



스파르타코딩클럽 4주차



매 주차 강의자료 시작에 PDF파일과 영상 링크를 올려두었어요!

▼ PDF 강의자료 다운받기

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/207bea08-5bae-4b20-baed-fa0210d07294/week04.pdf>

▼ 영상강의 참고하기

- [4주차 복습용 영상강의 링크](#)



모든 토글을 열고 닫는 단축키

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

Mac : **⌘** + **option** + **t**

목차



수업 목표



체크인



배우고 적용하기 - 전반부



5분꿀팁 - 프로그래밍은 문제 해결#3_디버깅

웹 기초 동작 원리 - 4주차: Flask, 모두의 책 리뷰, 나홀로 메모장

[15분] API 란 무엇일까?

[5분] 시작하기 전에 작업 공간 정비하기

[1시간] Flask 기초 - API 만들기 (화면 코드를 보내주는 API)

[0.5 시간] Flask 기초 - API 만들기 (JSON 형식 데이터를 보내주는 API)

[1시간] :모두의 책리뷰 완성하기



문제뱅크



배우고 적용하기 - 후반부



5분꿀팁 - 도구 잘 사용하기

[1.5시간] 나홀로메모장 완성하기



소화 타입

숙제 설명

복습 도우미

[1시간] 스스로 소화 타입 시작



체크아웃

[설치] - 다음 시간을 위해 미리 설치해와야 할 것들



수업 목표

1. API 만들기 - API 설계하고, Flask 프레임워크를 사용해 만들기
2. API 사용하기 - 클라이언트 코드에서 API를 사용해 데이터를 보여주기
3. 프로젝트 두 개 완성시키기 - '모두의 책 리뷰', '나홀로 메모장'

전반 3시간

✓ 체크인



튜터님은 체크인과 함께 출석 체크(링크)를 진행해주세요!

스파르타코딩클럽 출석체크

<http://spartacodingclub.shop/attendance>

▼ "15초 체크인"

- 튜터님은 타이머를 띄워주세요! (링크)
- 본인의 감정상태와 오늘 있었던 소소한 일을 공유하는 시간입니다.
 - 예. 지난주에 야구를 보러 갔어요. 갑자기 야구 순위 스크래핑을 해보고 싶더라고요. 배운 걸 바로 적용해볼 수 있으니 신나요! 오늘은 무얼 배워보게 될까요?

💡 배우고 적용하기 - 전반부

💡 5분 꿀팁 - 프로그래밍은 문제 해결#3_디버깅

▼ 디버깅(Debugging)

프로그래밍을 시작하자마자, 우리는 생각했던대로 프로그램 하는 것이 쉽지 않다는 것을 알게 되었죠. **디버깅은 발견해내야만 하는 것입니다.**

저는 정확한 순간을 기억합니다. 내 인생의 많은 부분을 내가 만든 그 프로그램에서 실수를 찾는데 쓸 것이라는 것을 깨달았을 때를.

— 초기 컴퓨터 EDSAC 개발자 Maurice Wilkes 교수,
1979년 9월 23일 "The Design and Use of the EDSAC," 강의 중에서

이 버그를 고치는 데 긴 시간이 걸린다면 왜 그런지 자문하라.
다음 번에는 이 버그를 좀 더 쉽게 고칠 수 있도록 할 수 있는 뭔가가 있을까?

— 앤드류 헌트, 데이비드 토머스, 『실용주의 프로그래머』, 김창준, 정지호 옮김, 인사이트(2005), p167

- ? 디버깅(Debugging)이 무엇인가요?

! 디버깅은 버그를 찾아내어 고치는 과정입니다.

버그(bug)는 소프트웨어가 오작동하거나 예상치 못한 잘못된 결과를 내고, 오류가 발생하는 등의 문제를 말하죠. 설계(Design)가 잘못되었거나, 프로그램 소스 코드에서 오류가 났기 때문에 생기죠.

! 프로그램 만들기는 설계(Design), 구현(Coding), 테스팅(Testing), 디버깅(버그 찾기, Debugging) 과정을 거치게 됩니다. 그만큼 디버깅은 프로그래밍할 때 빼먹을 수 없는 중요한 요소죠.

- 이제 여러분들이 디버깅의 세계 🌎 입문용 간단한 가이드를 드리겠습니다.
인생은 실전! 앞으로 실습하고 내 프로젝트를 진행하면서 디버깅을 마음껏 해보세요!

▼ 1. 당황하지 말고 잠시 시간을 두기

- 버그를 발견하면, 잠시 멈춰서 버그를 어떻게 하면 빠르게 찾을 수 있을까? 내가 맞닥뜨렸던 버그인가? 하고 생각해봅시다. 이 경험이 쌓이면, 디버깅을 조금 더 효율적이고 빠르게 할 수 있을 거예요.
- 먼저 심호흡 합니다. 버그는 어디서나 발생할 수 있고, 디버깅은 프로그래밍 하는 과정이니 일단 편하게 마음을 가져옵니다.

▼ 2-1. 여러 메시지 읽어보기

- 버그가 발생한 부분에 여러 메시지가 있다면 찬찬히 읽어봅니다.

- 이미 에러 메시지가 많은 정보를 알려주고 있는 경우가 많습니다. 왜 에러가 발생했는지 이유와 함께 에러가 발생한 코드의 라인 수를 같이 보여주는 경우가 많습니다. 해당 라인으로 가서 앞 뒤 코드를 읽어봅니다.
- Javascript에서 에러 메시지는 웹 브라우저의 개발자도구 console창에서 확인할 수 있습니다. 많이 발생하는 에러 메시지들은 아래와 같습니다.

Top 10 Most Common JavaScript Error Messages - dummies

To help you decipher the error messages that JavaScript throws your way, here's a list of the ten most common errors and what they mean: Syntax error: This load-time error means that JavaScript has detected improper syntax in a statement. The error message almost always tells you the exact line and character where the error ...

<https://www.dummies.com/web-design-development/top-10-common-javascript-error-messages/>

- Python에서 에러 메시지는 PyCharm 하단 Run 창에서 확인할 수 있습니다. ([Traceback](#)으로 시작하는 메시지)

Understanding the Python Traceback - Real Python

Python prints a traceback when an exception is raised in your code. The traceback output can be a bit overwhelming if you're seeing it for the first time or you don't know what it's telling you. But the Python traceback has a wealth of information that can help

<https://realpython.com/python-traceback/>



9.2 오류를 고치는 방법 | 파이썬 프로그래밍 입문서 (가제)

오류를 찾아 고치려면 추리소설 탐정처럼 하면 된다. 오류가 발생한 현장(코드)를 꼼꼼이 살펴보고, 프로그램이 죽기 전에 남긴 오류 메시지를 해석하고, 로그 기록에 따라 프로그램의 실행 흐름을 추적하고, 테스트 환경을 꾸며 오류를 재현해보면 대부분의 오류를 해결할 수 있다. 버그와 디버그 프로그램 <https://python.bakyeono.net/chapter-9-2.html#922-%EC%98%A4%EB%A5%98-%EB%A9%94%EC%8B%9C%EC%A7%80>



▼ 2-2. 에러 메시지가 없는데 내가 원하는 결과와 다르다면

- 버그가 발생한 부분에 에러 메시지가 없고, 내가 원하는 결과와 다르게 작업이 되었다면 직접 단계별로 데이터가 어떻게 변하는지 확인해야합니다.
- 또 버그가 발생한 상황을 그대로 만들어보는게 중요하죠. 이런 것을 '버그 재현'한다고 표현합니다.
- 여러가지 방법이 있지만 우리가 배우는 단계에서는 단계별로 데이터가 어떻게 변하는지 값을 '출력해서 디버깅하기 (Debugging by Printing)'가 유용합니다. 아래 단계에서 확인해볼게요.

▼ 3. '출력해서 디버깅하기(Debugging by Printing)'와 디버거(Debugger)

- 버그가 발생한 부분이 명확하지 않다면 소스 코드에서 어떤 단계까지 제대로 동작하고 어떤 단계부터는 제대로 동작하지 않는지 찾아내야합니다.
 - 이때 사용하는 것이 '출력해서 디버깅하기(Debugging by Printing)'와 디버거(Debugger)입니다.
 - 출력해서 디버깅하기(Debugging by Printing): 내가 컴퓨터에게 명령어를 내려서, 내가 가진 데이터가 바뀔 때마다 그 데이터를 출력해보는 것입니다. 어떤 부분까지 내 의도대로 동작했고, 어떤 부분부터 원하는 대로 동작하지 않았는지 확인할 수 있죠.
- 디버거(Debugger)를 사용하면 이런 동작을 조금 더 편리하게 할 수 있습니다.
- 디버거(Debugger): 어느 부분에서 버그가 발생했는지 쉽게 파악하기 위한 도구입니다. 우리가 사용하고 있는 크롬 개발자 도구와 PyCharm에도 디버거가 있습니다.

▼ Chrome Debugger 튜토리얼

Get Started with Debugging JavaScript in Chrome DevTools

This tutorial teaches you the basic workflow for debugging any JavaScript issue in DevTools. Read on, or watch the video version of this tutorial, below. Finding a series of actions that consistently reproduces a bug is always the first step to

<https://developers.google.com/web/tools/chrome-devtools/javascript>



▼ PyCharm Debugger 튜토리얼

디버거 - 기능 | PyCharm

PyCharm은 Python, virtualenv, Anaconda 또는 Conda env와는 관계없이 로컬 컴퓨터에서 실행되는 코드를 디버그할 수 있습니다. PyCharm Professional Edition의 경우 Docker 컨테이너, VM 내에서 또는 SSH를 통해 원격 호스트에서 실행되는 코드도 디버그할 수 있습니다.

JB <https://www.jetbrains.com/ko-kr/pycharm/features/debugger.html>

PyCharm

The Python IDE
for Professional Developers

JET BRAINS

Getting Started with PyCharm 6/8: Debugging

This video is the part of Getting Started with PyCharm video series by PyCharm Technical Advocate Paul Everitt. In this video you'll learn how to debug your ...

YouTube <https://youtu.be/QJtWxm12Eo0>

PyCharm

Debugging

Paul Everitt
PyCharm Developer Advocate at JetBrains

▼ 4. 질문하기 - 새로운 관점으로 보기

- 오랫동안 고민해도 버그가 보이지 않는다면, 터널비전(터널 속으로 들어간 것처럼 시야가 극도로 좁아지는 것)에 빠진 것일 수도 있어요. 내 눈에는 안 보였던 문제가 다른 사람에겐 금방 보일 수도 있지요.
- 이럴 때 새로운 관점이 필요합니다. 3주차 5분 꿀팁에서 배운 [질문 잘하기](#) 를 참고해 스스로에게, 다른 사람에게 질문해보세요!

☞ [모음] 5분 꿀팁 & 단축키

▼ 참고. 비한국어 자료를 한국어로 보고 싶을 때 - 한글 자동번역

- 유튜브 영상 자막 한글 자동번역 <https://support.google.com/youtube/answer/6373554?hl=ko>
- 크롬 자막 번역 확장프로그램 <https://chrome.google.com/webstore/detail/google-translate/aapbdbdomjkkjkaonfhkkfkfjllcleb?hl=ko>
- 앞으로 더 프로그밍을 하게 되면, 로깅(logging) - 프로그램 실행되는 과정을 나중에도 볼 수 있도록 로그로 기록 합니다.

WEB 기초 동작 원리 - 4주차: Flask, 모두의 책 리뷰, 나홀로 메모장

▼ 지난주 복습 - 3주차 개념

- 우리가 앞으로 만들 API에서 사용하는 프로그래밍 언어인 파이썬을 배워봤습니다. 가상환경을 만들어서 패키지를 설치했습니다.
- **CRUD** : 기본적인 데이터 처리 기능인 **Create, Read, Update, Delete** 의 두문자를 따서 **CRUD** 라고 합니다.
- MongoDB : No SQL의 한 종류인 MongoDB를 pymongo 패키지를 이용해 python으로 조작해보았습니다. mongoDB에 데이터를 CRUD 하는 것을 배웠습니다.

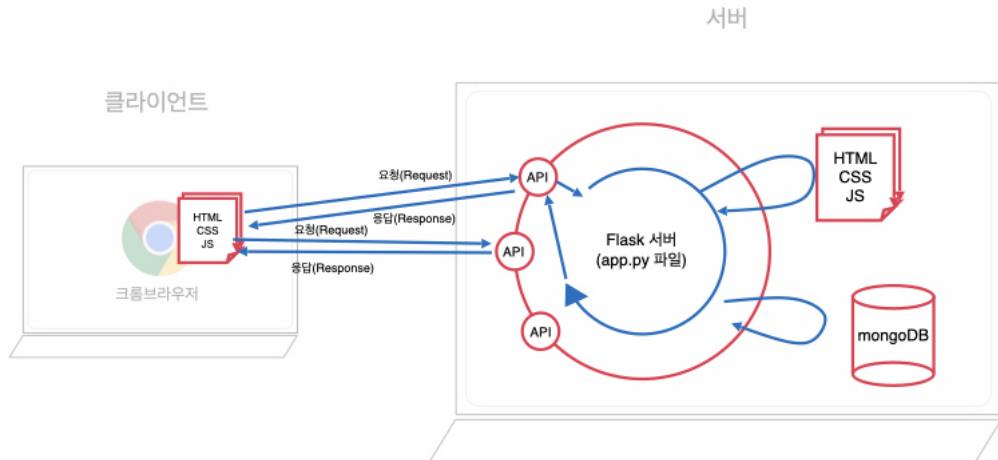


- HTTP : 웹의 통신규약. 클라이언트는 HTTP에서 요청을 하는 쪽이고, 서버는 HTTP에서 요청을 받아 응답하는 쪽입니다.
- API를 사용할 땐, 미리 정해둔 **약속**을 따라야 작동합니다. 약속들은 API 페이지(문서)에 적혀있습니다.

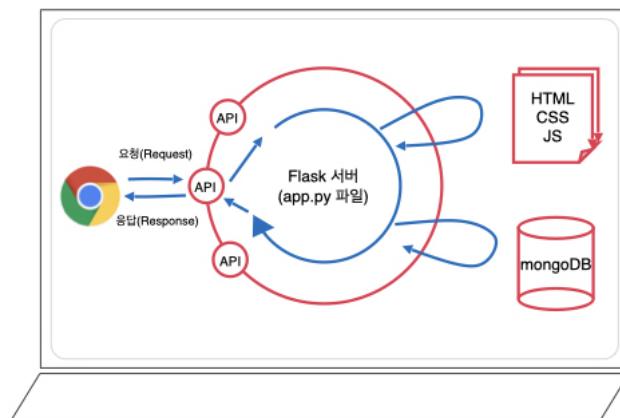


서버(백엔드) API를 만들어봅니다!
그리고 HTML 화면과 mongoDB까지 연동해서 두 가지 프로젝트를 만듭니다.

1. 모두의 책 리뷰 ([링크](#))
2. 나홀로 메모장 ([링크](#))



헷갈리지 말기!
우리는 컴퓨터 한 대를 서버로도 쓰고, 클라이언트로도 쓸 거에요.
이것을 바로 "로컬 개발환경"이라고 한답니다! 그림으로 보면, 대략 이렇습니다.



[15분] API 란 무엇일까?

▼ 1) API 가 무엇일까?



API(Application Programming Interface)는
응용 프로그램(application, 애플리케이션)에서 기능을 사용하거나 데이터를 주고 받기 위한 기능을 이야기합니다.

- 0주차 이론 강의를 떠올려보면서 API 개념을 복습해보죠!
- **스파르타코딩클럽 0주차 사전과제**
- 클라이언트가 서버에 데이터를 요청할 때, 아무렇게나 하면 될까요? 응행을 서버, 고객을 클라이언트라고 해봅시다.

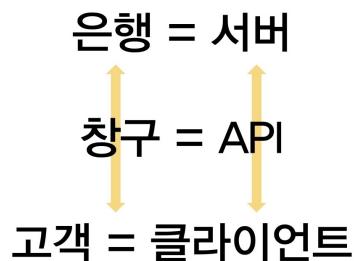
Q. 은행에 가서 내 통장 잔고를 확인하는 방법은?

1. 아무나에게 가서 물어본다
2. 대출 창구에 간다.
3. 크게 소리친다.
4. 번호표를 뽑고 입출금 창구에 가서 주민등록증을 보여준다.

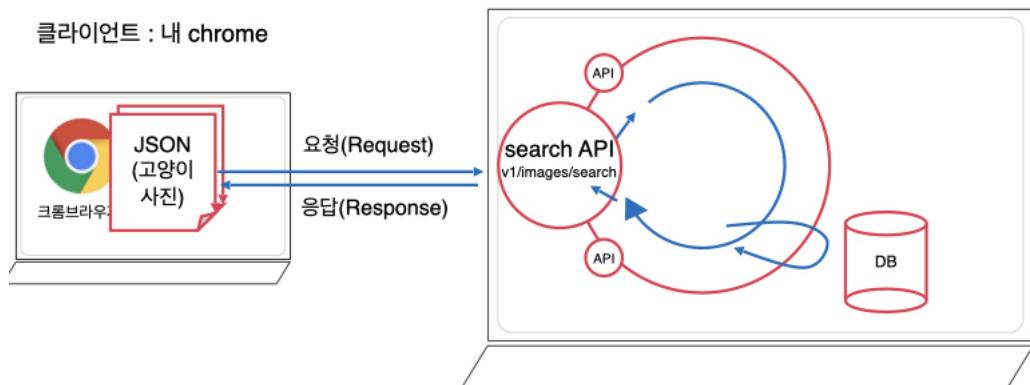


은행이 수많은 데이터를 가지고 있지만,
정해진 업무를 하는 창구에서 정해진 방식으로 소통해야하죠

- 고객(클라이언트)은 정해진 창구(API)를 통해, 은행(서버)에 요청합니다.



- 2주차에 사용한 랜덤 고양이 사진 API([링크](#) / [문서](#))를 살펴볼까요?
 1. **클라이언트:** 나의 chrome 창에서
 2. **요청(Request):** 요청 URL을 입력하고 엔터를 치는 것은 아래와 같은 의미예요.
 - 요청(Request) URL: <https://api.thecatapi.com/v1/images/search>
 - api.thecatapi.com 이라는 서버 주소의
 - v1/images/search 인 API에
 - 브라우저 주소창에서 엔터치기 = GET 요청을 보내기
 3. **서버 기능:** 서버는 정해진 동작인 **랜덤 고양이 사진 데이터를 조회**를 실행하고
 4. **응답(Response)** : 고양이 사진 데이터(JSON)를 클라이언트에 응답해줍니다.



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'])

이런 약속은 모두 API 문서에 적어둡니다!

[서울시 권역별 실시간 대기환경 현황](#)

[서울시 공공자전거 실시간 대여정보](#)

[랜덤 고양이 사진 API Doc](#)

[5분] 시작하기 전에 작업 공간 정비하기

- ▼ 2) 프로젝트를 보관할 폴더 네 개 만들고 시작하기

웹개발의 꽃, 백엔드-프론트엔드를 연결하는 일이 익숙해지도록,
연습 프로젝트 → 모두의 책리뷰 → 나홀로 메모장 → 마이 페이보릿 무비스타
총 4번에 걸쳐 반복 실습을 진행 할 예정입니다. 실습을 다 해내고 나면 아-주 익숙해질거예요!

- 코드 관리를 위해 미리 아래와 같이 폴더구조를 만들고 시작합니다!

sparta 폴더 안 → projects 폴더를 만들고 그 안에 → 폴더 4개 만들기

sparta > projects >			
	이름	수정한 날짜	유형
↑	alonememo	2020-05-08 오전 12:46	파일 폴더
↑	bookreview	2020-05-07 오후 11:24	파일 폴더
↑	moviestar	2020-05-07 오후 11:24	파일 폴더
↑	prac	2020-05-08 오전 8:48	파일 폴더

- alonememo 폴더 : "나홀로 메모장" 관련 코드를 작성합니다. (오늘 실습)
- bookreview : "모두의책리뷰" 관련 코드를 작성합니다. (오늘 실습)
- moviestar : "마이 페이보릿 무비스타" 관련 코드를 작성합니다. (다음주 실습)
- prac : Flask기초를 연습 할 폴더 (오늘 실습)

[1시간] Flask 기초 - API 만들기 (화면 코드를 보내주는 API)

앞으로 [튜터님과 함께] 단계별로 차근차근 만들어 나갈 거에요.
 sparta > projects > prac 이 프로젝트 폴더!

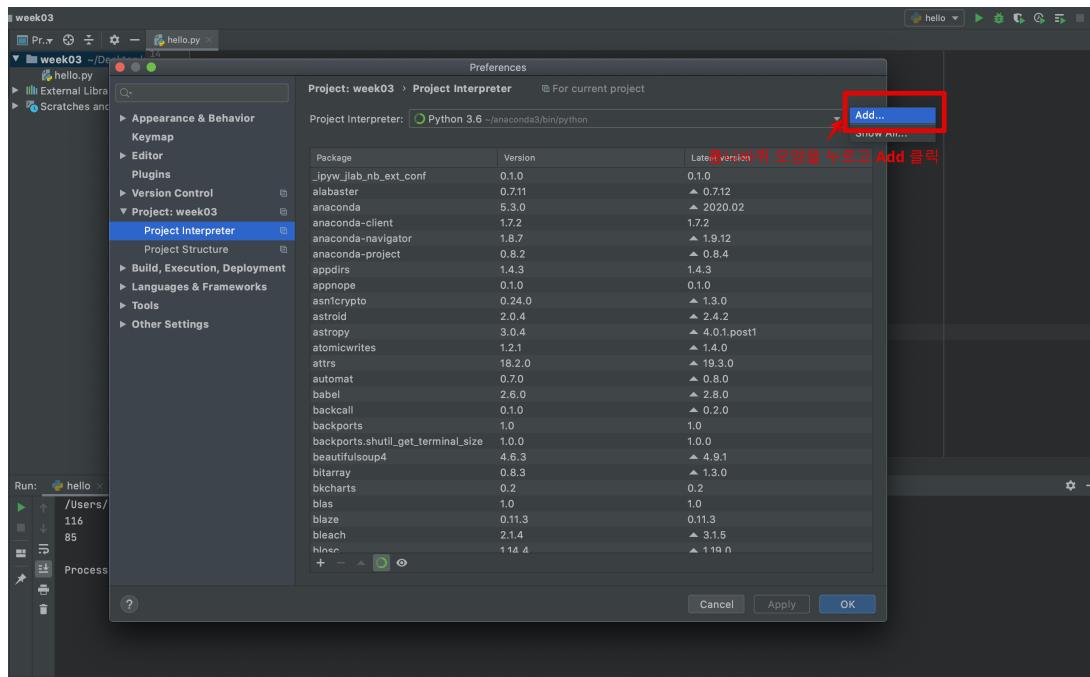
▼ 3) Flask 프로젝트 설정 & 구조 확인하기

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

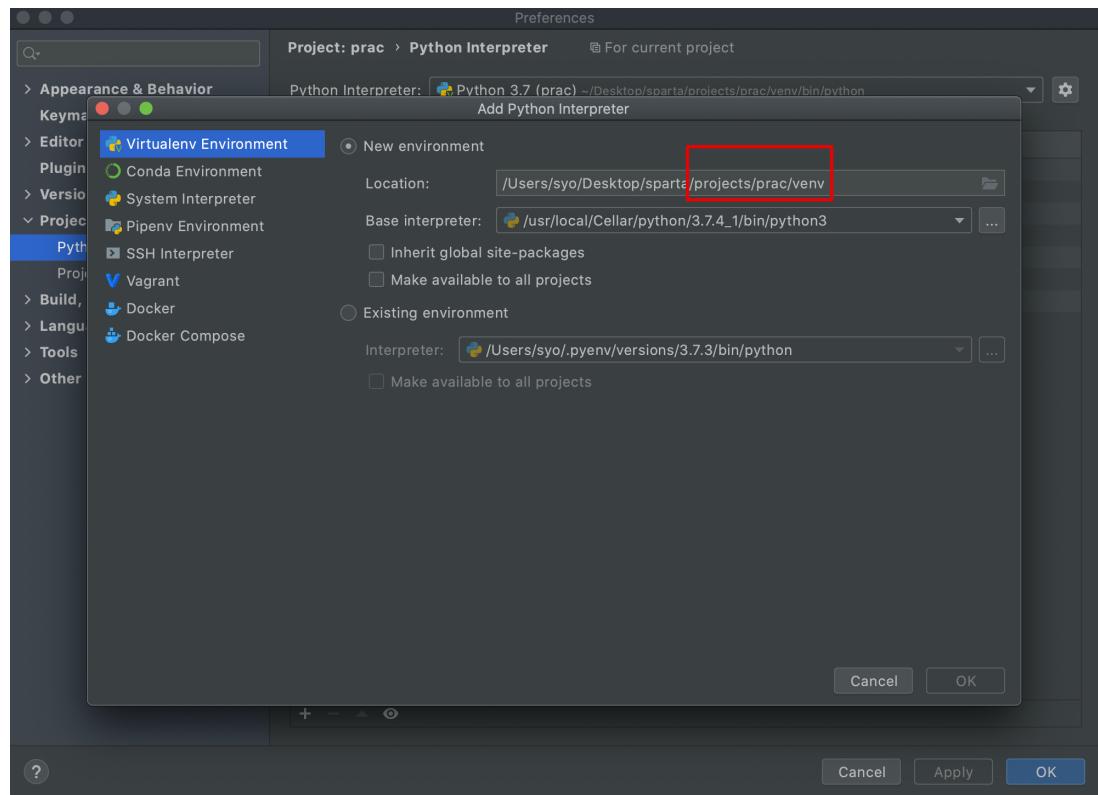
▼ 1. 가상 환경 설치 및 확인

- Windows : File → setting → project interpreter
Mac : Preference → Project Interpreter로 접근

- 톱니바퀴 add 버튼 클릭

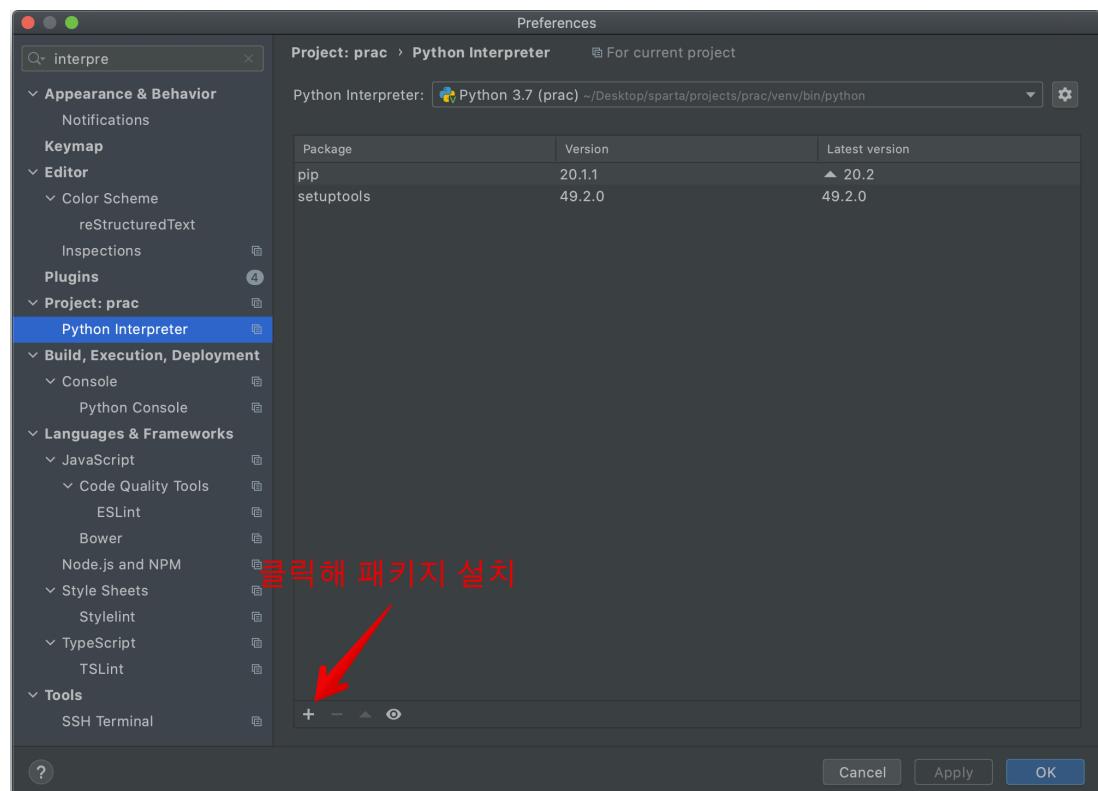


- virtual environment - New environment 선택
 - Location을 현재폴더/venv로 설정해주세요. 현재 프로젝트 폴더(prac) 아래에 가상환경 폴더(venv)가 생성이 됩니다.

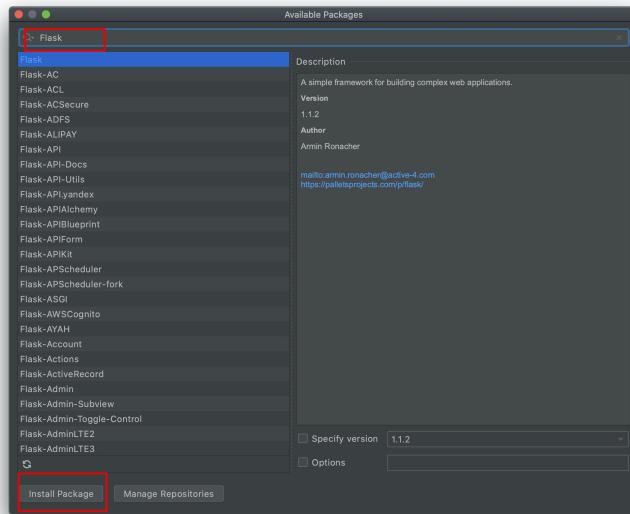


▼ 2. pip(python install package) 사용 - Flask 패키지 설치해보기

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



- flask 로 검색한 후, Install package 클릭



▼ 3. Flask 기본 폴더 구조 만들기

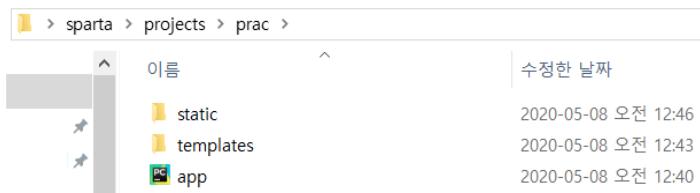
Flask 를 사용할 때는 기본 설정을 지켜주어야 합니다.
프로젝트 폴더 안에, 폴더를 만들어주세요!

static 폴더 : 이미지, css파일을 넣어둡니다.

templates 폴더 : html파일을 넣어둡니다.

app.py 파일

실습을 해가면서 각 폴더의 역할을 알아볼겁니다!



▼ 4) Flask 기본 실행

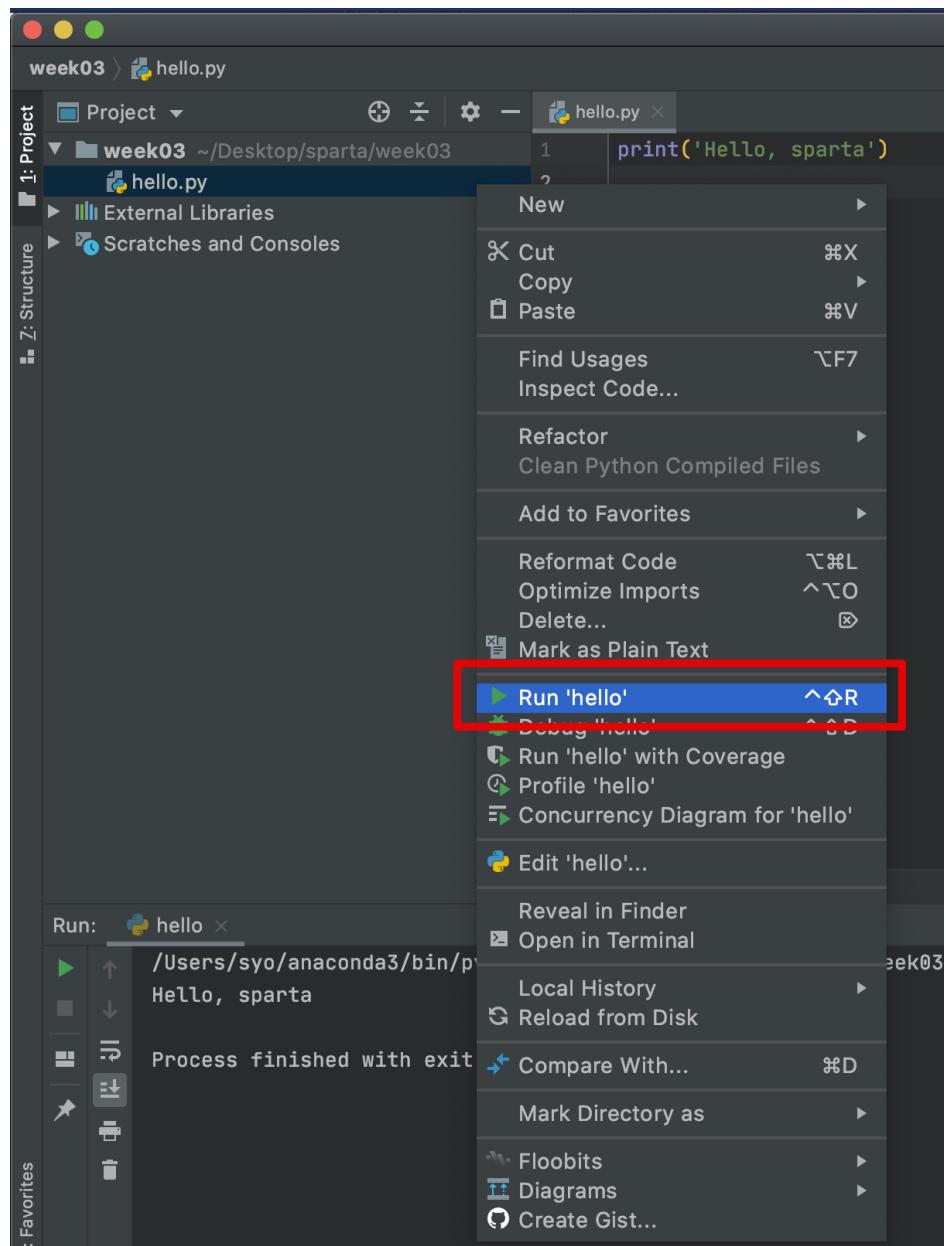
- Flask 는 웹을 만들고 서버를 구동시키기 편하게 하는 프레임워크(Framework)입니다. 서버를 구동하려면 필요한 복잡한 일들을 간편하게 쓸 수 있습니다. ([Flask 공식 문서](#) / [비공식 한글 번역문서](#))

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

- prac 폴더 안에 `app.py` 파일을 아래 분홍 형광펜 부분을 추가하고 실행해보세요.

▼ [지난 시간 복습] Pycharm에서 파일 실행하기

- 단축키 : Windows - `ctrl + shift + F10` / Mac - `ctrl+ shift + r`
- 또는 화면에서 클릭으로 실행
 - 우리는 `app.py` 를 실행할꺼니까 `Run app` 이라고 보이겠죠?



- 통상적으로 flask 서버를 실행시키는 파일은 app.py라고 이름 짓습니다!

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

@app.route('/')
def hello_world():
    return 'Hello World!'

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

2. 크롬 화면에서 웹 페이지를 확인하기

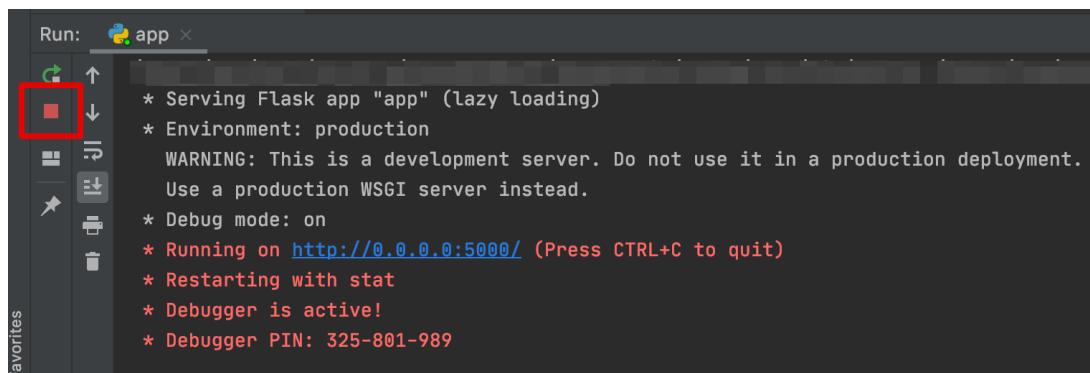
- 크롬에서 <http://0.0.0.0:5000/> 또는 [http://localhost:5000/](http://localhost:5000)으로 접속해보세요.
- 화면에 `Hello World!`라는 메시지가 보이시나요? 그렇다면 성공한 것! 🎉
- 어떻게 이런 일이 일어난 걸까요?

`@app.route('/')` 안의 `/` 가 URL 세부 주소가 됩니다. <http://localhost:5000/>에서 `/`로 접속하면, home 함수를 실

행합니다.

3. 서버 실행을 종료하는 방법

- 아래처럼 화면에서 stop 빨간 버튼을 누르거나, **ctrl + c**로 종료할 수 있습니다.



▼ 5) Flask로 URL 만들기

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



`@app.route(URL경로)`에서 URL경로와 함수명은 유일해야합니다.

여러 가지가 있다면 어떤 함수를 실행해야 할지 알 수 없겠죠?

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

@app.route('/')
def hello_world():
    return 'Hello World!'

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

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

- <http://localhost:5000/mypage>로 접속해보세요! 함수와 `@app.route('/mypage')`를 바꿔주었더니 뭐가 달라졌나요?

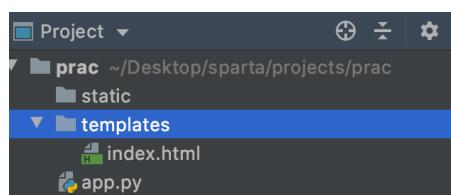
▼ 6) Flask로 화면 띄우기 - HTML 파일 사용



templates 폴더에는

HTML 파일을 담아둡니다. 실행할 때에는 이 폴더에서 화면을 불러오죠.

1. 간단한 index.html 파일을 templates 안에 만들기



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

```

<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>
</head>
<body>
    <h1>서버를 만들었다!</h1>
</body>
</html>

```

2. HTML 파일 불러오기

- app.py 파일을 아래 분홍 형광펜 부분을 수정/추가하고 실행해보세요.

 Flask 내장 함수 render_template를 이용해 HTML 파일 불러옵니다.
바로 이게 프레임워크의 위력!

```

from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home(): # 함수명 설정 - 이름만 보고 접속되는 페이지를 확인할 수 있게!
    return render_template('index.html')

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

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

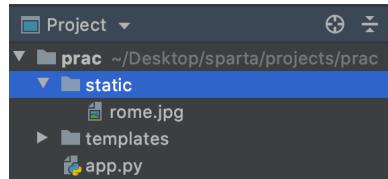
```

▼ 7) Flask로 화면 띄우기 - HTML 화면에 이미지를 불러오기

 static 폴더는
이미지나 css파일과 같은 정적 파일을 담아두는 역할을 하지요!

1. 아래 이미지 다운로드 받고 static 폴더에 넣습니다.

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/bd59a681-63a4-4dab-86cb-eacfc40b0b82/rome.jpg>



2. Flask에서 HTML을 부를 때, 저장된 이미지 파일을 불러오기

 이 HTML 파일은 Flask에서 읽어주기 때문에,
Flask에 미리 정의된 규칙대로 경로를 입력해줘야합니다.

이제 왜 static, templates 폴더가 존재하는지, 이해하시겠나요?

```

<!DOCTYPE html>
<html lang="ko">
  <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>Title</title>
  </head>
  <body>
    <h1>서버를 만들었다!</h1>
    
  </body>
</html>

```

[0.5 시간] : Flask 기초 - API 만들기 (JSON 형식 데이터를 보내주는 API)

▼ 8) 들어가기 전에 - GET, POST 요청 타입



리마인드!

은행의 창구가 API와 같다느 것을 기억하시나요?

같은 예금 창구에서도 개인 고객이냐 기업 고객이냐에 따라 처리하는 것이 다른 것처럼,

클라이언트가 요청 할 때에도, "방식"이 존재합니다.

HTTP라는 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.



GET, POST 방식

여러 방식([링크](#))이 존재하지만 우리는 가장 많이 쓰이는 GET, POST 방식에 대해 다루겠습니다.

- * 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

▼ 9) [💡튜터와 함께] API 맛보기



리마인드!

우리가 사용하는 API는 아래 적힌 모든 것을 미리 약속해두고 그대로 동작합니다.

1. 요청 정보 : 요청 URL, 요청 방식 (GET / POST /...)
2. 서버가 제공할 기능 : 예. 회원 데이터 조회(Read), 주문데이터 생성(Create) 등
3. 응답 데이터 : 응답 데이터 형식 (어떤 key로 어떤 데이터를 줄지, 예. response['img'])

위 세 가지 정보를 잘 기억해두세요!

이제 기본 API를 만들고, 클라이언트에서 요청해서 API를 사용해보겠습니다.

직접 만들고, 바로 사용하고!

▼ [💻코드 API 만들기 01-1] GET 방식 API 만들기



만들 API 정보

A. 요청 정보

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

B. 서버가 제공할 기능 : 클라이언트에게 정해진 메시지를 보냄

C. 응답 데이터 : (JSON 형식) 'result'='success', 'msg'='이 요청은 GET!'

- `app.py`에 분홍 형광펜 부분을 추가해주세요. 그리고 파일을 실행시키기!

```
from flask import Flask, render_template, jsonify, request

app = Flask(__name__)

## HTML을 주는 부분
@app.route('/')
def home():
    return render_template('index.html')

## API 역할을 하는 부분
@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!'})

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

▼ [💻 코드 API 만들기 01-2] GET 방식 API 사용하기

- 크롬 개발자도구 - console창을 열어 아래 코드를 실행해주세요.



사용할 API 정보

A. 요청 정보

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

B. 서버가 제공할 기능 : 클라이언트에게 정해진 메시지를 보냄

C. 응답 데이터 : (JSON 형식) 'result'='success', 'msg'='이 요청은 GET!'



Ajax를 사용하기 위해서,

앞서 만들어둔 `index.html` (ajax import 되어있음)을 크롬에 띄워주세요.

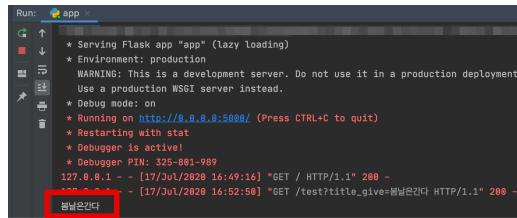
이제 웹 페이지 소스를 응답해주는 Flask 서버를 만들고 실행시켜두었으니,
크롬에 `localhost:5000/`를 입력하면 화면이 띄워지겠죠?

```
$.ajax({
  type: "GET",
  url: "/test?title_give=봄날은간다",
  data: {},
  success: function(response){
    console.log(response)
  }
})
```

- 크롬 console 화면에 어떤 메시지가 뜨나요?

```
> $.ajax({
    type: "GET",
    url: "/test?title_give=봄날은간다",
    data: {},
    success: function(response){
        console.log(response)
    }
})
<-- {readyState: 1, getResponseHeader: f, getAllRe
rs: f, setRequestHeader: f, overrideMimeType:
}
    ▶ {msg: "이 요청은 GET!", result: "success"}
```

- Flask 프로젝트 화면에는 어떤 메시지가 뜨나요?



```
Run: app
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 325-081-989
127.0.0.1 - [17/Jul/2020 16:49:16] "GET / HTTP/1.1" 200 -
127.0.0.1 - [17/Jul/2020 16:52:58] "GET /test?title_give=봄날은간다 HTTP/1.1" 200 -
봄날은간다
```



어? '봄날은간다'는 어떻게 출력된 걸까요?

바로 클라이언트가 보내준 데이터인 `title_give`라는 key의 값입니다.
(요청 URL의 ? 뒤에 'title_give'=봄날은간다)

서버에서는 요청 데이터를 key 이름인 `title_give`로 찾아서 변수에 저장 후, 출력했네요.
(`request.args.get('title_give')`)

▼ [💻 코드 API 만들기 02-1] POST 방식 API 만들기



만들 API 정보

A. 요청 정보

- 요청 URL = `/test`, 요청 방식 = `POST`
- 요청 데이터 : 없음

B. 서버가 제공할 기능 : 클라이언트에게 정해진 메시지를 보냄

C. 응답 데이터 : (JSON 형식) 'result'='success', 'msg'='이 요청은 POST!'

- `app.py`에 분홍 형광펜 부분을 추가해주세요. 서버가 실행되어 있다면, 자동으로 반영될 겁니다.

```
from flask import Flask, render_template, jsonify, request

app = Flask(__name__)

## HTML을 주는 부분
@app.route('/')
def home():
    return render_template('index.html')

## API 역할을 하는 부분
@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!'})
```

```

@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!'})

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

```

▼ [💻 코드 API 만들기 02-2] POST 방식 API 사용하기



사용할 API 정보

A. 요청 정보

- 요청 URL = `/test`, 요청 방식 = `POST`
- 요청 데이터 : 없음

B. 서버가 제공할 기능 : 클라이언트에게 정해진 메시지를 보냄

C. 응답 데이터 : (JSON 형식) 'result'= 'success', 'msg'= '이 요청은 POST!'

- 크롬 개발자도구 - console창을 열어 아래 코드를 실행해주세요.
- 역시 Ajax를 쓰기 위해서, 크롬에 `localhost:5000/` 를 입력해서 뜨는 `index.html` 을 사용해보겠습니다.

```

$.ajax({
  type: "POST",
  url: "/test",
  data: { title_give:"봄날은간다" },
  success: function(response){
    console.log(response)
  }
})

```

- 크롬 console 화면에 어떤 메시지가 뜨나요?

```

> $.ajax({
  type: "POST",
  url: "/test",
  data: { title_give:'봄날은간다' },
  success: function(response){
    console.log(response)
  }
})
<   readyState: 1, getResponseHeader: f, get
  rs: f, setRequestHeader: f, overrideMimeType:
    > {msg: "이 요청은 POST!", result: "success"}

```

- Flask 프로젝트 화면에는 어떤 메시지가 뜨나요?

```

127.0.0.1 - - [17/Jul/2020 16:49:16] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Jul/2020 16:52:50] "GET /test?title_give=봄날은간다"
127.0.0.1 - - [17/Jul/2020 17:03:33] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Jul/2020 17:05:57] "POST /test HTTP/1.1" 200 -

```



POST 방식에서는 요청 데이터를 요청하는 방식과
서버에서 요청 데이터를 확인하는 코드가 조금 다르게 생겼죠?

클라이언트쪽 JQuery 코드에서 `data` 항목에 넣어주고
서버에서는 요청 데이터를 `request.form['title_give']` 로 확인합니다.



위에서는 요청 데이터를 `print` 로 출력해 확인했습니다.
실전에선 요청 데이터로 여러가지 작업을 할 수 있겠죠?
(예. 요청 데이터 `title_give` 인 제목의 영화 정보만 보내주기)

[1시간] : 모두의 책리뷰 완성하기



앞으로 [👉튜터님과 함께] 다시 한 번 차근차근 만들어 나가볼거에요.
`sparta > projects > bookreview` 가 프로젝트 폴더!

▼ 10) 문제 분석 - 완성작 분석하기

👉 완성작 보러가기

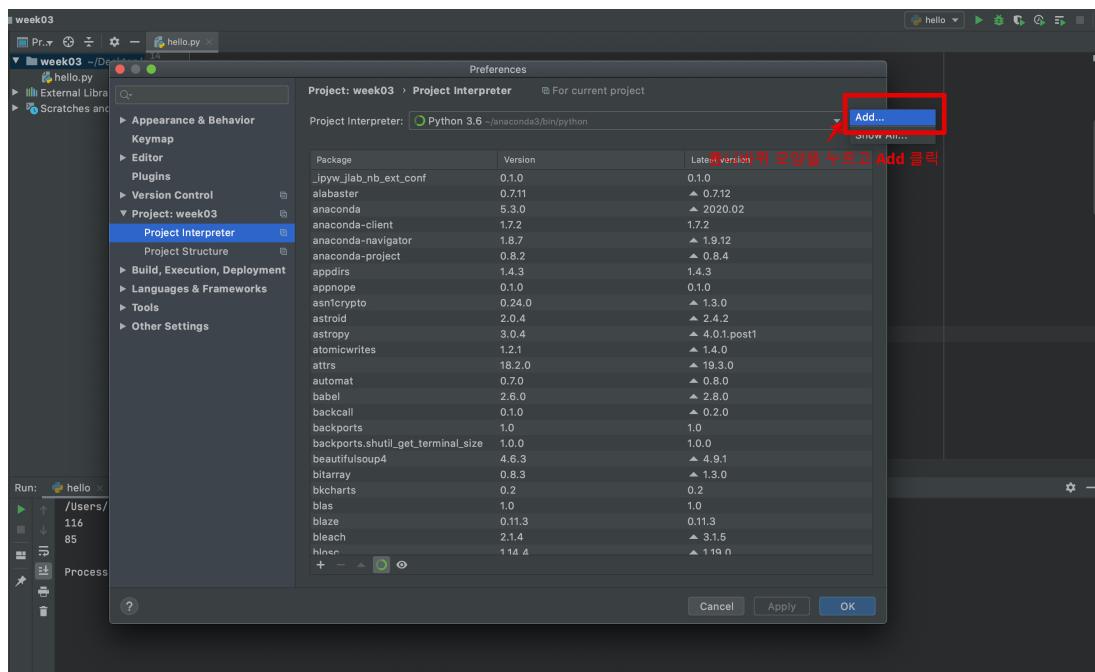
- 자유롭게 사용하며 어떤 기능이 있는지 관찰해봅시다
- 기능
 - 리뷰 작성하기 (Create)
 - 리뷰 보기 (Read)

▼ 11) 프로젝트 설정 - mongoDB 실행 확인 + 새 프로젝트니까, 가상환경 설정하기

- 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!
- Pycharm에서 File - Open에서 `sparta/projects/bookreview` 폴더를 선택

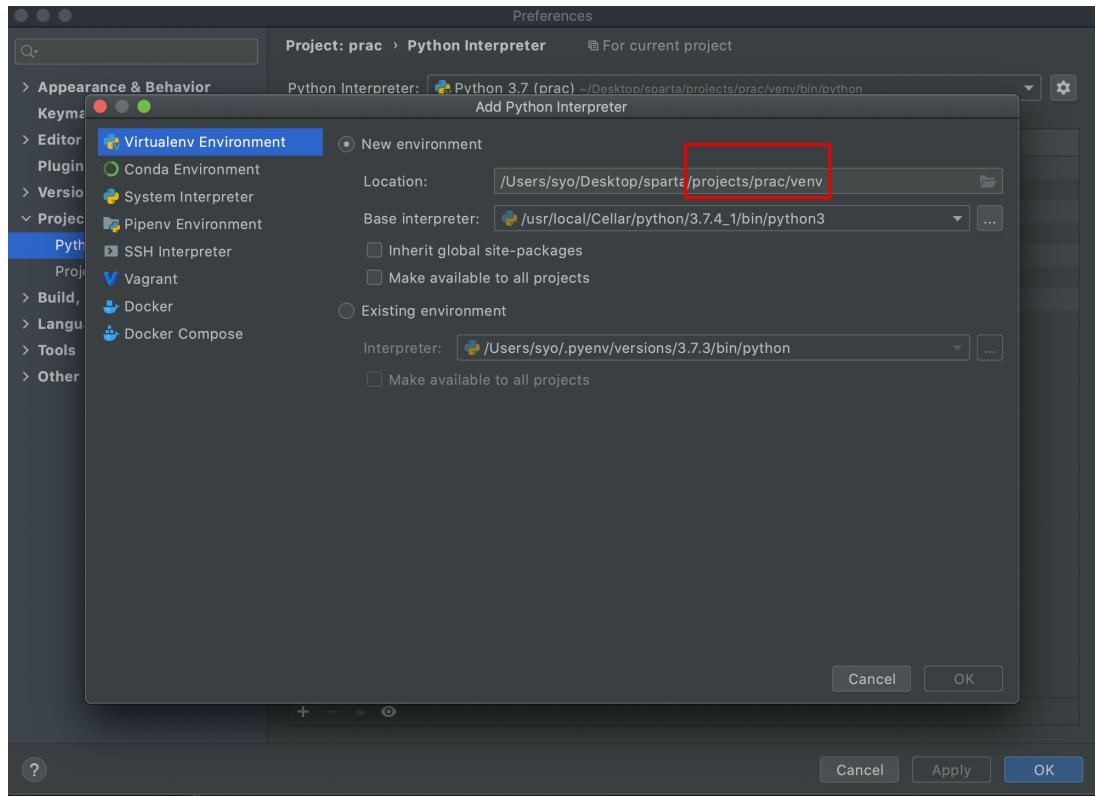
▼ 1. 가상 환경 설치 및 확인

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



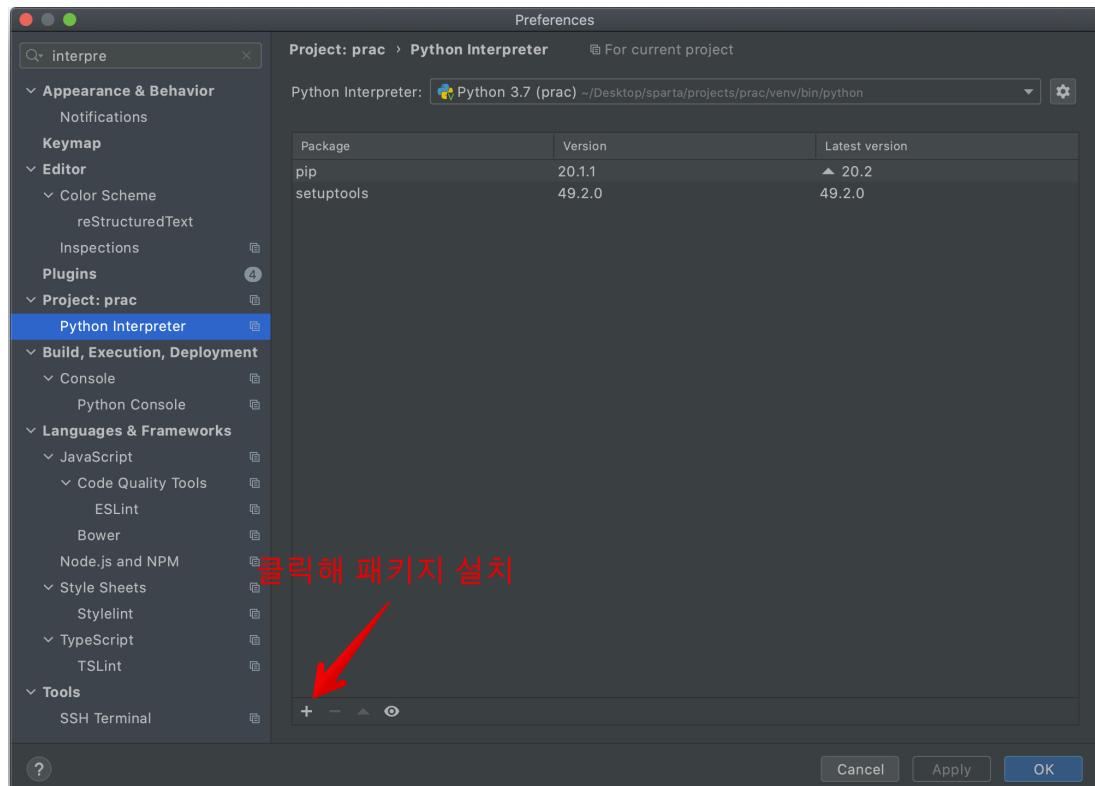
- virtual environment - New environment 선택

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

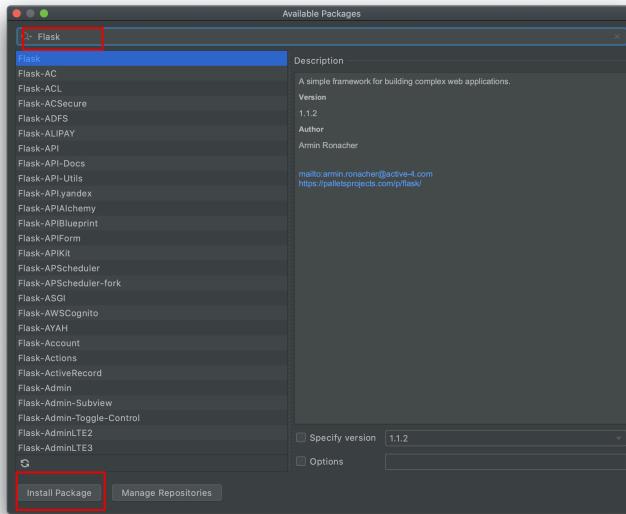


▼ 2. pip(python install package) 사용 - Flask 패키지 설치해보기

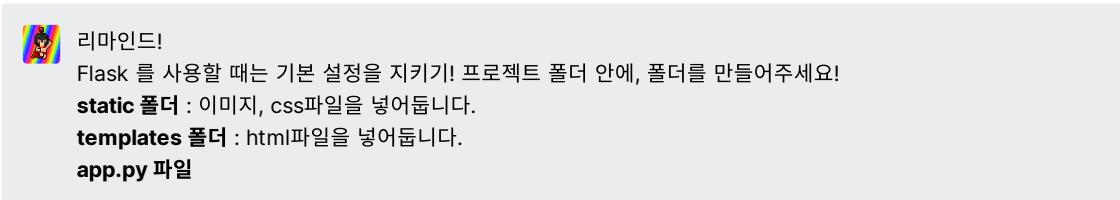
- Flask를 사용하기 위해서 패키지를 설치해보겠습니다.
- project interpreter 화면에서 + 버튼을 누르면 아래 창이 뜹니다!



- flask로 검색한 후, Install package 클릭



▼ 3. Flask 기본 폴더 구조 만들기



- 아래 모양처럼 만들어주세요!



▼ 12) 프로젝트 준비 - `app.py` 기본 준비하기 & 디버깅(Debugging)

- bookreview 폴더 안에 `app.py` 파일 만들기

```
from flask import Flask, render_template, jsonify, request
from pymongo import MongoClient # pymongo를 임포트 하기(패키지 인스톨 먼저 해야겠죠?)

app = Flask(__name__)

client = MongoClient('localhost', 27017) # mongoDB는 27017 포트로 돌아갑니다.
db = client.dbsparta # 'dbsparta'라는 이름의 db를 만듭니다.

## HTML을 주는 부분
@app.route('/')
def home():
    return render_template('index.html')

## API 역할을 하는 부분
@app.route('/review', methods=['POST'])
def write_review():
```

```

# 1. 클라이언트가 준 title, author, review 가져오기.
# 2. DB에 정보 삽입하기
# 3. 성공 여부 & 성공 메시지 반환하기
return jsonify({'result': 'success', 'msg': '이 요청은 POST!'})

@app.route('/review', methods=['GET'])
def read_reviews():
    return jsonify({'result': 'success', 'msg': '이 요청은 GET!'})

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

```

2. 에러 분석

- 위 코드를 쓰면, 두번째 줄(`from pymongo...`)에 빨간 밑줄이 뜰 거예요. Pycharm에서 빨간 밑줄은 꼭 고치고 넘어가야 하는 에러를 의미해요.

```
from pymongo import MongoClient # pymongo를 임포트 하기(패키지 인스톨 먼저 해야겠죠?)
```

- 일단은 실행해봅시다!

우리는 아직 배우는 과정이니까 디버깅(에러 고치기)을 익히기 위해 일단은 에러를 고치지 않고 실행시켜볼게요. 실행시키면 `Traceback`으로 시작되는 에러 메시지가 Pycharm 하단 창에 뜰 거예요.

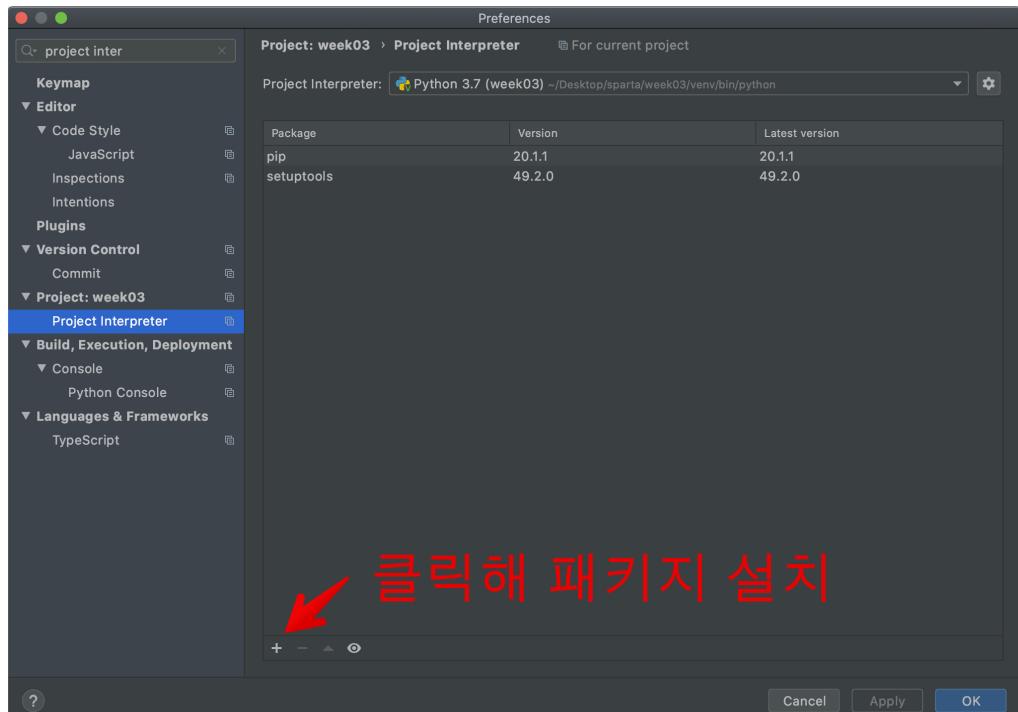
```

Traceback (most recent call last):
  File "/.../sparta/projects/bookreview/venv/lib/python3.7/site-packages/flask/cli.py",
    __import__(module_name)
  File ".../sparta/projects/bookreview/app.py", line 2, in <module>
    from pymongo import MongoClient # pymongo를 임포트 하기(패키지 인스톨 먼저 해야겠죠?)
ModuleNotFoundError: No module named 'pymongo'
```

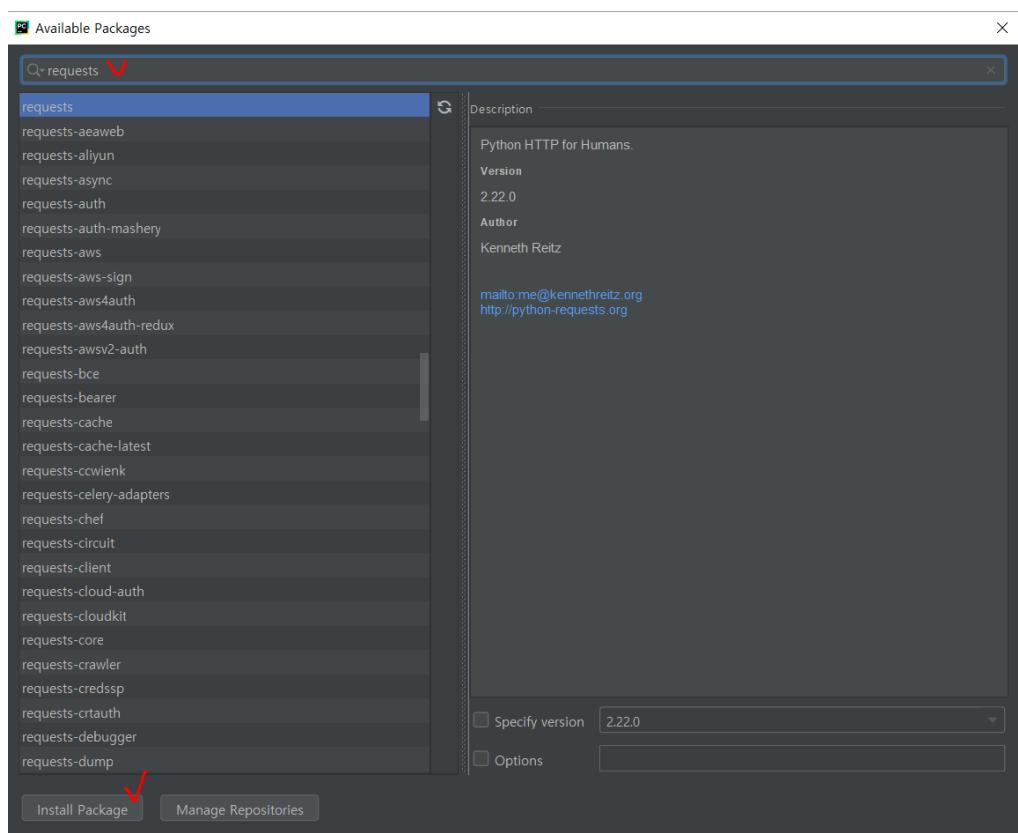
- 파이썬에서는 기본적으로 에러 메세지가 `Traceback`로 시작되어 쭉 뜨게 되어있습니다. 맨 마지막 줄에 ~Error라고 나오죠? 이 에러 메시지(`ModuleNotFoundError: No module named 'pymongo'`)를 구글링해보면 해결방법이 나올 거예요. 앞으로 실습을 하거나 내 프로젝트를 할 때 여러 번 해보면서 익숙해질겁니다!

3. 에러 해결하기

- `ModuleNotFoundError: No module named 'pymongo'` 에러가 발생하는 이유는 `pymongo` 패키지가 설치되어 있지 않기 때문이에요.
- Project Interpreter 창에서 `pymongo` 패키지를 설치하겠습니다.
 - ▼ 참고. Pycharm에서 패키지 설치하기
 - project interpreter 화면에서 + 버튼을 누르면 아래 창이 뜹니다!



- `pymongo` 로 검색한 후, Install package 클릭



4. 에러 해결했는지 확인하기

- `app.py` 를 실행해보세요. 이제 Traceback 에러 메시지가 뜨지 않죠? 코드에서도 빨간 밑줄이 사라졌을거에요.

▼ 13) 프로젝트 준비 - `index.html` 기본 준비하기



HTML 코드는 미리 준비해두었어요.
templates 폴더 아래 index.html에 다음 내용을 복사 붙여넣기 합니다.

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

    <head>
        <!-- Webpage Title -->
        <title>모두의 책리뷰 | 스파르타코딩클럽</title>

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

        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
              integrity="sha384-Gn5384xqQ1aoXA+058RXPxPg6fy4IwvTNh0E263XmFcJ1SAwiGgFAW/dAiS6JXm"
              crossorigin="anonymous">

        <!-- JS -->
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
               integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
               crossorigin="anonymous"></script>

        <!-- 구글폰트 -->
        <link href="https://fonts.googleapis.com/css?family=Do+Hyeon&display=swap" rel="stylesheet">

        <script type="text/javascript">

            $(document).ready(function () {
                $("#reviews-box").html("");
                showReview();
            });

            function makeReview() {
                // 1. 제목, 저자, 리뷰 내용을 가져옵니다.
                // 2. 제목, 저자, 리뷰 중 하나라도 입력하지 않았을 경우 alert를 띄웁니다.
                // 3. POST /review에 저장을 요청합니다.
                $.ajax({
                    type: "POST",
                    url: "/review",
                    data: {},
                    success: function (response) {
                        if (response["result"] == "success") {
                            alert(response["msg"]);
                            window.location.reload();
                        }
                    }
                })
            }

            function showReview() {
                // 1. 리뷰 목록을 서버에 요청하기
                // 2. 요청 성공 여부 확인하기
                // 3. 요청 성공했을 때 리뷰를 올바르게 화면에 나타내기
                $.ajax({
                    type: "GET",
                    url: "/review",
                    data: {},
                    success: function (response) {
                        if (response["result"] == "success") {
                            alert(response["msg"]);
                        } else {
                            alert("리뷰를 받아오지 못했습니다");
                        }
                    }
                })
            }

            function validateLength(obj) {
                let content = $(obj).val();
                if (content.length > 140) {
                    alert("리뷰는 140자까지 기록할 수 있습니다.");
                    $(obj).val(content.substring(0, content.length - 1));
                }
            }
        </script>

        <style type="text/css">
            * {
                font-family: "Do Hyeon", sans-serif;

```

```

        }

        h1,
        h5 {
            display: inline;
        }

        .info {
            margin-top: 20px;
            margin-bottom: 20px;
        }

        .review {
            text-align: center;
        }

        .reviews {
            margin-top: 100px;
        }
    </style>
</head>

<body>
    <div class="container">
        
        <div class="info">
            <h1>읽은 책에 대해 말씀해주세요.</h1>
            <p>다른 사람을 위해 리뷰를 남겨주세요! 다 같이 좋은 책을 읽는다면 다 함께 행복해질 수 있지 않을까요?</p>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">제목</span>
                </div>
                <input type="text" class="form-control" id="title">
            </div>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">저자</span>
                </div>
                <input type="text" class="form-control" id="author">
            </div>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">리뷰</span>
                </div>
                <textarea class="form-control" id="bookReview"
                    cols="30"
                    rows="5" placeholder="140자까지 입력할 수 있습니다." onkeyup="validateLength(this)"></textarea>
            </div>
            <div class="review">
                <button onclick="makeReview()" type="button" class="btn btn-primary">리뷰 작성하기</button>
            </div>
        </div>
        <div class="reviews">
            <table class="table">
                <thead>
                    <tr>
                        <th scope="col">제목</th>
                        <th scope="col">저자</th>
                        <th scope="col">리뷰</th>
                    </tr>
                </thead>
                <tbody id="reviews-box">
                    <tr>
                        <td>왕초보 8주 코딩</td>
                        <td>김르단</td>
                        <td>역시 왕초보 코딩교육의 명가답군요. 따라하다보니 눈 깜빡할 사이에 8주가 지났습니다.</td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
</body>

</html>

```

▼ 14) API 만들고 사용하기 - 제목, 저자, 리뷰 정보 저장하기(Create → POST)



서버에서 API를 만들었으면, API 를 사용하는 클라이언트 쪽이 있겠죠?

아래 클라이언트/서버 코드는 쌍을 이루고 있습니다.
prac 프로젝트에서도 API클라이언트 세트로 사용했었습니다.

▼ 1. 클라이언트와 서버 확인하기

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



만들어져 있는 API 정보

- 1. 요청 정보 :** 요청 URL= `/review` , 요청 방식 = `POST`
- 2. 서버가 제공할 기능 :** 클라이언트에게 정해진 메시지를 보냄
- 3. 응답 데이터 :** (JSON 형식) 'result'= 'success', 'msg'= '리뷰가 성공적으로 작성되었습니다.'

[서버 코드 - `app.py`]

```
## API 역할을 하는 부분
@app.route('/review', methods=['POST'])
def write_review():
    # 1. 클라이언트가 준 title, author, review 가져오기.
    # 2. DB에 정보 삽입하기
    # 3. 성공 여부 & 성공 메시지 반환하기
    return jsonify({'result': 'success', 'msg': '리뷰가 성공적으로 작성되었습니다.'})
```

[클라이언트 코드 - `index.html`]

```
function makeReview() {
    // 1. 제목, 저자, 리뷰 내용을 가져옵니다.
    // 2. 제목, 저자, 리뷰 중 하나라도 입력하지 않았을 경우 alert를 띄웁니다.
    // 3. POST /review 에 저장을 요청합니다.
    $.ajax({
        type: "POST",
        url: "/review",
        data: { },
        success: function (response) {
            if (response["result"] == "success") {
                alert(response["msg"]);
                window.location.reload();
            }
        }
    })
}
```



동작 테스트

'리뷰 시작하기' 버튼을 눌렀을 때, '리뷰가 성공적으로 작성되었습니다!'라는 내용의 alert창이 뜨면 클라이언트 코드와 서버 코드가 연결 되어있는 것입니다.

▼ 2. 서버부터 만들기



API 는 약속이라고 했습니다. API를 먼저 만들어보죠!

리뷰를 작성하기 위해 필요한 정보는 다음 세 가지입니다.

- 제목(title)
- 저자(author)
- 리뷰(review)

따라서 API 기능은 다음 세 단계로 구성되어야 합니다.

1. 클라이언트가 준 title, author, review 가져오기.
2. DB에 정보 삽입하기
3. 성공 여부 & 성공 메시지 반환하기



정리하면, 만들 API 정보는 아래와 같습니다.

A. 요청 정보

- 요청 URL= `/review` , 요청 방식 = `POST`
- 요청 데이터 : 제목(title), 저자(author), 리뷰(review)

B. 서버가 제공할 기능 : 클라이언트에게 보낸 요청 데이터를 데이터베이스에 생성(Create)하고, 저장이 성공했다고 응답 데이터를 보냄

C. 응답 데이터 : (JSON 형식) 'result'= 'success', 'msg'= '리뷰가 성공적으로 작성되었습니다.'

```
@app.route('/review', methods=['POST'])
def write_review():
    # title_receive로 클라이언트가 준 title 가져오기
    title_receive = request.form['title_give']
    # author_receive로 클라이언트가 준 author 가져오기
    author_receive = request.form['author_give']
    # review_receive로 클라이언트가 준 review 가져오기
    review_receive = request.form['review_give']

    # DB에 삽입할 review 만들기
    review = {
        'title': title_receive,
        'author': author_receive,
        'review': review_receive
    }
    # reviews에 review 저장하기
    db.reviews.insert_one(review)
    # 성공 여부 & 성공 메시지 반환
    return jsonify({'result': 'success', 'msg': '리뷰가 성공적으로 작성되었습니다.'})
```

▼ 3. 클라이언트 만들기



API 는 약속이라고 했습니다. API를 사용할 클라이언트를 만들어보죠!

리뷰를 작성하기 위해 필요한 정보는 다음 세 가지입니다.

- 제목(title)
- 저자(author)
- 리뷰(review)

따라서 클라이언트 코드는 다음 세 단계로 구성되어야 합니다.

1. input에서 title, author, review 가져오기
2. 입력값이 하나라도 없을 때 alert 띄우기.
3. Ajax로 서버에 저장 요청하고, 화면 다시 로딩하기



사용할 API 정보

A. 요청 정보

- 요청 URL = `/review`, 요청 방식 = `POST`
- 요청 데이터 : 제목(title), 저자(author), 리뷰(review)

B. 서버가 제공할 기능 : 클라이언트에게 보낸 요청 데이터를 데이터베이스에 생성(Create)하고, 저장이 성공했다고 응답 데이터를 보냄

C. 응답 데이터 : (JSON 형식) 'result' = 'success', 'msg' = '리뷰가 성공적으로 작성되었습니다.'

```
function makeReview() {  
    // 1. 화면에 입력어 있는 제목, 저자, 리뷰 내용을 가져옵니다.  
    let title = $("#title").val();  
    let author = $("#author").val();  
    let review = $("#bookReview").val();  
  
    // 2. 제목, 저자, 리뷰 중 하나라도 입력하지 않았을 경우 alert를 띄웁니다.  
    if (title == "") {  
        alert("제목을 입력해주세요");  
        $("#title").focus();  
        return;  
    } else if (author == "") {  
        alert("저자를 입력해주세요");  
        $("#author").focus();  
        return;  
    } else if (review == "") {  
        alert("리뷰를 입력해주세요");  
        $("#bookReview").focus();  
        return;  
    }  
  
    // 3. POST /review 에 저장(Create)을 요청합니다.  
    $.ajax({  
        type: "POST",  
        url: "/review",  
        data: { title_give: title, author_give: author, review_give: review },  
        success: function (response) {  
            if (response["result"] == "success") {  
                alert(response["msg"]);  
                window.location.reload();  
            }  
        }  
    })  
}
```

▼ 4. 완성 확인하기



동작 테스트

제목, 저자, 리뷰를 작성하고 '리뷰 작성하기' 버튼을 눌렀을 때,
'리뷰가 성공적으로 작성되었습니다.'라는 alert가 뜨는지 확인합니다.

▼ 15) API 만들고 사용하기 - 저장된 리뷰를 화면에 보여주기(Read → GET)



아래 클라이언트/서버 코드도 쌍을 이루고 있습니다.

▼ 1. 클라이언트와 서버 확인하기

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



만들어져 있는 API 정보

1. 요청 정보 : 요청 URL = `/review`, 요청 방식 = `GET`
2. 서버가 제공할 기능 : 클라이언트에게 정해진 메시지를 보냄
3. 응답 데이터 : (JSON 형식) 'result' = 'success'

[서버 코드 - `app.py`]

```
@app.route('/review', methods=['GET'])
def read_reviews():
    # 1. 모든 reviews의 문서를 가져온 후 list로 변환합니다.
    # 2. 성공 메시지와 함께 리뷰를 보냅니다.
    return jsonify({'result': 'success'})
```

[클라이언트 코드 - `index.html`]

```
function showReview() {
    // 1. 리뷰 목록을 서버에 요청하기
    // 2. 요청 성공 여부 확인하기
    // 3. 요청 성공했을 때 리뷰를 올바르게 화면에 나타내기
    $.ajax({
        type: "GET",
        url: "/review",
        data: {},
        success: function (response) {
            if (response['result'] == 'success') {
                alert('리뷰를 받아왔습니다.');
                // 2. 성공했을 때 리뷰를 올바르게 화면에 나타내기
            } else {
                alert('리뷰를 받아오지 못했습니다');
            }
        }
    })
}
```



동작 테스트

화면을 새로고침 했을 때, '리뷰를 받아왔습니다.'라는 내용의 alert창이 뜨면
클라이언트 코드와 서버 코드가 연결 되어있는 것입니다.

▼ 2. 서버부터 만들기



API 는 약속이라고 했습니다. API를 먼저 만들어보죠!

API 기능은 다음 단계로 구성되어야 합니다.

1. DB에서 리뷰 정보 모두 가져오기
2. 성공 여부 & 리뷰 목록 반환하기



정리하면, 만들 API 정보는 아래와 같습니다.

A. 요청 정보

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

B. 서버가 제공할 기능 : 데이터베이스에 리뷰 정보를 조회(Read)하고, 성공 메시지와 리뷰 정보를 응답 데이터를 보냄

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

```

@app.route('/review', methods=['GET'])
def read_reviews():
    # 1. DB에서 리뷰 정보 모두 가져오기
    reviews = list(db.reviews.find({}, {'_id': 0}))
    # 2. 성공 여부 & 리뷰 목록 반환하기
    return jsonify({'result': 'success', 'reviews': reviews})

```

▼ 3. 클라이언트 만들기



API 는 약속이라고 했습니다. API를 사용할 클라이언트를 만들어보죠!

리뷰를 작성하기 위해 필요한 정보는 다음 세 가지입니다.

- 제목(title)
- 저자(author)
- 리뷰(review)

따라서 클라이언트 코드는 다음 세 단계로 구성되어야 합니다.

1. 리뷰 목록을 서버에 요청하기
2. 요청 성공 여부 확인하기
3. 요청 성공했을 때 리뷰를 올바르게 화면에 나타내기



사용할 API 정보는 아래와 같습니다.

A. 요청 정보

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

B. 서버가 제공할 기능 : 데이터베이스에 리뷰 정보를 조회(Read)하고, 성공 메시지와 리뷰 정보를 응답 데이터를 보냄

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

```

function showReview() {
    // 1. 리뷰 목록을 서버에 요청하기
    $.ajax({
        type: "GET",
        url: "/review",
        data: {},
        success: function (response) {
            // 2. 요청 성공 여부 확인하기
            if (response["result"] == "success") {
                let reviews = response["reviews"];
                // 3. 요청 성공했을 때 리뷰를 올바르게 화면에 나타내기
                for (let i = 0; i < reviews.length; i++) {
                    makeCard(reviews[i]["title"], reviews[i]["author"], reviews[i]["review"]);
                }
            } else {
                alert("리뷰를 받아오지 못했습니다.");
            }
        }
    })
}

function makeCard(title, author, review) {
    let tempHtml = `<tr>
        <td>${title}</td>
        <td>${author}</td>
        <td>${review}</td>
    </tr>`;
    $("#reviews-box").append(tempHtml);
}

```

▼ 4. 완성 확인하기



동작 테스트

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

▼ 16) 전체 완성 코드!



프로젝트 API 정보는 아래와 같습니다.

[리뷰 저장하기(Create)]

A. 요청 정보

- 요청 URL = `/review`, 요청 방식 = `POST`
- 요청 데이터 : 제목(title), 저자(author), 리뷰(review)

B. 서버가 제공할 기능 : 클라이언트에게 보낸 요청 데이터를 데이터베이스에 생성(Create)하고, 저장이 성공했다고 응답 데이터를 보냄

C. 응답 데이터 : (JSON 형식) 'result'='success', 'msg'='리뷰가 성공적으로 작성되었습니다.'

[리뷰 보여주기(Read)]

A. 요청 정보

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

B. 서버가 제공할 기능 : 데이터베이스에 리뷰 정보를 조회(Read)하고, 성공 메시지와 리뷰 정보를 응답 데이터를 보냄

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

▼ [💻 코드 - 서버 `app.py`]

```
from flask import Flask, render_template, jsonify, request
from pymongo import MongoClient # pymongo를 임포트 하기(패키지 인스톨 먼저 해야겠죠?)

app = Flask(__name__)

client = MongoClient('localhost', 27017) # mongoDB는 27017 포트로 돌아갑니다.
db = client.dbsparta # 'dbsparta'라는 이름의 db를 만듭니다.

## HTML을 주는 부분
@app.route('/')
def home():
    return render_template('index.html')

## API 역할을 하는 부분
@app.route('/review', methods=['POST'])
def write_review():
    # title_receive로 클라이언트가 준 title 가져오기
    title_receive = request.form['title_give']
    # author_receive로 클라이언트가 준 author 가져오기
    author_receive = request.form['author_give']
    # review_receive로 클라이언트가 준 review 가져오기
    review_receive = request.form['review_give']

    # DB에 삽입할 review 만들기
    review = {
        'title': title_receive,
        'author': author_receive,
        'review': review_receive
    }
    # reviews에 review 저장하기
    db.reviews.insert_one(review)
    # 성공 여부 & 성공 메시지 반환
    return jsonify({'result': 'success', 'msg': '리뷰가 성공적으로 작성되었습니다.'})

@app.route('/review', methods=['GET'])
def read_reviews():
    # 1. DB에서 리뷰 정보 모두 가져오기
    reviews = list(db.reviews.find({}, {'_id': 0}))
```

```

# 2. 성공 여부 & 리뷰 목록 반환하기
return jsonify({'result': 'success', 'reviews': reviews})

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

```

▼ [코드 - 클라이언트 `index.html`]

```

<!DOCTYPE html>
<html lang="ko">

    <head>
        <!-- Webpage Title -->
        <title>모두의 책리뷰 | 스파르타코딩클럽</title>

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

        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
              integrity="sha384-Gn5zXq1oWXA+058RXPxPg6fy4IWvTNh0E263XmFcJ1SAwiGgFAW/dAiS6JXm"
              crossorigin="anonymous">

        <!-- JS -->
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/umd/popper.min.js"
               integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
               crossorigin="anonymous"></script>

        <!-- 구글폰트 -->
        <link href="https://fonts.googleapis.com/css?family=Do+Hyeon&display=swap" rel="stylesheet">

        <script type="text/javascript">

            $(document).ready(function () {
                $("#reviews-box").html("");
                showReview();
            });

            function makeReview() {
                // 1. 제목, 저자, 리뷰 내용을 가져옵니다.
                let title = $("#title").val();
                let author = $("#author").val();
                let review = $("#bookReview").val();

                // 2. 제목, 저자, 리뷰 중 하나라도 입력하지 않았을 경우 alert를 띄웁니다.
                if (title === "") {
                    alert("제목을 입력해주세요");
                    $("#title").focus();
                    return;
                } else if (author === "") {
                    alert("저자를 입력해주세요");
                    $("#author").focus();
                    return;
                } else if (review === "") {
                    alert("리뷰를 입력해주세요");
                    $("#bookReview").focus();
                    return;
                }

                // 3. POST /review에 저장을 요청합니다.
                $.ajax({
                    type: "POST",
                    url: "/review",
                    data: {title_give: title, author_give: author, review_give: review},
                    success: function (response) {
                        if (response["result"] === "success") {
                            alert(response["msg"]);
                            window.location.reload();
                        }
                    }
                })
            }

            function showReview() {
                // 1. 리뷰 목록을 서버에 요청하기
                $.ajax({
                    type: "GET",
                    url: "/review",
                    data: {},
                    success: function (response) {
                        // 2. 요청 성공 여부 확인하기

```

```

        if (response["result"] == "success") {
            let reviews = response["reviews"];
            // 3. 요청 성공했을 때 리뷰를 물바르게 화면에 나타내기
            for (let i = 0; i < reviews.length; i++) {
                makeCard(reviews[i]["title"], reviews[i]["author"], reviews[i]["review"]);
            }
        } else {
            alert("리뷰를 받아오지 못했습니다.");
        }
    })
}

function makeCard(title, author, review) {
    let tempHtml = `<tr>
        <td>${title}</td>
        <td>${author}</td>
        <td>${review}</td>
    </tr>`;
    $("#reviews-box").append(tempHtml);
}

function validateLength(obj) {
    let content = $(obj).val();
    if (content.length > 140) {
        alert("리뷰는 140자까지 기록할 수 있습니다.");
        $(obj).val(content.substring(0, content.length - 1));
    }
}
</script>

<style type="text/css">
    * {
        font-family: "Do Hyeon", sans-serif;
    }

    h1,
    h5 {
        display: inline;
    }

    .info {
        margin-top: 20px;
        margin-bottom: 20px;
    }

    .review {
        text-align: center;
    }

    .reviews {
        margin-top: 100px;
    }
</style>
</head>

<body>
    <div class="container">
        
        <div class="info">
            <h1>읽은 책에 대해 말씀해주세요.</h1>
            <p>다른 사람을 위해 리뷰를 남겨주세요! 다 같이 좋은 책을 읽는다면 다 함께 행복해질 수 있지 않을까요?</p>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">제목</span>
                </div>
                <input type="text" class="form-control" id="title">
            </div>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">저자</span>
                </div>
                <input type="text" class="form-control" id="author">
            </div>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">리뷰</span>
                </div>
                <textarea class="form-control" id="bookReview"
                    cols="30"
                    rows="5" placeholder="140자까지 입력할 수 있습니다." onkeyup="validateLength(this)"></textarea>
            </div>
            <div class="review">
                <button onclick="makeReview()" type="button" class="btn btn-primary">리뷰 작성하기</button>
            </div>
        </div>
    </div>

```

```

<div class="reviews">
  <table class="table">
    <thead>
      <tr>
        <th scope="col">제목</th>
        <th scope="col">저자</th>
        <th scope="col">리뷰</th>
      </tr>
    </thead>
    <tbody id="reviews-box">
      <tr>
        <td>왕초보 8주 코딩</td>
        <td>김르탄</td>
        <td>역시 왕초보 코딩교육의 명가답군요. 따라하다보니 눈 깜빡할 사이에 8주가 지났습니다.</td>
      </tr>
    </tbody>
  </table>
</div>
</body>

</html>

```



와, 우리 손으로 직접 API 를 만들고 사용해봤어요!
벌써 연습 프로젝트와 모두의 책리뷰 두 가지 프로젝트를 완성했네요!

신난다! 이제 후반부에서는 한번 더 반복! 나홀로메모장을 완성해볼게요.

😎 문제뱅크

- 9) [👉튜터와 함께] API 맛보기 를 빠르게 다시 한 번 복습해보세요.

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 - 도구 잘 사용하기

▼ 도구 잘 사용하기

- 도구를 손처럼 사용한다면, 생산성이 올라간답니다! 하나의 도구를 강력하게 잘 사용해보세요. 도구 단축키와 다양한 기능들을요!
- 우리가 사용하는 Pycharm 과 크롬 개발자 도구를 잘 사용하기 위한 간단한 튜토리얼을 소개합니다.
- Pycharm 시작하기

리소스 - 문서 | PyCharm

리팩토링, 디버거, 코드 완성, 즉석 코드 분석, 코딩 생산성을 지원하는 지능적인 Python IDE

🔗 <https://www.jetbrains.com/ko-kr/pycharm/documentation/>

PyCharm

The Python IDE
for Professional Developers



- 크롬 개발자 도구 사용하기

Chrome DevTools | Google Developers

Chrome DevTools는 Google Chrome에 내장되어있는 웹 저작 및 디버깅 도구셋입니다. DevTools를 이용하여 사이트를 반복하고, 디버깅하고, 프로파일링할 수 있습니다. 많은 DevTools 문서는 최신 Chrome 피쳐를 제공하는 Note: Chrome Canary 를 기반으로 작성했습니다. 완전히 반응하는 모바일 우선 웹 경험을 만드는

👉 <https://developers.google.com/web/tools/chrome-devtools?hl=ko>



크롬 개발자 도구 101

썸머노트를 개발하며 늘 사용하는 크롬 개발자 도구에 대해 한 번도 제대로 공부해본 적이 없었는데, 마침 아살(@ahastudio)님이 "아듀 2018"이라는 일종의 Advent Calender를 주최하는 것을 보고 재미있겠다 싶어 이걸 기회 삼아 크롬 개발자 도구의 기초 내용을 정리해보게 되었다. 크롬 개발자 도구에 대한 정보는

👉 <https://lqez.github.io/blog/chrome-dev-tool-101.html>



Chrome으로 디버깅하기

좀 더 복잡한 코드를 작성하기 전에, 디버깅이란 것에 대해 이야기해봅시다. 디버깅(debugging)은 스크립트 내 에러를 검출해 제거하는 일련의 과정을 의미합니다. 모던 브라우저와 호스트 환경 대부분은 개발자 도구 안에 UI 형태로 디버깅 툴을 구비해 놓습니다. 디버깅 툴을 사용하면 디버깅이 훨씬 쉬워지고, 실행 단계마다 어

👉 <https://ko.javascript.info/debugging-chrome>



[1.5시간] 나홀로메모장 완성하기



앞으로 [👉튜터님과 함께] 단계별로 차근차근 만들어 나갈 거예요.
sparta > projects > alonememo 가 프로젝트 폴더!

▼ 17) 문제 분석 - 완성작부터 보기!

👉 완성작 보러가기

- 기능
 - 포스팅하기 - 카드 생성(Create)
 - 리스팅하기 - 저장된 카드 보여주기(Read)

▼ 18) 프로젝트 설계 - 만들 API 설계



포스팅API - 카드 생성 (Create)

A. 요청 정보

- 요청 URL = `/memo`, 요청 방식 = `POST`
- 요청 데이터 : 제목(url_give), 코멘트(comment_give)

B. 서버가 제공할 기능

- URL의 meta태그 정보를 바탕으로 제목, 설명, 이미지URL 스크래핑
- (제목, 설명, URL, 이미지URL, 코멘트) 정보를 모두 DB에 저장

C. 응답 데이터

- API가 정상적으로 작동하는지 클라이언트에게 알려주기 위해서 성공 메시지 보내기
- (JSON 형식) 'result' = 'success'



리스팅API - 저장된 카드 보여주기 (Read)

A. 요청 정보

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

B. 서버가 제공할 기능

- DB에 저장돼있는 모든 (제목, 설명, URL, 이미지URL, 코멘트) 정보를 가져오기

C. 응답 데이터

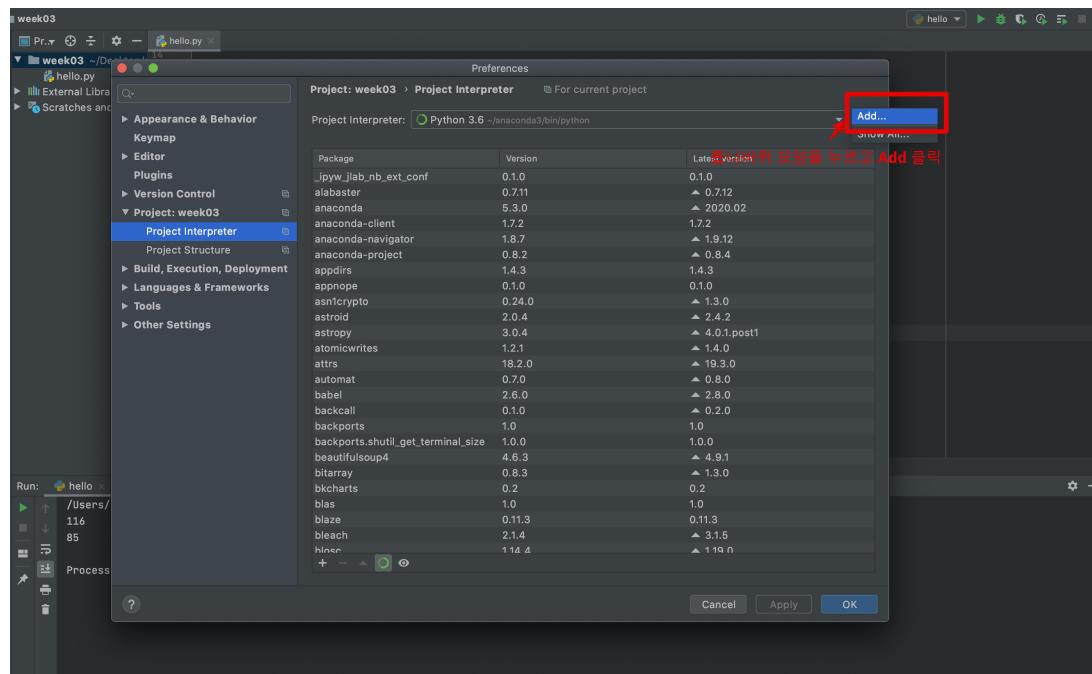
- API 동작 잘했다는 성공 메시지, 아티클(기사)들의 정보(제목, 설명, URL, 이미지URL, 코멘트)
- (JSON 형식) 'result' = 'success', 'articles': 아티클 정보

▼ 19) 프로젝트 설정 - 새 프로젝트니까, 가상환경 설정하기 + mongoDB 실행 확인

- Pycharm에서 File - Open에서 `sparta/projects/alonememo` 폴더를 선택

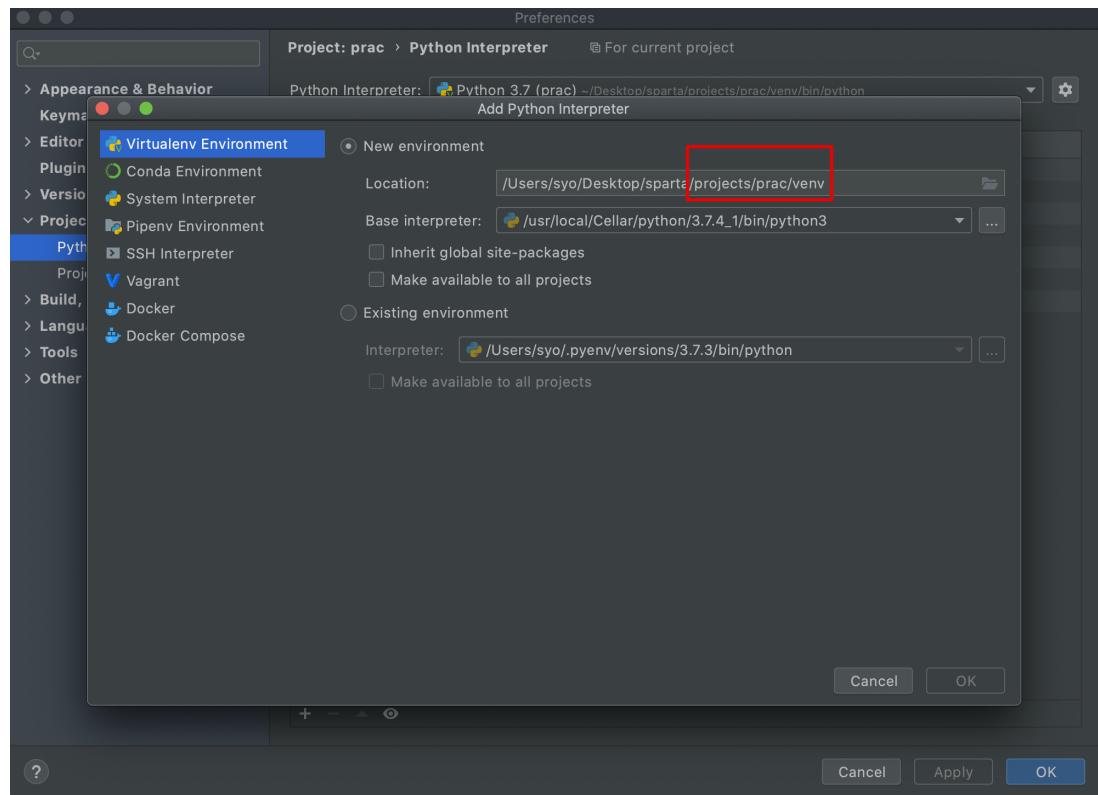
▼ 1. 가상 환경 설치 및 확인

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



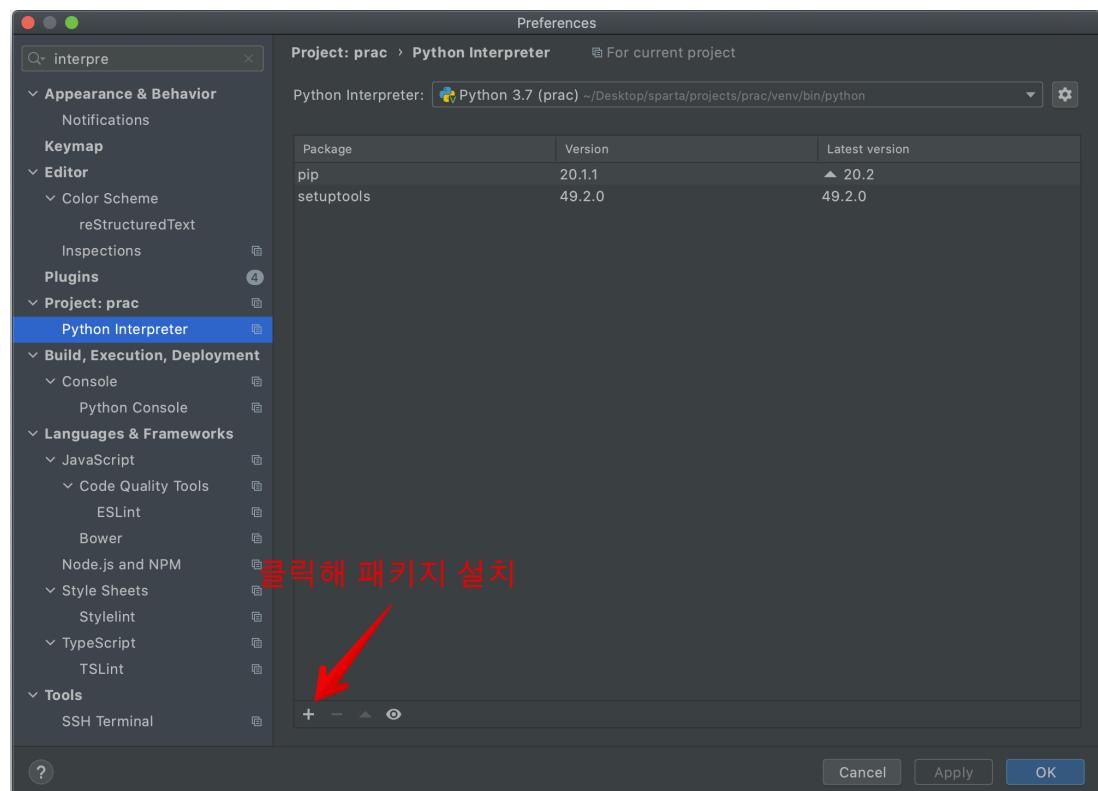
- virtual environment - New environment 선택

- Location을 `현재폴더/venv`로 설정해주세요. 현재 프로젝트 폴더(prac) 아래에 가상환경 폴더(venv)가 생성이 됩니다. 우리는 `projects/alonememo/venv`로 보이겠죠?

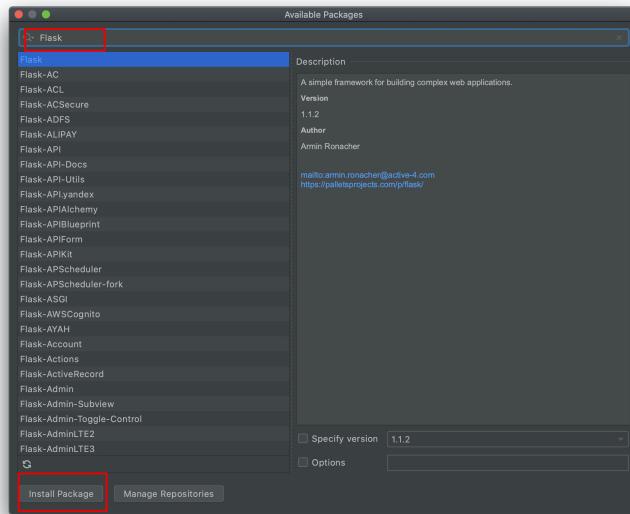


▼ 2. pip(python install package) 사용 - Flask 패키지 설치해보기

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

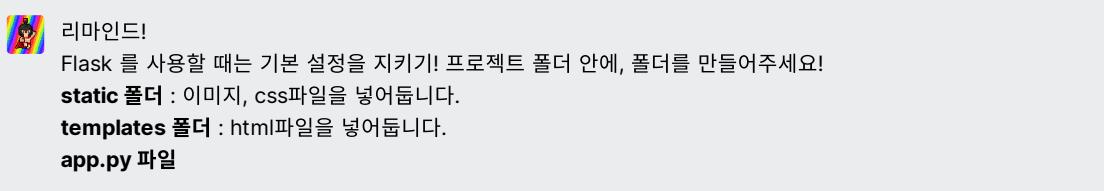


- `flask` 로 검색한 후, Install package 클릭

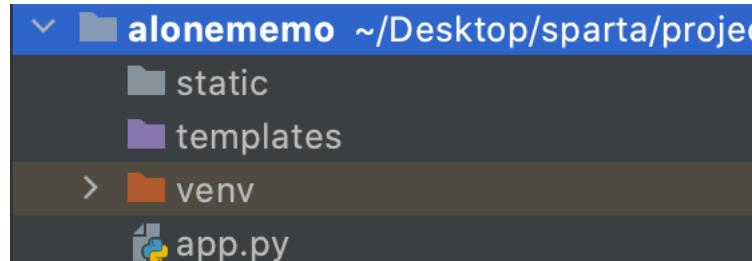


- 위 방법처럼 `pymongo` 와 `requests` 도 패키지 설치 해주세요!

▼ 3. Flask 기본 폴더 구조 만들기



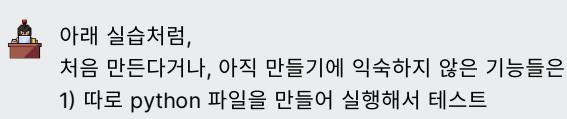
- 아래 모양처럼 만들어주세요!



- 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!

▼ 20) 프로젝트 준비 - URL 에서 페이지 정보 가져오기(**meta태그 스크래핑**)



그럼, meta tag가 뭔지 공부해볼까요?

▼ 어떤 정보를 스크래핑으로 가져와야 하나요?



우리는 기사URL만 입력했는데, 자동으로 카드 정보가 보여집니다.

함께 확인해볼까요?

<http://spartacodingclub.shop/>

루이싱 커피, 2년 내 매장 10000개 오픈한 다 - 'Startup's Story Platform'

스타벅스를 벤치마킹해 중국서 가장 강력한 경쟁자로 떠오른 루이싱커피(瑞幸咖啡, Luckin coffee, 이하 루이싱)가 2021년까지 10,000 개의 매장을 오픈하며 수익성 개선을 위해 비 커피 부문까지 사업을 확장한다. 첸즈야(Qian Zhiya, 錢治亞) 루이싱 대표는 29일 세계 커피산업 포럼에서 이와 같은 회사의 계획을 밝혔다. 첸즈야 대표는 ...

아티클의 url을 입력하면, 타이틀, 이미지, 설명을 크롤링해오고, 내 코멘트와 함께 저장합니다.

Hackers are stealing years of call records from hacked cell networks – TechCrunch

Security researchers say they have uncovered a massive espionage campaign involving the theft of call records from hacked cell network providers to conduct targeted surveillance on individuals of interest. The hackers have systematically broken in to more than 10 cell networks around the world to ...

하나 더!



바로 '기사 제목', '썸네일 이미지', '내용'입니다.

이 정보는 'meta'태그를 스크래핑해서 얻을 수 있습니다.

meta태그가 무엇이고, 어떻게 스크래핑 하는지, 함께 살펴볼까요?

▼ meta 태그가 무엇일까요?

- (링크)에 접속한 뒤 크롬 개발자 도구를 이용해 HTML의 생김새를 살펴볼까요?
- meta 태그는, `<head>~</head>` 부분에 들어가는, 눈으로 보이는 것(body) 외에 사이트의 속성을 설명해주는 태그들입니다.
예) 구글 검색 시 표시 될 설명문, 사이트 제목, 카톡 공유 시 표시 될 이미지 등
- 우리는 그 중 og:image / og:title / og:description 을 크롤링 할 예정입니다.



▼ meta 태그 스크래핑 하기

[💻 코드 메타태그 스크래핑 01] - 기본 준비

- 연습을 위해, week04 에 `meta_prac.py` 파일을 만들어봅니다. 기본 준비를 합니다.

```
import requests
from bs4 import BeautifulSoup

url = 'https://platum.kr/archives/120958'

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36'
}
data = requests.get(url, headers=headers)

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

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

[💻 코드 메타태그 스크래핑 02]

- `select_one`을 이용해 meta tag를 먼저 가져와봅니다.

```
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)
```

[💻 코드 메타태그 스크래핑 03]

- 가져온 meta tag의 `content`를 가져와봅시다.

 튜터는 이미 방법을 알고 있지만, 함께 구글링해봅시다!

```
url_image = og_image['content']
url_title = og_title['content']
url_description = og_description['content']

print(url_image)
print(url_title)
print(url_description)
```

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

 2주차에 만들었던 나홀로메모장 HTML에 우리가 사용할 함수를 만들어두었어요.
templates 폴더 안 `index.html`에 다음 내용을 복사해 넣어씌웁니다.

▼ [💻 코드 - `index.html`] - 2주차에 만들었던 나홀로메모장 HTML

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

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

        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
              integrity="sha384-Gn5Bf9tH/B/KDkCePeYwPiBzW3mgPvhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
              crossorigin="anonymous">

        <!-- JS -->
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
               integrity="sha384-9eNzUCdb4fD3L0g0Z8uAAButmignewqtmH06ODjezGqZD0Q&family=Stylish&display=swap" rel="stylesheet">
```

```

<title>스파르타코딩클럽 | 나홀로 메모장</title>

<!-- style -->
<style type="text/css">
    * {
        font-family: "Stylish", sans-serif;
    }

    .wrap {
        width: 900px;
        margin: auto;
    }

    .comment {
        color: blue;
        font-weight: bold;
    }

    #post-box {
        width: 500px;
        margin: 20px auto;
        padding: 50px;
        border: black solid;
        border-radius: 5px;
    }
</style>
<script>
    $(document).ready(function () {
        $("#cards-box").html("");
        showArticles();
    });

    function openClose() {
        // id 값 post-box의 display 값이 block 이면(= 눈에 보이면)
        if ($("#post-box").css("display") == "block") {
            // post-box를 가리고
            $("#post-box").hide();
            // 다시 버튼을 클릭하면, 박스 열기를 할 수 있게 텍스트 바꿔두기
            $("#btn-post-box").text("포스팅 박스 열기");
        } else {
            // 아니면(눈에 보이지 않으면) post-box를 펴라
            $("#post-box").show();
            // 다시 버튼을 클릭하면, 박스 닫기를 할 수 있게 텍스트 바꿔두기
            $("#btn-post-box").text("포스팅 박스 닫기");
        }
    }

    function postArticle() {
        $.ajax({
            type: "POST",
            url: "/memo",
            data: {},
            success: function (response) { // 성공하면
                if (response["result"] == "success") {
                    alert(response["msg"]);
                }
            }
        })
    }

    function showArticles() {
        $.ajax({
            type: "GET",
            url: "/memo",
            data: {},
            success: function (response) {
                if (response["result"] == "success") {
                    alert(response["msg"]);
                }
            }
        })
    }

    function makeCard(url, title, desc, comment, image) {
    }
</script>

</head>

<body>
    <div class="wrap">
        <div class="jumbotron">
            <h1 class="display-4">나홀로 링크 메모장!</h1>
            <p class="lead">중요한 링크를 저장해두고, 나중에 볼 수 있는 공간입니다</p>

```

```

<hr class="my-4">
<p class="lead">
    <button onclick="openClose()" id="btn-post-box" type="button" class="btn btn-primary">포스팅 박스 열기
    </button>
</p>
</div>
<div id="post-box" class="form-post" style="display:none">
    <div>
        <div class="form-group">
            <label for="post-url">아티클 URL</label>
            <input id="post-url" class="form-control" placeholder="">
        </div>
        <div class="form-group">
            <label for="post-comment">간단 코멘트</label>
            <textarea id="post-comment" class="form-control" rows="2"></textarea>
        </div>
        <button type="button" class="btn btn-primary" onclick="postArticle()">기사저장</button>
    </div>
</div>
<div id="cards-box" class="card-columns">
    <div class="card">
        
        <div class="card-body">
            <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
            <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁</p>
            <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
    </div>
    <div class="card">
        
        <div class="card-body">
            <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
            <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁</p>
            <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
    </div>
    <div class="card">
        
        <div class="card-body">
            <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
            <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁</p>
            <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
    </div>
    <div class="card">
        
        <div class="card-body">
            <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
            <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁</p>
            <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
    </div>
    <div class="card">
        
        <div class="card-body">
            <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
            <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁</p>
            <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
    </div>
</div>
</body>
</html>

```



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

▼ [💻 코드 - `app.py`]

```
from flask import Flask, render_template, jsonify, request
import requests
from bs4 import BeautifulSoup
from pymongo import MongoClient # pymongo를 임포트 하기(패키지 인스톨 먼저 해야겠죠?)

app = Flask(__name__)

client = MongoClient('localhost', 27017) # mongoDB는 27017 포트로 돌아갑니다.
db = client.dbsparta # 'dbsparta'라는 이름의 db를 만들거나 사용합니다.

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

@app.route('/memo', methods=['POST'])
def post_article():
    # 1. 클라이언트로부터 데이터를 받기
    # 2. meta tag를 스크래핑하기
    # 3. mongoDB에 데이터 넣기
    return jsonify({'result': 'success', 'msg': 'POST 연결되었습니다!'})

@app.route('/memo', methods=['GET'])
def read_articles():
    # 1. mongoDB에서 _id 값을 제외한 모든 데이터 조회해오기(Read)
    # 2. articles라는 키 값으로 articles 정보 보내주기
    return jsonify({'result': 'success', 'msg': 'GET 연결되었습니다!'})

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

▼ 22) API 만들고 사용하기 - 포스팅API (Create → POST)



우리가 만들 API 두 가지

- 1) 포스팅API - 카드 생성 (Create) : 클라이언트에서 받은 url, comment를 이용해서 페이지 정보를 찾고 저장하기
- 2) 리스트링API - 저장된 카드 보여주기 (Read)



설계한 API - 포스팅API - 카드 생성 (Create)

A. 요청 정보

- 요청 URL= `/memo` , 요청 방식 = `POST`
- 요청 데이터 : URL(url_give), 코멘트(comment_give)

B. 서버가 제공할 기능

- URL의 meta태그 정보를 바탕으로 제목, 설명, 이미지URL 스크래핑
- (제목, 설명, URL, 이미지URL, 코멘트) 정보를 모두 DB에 저장

C. 응답 데이터

- API가 정상적으로 작동하는지 클라이언트에게 알려주기 위해서 성공 메시지 보내기
- (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('/memo', methods=['POST'])
def post_articles():
    # 1. 클라이언트로부터 데이터를 받기
    # 2. meta tag를 스크래핑하기
    # 3. mongoDB에 데이터 넣기
    return jsonify({'result': 'success', 'msg': 'POST 연결되었습니다!'})

```

[클라이언트 코드 - [index.html](#)]

```

function postArticle() {
    $.ajax({
        type: "POST",
        url: "/memo",
        data: {},
        success: function (response) { // 성공하면
            if (response['result'] == 'success') {
                alert(response['msg']);
            }
        }
    })
}

<button type="button" class="btn btn-primary" onclick="postArticle()">기사저장</button>

```



동작 테스트

'기사저장' 버튼을 클릭했을 때, 'POST 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!

메모를 작성하기 위해 서버가 전달받아야하는 정보는 다음 두 가지입니다.

- URL(url_give)
- 코멘트(comment_give)

그리고 URL를 meta tag를 스크래핑해서 아래 데이터를 저장(Create)합니다.

- URL(url)
- 제목(title)
- 설명(desc)
- 이미지URL(image)
- 코멘트(comment)

따라서 서버 로직은 다음 단계로 구성되어야 합니다.

1. 클라이언트로부터 데이터를 받기.
2. meta tag를 스크래핑하기
3. mongoDB에 데이터를 넣기

```

@app.route('/memo', methods=['POST'])
def post_article():
    # 1. 클라이언트로부터 데이터를 받기
    url_receive = request.form['url_give'] # 클라이언트로부터 url을 받는 부분
    comment_receive = request.form['comment_give'] # 클라이언트로부터 comment를 받는 부분

    # 2. meta tag를 스크래핑하기
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36'
    }
    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"]')

    url_title = og_title['content']

```

```

url_description = og_description['content']
url_image = og_image['content']

article = {'url': url_receive, 'title': url_title, 'desc': url_description, 'image': url_image,
           'comment': comment_receive}

# 3. mongoDB에 데이터를 넣기
db.articles.insert_one(article)

return jsonify({'result': 'success'})

```

▼ 3) 클라이언트 만들기



API 는 약속이라고 했습니다. API를 사용할 클라이언트를 만들어보죠!

메모를 작성하기 위해 서버에게 주어야하는 정보는 다음 두 가지입니다.

- URL (url_give) : meta tag를 가져온 url
- comment (comment_give) : 유저가 입력한 코멘트

따라서 클라이언트 로직은 다음 단계로 구성되어야 합니다.

1. 유저가 입력한 데이터를 #post-url과 #post-comment에서 가져오기
2. /memo에 POST 방식으로 메모 생성 요청하기
3. 성공 시 페이지 새로고침하기

```

function postArticle() {
    // 1. 유저가 입력한 데이터를 #post-url과 #post-comment에서 가져오기
    let url = $("#post-url").val();
    let comment = $("#post-comment").val();

    // 2. memo에 POST 방식으로 메모 생성 요청하기
    $.ajax({
        type: "POST", // POST 방식으로 요청하겠다.
        url: "/memo", // /memo라는 url에 요청하겠다.
        data: { url_give: url, comment_give: comment}, // 데이터를 주는 방법
        success: function(response){ // 성공하면
            if (response["result"] == "success") {
                alert("포스팅 성공!");
                // 3. 성공 시 페이지 새로고침하기
                window.location.reload();
            } else {
                alert("서버 오류!")
            }
        }
    })
}

```

▼ 4) 완성 확인하기



동작 테스트

<https://platum.kr/archives/129737> ← 이 URL을 입력하고 기사저장을 눌렀을 때, '포스팅 성공!' alert창이 뜨는지 확인합니다.

(우리는 스크래핑을 사용해 정보를 저장하고 있으니까, meta tag 가 있는 사이트만 저장이 제대로 되겠죠?)

참고!

지금은 카드가 보이지 않습니다. 아직 카드를 보여주는 리스트ng API 를 만들지 않았기 때문이죠.

▼ 23) API 만들고 사용하기 - 리스트ng API (Read → GET)



우리가 만들 API 두 가지

- 1) 포스팅API - 카드 생성 (Create) : 클라이언트에서 받은 url, comment를 이용해서 페이지 정보를 찾고 저장하기
- 2) 리스트ng API - 저장된 카드 보여주기 (Read)



설계한 API - 리스트API - 저장된 카드 보여주기 (Read)

A. 요청 정보

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

B. 서버가 제공할 기능

- DB에 저장돼있는 모든 (제목, 설명, URL, 이미지URL, 코멘트) 정보를 가져오기

C. 응답 데이터

- API 동작 잘했다는 성공 메시지, 아티클(기사)들의 정보(제목, 설명, URL, 이미지URL, 코멘트)
- (JSON 형식) 'result' = 'success', 'articles': 아티클 정보

▼ 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('/memo', methods=['GET'])
def read_articles():
    # 1. 모든 document 찾기 & _id 값은 출력에서 제외하기
    # 2. articles라는 키 값으로 영화정보 내려주기
    return jsonify({'result':'success', 'msg':'GET 연결되었습니다!'})
```

[클라이언트 코드 - `index.html`]

```
function showArticles() {
    $.ajax({
        type: "GET",
        url: "/memo",
        data: {},
        success: function (response) {
            if (response["result"] == "success") {
                alert(response["msg"]);
            }
        }
    })
}
```



동작 테스트

새로고침했을 때, 'GET 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!

메모를 보여주기 위해 서버가 추가로 전달받아야하는 정보는 없습니다. 조건없이 모든 메모를 보여줄 꺼니까요!

따라서 서버 로직은 다음 단계로 구성되어야 합니다.

1. mongoDB에서 _id 값을 제외한 모든 데이터 조회해오기 (Read)
2. articles라는 키 값으로 articles 정보 보내주기

```
@app.route('/memo', methods=['GET'])
def read_articles():
    # 1. mongoDB에서 _id 값을 제외한 모든 데이터 조회해오기 (Read)
    result = list(db.articles.find({}, {'_id': 0}))
```

```
# 2. articles라는 키 값으로 article 정보 보내주기
return jsonify({'result': 'success', 'articles': result})
```

▼ 3) 클라이언트 만들기



API 는 약속이라고 했습니다. API를 사용할 클라이언트를 만들어보죠!

메모를 작성하기 위해 서버에게 주어야하는 정보는 없습니다. 조건없이 모든 메모를 가져오기 때문입니다.

따라서 클라이언트 로직은 다음 단계로 구성되어야 합니다.

1. /memo에 GET 방식으로 메모 정보 요청하고 articles로 메모 정보 받기
2. , makeCard 함수를 이용해서 카드 HTML 붙이기
(→ 2주차 Ajax 연습과 같습니다!)

```
function showArticles() {
    $.ajax({
        type: "GET",
        url: "/memo",
        data: {},
        success: function(response){
            let articles = response["articles"];
            for (let i = 0; i < articles.length; i++) {
                makeCard(articles[i]["image"], articles[i]["url"], articles[i]["title"], articles[i]["desc"], articles[i]["comment"])
            }
        }
    })
}

function makeCard(image, url, title, desc, comment) {
    let temp_html = `<div class="card">
        
        <div class="card-body">
            <a href="${url}" target="_blank" class="card-title">${title}</a>
            <p class="card-text">${desc}</p>
            <p class="card-text comment">${comment}</p>
        </div>
    </div>`;
    $("#cards-box").append(temp_html);
}
```

▼ 4) 완성 확인하기



동작 테스트

새로고침했을 때, 앞 포스팅 API를 만들고 테스트했던 메모가 보이면 성공입니다.

참고.

card가 정렬되는 순서는 위에서 아래로 채워지고, 왼쪽부터 오른쪽으로 순서대로 채워집니다. 부트스트랩 컴퍼넌트 페이지에 적혀있어요. "Cards are ordered from top to bottom and left to right." ([컴퍼넌트 페이지 링크](#))

▼ 24) 전체 완성 코드

▼ 클라이언트 코드 [index.html](#)

```
<!Doctype html>
<html lang="ko">

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

        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
              integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJ1SAwiGgFAW/dAiS6JXm"
              crossorigin="anonymous">

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

```

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
        integrity="sha384-ApNbghB+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
        crossorigin="anonymous"></script>

<!-- 구글폰트 -->
<link href="https://fonts.googleapis.com/css?family=Stylish&display=swap" rel="stylesheet">

<title>스파르타코딩클럽 | 나홀로 메모장</title>

<!-- style -->
<style type="text/css">
    * {
        font-family: "Stylish", sans-serif;
    }

    .wrap {
        width: 900px;
        margin: auto;
    }

    .comment {
        color: blue;
        font-weight: bold;
    }

    #post-box {
        width: 500px;
        margin: 20px auto;
        padding: 50px;
        border: black solid;
        border-radius: 5px;
    }
</style>
<script>
    $(document).ready(function () {
        $("#cards-box").html("");
        showArticles();
    });

    function openClose() {
        // id 값 post-box의 display 값이 block 이면(= 눈에 보이면)
        if ($("#post-box").css("display") == "block") {
            // post-box를 가리고
            $("#post-box").hide();
            // 다시 버튼을 클릭하면, 박스 열기를 할 수 있게 텍스트 바꿔두기
            $("#btn-post-box").text("포스팅 박스 열기");
        } else {
            // 아니면(눈에 보이지 않으면) post-box를 펴라
            $("#post-box").show();
            // 다시 버튼을 클릭하면, 박스 닫기를 할 수 있게 텍스트 바꿔두기
            $("#btn-post-box").text("포스팅 박스 닫기");
        }
    }

    function postArticle() {
        let url = $("#post-url").val();
        let comment = $("#post-comment").val();

        // 2. memo에 POST 방식으로 메모 생성 요청하기
        $.ajax({
            type: "POST", // POST 방식으로 요청하겠다.
            url: "/memo", // /memo라는 url에 요청하겠다.
            data: {url_give: url, comment_give: comment}, // 데이터를 주는 방법
            success: function (response) { // 성공하면
                if (response["result"] == "success") {
                    alert("포스팅 성공!");
                    // 3. 성공 시 페이지 새로고침하기
                    window.location.reload();
                } else {
                    alert("서버 오류!")
                }
            }
        })
    }

    function showArticles() {
        $.ajax({
            type: "GET",
            url: "/memo",
            data: {},
            success: function (response) {
                let articles = response["articles"];
                console.log(articles);
                for (let i = 0; i < articles.length; i++) {
                    makeCard(articles[i]["image"], articles[i]["url"], articles[i]["title"], articles[i]["desc"])
                }
            }
        })
    }
</script>

```

```

        }
    })
}

function makeCard(image, url, title, desc, comment) {
    let tempHtml = `<div class="card">
        
        <div class="card-body">
            <a href="${url}" target="_blank" class="card-title">${title}</a>
            <p class="card-text">${desc}</p>
            <p class="card-text comment">${comment}</p>
        </div>
    </div>`;
    $("#cards-box").append(tempHtml);
}

</script>

</head>

<body>
    <div class="wrap">
        <div class="jumbotron">
            <h1 class="display-4">나홀로 링크 메모장!</h1>
            <p class="lead">중요한 링크를 저장해두고, 나중에 볼 수 있는 공간입니다</p>
            <hr class="my-4">
            <p class="lead">
                <button onclick="openClose()" id="btn-post-box" type="button" class="btn btn-primary">포스팅 박스 열기
            </button>
            </p>
        </div>
        <div id="post-box" class="form-post" style="display:none">
            <div>
                <div class="form-group">
                    <label for="post-url">아티클 URL</label>
                    <input id="post-url" class="form-control" placeholder="">
                </div>
                <div class="form-group">
                    <label for="post-comment">간단 코멘트</label>
                    <textarea id="post-comment" class="form-control" rows="2"></textarea>
                </div>
                <button type="button" class="btn btn-primary" onclick="postArticle()">기사저장</button>
            </div>
        </div>
        <div id="cards-box" class="card-columns">
            <div class="card">
                
                <div class="card-body">
                    <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
                    <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁</p>
                    <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
                </div>
            </div>
            <div class="card">
                
                <div class="card-body">
                    <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
                    <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁</p>
                    <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
                </div>
            </div>
            <div class="card">
                
                <div class="card-body">
                    <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
                    <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁</p>
                    <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
                </div>
            </div>
            <div class="card">
                
                <div class="card-body">
                    <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
                    <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁</p>
                    <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
                </div>
            </div>
            <div class="card">
                
                <div class="card-body">
                    <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
                    <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁</p>
                    <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
                </div>
            </div>
        </div>
    </div>

```

```

        src="https://www.eurail.com/content/dam/images/eurail/Italy%20CP%20Promo%20Block.adaptive.767
        alt="Card image cap">
    <div class="card-body">
        <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
        <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁<br/>
        <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
    </div>
</div>
<div class="card">
    
    <div class="card-body">
        <a href="#" class="card-title">여기 기사 제목이 들어가죠</a>
        <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁<br/>
        <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
    </div>
</div>
</div>
</body>

</html>

```

▼ 서버 코드 [app.py](#)

```

from flask import Flask, render_template, jsonify, request
import requests
from bs4 import BeautifulSoup
from pymongo import MongoClient # pymongo를 임포트 하기(패키지 인스톨 먼저 해야겠죠?)

app = Flask(__name__)

client = MongoClient('localhost', 27017) # mongoDB는 27017 포트로 돌아갑니다.
db = client.dbsparta_ninework # 'dbsparta'라는 이름의 db를 만들거나 사용합니다.

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

@app.route('/memo', methods=['POST'])
def post_article():
    # 1. 클라이언트로부터 데이터를 받기
    url_receive = request.form['url_give'] # 클라이언트로부터 url을 받는 부분
    comment_receive = request.form['comment_give'] # 클라이언트로부터 comment를 받는 부분

    # 2. meta tag를 스크래핑하기
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36'}
    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"]')

    url_title = og_title['content']
    url_description = og_description['content']
    url_image = og_image['content']

    article = {'url': url_receive, 'title': url_title, 'desc': url_description, 'image': url_image,
               'comment': comment_receive}

    # 3. mongoDB에 데이터를 넣기
    db.articles.insert_one(article)

    return jsonify({'result': 'success'})

@app.route('/memo', methods=['GET'])
def read_articles():
    # 1. mongoDB에서 _id 값을 제외한 모든 데이터 조회해오기 (Read)
    result = list(db.articles.find({}, {'_id': 0}))
    # 2. articles라는 키 값으로 article 정보 보내주기
    return jsonify({'result': 'success', 'articles': result})

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

```



와! 우리 API를 사용한 똑딱 완성했어요.
아직 헷갈려도 괜찮아요. 다음주 다시 한 번 반복할 거니까요.

신난다! 내가 만든 프로젝트가 벌써 세 개가 되었어요.

😎 문제뱅크

- 오늘 문제뱅크는 쉬어갈게요. 숙제를 하는 것에 집중해주세요!

💡 소화 타임



소화타임 = 복습 & 숙제 시작 타임!
인생은 실전! 튜터에게 질문하면서 숙제를 시작해보세요.

▼ 오늘은 소화 타임이 30분 늘었네요?

- 소화하실 게 많을 거 같아 늘려두었습니다. (오늘 거의 무한리필이었죠?)
- 4주 동안 숨가쁘게 달려오느라 수고하셨습니다. 이제 웹 개발 과정 A-Z까지를 한 번 훑어봤어요.👏
- 다음주에는 프로젝트 기획안을 수립하고 본격적으로 달리게 됩니다. 그 전에, 1~4주차 내용을 스스로 정리할 수 있는 시간을 마련하였습니다.
- 오늘의 숙제는 튜터님의 충분한 도움을 받아 완성하고 가시면 좋을거 같아요.
- 만약 숙제를 어디서부터 손대야 할지 모르겠다면 <모두의 책 리뷰>를 처음부터 혼자 완성해보세요. 숙제와 아주 유사하답니다.

숙제 설명



달달한 맛 vs 짭짤한 맛 중 하나만 골라서 하시면 됩니다! :-)
숙제는 되도록 수업 D-1 까지 하기!

1. 공통 숙제 - 내 프로젝트 생김새 완성하기!

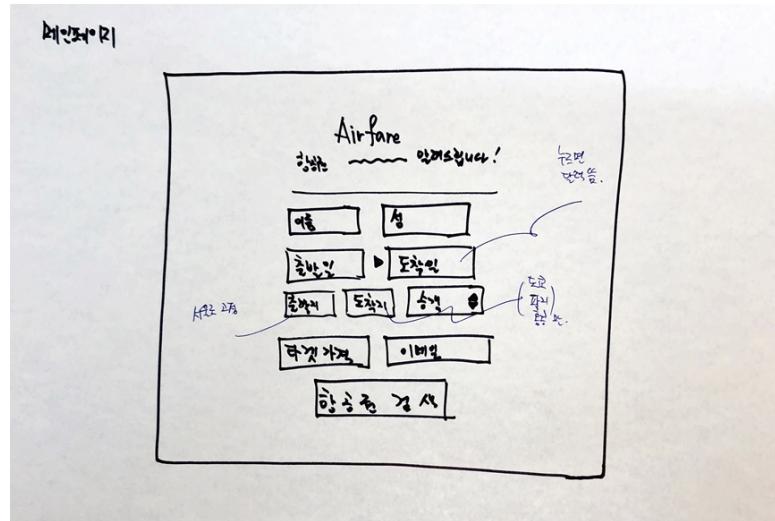
- 1주차, 3주차 내 프로젝트에 대해서 상상했었죠? 다음 시간에 내 프로젝트를 간단하게 소개하는 시간을 가질 거에요.



내가 만들 "프로젝트 생김새(레이아웃)"를 그려서 슬랙 메시지로 올려주세요!
챗봇 등 프론트페이지가 없는 경우, 간단한 로직과 기획을 글로 작성해주세요.

일단 생김새가 완성되어야 API를 구체적으로 생각해볼 수 있고,
그래야 **프로젝트 완성 가능성**이 무척 올라간답니다!

▼ 프로젝트 생김새(레이아웃)은 이런 느낌!



- 아이디어 창고 [스파르타코딩클럽 포트폴리오 페이지](#)
- 혹시 프로젝트를 꼭 같이 하고 싶은 분이 있나요? 개별 프로젝트를 권장합니다만, 같이 하고 싶은 분이 있으면 튜터님과 상의해서 팀으로 진행해도 무방합니다!

2. 원페이지 쇼핑몰 완성하기

[숙제 - 달달한 맛 🍫] - 원페이지 쇼핑몰에 API 기능 붙이기 (1시간 예상)



쇼핑몰 기능 API를 사용하는 Flask 프로젝트를 만들어보세요.
(아래 API가 만들어져있는 `app.py` 코드를 사용하면 됩니다.)

쇼핑몰은 두 가지 기능을 수행해야 합니다.

- 1) 주문하기(POST): 정보 입력 후 '주문하기' 버튼클릭 시 주문목록에 추가
- 2) 주문내역보기(GET): 페이지 로딩 후 하단 주문 목록이 자동으로 보이기

아래 완성본을 참고해주세요!

<http://spartacodingclub.shop/homework>

▼ 사용할 코드 `app.py`]

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

from pymongo import MongoClient

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

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

# 주문하기(POST) API
@app.route('/order', methods=['POST'])
def save_order():
    name_receive = request.form['name_give']
    count_receive = request.form['count_give']
    address_receive = request.form['address_give']
    phone_receive = request.form['phone_give']

    doc = {
        'name': name_receive,
        'count': count_receive,
```

```

        'address': address_receive,
        'phone': phone_receive
    }

    db.homework.insert_one(doc)

    return jsonify({'result': 'success'})

# 주문 목록보기(Read) API
@app.route('/order', methods=['GET'])
def view_orders():
    orders = list(db.homework.find({}, {'_id': 0}))
    return jsonify({'result': 'success', 'orders': orders})

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

```

▼ 사용할  코드 [index.html](#)]

```

<!DOCTYPE html>
<html lang="ko">
    <head>
        <!-- Webpage Title -->
        <title>나홀로 쇼핑몰</title>

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

        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
              integrity="sha384-Gn5IHeDmOsO0O6RvQn7sXN0/4wLc372fNqGx+41rGcQfHQA0PCwQJhj" crossorigin="anonymous">

        <!-- JS -->
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@1.12.9/dist/umd/popper.min.js"
               integrity="sha384-A9+qxDWeQ8D8eIw+Z1j9AcWQzJ4�f/DkSuNxpujewRYJzqX3vXW1t0puL4d" crossorigin="anonymous"></script>

        <!-- 구글폰트 -->
        <link href="https://fonts.googleapis.com/css?family=Stylish&display=swap" rel="stylesheet">

        <script type="text/javascript">
            function isCellPhone(p) {
                let regExp = /^(01[0|1|6|7|8|9]{1}|02|0[3-9]{1}[0-9]{1})[-][0-9]{3,4}[-][0-9]{4}$/;
                return regExp.test(p);
            }

            function order() {
                let name = $("#order-name").val();
                let count = $("#order-count").val();
                let address = $("#order-address").val();
                let phone = $("#order-phone").val();

                if (name == "") {
                    alert("이름을 입력해주세요")
                    $("#order-name").focus()
                    return
                } else if (count == "") {
                    alert("수량을 입력해주세요")
                    $("#order-count").focus()
                    return
                } else if (address == "") {
                    alert("주소를 입력해주세요")
                    $("#order-address").focus()
                    return
                } else if (phone == "") {
                    alert("휴대폰번호를 입력해주세요")
                    $("#order-phone").focus()
                    return
                } else if (!isCellPhone(phone)) {
                    alert("휴대폰번호 입력 형식이 틀립니다. \n 010-0000-0000으로 입력해주세요")
                    return
                }

                // 주문하기 API를 사용해 여기를 채워주세요
            }

            $(document).ready(function () {
                $("#orders-box").html("");
                showOrders();
            });
        </script>

```

```

    });

    function showOrders() {
        // 주문 목록보기 API 를 사용해 여기를 채워주세요
    }

    function makeOrderRow(name, count, address, phone) {
        let tempHtml = `<tr>
            <td>${name}</td>
            <td>${count}</td>
            <td>${address}</td>
            <td>${phone}</td>
        </tr>`;
        $("#orders-box").append(tempHtml);
    }
</script>

<style type="text/css">
    * {
        font-family: "Stylish", sans-serif;
    }

    h1, h5 {
        display: inline;
    }

    .wrap {
        width: 500px;
        margin: auto;
    }

    .img {
        background-image: url("https://www.conscious-skincare.com/wp-content/uploads/2016/02/glc-candle-lit-with-ne
        background-size: cover;
        background-position: center;
        width: 500px;
        height: 300px;
    }

    .info {
        margin-top: 20px;
        margin-bottom: 20px;
    }

    .order {
        text-align: center;
    }

    .orders {
        margin-top: 100px;
    }
</style>
</head>
<body>
    <div class="wrap">
        <div class="img"></div>
        <div class="info">
            <h1>양초를 팝니다</h1>
            <h5>가격: 3,000 원 / 개</h5>
            <p>이 양초는 사실 특별한 힘을 가지고 있어요. 양초를 켜고 소원을 빌면 짜자잔 워든지 이뤄지게 된답니다. 하나 사가세요! 대나무 향이 아주 좋아요</p>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">주문자 이름</span>
                </div>
                <input type="text" class="form-control" id="order-name">
            </div>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <label class="input-group-text" for="order-count">수량</label>
                </div>
                <select class="custom-select" id="order-count">
                    <option selected value=""> -- 수량을 선택하세요 --</option>
                    <option value="1">1</option>
                    <option value="2">2</option>
                    <option value="3">3</option>
                </select>
            </div>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">주소</span>
                </div>
                <input type="text" class="form-control" id="order-address">
            </div>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">전화번호</span>
                </div>

```

```

        </div>
        <input type="text" class="form-control" id="order-phone">
    </div>
    <div class="order">
        <button onclick="order()" type="button" class="btn btn-primary">주문하기</button>
    </div>
</div>
<div class="orders">
    <table class="table">
        <thead>
            <tr>
                <th scope="col">이름</th>
                <th scope="col">수량</th>
                <th scope="col">주소</th>
                <th scope="col">전화번호</th>
            </tr>
        </thead>
        <tbody id="orders-box">
            <tr>
                <td>박근단</td>
                <td>3</td>
                <td>스파르타국 코딩시 프론트구</td>
                <td>010-1234-5678</td>
            </tr>
        </tbody>
    </table>
</div>
</body>
</html>

```

[숙제 - 짭짤한 맛 🍜] - 원페이지 쇼핑몰 완성하기



1주차에 완성한 쇼핑몰을 완성해주세요!

쇼핑몰은 두 가지 기능을 수행해야 합니다.

- 1) 주문하기(POST): 정보 입력 후 '주문하기' 버튼클릭 시 주문목록에 추가
- 2) 주문내역보기(GET): 페이지 로딩 후 하단 주문 목록이 자동으로 보이기

아래 완성본을 참고해주세요!

<http://spartacodingclub.shop/homework>



힌트:

<모두의책리뷰>랑 아주아주 유사하답니다!

먼저 API 를 어떻게 만들지 설계하고, 차근차근 만들어보세요.

웹 페이지 하단에 주문정보를 붙일 때에는, 보트스트랩 Table(링크)를 이용하세요!

[숙제 제출]

- 내 Github 에 올리면(push) 됩니다. 슬랙에 github 봇을 제대로 붙여두었다면 자동으로 알람이 갈 거에요. 만약 제대로 안 뜬다면, Github Repository URL 을 슬랙에 공유해주세요.

복습 도우미

- 이번주 배운 내용을 빠르게 복습할 수 있게 준비했어요! 이번주에도 새로운 재밌는 것들 많이 배웠죠! 😊 다시 한번 복습하면서 빠르게 되짚어보세요.

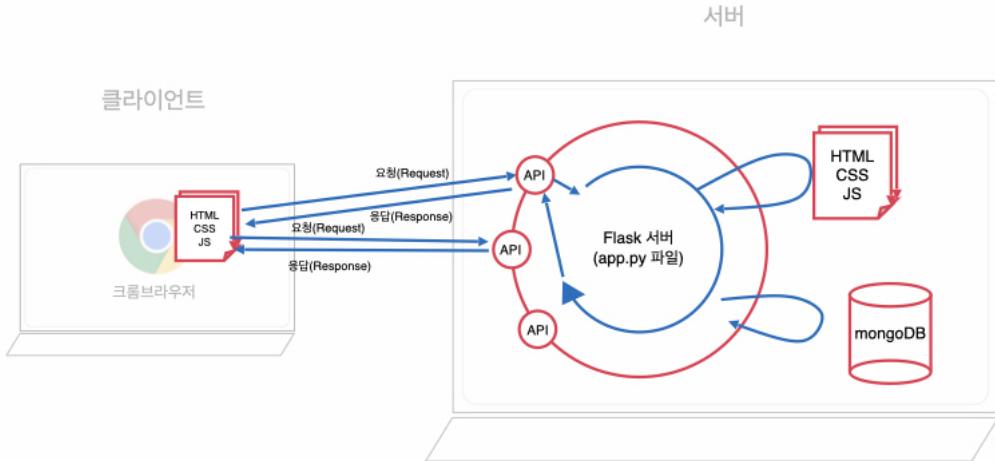
▼ 웹 기초 동작 원리



서버(백엔드) API를 만들어보았습니다.

그리고 HTML 화면과 mongoDB까지 연동해서 두 가지 프로젝트를 만들었죠.

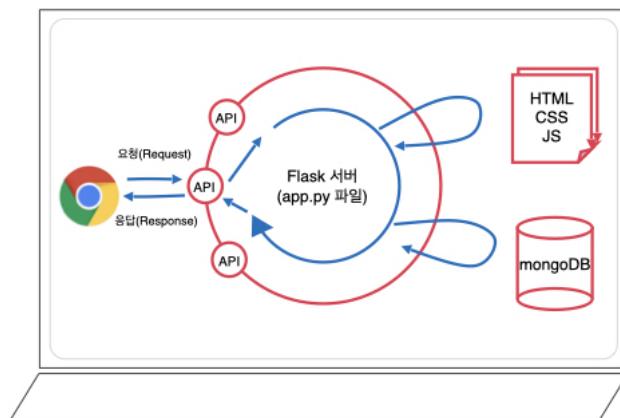
1. 모두의 책 리뷰 ([링크](#))
2. 나홀로 메모장 ([링크](#))



헷갈리지 말기!

우리는 컴퓨터 한 대를 서버로도 쓰고, 클라이언트로도 사용했습니다.

이것을 바로 "로컬 개발환경"이라고 했죠!



▼ 전반부 주요 키워드

- 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(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

▼  후반부 주요 키워드

- meta 태그: <head>~</head> 부분에 들어가는, 눈으로 보이는 것(body) 외에 사이트의 속성을 설명해주는 태그들입니다.
예) 구글 검색 시 표시 될 설명문, 사이트 제목, 카톡 공유 시 표시 될 이미지 등

[1시간] 스스로 소화 타임 시작

- 오늘 배운 것을 복습하고, 본격적으로 숙제를 시작합니다. 모르는 것이 있으면 튜터에게 질문합니다.

체크아웃

▼ "15초 체크아웃"을 진행합니다.

- 튜터는 타이머를 띄워주세요! ([링크](#)).
- 체크인처럼, 현재 본인의 감정상태와 수업후기에 관해 이야기합니다.
 - 예. 오늘 여러 프로젝트를 빠르게 만들어봐서 재밌었어요!
 - 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>

[설치] - 다음 시간을 위해 미리 설치해와야 할 것들

- 없음!