폴더 : html파일을 넣어둡니다.

app.py 파일

- 아래 모양처럼 만들어주세요!



- mongoDB 사용할거니, mongoDB가 켜져 있는지 확인해보세요..
 - localhost:27017 접속하면 `It looks like you are trying to access MongoDB over HTTP on the native driver port.` 메시지가 뜨는지 확인!
- ▼ 3) 프로젝트 준비 - 프로젝트에서 사용할 데이터 넣기(웹 스크래핑)



API를 설계하고 만드는 것에 집중할 수 있게,

1) 사용할 데이터를 웹 스크래핑해서, 2) 데이터베이스에 저장하는 코드를 미리 작성해두었어요.

- **moviestar** 폴더 안에 `init_db.py` 파일을 만들어 아래 코드를 복사-붙여넣기해주세요.

▼ [코드]

```

import requests
from bs4 import BeautifulSoup

from pymongo import MongoClient

client = MongoClient('localhost', 27017)
db = client.dbsparta

# DB에 저장할 영화인들의 출처 url을 가져옵니다.
def get_urls():
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.'
    }
    data = requests.get('https://movie.naver.com/movie/sdb/rank/rpeople.nhn', headers=headers)

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

    trs = soup.select('#old_content > table > tbody > tr')

    urls = []
    for tr in trs:
        a = tr.select_one('td.title > a')
        if a is not None:
            base_url = 'https://movie.naver.com/'
            url = base_url + a['href']
            urls.append(url)

    return urls

# 출처 url로부터 영화인들의 사진, 이름, 최근작 정보를 가져오고 mystar 컬렉션에 저장합니다.
def insert_star(url):
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.'
    }
    data = requests.get(url, headers=headers)

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

    name = soup.select_one('#content > div.article > div.mv_info_area > div.mv_info.character > h3 > a').text
    img_url = soup.select_one('#content > div.article > div.mv_info_area > div.poster > img')['src']
    recent_work = soup.select_one(
        '#content > div.article > div.mv_info_area > div.mv_info.character > dl > dd > a:nth-child(1)').text

    doc = {
        'name': name,
        'img_url': img_url,
        'recent': recent_work,
        'url': url,
        'like': 0
    }

    db.mystar.insert_one(doc)

```

```

        }

        db.mystar.insert_one(doc)
        print('완료!', name)

        # 기존 mystar 콜렉션을 삭제하고, 출처 url들을 가져온 후, 크롤링하여 DB에 저장합니다.
    def insert_all():
        db.mystar.drop() # mystar 콜렉션을 모두 지워줍니다.
        urls = get_urls()
        for url in urls:
            insert_star(url)

        ### 실행하기
    insert_all()

```

- `init_db.py` 파일을 실행하면 내 mongoDB의 `mystar` collection에 영화인 정보가 저장됩니다.
- mystar 콜렉션에 데이터가 저장된 모습은 다음과 같습니다.

| | _id | name | img_url | recent | url | like |
|---|---------------------|--------|----------------------------------|-----------------|----------------------------------|------|
| 1 | ObjectId("5e686e... | 김다미 | https://search.pstatic.net/co... | 안녕, 나의 소울메이트... | https://movie.naver.com//movi... | 0 |
| 2 | ObjectId("5e686e... | 봉준호 | https://search.pstatic.net/co... | 기생충 | https://movie.naver.com//movi... | 0 |
| 3 | ObjectId("5e686e... | 연상호 | https://search.pstatic.net/co... | 반도 | https://movie.naver.com//movi... | 0 |
| 4 | ObjectId("5e686e... | 신현빈 | https://search.pstatic.net/co... | 클로젯 | https://movie.naver.com//movi... | 0 |
| 5 | ObjectId("5e686e... | 마고 로비 | https://search.pstatic.net/co... | 더 수어사이드 스쿼드 | https://movie.naver.com//movi... | 0 |
| 6 | ObjectId("5e686e... | 황우슬혜 | https://search.pstatic.net/co... | 히트맨 | https://movie.naver.com//movi... | 0 |
| 7 | ObjectId("5e686e... | 샘 멘데스 | https://search.pstatic.net/co... | 1917 | https://movie.naver.com//movi... | 0 |
| 8 | ObjectId("5e686e... | 홍기준 | https://search.pstatic.net/co... | 신의 한 수: 귀수편 | https://movie.naver.com//movi... | 0 |
| 9 | ObjectId("5e686e... | 시얼사 로넌 | https://search.pstatic.net/co... | 프렌치 디스패치 | https://movie.naver.com//movi... | 0 |

▼ 4) 프로젝트 준비 - `index.html`, `app.py` 준비하기

moviestar > templates 폴더 안 index.html에 다음 내용을 복사해 덮어씁니다.

▼ [코드 - `index.html`]

```

<!DOCTYPE html>
<html lang="ko">
    <head>
        <meta charset="UTF-8"/>
        <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
        <title>마이 페이보릿 무비스타 | 프론트-백엔드 연결 마지막 예제!</title>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.8.0/css/bulma.min.css"/>
        <script defer src="https://use.fontawesome.com/releases/v5.3.1/js/all.js"></script>
        <style>
            .center {
                text-align: center;
            }

            .star-list {
                width: 500px;
                margin: 20px auto 0 auto;
            }

            .star-name {
                display: inline-block;
            }

            .star-name:hover {
                text-decoration: underline;
            }

            .card {
                margin-bottom: 15px;
            }
        </style>

```

```

<script>
    $(document).ready(function () {
        // index.html 로드가 완료되면 자동으로 showStar() 함수를 호출합니다.
        showStar();
    });

    function showStar() {
        $.ajax({
            type: 'GET',
            url: '/api/list',
            data: {},
            success: function (response) {
                if (response['result'] == 'success') {
                    let msg = response['msg'];
                    alert(msg);
                }
            }
        });
    }

    function likeStar(name) {
        $.ajax({
            type: 'POST',
            url: '/api/like',
            data: {},
            success: function (response) {
                if (response['result'] == 'success') {
                    let msg = response['msg'];
                    alert(msg);
                }
            }
        });
    }

    function deleteStar(name) {
        $.ajax({
            type: 'POST',
            url: '/api/delete',
            data: {},
            success: function (response) {
                if (response['result'] == 'success') {
                    let msg = response['msg'];
                    alert(msg);
                }
            }
        });
    }

    </script>
</head>
<body>
    <section class="hero is-warning">
        <div class="hero-body">
            <div class="container center">
                <h1 class="title">
                    마이 페이보릿 무비스타!
                </h1>
                <h2 class="subtitle">
                    순위를 매겨봅시다
                </h2>
            </div>
        </div>
    </section>
    <div class="star-list" id="star-box">
        <div class="card">
            <div class="card-content">
                <div class="media">
                    <div class="media-left">
                        <figure class="image is-48x48">
                            
                        </figure>
                    </div>
                    <div class="media-content">
                        <a href="#" target="_blank" class="star-name title is-4">김다미 (좋아요: 3)</a>
                        <p class="subtitle is-6">안녕, 나의 소울메이트(가제)</p>
                    </div>
                </div>
            </div>
            <div class="card-footer">
                <a href="#" onclick="likeStar('김다미')" class="card-footer-item has-text-info">
                    위로!
                <span class="icon">
                    <i class="fas fa-thumbs-up"></i>
                </span>
            </div>
        </div>
    </div>

```

```

        </a>
        <a href="#" onclick="deleteStar('김다미')" class="card-footer-item has-text-danger">
            삭제
            <span class="icon">
                <i class="fas fa-ban"></i>
            </span>
        </a>
    </div>
</div>
</body>
</html>

```



moviestar 폴더 안 `app.py`에 다음 내용을 복사해 덮어씌웁니다.

▼ [💻 코드 - `app.py`]

```

from pymongo import MongoClient

from flask import Flask, render_template, jsonify, request

app = Flask(__name__)

client = MongoClient('localhost', 27017)
db = client.dbsparta

# HTML 화면 보여주기
@app.route('/')
def home():
    return render_template('index.html')

# API 역할을 하는 부분
@app.route('/api/list', methods=['GET'])
def show_stars():
    # 1. db에서 mystar 목록 전체를 검색합니다. ID는 제외하고 like 가 많은 순으로 정렬합니다.
    # 참고) find({},{ '_id':False}), sort()를 활용하면 굿!
    # 2. 성공하면 success 메시지와 함께 stars_list 목록을 클라이언트에 전달합니다.
    return jsonify({'result': 'success', 'msg': 'list 연결되었습니다!'})

@app.route('/api/like', methods=['POST'])
def like_star():
    # 1. 클라이언트가 전달한 name_give를 name_receive 변수에 넣습니다.
    # 2. mystar 목록에서 find_one으로 name이 name_receive와 일치하는 star를 찾습니다.
    # 3. star의 like 에 1을 더해준 new_like 변수를 만듭니다.
    # 4. mystar 목록에서 name이 name_receive인 문서의 like 를 new_like로 변경합니다.
    # 참고: '$set' 활용하기!
    # 5. 성공하면 success 메시지를 반환합니다.
    return jsonify({'result': 'success', 'msg': 'like 연결되었습니다!'})

@app.route('/api/delete', methods=['POST'])
def delete_star():
    # 1. 클라이언트가 전달한 name_give를 name_receive 변수에 넣습니다.
    # 2. mystar 목록에서 delete_one으로 name이 name_receive와 일치하는 star를 제거합니다.
    # 3. 성공하면 success 메시지를 반환합니다.
    return jsonify({'result': 'success', 'msg': 'delete 연결되었습니다!'})

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

```



동작 테스트

`app.py`를 실행시켜 서버를 켜고 화면을 띄웠을 때, 'list 연결되었습니다.'라는 메시지가 뜨면 동작하는 것입니다.

[1시간] 마이 페이보릿 무비스타 - API(GET) 만들어 사용하기

▼ 5) 문제 분석 - 화면과 동작 살펴보기

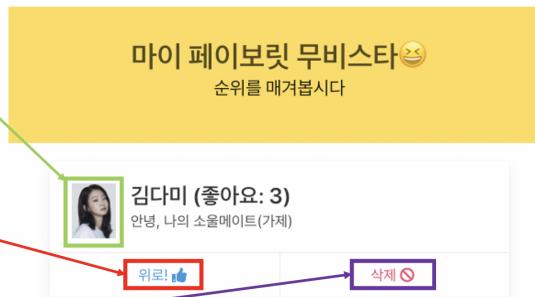
- 우리가 만들 기능은 영화인 정보를 카드로 보여주기(**Read**)입니다.
- 화면에 어떤 데이터가 어떤 부분에 보여지는지 영화인 카드 화면 코드를 보며 분석해보겠습니다.

- 영화인 이름
- 영화인 이미지 : 이미지 src 속성
- 좋아요 개수
- 최근 작품 내용이 들어가는 부분



`index.html` 을 크롬에서 실행시켜 크롬 개발자도구 - 검사하기(Inspector)로 어떤 요소에 어떤 데이터가 보일지 분석해보세요.

```
<div class="card">
  <div class="card-content">
    <div class="media">
      <div class="media-left">
        <figure class="image is-48x48">
          
        </figure>
      </div>
      <div class="media-content">
        <a href="#" target="_blank" class="star-name title is-4" href="#">김다미 (좋아요: 3)
        <p class="subtitle is-6">안녕, 나의 소울메이트(가제)</p>
      </div>
    </div>
    <footer class="card-footer">
      <a href="#" onclick="like_star('김다미')" class="card-footer-item has-text-info">
        위로!
        <span class="icon">
          <i class="fas fa-thumbs-up"></i>
        </span>
      </a>
      <a href="#" onclick="delete_star('김다미!')" class="card-footer-item has-text-danger">
        삭제
        <span class="icon">
          <i class="fas fa-ban"></i>
        </span>
      </a>
    </footer>
  </div>
```



김다미 (좋아요: 3)

순위를 매겨봅시다

김다미 (좋아요: 3)
안녕, 나의 소울메이트(가제)

위로! 🌟

삭제 ❌

```
<!-- 다음 코드가 하나의 카드를 이루는 div 입니다. -->
<div class="card">
  <div class="card-content">
    <div class="media">
      <div class="media-left">
        <figure class="image is-48x48">
          
        </figure>
      </div>
      <div class="media-content">
        <a href="https://movie.naver.com//movie/bi/pi/basic.nhn?st=1&code=397373" target="_blank" class="star-name title is-4" href="#">김다미
        <p class="subtitle is-6">안녕, 나의 소울메이트(가제)</p>
      </div>
    </div>
    <footer class="card-footer">
      <a href="#" onclick="likeStar('김다미')" class="card-footer-item has-text-info">
        위로!
        <span class="icon">
          <i class="fas fa-thumbs-up"></i>
        </span>
      </a>
      <a href="#" onclick="deleteStar('김다미!')" class="card-footer-item has-text-danger">
        삭제
        <span class="icon">
          <i class="fas fa-ban"></i>
        </span>
      </a>
    </footer>
  </div>
```

▼ 6) API 만들고 사용하기 - 영화인 조회 API (Read → GET)



만들 API

1. 조회(Read) 기능: 영화인 정보 전체를 조회
2. 좋아요(Update) 기능: 클라이언트에서 받은 이름(name_give)으로 찾아서 좋아요(like)를 증가
3. 삭제(Delete) 기능: 클라이언트에서 받은 이름(name_give)으로 영화인을 찾고, 해당 영화인을 삭제



정리하면, 만들 API 정보는 아래와 같습니다.

A. 요청 정보

- 요청 URL = `/api/list`, 요청 방식 = `GET`
- 요청 데이터 : 없음

B. 서버가 제공할 기능 : 데이터베이스에 영화인 정보를 조회(Read)하고, 영화인 정보를 응답 데이터로 보냄

C. 응답 데이터 : (JSON 형식) 'result'='success', 'stars_list'= 영화인 정보 리스트

▼ 1. 클라이언트와 서버 연결 확인하기

- 여기서는 미리 적혀 있는 쌍으로 되어있는 서버-클라이언트 코드를 확인하고 갈게요.
- 분홍 형광펜 부분이 서로 어떻게 매칭되는지 확인해보세요!

[서버 코드 - `app.py`]

```
@app.route('/api/list', methods=['GET'])
def show_stars():
    # 1. db에서 mystar 목록 전체를 검색합니다. ID는 제외하고 like 가 많은 순으로 정렬합니다.
    # 참고) find({}, {'_id': False}), sort()를 활용하면 굿!
    # 2. 성공하면 success 메시지와 함께 stars_list 목록을 클라이언트에 전달합니다.
    return jsonify({'result': 'success', 'msg': 'list 연결되었습니다!'})
```

[클라이언트 코드 - `index.html`]

```
function showStar(){
    // 1. #star_box의 내부 html 태그를 모두 삭제합니다.
    // 2. 서버에 1) GET 방식으로, 2) /api/list 라는 주소로 stars_list를 요청합니다.
    // 3. 서버가 돌려준 stars_list를 stars라는 변수에 저장합니다.
    // 4. for 문을 활용하여 stars 배열의 요소를 차례대로 조회합니다.
    // 5. stars[i] 요소의 name, url, img_url, recent_work, like 키 값을 활용하여 값을 조회합니다.
    // 6. 영화인 카드를 만듭니다.
    // 7. #star-box에 tempHtml을 붙입니다.
    $.ajax({
        type: 'GET',
        url: '/api/list',
        data: {},
        success: function (response) {
            let msg = response['msg'];
            alert(msg);
        }
    });
}
```



동작 테스트

새로고침했을 때, 'list 연결되었습니다.'라는 메시지가 뜨면 동작하는 것입니다.

▼ 2. 서버부터 만들기



API 는 약속이라고 했습니다. 위에 미리 설계해 둔 API 정보를 보고 만들어보죠!

영화인 정보 전체를 조회하기 위해 서버가 받을 정보는 없습니다. 조건없이 모든 정보를 보여줄 것이니까요!

따라서 서버 로직은 다음 단계로 구성되어야 합니다.

1. mystar 목록 전체를 검색합니다. ID는 제외하고 like 가 많은 순으로 정렬합니다.
2. 성공하면 success 메시지와 함께 stars_list 목록을 클라이언트에 전달합니다.

```
@app.route('/api/list', methods=['GET'])
def stars_list():
    # 1. db에서 mystar 목록 전체를 검색합니다. ID는 제외하고 like 가 많은 순으로 정렬합니다.
    # 참고) find({},{ '_id':False}), sort()를 활용하면 굿!
    stars = list(db.mystar.find({}, {'_id': False}).sort('like', -1))
    # 2. 성공하면 success 메시지와 함께 stars_list 목록을 클라이언트에 전달합니다.
    return jsonify({'result': 'success', 'stars_list': stars})
```

▼ 3. 클라이언트 만들기



API 는 약속이라고 했습니다. API를 사용할 클라이언트를 만들어보죠!

영화인 정보 전체를 조회하기 위해 서버가 받을 정보는 없습니다. 조건없이 모든 정보를 보여줄 것이니까요!

따라서 클라이언트 로직은 다음 단계로 구성되어야 합니다.

1. #star_box의 내부 html 태그를 모두 삭제합니다.
2. 서버에 1) GET 방식으로, 2) /api/list 라는 주소로 stars_list를 요청합니다.
3. 서버가 돌려준 stars_list를 stars라는 변수에 저장합니다.
4. for 문을 활용하여 stars 배열의 요소를 차례대로 조회합니다.
5. stars[i] 요소의 name, url, img_url, recent, like 키 값을 활용하여 값 조회합니다.
6. 영화인 카드 코드 만들어 #star-box에 붙이기

```
function showStar() {
    // 1. #star_box의 내부 html 태그를 모두 삭제합니다.
    $('#star-box').empty()

    // 2. 서버에 1) GET 방식으로, 2) /api/list 라는 주소로 star_list를 요청합니다.
    $.ajax({
        type: "GET",
        url: "/api/list",
        data: {},
        success: function (response) {
            // 3. 서버가 돌려준 star_list를 star라는 변수에 저장합니다.
            let stars = response['stars_list']
            // 4. for 문을 활용하여 star 배열의 요소를 차례대로 조회합니다.
            for (let i = 0; i < stars.length; i++) {
                let star = stars[i]
                // 5. star[i] 요소의 name, url, img_url, recent, like 키 값을 활용하여 값을 조회합니다.
                let name = star['name']
                let url = star['url']
                let imgUrl = star['img_url']
                let recentWork = star['recent']
                let like = star['like']

                // 6. 영화인 카드를 만듭니다.
                let tempHtml = `<div class="card">
                    <div class="card-content">
                        <div class="media">
                            <div class="media-left">
                                <figure class="image is-48x48">
                                    
                                </figure>
                            </div>
                            <div class="media-content">
                                <a href="${url}" target="_blank" class="star-name title is-4">${name} (좋아요: ${like})</a>
                                <p class="subtitle is-6">${recentWork}</p>
                            </div>
                        </div>
                    </div>
                `
```

```
        </div>
        <footer class="card-footer">
            <a href="#" onclick="likeStar('${name}')" class="card-footer-item has-text-info">
                위로!
                <span class="icon">
                    <i class="fas fa-thumbs-up"></i>
                </span>
            </a>
            <a href="#" onclick="deleteStar('${name}')" class="card-footer-item has-text-danger">
                삭제
                <span class="icon">
                    <i class="fas fa-ban"></i>
                </span>
            </a>
        </footer>
    </div>

    // 7. #star-box에 temp_html을 불입니다.
    $('#star-box').append(tempHtml)
}
});
```

▼ 4. 완성 확인하기



동작 테스트

화면을 새로고침 했을 때 영화인 정보가 조회되는지 확인합니다.

[1.5시간] 마이 페이보릿 무비스타 - API(POST) 만들어 사용하기

▼ 7) 문제 분석 - 화면과 동작 살펴보기



만들 API

1. 조회(**Read**) 기능: 영화인 정보 전체를 조회
 2. 좋아요(**Update**) 기능: 클라이언트에서 받은 이름(name_give)으로 찾아서 좋아요(like)를 증가
 3. 삭제(**Delete**) 기능: 클라이언트에서 받은 이름(name_give)으로 영화인을 찾고, 해당 영화인을 삭제

- 어떤 동작과 어떤 화면 요소가 연결되는지 분석해보겠습니다.
 - 좋아요 기능: `likeStar` 함수, 위로 버튼
 - 삭제 기능: `deleteStar` 함수, 삭제 버튼



`index.html` 을 크롬에서 실행시켜 크롬 개발자도구 - 검사하기(Inspector)로 어떤 요소와 어떤 동작을 연결할지 분석해보세요.

```
<div class="card">
  <div class="card-content">
    <div class="media">
      <div class="media-left">
        <figure class="image is-48x48">
          
        </figure>
      </div>
      <div class="media-content">
        <a href="#" target="_blank" class="star-name title is-4" onclick="like_star('김다미')">김다미 (좋아요: 3)</a>
        <p class="subtitle is-6">한국, 나의 소울메이트(가족)</p>
      </div>
    </div>
  </div>
</div>
<div class="card-footer">
  <a href="#" onclick="like_star('김다미')" class="card-footer-item has-text-info">
    위로!
    <span class="icon">
      <i class="fas fa-thumbs-up"></i>
    </span>
  </a>
  <a href="#" onclick="delete_star('김다미')" class="card-footer-item has-text-danger">
    삭제
    <span class="icon">
      <i class="fas fa-ban"></i>
    </span>
  </a>
</div>
</div>
```

마이 페이보릿 무비스타 😊

순위를 매겨봅시다



1 길다미 (좋아요·3)

김나미 (홍야요: 3)



10

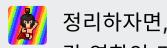


스파르타코딩클럽 5주차

```

<!-- 다음 코드가 하나의 카드를 이루는 div 입니다. -->
<div class="card">
  <div class="card-content">
    <div class="media">
      <div class="media-left">
        <figure class="image is-48x48">
          
        </figure>
      </div>
      <div class="media-content">
        <a href="https://movie.naver.com/move/bi/pi/basic.nhn?st=1&code=397373" target="_blank" class="star-name title is
          <p class="subtitle is-6">안녕, 나의 소울메이트(가제)</p>
        </div>
      </div>
    </div>
    <div class="card-footer">
      <a href="#" onclick="likeStar('김다미')" class="card-footer-item has-text-info">
        위로!
        <span class="icon">
          <i class="fas fa-thumbs-up"></i>
        </span>
      </a>
      <a href="#" onclick="deleteStar('김다미')" class="card-footer-item has-text-danger">
        삭제
        <span class="icon">
          <i class="fas fa-ban"></i>
        </span>
      </a>
    </div>
  </div>

```



정리하자면,
각 영화인 카드마다 '위로!', '삭제' 버튼이 있고,
a tag를 누르면 함수가 동작(onclick)하게 되어있습니다. (우리가 앞서 배운 버튼처럼 동작합니다.)

▼ 8) API 만들고 사용하기 - 좋아요 API (Update → POST)



정리하면, 만들 API 정보는 아래와 같습니다.

A. 요청 정보

- 요청 URL = `/api/like`, 요청 방식 = `POST`
- 요청 데이터 : 영화인 이름(name_give)

B. 서버가 제공할 기능 : 영화인 이름(요청 데이터)과 일치하는 영화인 정보의 좋아요 수를 한 개 증가시켜 데이터베이스에 업데이트하고(Update), 성공했다고 응답 메세지를 보냄

C. 응답 데이터 : (JSON 형식) 'result' = 'success'

▼ 9) API 만들고 사용하기 - 카드 삭제 API (Delete → POST)



만들 API

- 조회(Read) 기능:** 영화인 정보 전체를 조회
- 좋아요(Update) 기능:** 클라이언트에서 받은 이름(name_give)으로 찾아서 좋아요(like)를 증가
- 삭제(Delete) 기능:** 클라이언트에서 받은 이름(name_give)으로 영화인을 찾고, 해당 영화인을 삭제



정리하면, 만들 API 정보는 아래와 같습니다.

A. 요청 정보

- 요청 URL = `/api/delete`, 요청 방식 = `POST`
- 요청 데이터 : 영화인 이름(name_give)

B. 서버가 제공할 기능 : 영화인 이름(요청 데이터)과 일치하는 영화인 정보를 데이터베이스에서 삭제(Delete)하고, 성공했다고 응답 메세지를 보냄

↑ O B

C. 응답 데이터 : (JSON 형식) 'result' = 'success'

▼ 1) 클라이언트와 서버 연결 확인하기

- 여기서는 미리 적혀 있는 쌍으로 되어있는 서버-클라이언트 코드를 확인하고 갈게요.
- 분홍 형광펜 부분이 서로 어떻게 매칭되는지 확인해보세요!

[서버 코드 - `app.py`]

```
@app.route('/api/delete', methods=['POST'])
def delete_star():
    # 1. 클라이언트가 전달한 name_give를 name_receive 변수에 넣습니다.
    # 2. mystar 목록에서 delete_one으로 name이 name_receive와 일치하는 star를 제거합니다.
    # 3. 성공하면 success 메시지를 반환합니다.
    return jsonify({'result': 'success', 'msg': 'delete 연결되었습니다!'})
```

[클라이언트 코드 - `index.html`]

```
function deleteStar(name){
    // 1. 서버에 1) POST 방식으로, 2) /api/delete 라는 url에, 3) name_give라는 이름으로 name을 전달합니다.
    // 참고) POST 방식으로 data: {'name_give': name} 과 같은 양식이 되어야합니다!
    // 2. '삭제 완료! 안녕!' 얼럿을 띠웁니다.
    // 3. 변경된 정보를 반영하기 위해 새로고침합니다.
    $.ajax({
        type: 'POST',
        url: '/api/delete',
        data: {},
        success: function (response) {
            if (response['result'] == 'success') {
                let msg = response['msg'];
                alert(msg);
            }
        }
    });
}
```



동작 테스트

'삭제' 버튼을 눌렀을 때, 'delete 연결되었습니다!' alert창이 뜨면
클라이언트 코드와 서버 코드가 연결 되어있는 것입니다.

▼ 2) 서버부터 만들기



API 는 약속이라고 했습니다. 위에 미리 설계해 둔 API 정보를 보고 만들어보죠!

영화인 카드를 삭제하기 위해 필요한 정보는 다음과 같습니다.
- 영화인의 이름 (name_give)

따라서 서버 로직은 다음 단계로 구성되어야 합니다.

1. 클라이언트가 전달한 name_give를 name_receive 변수에 넣기
2. mystar 에서 delete_one으로 name이 name_receive와 일치하는 star를 제거
3. 성공하면 success 메시지를 반환

```

@app.route('/api/delete', methods=['POST'])
def delete_star():
    # 1. 클라이언트가 전달한 name_give를 name_receive 변수에 넣습니다.
    name_receive = request.form['name_give']
    # 2. mystar 목록에서 delete_one으로 name이 name_receive와 일치하는 star를 제거합니다.
    db.mystar.delete_one({'name': name_receive})
    # 3. 성공하면 success 메시지를 반환합니다.
    return jsonify({'result': 'success'})

```

▼ 3) 클라이언트 만들기



API 는 약속이라고 했습니다. API를 사용할 클라이언트를 만들어보죠!

영화인 카드를 삭제하기 위해 필요한 정보는 다음과 같습니다.

- 영화인의 이름 (name_give)

따라서 클라이언트 로직은 다음 단계로 구성되어야 합니다.

1. 서버에

- 1) POST 방식으로,
 - 2) /api/delete 라는 url에,
 - 3) name_give라는 이름으로 name을 전달
(참고) POST 방식이므로 data: {'name_give': name}
2. '삭제 완료! 안녕!' alert창 띄우기
 3. 변경된 정보를 반영하기 위해 새로고침

```

function deleteStar(name){
    // 1. 서버에 1) POST 방식으로, 2) /api/delete 라는 url에, 3) name_give라는 이름으로 name을 전달합니다.
    // 참고) POST 방식이므로 data: {'name_give': name} 과 같은 양식이 되어야합니다!
    $.ajax({
        type: "POST",
        url: "/api/delete",
        data: { 'name_give': name },
        success: function (response) {
            if (response['result'] == 'success') {
                // 2. '삭제 완료! 안녕!' 얼럿을 띄웁니다.
                alert('삭제 완료! 안녕!')
                // 3. 변경된 정보를 반영하기 위해 새로고침합니다.
                window.location.reload()
            }
        }
    });
}

```

▼ 4) 완성 확인하기



동작 테스트

삭제 버튼을 눌렀을 때 영화인 카드가 삭제되는지 확인합니다.

▼ 10) 전체 완성 코드

▼ [코드 - [index.html](#)]

```

<!DOCTYPE html>
<html lang="ko">
    <head>
        <meta charset="UTF-8"/>
        <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
        <title>마이 페이보릿 무비스타 | 프론트-백엔드 연결 마지막 예제!</title>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.8.0/css/bulma.min.css"/>
        <script defer src="https://use.fontawesome.com/releases/v5.3.1/js/all.js"></script>
        <style>
            .center {
                text-align: center;
            }

            .star-list {

```

```

        width: 500px;
        margin: 20px auto 0 auto;
    }

    .star-name {
        display: inline-block;
    }

    .star-name:hover {
        text-decoration: underline;
    }

    .card {
        margin-bottom: 15px;
    }

```

</style>

<script>

```

$(document).ready(function () {
    // index.html 로드가 완료되면 자동으로 showStar() 함수를 호출합니다.
    showStar();
});

function showStar() {
    // 1. #star_box의 내부 html 태그를 모두 삭제합니다.
    $('#star-box').empty()

    // 2. 서버에 1) GET 방식으로, 2) /api/list 라는 주소로 star_list를 요청합니다.
    $.ajax({
        type: "GET",
        url: "/api/list",
        data: {},
        success: function (response) {
            // 3. 서버가 돌려준 star_list를 star라는 변수에 저장합니다.
            let stars = response['stars_list']
            // 4. for 문을 활용하여 star 배열의 요소를 차례대로 조회합니다.
            for (let i = 0; i < stars.length; i++) {
                let star = stars[i]
                // 5. star[i] 요소의 name, url, img_url, recent, like 키 값을 활용하여 값을 조회합니다.
                let name = star['name']
                let url = star['url']
                let imgUrl = star['img_url']
                let recentWork = star['recent']
                let like = star['like']

                // 6. 영화인 카드를 만듭니다.
                let tempHtml = `<div class="card">
                    <div class="card-content">
                        <div class="media">
                            <div class="media-left">
                                <figure class="image is-48x48">
                                    
                                </figure>
                            </div>
                            <div class="media-content">
                                <a href="${url}" target="_blank" class="star-name title is-4">${name} (좋아요: ${like})</a>
                                <p class="subtitle is-6">${recentWork}</p>
                            </div>
                        </div>
                    </div>
                    <footer class="card-footer">
                        <a href="#" onclick="likeStar('${name}')" class="card-footer-item has-text-info">
                            위로!
                            <span class="icon">
                                <i class="fas fa-thumbs-up"></i>
                            </span>
                        </a>
                        <a href="#" onclick="deleteStar('${name}')" class="card-footer-item has-text-danger">
                            삭제
                            <span class="icon">
                                <i class="fas fa-ban"></i>
                            </span>
                        </a>
                    </footer>
                </div>`;

                // 7. #star-box에 temp_html을 붙입니다.
                $('#star-box').append(tempHtml)
            }
        }
    });
}

function likeStar(name) {
    $.ajax({

```

```

        type: "POST",
        url: "/api/like",
        data: {'name_give': name},
        success: function (response) {
            if (response['result'] == 'success') {
                // 2. '좋아요 완료!' 알럿을 띵웁니다.
                alert('좋아요 완료!')
                // 3. 변경된 정보를 반영하기 위해 새로고침합니다.
                window.location.reload()
            }
        }
    });
}

function deleteStar(name) {
    // 1. 서버에 1) POST 방식으로, 2) /api/delete 라는 url에, 3) name_give라는 이름으로 name을 전달합니다.
    // 참고) POST 방식이므로 data: {'name_give': name} 과 같은 양식이 되어야합니다!
    $.ajax({
        type: "POST",
        url: "/api/delete",
        data: {'name_give': name},
        success: function (response) {
            if (response['result'] == 'success') {
                // 2. '삭제 완료! 안녕!' 알럿을 띵웁니다.
                alert('삭제 완료! 안녕!')
                // 3. 변경된 정보를 반영하기 위해 새로고침합니다.
                window.location.reload()
            }
        }
    });
}

</script>
</head>
<body>
<section class="hero is-warning">
    <div class="hero-body">
        <div class="container center">
            <h1 class="title">
                마이 페이보릿 무비스타😎
            </h1>
            <h2 class="subtitle">
                순위를 매겨봅시다
            </h2>
        </div>
    </div>
</section>
<div class="star-list" id="star-box">
    <div class="card">
        <div class="card-content">
            <div class="media">
                <div class="media-left">
                    <figure class="image is-48x48">
                        
                    </figure>
                </div>
                <div class="media-content">
                    <a href="#" target="_blank" class="star-name title is-4">김다미 (좋아요: 3)</a>
                    <p class="subtitle is-6">안녕, 나의 소울메이트(가제)</p>
                </div>
            </div>
        </div>
        <footer class="card-footer">
            <a href="#" onclick="likeStar('김다미')" class="card-footer-item has-text-info">
                좋아요!
            </a>
            <span class="icon">
                <i class="fas fa-thumbs-up"></i>
            </span>
            </a>
            <a href="#" onclick="deleteStar('김다미')" class="card-footer-item has-text-danger">
                삭제
            <span class="icon">
                <i class="fas fa-ban"></i>
            </span>
            </a>
        </footer>
    </div>
</div>
</body>
</html>

```

▼ [ 코드 -  app.py]

```

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

client = MongoClient('localhost', 27017)
db = client.dbsparta

# HTML 화면 보여주기
@app.route('/')
def home():
    return render_template('index.html')

# API 역할을 하는 부분
@app.route('/api/list', methods=['GET'])
def show_stars():
    # 1. db에서 mystar 목록 전체를 검색합니다. ID는 제외하고 like 가 많은 순으로 정렬합니다.
    # 참고) find({}, {'_id': False}), sort()를 활용하면 굿!
    stars = list(db.mystar.find({}, {'_id': False}).sort('like', -1))
    # 2. 성공하면 success 메시지와 함께 stars_list 목록을 클라이언트에 전달합니다.
    return jsonify({'result': 'success', 'stars_list': stars})

@app.route('/api/like', methods=['POST'])
def like_star():
    # 1. 클라이언트가 전달한 name_give를 name_receive 변수에 넣습니다.
    name_receive = request.form['name_give']

    # 2. mystar 목록에서 find_one으로 name이 name_receive와 일치하는 star를 찾습니다.
    star = db.mystar.find_one({'name': name_receive})
    # 3. star의 like 에 1을 더해준 new_like 변수를 만듭니다.
    new_like = star['like'] + 1

    # 4. mystar 목록에서 name이 name_receive인 문서의 like 를 new_like로 변경합니다.
    # 참고: '$set' 활용하기!
    db.mystar.update_one({'name': name_receive}, {'$set': {'like': new_like}})

    # 5. 성공하면 success 메시지를 반환합니다.
    return jsonify({'result': 'success'})

@app.route('/api/delete', methods=['POST'])
def delete_star():
    # 1. 클라이언트가 전달한 name_give를 name_receive 변수에 넣습니다.
    name_receive = request.form['name_give']
    # 2. mystar 목록에서 delete_one으로 name이 name_receive와 일치하는 star를 제거합니다.
    db.mystar.delete_one({'name': name_receive})
    # 3. 성공하면 success 메시지를 반환합니다.
    return jsonify({'result': 'success'})

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

```



와, API 사용하고 만들기 한 번 더 복습하니 조금은 익숙해지죠?
마이페이지보릿 무비스타 프로젝트도 완성!

후반 3시간

[월수/화목반] : 체크인 & 출석체크



튜터님은 체크인과 함께 출석 체크(링크)를 진행해주세요!

스파르타코딩클럽 출석체크

<http://spartacodingclub.shop/attendance>

▼ "15초 체크인" 진행합니다

- 튜터는 타이머를 띄워주세요! ([링크](#))
- 본인의 감정상태와 오늘 있었던 소소한 일을 공유하는 시간입니다.

배우고 적용하기 - 후반부

5분 꿀팁 - 기록하며 배우기

▼ 개발일지 - TIL(Today I Learned)

- **?** TIL(Today I Learned) 이 뭔가요? 일일 개발일지 같은 건가요?
 - **!** 네, 맞습니다! 오늘 내가 배운 것(Today I Learned) 을 기록하는 거죠. 오늘 공부하면서 배운 걸 자유롭게 적는 것입니다. 기록하면서 내가 배운 내용이 한 번 더 정리가 되겠죠? 우리가 수업시간에 **소화타임**을 가지는 것과 비슷하죠.
- 개발자들은 Github repository에 적기도 하고 블로그를 사용하기도 합니다. 일일커밋(Git에 매일매일 commit과 함께 진행하기도 하죠.
- TIL은 아래처럼 토픽별로 적을 수도 있고

jbranchaud/til

Today I Learned A collection of concise write-ups on small things I learn day to day across a variety of languages and technologies. These are things that don't really warrant a full blog post. These are things I've picked up by Learning In Public™ and pairing with smart people at

 <https://github.com/jbranchaud/til>



ohahohah/TIL

Today I learned :+1: / 자유롭게 배운 내용을 정리함.. Contribute to ohahohah/TIL development by creating an account on GitHub.

 <https://github.com/ohahohah/TIL>



- 일자별로 적을 수도 있어요

6개월간의 TIL 회고 (꾸준히 하면 좋은 일이 생긴다)

2016년 퇴사 후 비전공자로 개발공부를 하면서 여러번 좌절하고 중간에 도망(?)을 가기도 했었다. github 커밋로그를 보면 학습 과정의 부침을 그대로 볼 수 있는데 아주 간단하게 정리하면 아래와 같다. 과거 : 일문과 출

업 후 스타트업에서 여러가지 일을 맡아서 할 (마케팅, 정산, 고객 CS, 서비스 운영 등등) 2016년 9월 : 프론트

 <https://wayhome25.github.io/til/2017/08/14/TIL-for-6-months/>



- TIL하고 나서 한 달, 6개월, 1년 어땠는지 회고를 하기도 합니다.

TIL을 1년동안 진행하며

저는 개발자분들의 블로그나 NHN이나 네이버, 우아한형제들같은 회사 기술 블로그를 굉장히 좋아합니다. 개발 공부를 이제 막 시작한, 1년 전 어느날도 평소와 마찬가지로 여러 기술블로그들을 탐닉하는 와중.. 우연히 1일1카피이라는 글을 봤습니다. 또 어디선가는 TIL이라는 글도 봤습니다. 이 TIL과 일일커밋에 관한 포스팅들이 https://junwoo45.github.io/2019-09-10-til_후기/?fbclid=IwAR2GIPJ5NUzilp-NAtsvqfu1RcN2C9JrUFPWZ_itLqKVbjM98GQpC5yscSg

- [AngularJS-Atom](#) : Angular-UI 팀에서 매인테이너로 참여하는 Atom 패키지
- [atom](#) : Atom 에디터
- [electron-starter](#) : Electron(구 Atom Shell)
- [iojs-ko](#) : io.js 한국어 번역팀
- [FRENDS](#) : FRENDS 사이트
- [involved](#) : 개인 프로젝트
- [ansible-prototype](#) : 아는 사람들과 진행하는 사이드 프로젝트의 프로토타입
- [Socket.IO Examples](#) : 2년 동안 버려뒀던 개인 프로젝트

[✍ 실행하기]

- 우리는 앞으로 '내 프로젝트'를 진행하면서 개발일지를 적어볼 거예요.
여기에 프로젝트 진행하면서 배운 것도 함께 적어볼까요?. 꼭 매일매일 적을 필요는 없어요. 공부할 때마다 조금씩이라도 적어 나가면 됩니다.

[1.5시간]: 프로젝트 개요 & 목표 설정

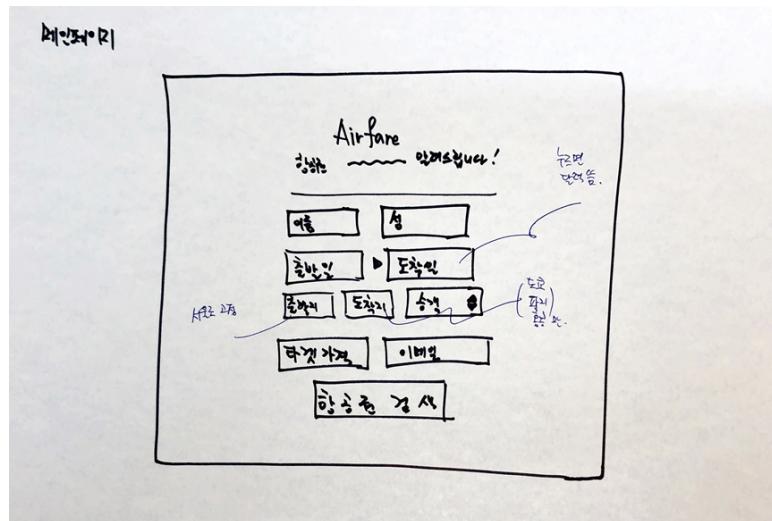
▼ 11) 프로젝트 1차 기획안 작성 & 발표



일단 말하고 나면, 이루어진답니다! :-)
내가 만든 기획안은 프로젝트 기간동안 나침반 역할을 해줄거에요!

- 공유 가능한 블로그(네이버 블로그 추천!)에 사진+글로 정리하고, 슬랙에 공유합니다!
 - 기획안을 바탕으로 6~8주차에 계속해서 개발일지를 작성할 예정입니다.
 - 본인이 익숙한, 슬랙에 공개 가능한 블로그(Velog, Medium) 등을 사용하셔도 좋습니다.
- 기획안에 꼭! 들어가야 하는 내용들 ([예시1](#), [예시2](#), [예시3](#))
 1. 프로젝트 이름/설명
 2. 프로젝트 생김새(레이아웃) = **꼭 그림으로 그려주세요!**
(챗봇 등 화면이 없는 프로젝트는 패스)

▼ 이렇게!



3. 개발해야 하는 기능들
4. 프로젝트에 필요한 데이터 소스 (3주차 공통 숙제 참고)

[1.5시간]: 투터 개별 피드백

▼ 12) 개발 범위 & 우선순위 정하기

- 투터와 10분 씩 개별 면담을 통해 프로젝트의 1) 개발 범위와 2) 기능 우선순위에 대해 함께 검토해서 3) 다음 시간까지 해올 **Todo 리스트***를 선정합니다.



Todo 리스트*

아래 예시처럼 할 일을 정리해보기로 해요!

한번 속 읽어보면 "뭐부터 해야할까?" 정리하는 데 도움이 될거에요!

▼ '한 걸음 더' 페이지를 보고 필요한 기술이 있는지 확인해보기

한 걸음 더

▼ 프로젝트와 관련 있는 API 사용법을 공부해오기

- [공공데이터포털](#), [서울 열린데이터광장](#), [네이버](#), [카카오](#), [구글](#), [유튜브](#), [페이스북](#), ... 등등의 API 사용법을 유튜브 영상 또는 구글링으로 공부해오기
- [링크](#)에서 내가 필요한 서비스가 API를 제공하는지 확인해봅니다. (Behance, Gmail, Google Analytics, Riot Games, Dota 2, Covid-19 등등 무궁무진한 API가 있습니다!)

▼ 웹 스크래퍼(크롤러)가 필요하다면 어떻게 만들지 구상해오기

- 예를 들어 면세점 정보를 모아서 종합 할인가를 보여주는 서비스를 만들기 위해서는 Selenium이라는 패키지의 사용법을 알아보아야 합니다. (위의 '한 걸음 더' 페이지를 참고하세요!)

▼ 특정 기술이 필요하다면 그에 관해 조사해오기

- 그 외 프로젝트에 필요한 기술이 있다면 조사해오고, 어떻게 공부해서 사용할지에 대한 계획이 필요합니다!



와, 이제 내 프로젝트가 어떻게 만들어질지 상상되시나요?

두근두근 나만의 프로젝트 완성이 기대되네요!

소화 타임



소화타임 = 복습 & 숙제 시작 타임!

숙제 설명

[숙제 - 단짠단짠 🍔🍜] 프로젝트 기획안 완성!

- 오늘 튜터님에게 받은 피드백을 바탕으로 최종 프로젝트 기획안을 완성해옵니다.
- 공유 가능한 블로그에 사진+글로 정리하고, 링크를 슬랙에 공유합니다! (티스토리, 네이버 블로그, Medium, velog 모두 좋습니다!)
- 최종 프로젝트 기획안에는 다음 내용을 넣어주세요.



스파르타코딩클럽 내 프로젝트 기획안

- 프로젝트 이름/설명
- 프로젝트 생김새(레이아웃)
- 개발해야 하는 기능 (우선순위별 정렬)
- 프로젝트에 필요한 데이터 소스 (3주차 공통 숙제 참고)
- 다음 시간까지 해야 하는 Todo 리스트

[💡 한 걸음 더!]

- 기획안이 완성되었다면, 개발 Todo 리스트 1순위를 진행해보세요! **바로 내 프로젝트로 뛰어들기!**
- 개발하며 TIL도 적어보세요.

복습 도우미

- 이번주는 4주차 배운 것 복습입니다! 복습하니 더 재미있네요! 😊 총 복습한다는 느낌으로 내용을 빠르게 훑어보세요.

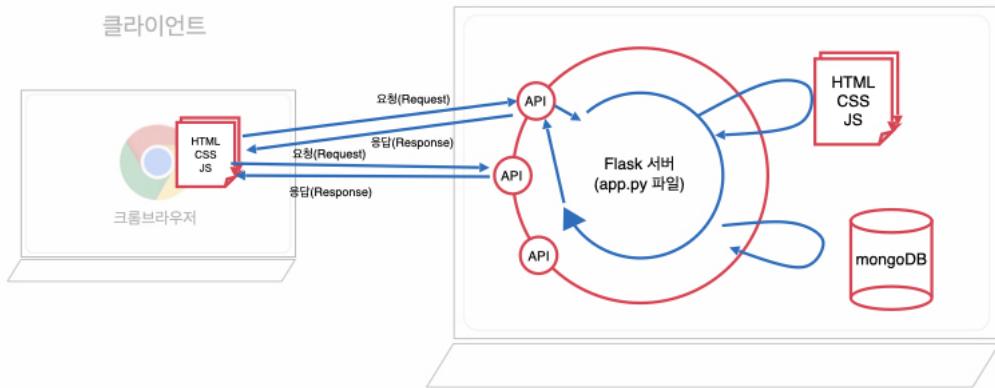
▼ 웹 기초 동작 원리



오늘은 머리 속 퍼즐을 맞추기 위해, 저번주와 비슷한 난이도의 프로젝트를 실습했습니다.

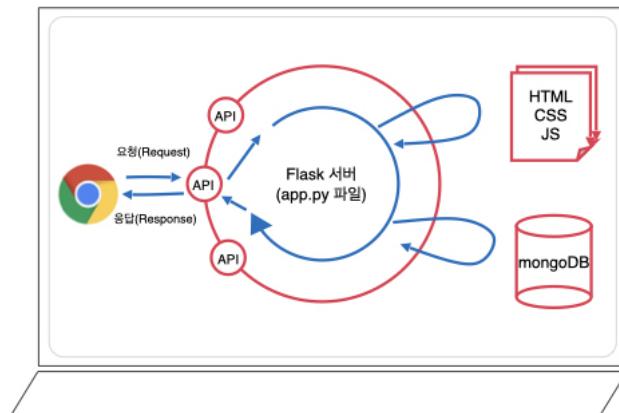
여기까지 배웠다면, 이제 6~8주차 프로젝트 준비 완료!!

서버



우리 프로젝트는 클라이언트(요청하는 쪽)과 서버(응답하는 쪽)이 한 컴퓨터에 있는 **로컬 개발환경**으로 구성이 되어 있습니다!

다음주에 서버와 클라이언트가 다른 컴퓨터에 있는 실제 서비스 환경처럼 프로젝트를 진행해볼 거에요.



▼ 지난주(4주차) 복습 - API, Flask, GET / POST 방식 요청

- API(Application Programming Interface): 응용 프로그램(application, 애플리케이션)에서 기능을 사용하거나 데이터를 주고 받기 위한 기능
 - 우리가 사용하는 API는 아래 적힌 모든 것을 미리 약속해두고 그대로 동작합니다.
 1. **요청 정보** : 요청 URL, 요청 방식 (GET / POST ...)
 2. **서버가 제공할 기능** : 데이터 조회(Read), 데이터 생성(Create) 등
 3. **응답 데이터** : 응답 데이터 형식 (어떤 key 로 어떤 데이터를 줄지, 예. response['img'])
- Flask : 웹을 만들고 서버를 구동시키기 편하게 하는 프레임워크(Framework)([Flask 공식 문서](#) / [비공식 한글 번역문서](#))
 - `@app.route('/')` 부분을 수정해서 URL을 부여할 수 있습니다
 - **templates 폴더** : HTML 파일을 담아둡니다. 실행할 때에는 이 폴더에서 화면을 불러오죠.
 - **static 폴더** : 이미지나 css파일과 같은 정적 파일을 담아두는 역할을 하지요!
- 클라이언트 요청 방식 - GET, POST
 - GET → 통상적으로! 데이터 조회(Read)를 요청할 때
예) 영화 목록 조회
 - **데이터 전달** : URL 뒤에 물음표를 붙여 key=value로 전달
 - 예: google.com?q=북극곰

- POST → 통상적으로! 데이터 생성(Create), 변경(Update), 삭제(Delete) 요청 할 때
예) 회원가입, 회원탈퇴, 비밀번호 수정
- 데이터 전달 : 바로 보이지 않는 HTML body에 key:value 형태로 전달

✓ 체크아웃

▼ "15초 체크아웃"을 진행합니다.

- 튜터는 타이머를 띄워주세요! ([링크](#)).
- 체크인처럼, 현재 본인의 감정상태와 수업후기에 관해 이야기합니다.
 - 예. 오늘 프로젝트를 만들면서 복습하니 Flask 프로젝트 만드는 거에 익숙해진 거 같아요.
 - 자유롭게 해주셔도 좋고, **KPT**에 맞춰해주셔도 좋아요.
 - Keep : 오늘 수업을 하면서 좋았던 것, 앞으로도 할 행동 / Problem : 아쉬워서 고쳐보면 좋을 것 / Try : 고치기 위해 내가 할 시도
 - 예) (Keep) 웹사이트를 빠르게 만들어서 좋았어요. (Problem) 아직 API 사용하는 부분이 헷갈려요. (Try) 집에 가서 숙제하면서 복습해볼게요!
- 튜터님은 체크아웃과 함께 [출석 체크](#)([링크](#))를 진행해주세요!
- 출석체크 - (변동이 있다면 다시 제출해주세요!)

스파르타코딩클럽 출석체크

<http://spartacodingclub.shop/attendance>

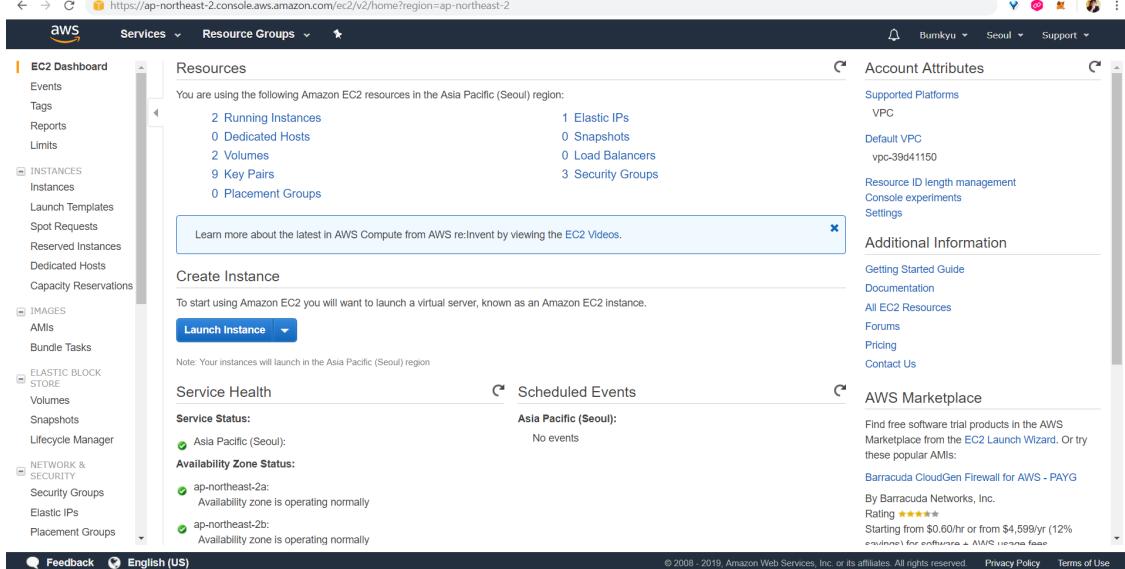
[가입/설치] - 다음 시간을 위해 미리 가입/설치해와야 할 것들

▼ AWS 가입하기 (승인까지 최대 24시간이 걸리니, 미리 해주세요!)

- 가입: <https://console.aws.amazon.com/console/home>
- 해외결제가 가능한 유효한 결제 수단을 넣어야 가입이 정상적으로 이루어집니다. Visa 또는 Master 겸용의 신용카드를 추천드립니다. 가입이 정상적으로 이루어지지 않을 경우 6주차 이후 수업을 진행할 수 없으므로 사전에 해외결제가 가능한지 반드시 확인 부탁드립니다.
- AWS는 개인에게 클라우드 환경의 가상서버를 제공합니다. 기본 사양의 서버(EC2)를 1년 동안 무료로 사용할 수 있으며, 이 후 월 1만 원정도의 금액이 결제될 수 있습니다.
- 가입 시 결제된 금액은 다시 반환됩니다. (일종의 결제 테스트 목적)

▼ 가입 후 아래와 같은 화면에 접속하면 성공!

- <https://ap-northeast-2.console.aws.amazon.com/ec2/v2/home?region=ap-northeast-2>



▼ Filezilla 설치하기

- 다운로드: <https://filezilla-project.org/download.php>
- 다운로드 클릭 후 가장 기본 버전 (스크린샷 기준 왼쪽) 다운로드!

Please select your edition of FileZilla Client

| | FileZilla | FileZilla with manual | FileZilla Pro |
|---------------------------------------|-----------|-----------------------|---------------|
| Standard FTP | Yes | Yes | Yes |
| FTP over TLS | Yes | Yes | Yes |
| SFTP | Yes | Yes | Yes |
| Comprehensive PDF manual | - | Yes | Yes |
| Amazon S3 | - | - | Yes |
| Backblaze B2 | - | - | Yes |
| Dropbox | - | - | Yes |
| Microsoft OneDrive | - | - | Yes |
| Google Drive | - | - | Yes |
| Google Cloud Storage | - | - | Yes |
| Microsoft Azure Blob and File Storage | - | - | Yes |
| WebDAV | - | - | Yes |
| OpenStack Swift | - | - | Yes |
| Box | - | - | Yes |
| Site Manager synchronization | - | - | Yes |

Download

Select

Select