



[스파르타코딩클럽] 웹개발 종합반 - 4주차



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

[수업 목표]

1. Flask 프레임워크를 활용해서 API를 만들 수 있다.
2. '모두의책리뷰' API를 만들고 클라이언트에 연결한다.
3. '나홀로메모장' API를 만들고 클라이언트와 연결한다.

- 01. 오늘 배울 것
- 02. 폴더 세팅
- 03. Flask시작하기 - 서버만들기
- 04. Flask시작하기 - HTML파일 주기
- 05. Flask시작하기 - 본격 API 만들기
- 06. [모두의책리뷰] - 프로젝트 세팅
- 07. [모두의책리뷰] - 뼈대 준비하기
- 08. [모두의책리뷰] - POST 연습(리뷰 저장)
- 09. [모두의책리뷰] - GET 연습(리뷰 보여주기)
- 10. [나홀로메모장] - 프로젝트 세팅
- 11. [나홀로메모장] - API 설계하기
- 12. [나홀로메모장] - 조각 기능 구현해보기
- 13. [나홀로메모장] - 뼈대 준비하기
- 14. [나홀로메모장] - POST 연습(메모하기)
- 15. [나홀로메모장] - GET 연습(보여주기)
- 16. 끝 & 숙제 설명

01. 오늘 배울 것

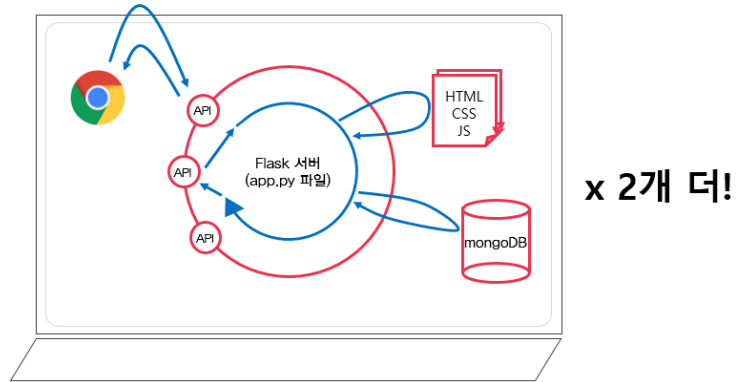
- ▼ 오늘 배울 것 이야기 - 4주차: Flask, 미니프로젝트1, 미니프로젝트2

이번 주 완성본 1. 모두의책리뷰 → [결과물 링크](#)

이번 주 완성본 2. 나홀로메모장 → [결과물 링크](#)

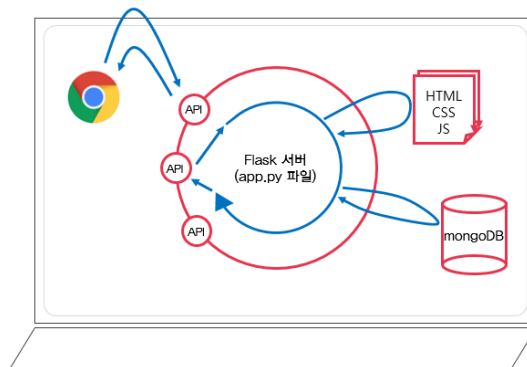


오늘은 HTML과 mongoDB까지 연동해서 서버를 만들어봅시다!



☞ 나중에 또 이야기하겠지만 헛갈리면 안되는 것!
우리는 컴퓨터가 한 대 잡아요... 그래서 같은 컴퓨터에다 서버도 만들고, 요청도 할 거예요. 즉, 클라이언트 = 서버가 되는 것이죠.

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



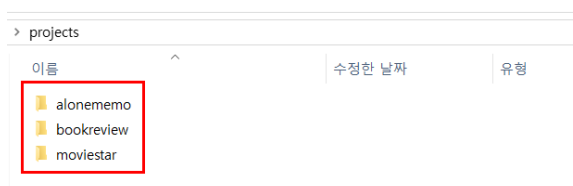
02. 폴더 세팅

▼ 1) 폴더 세 개 만들고 시작하기

☞ 웹개발의 꽃, 백엔드-프론트엔드를 연결하는 일이 익숙해지도록,
연습(week04) → 모두의책리뷰 → 나홀로메모장 → 마이페이지보릿무비스타


총 3번에 걸쳐 반복 실습을 진행 할 예정입니다. 다 해내고 나면 아~주 익숙해질거예요!

☞ 코드 관리를 위해 미리 아래와 같이 폴더구조를 만들고 시작합니다!
sparta 아래 → projects 아래 → 폴더 3개




- alonememo : "나홀로메모장" 관련 코드를 작성합니다. (오늘)
- bookreview : "모두의책리뷰" 관련 코드를 작성합니다. (오늘)
- moviestar : "마이페이보릿무비스타" 관련 코드를 작성합니다. (다음주)

03. Flask시작하기 - 서버만들기

 sparta → week04 폴더를 열고 시작! (new project로 열기!)

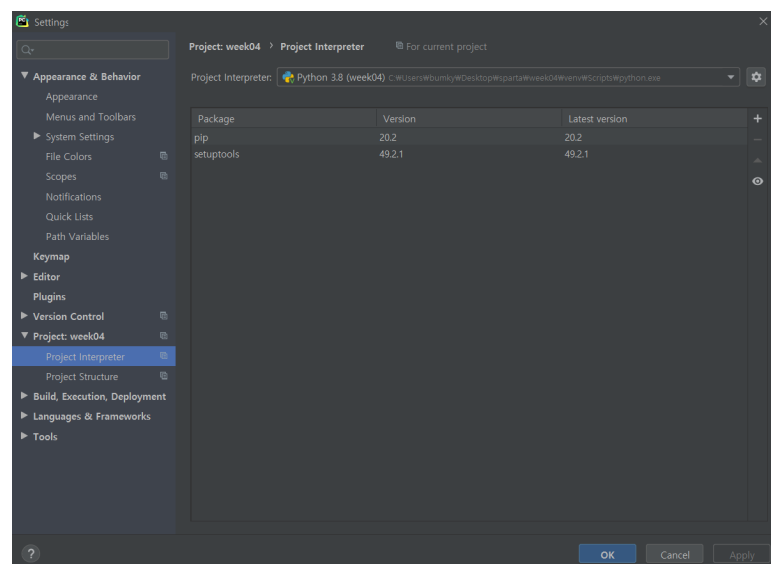
▼ 2) Flask 패키지 설치하고 시작!

 리마인드! 패키지 설치 화면 진입하기

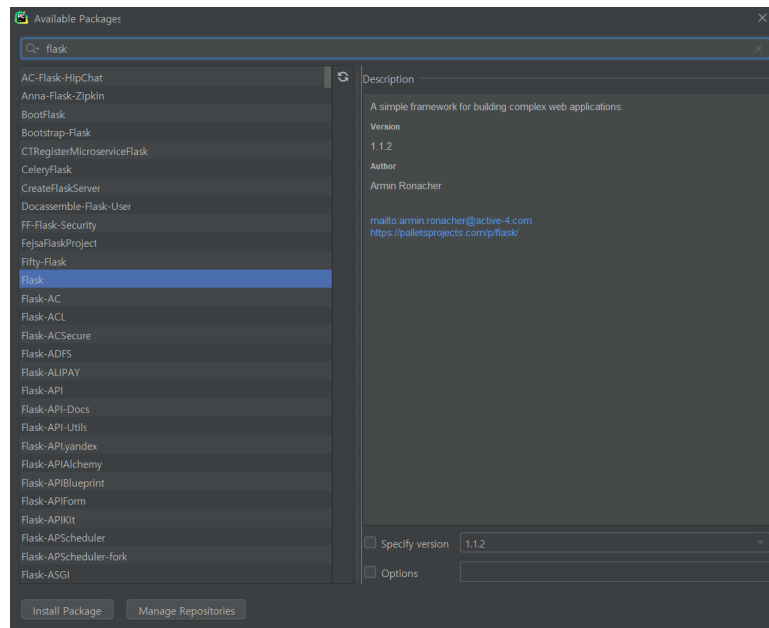
윈도우 : 좌상단File → setting → project interpreter

맥 : 좌상단Pycharm → Preference → Project Interpreter

- project interpreter 화면에서 + 버튼을 누르면 아래 창이 뜹니다!
(맥은 아래에, 윈도우는 오른쪽에 위치)



- flask 로 검색한 후, Install package 클릭



▼ 3) Flask 기초: 기본 실행

- Flask 프레임워크: 서버를 구동시켜주는 편한 코드 모음. 서버를 구동하려면 필요한 복잡한 일들을 쉽게 가져다 쓸 수 있습니다.

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

- `app.py` 파일을 만들어 아래 코드를 붙여넣어봅니다.

👉 파일 이름은 아무렇게나 해도 상관없지만,
통상적으로 flask 서버를 돌리는 파일은 `app.py`라고 이름 짓습니다!

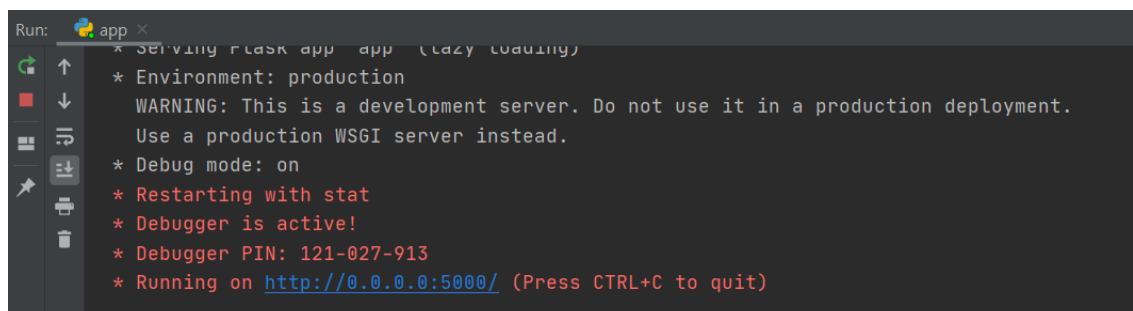
▼ [코드스니펫] - 01. flask 시작 코드

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

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

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

- 오른쪽 클릭 → 'Run app' 을 클릭하고, 터미널에 아래와 같은 메시지가 뜨면 실행 성공!



- 이제 크롬에서 <http://localhost:5000/> 으로 접속해보세요.

☞ 화면에 **Hello World!** 라는 메시지가 보이시나요? 그렇다면 성공한 것! 🌞

- 종료하는 방법

☞ 터미널 창을 클릭하시고, **ctrl + c** 을 누르시면 서버를 종료할 수 있습니다.

▼ 4) Flask 기초: URL 나뉘보기

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

☞ url 별로 함수명이 같거나, `route('/')` 내의 주소가 같으면 안됩니다.

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

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

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

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

04. Flask 시작하기 - HTML 파일 주기

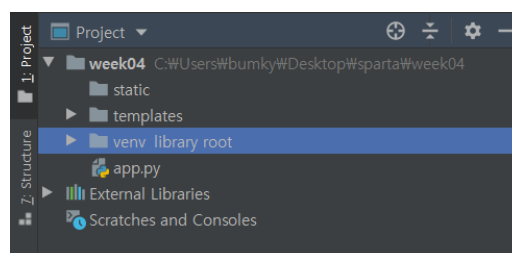
- ▼ 5) Flask 기초: 기본 폴더구조 - 항상 이렇게 세팅하고 시작!

☞ Flask 서버를 만들 때, 항상,

프로젝트 폴더 안에,
 └ **static** 폴더 (이미지, css파일을 넣어둡니다)
 └ **templates** 폴더 (html파일을 넣어둡니다)
 └ **app.py** 파일

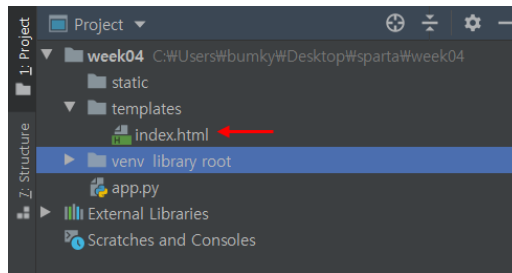
이렇게 세 개를 만들어두고 시작하세요. 이제 각 폴더의 역할을 알아보시다!

(꼭 참고!! venv는 실제로는 보이지만, **안보인다~**라고 생각하세요! 기억하시죠?)



- ▼ 6) Flask 기초: HTML 파일 불러오기

👉 **templates** 폴더의 역할을 알아보겠습니다.
HTML 파일을 담아두고, 불러오는 역할을 하죠!



1. 간단한 index.html 파일을 templates 안에 만들기

▼ [코드스니펫] - 02. index.html 예제코드

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
  <title>Document</title>
</head>
<body>
  <h1>서버를 만들었다!</h1>
</body>
</html>
```

2. html 파일 불러오기

👉 flask 내장함수 render_template를 이용합니다. 바로 이게 프레임워크의 위력!

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

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

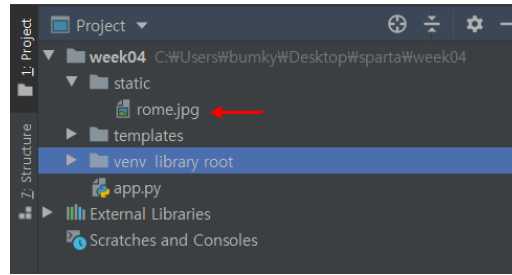
@app.route('/')
def home():
    return render_template('index.html')

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

▼ 7) Flask 기초: HTML 파일 내 이미지를 불러오기

👉 **static** 폴더의 역할을 알아보겠습니다.
이미지나 css파일과 같은 파일을 담아두는 역할을 하지요!

1. static 폴더에 이미지를 넣습니다. 구글 검색 → 아무 이미지나 다운로드!



2. flask 에서 html을 부를 때, 저장된 이미지 파일을 불러오는 법!

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

이제 왜 static, templates 폴더가 존재하는지, 이해하시겠나요?

05. Flask시작하기 - 본격 API 만들기

▼ 8) 들어가기 전에: GET, POST 요청타입 - 리마인드

☞ 리마인드!
은행의 창구가 API와 같다는 것을 기억하시나요?
같은 예금 창구에서도 개인 고객이나 기업 고객이나에 따라 처리하는 것이 다른 것처럼,

클라이언트가 요청 할 때에도, "방식"이 존재합니다.

HTTP 라는 통신 규약을 따른다는 거 잊지 않으셨죠? 클라이언트는 요청할 때 HTTP request method(요청 메소드)를 통해, 어떤 요청 종류인지 응답하는 서버 쪽에 정보를 알려주는 거예요.

☞ 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) GET, POST 요청에서 클라이언트의 데이터를 받는 방법

- 예를 들어, 클라이언트에서 서버에 title_give란 키 값으로 데이터를 들고왔다고 생각합시다.
(주민등록번호 라는 키 값으로 850120- .. 을 가져온 것과 같은 의미)

☞ 받은 값을 개발자가 볼 수 있게 print 로 찍어볼 수 있게 했습니다. 실전에선 print로 찍어주는 것 외에, 여러가지 작업을 할 수 있겠죠?

▼ [코스스니펫] - 04. GET 요청 API코드

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

▼ [코드스니펫] - 05. GET 요청 확인 Ajax코드

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

▼ [코드스니펫] - 06. POST 요청 API코드

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

▼ [코드스니펫] - 07. POST 요청 확인 Ajax코드

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

06. [모두의책리뷰] - 프로젝트 세팅



sparta → projects → bookreview 폴더를 열고 시작!

▼ 10) 문제 분석 - 완성작부터 보기!

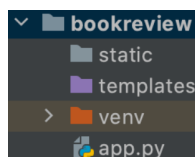
▼ [코드스니펫] - 08. 모두의책리뷰 보러가기

<http://spartacodingclub.shop/bookreview>

▼ 11) 프로젝트 설정 - flask 폴더 구조 만들기



static, templates 폴더 + app.py 만들기! 이젠 너무 익숙하죠?



07. [모두의책리뷰] - 뼈대 준비하기

▼ 12) 프로젝트 준비 - `app.py` 준비하기

▼ [코드스니펫] - 09. 모두의책리뷰-app.py

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

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

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

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

▼ 13) 프로젝트 준비 - index.html 준비하기

▼ [코드스니펫] - 10. 모두의책리뷰-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-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XcFcJLSajwGgFAW/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/ScQsAP7hUibX39j7fakPskvXusvfa0b4Q"
        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 /review 에 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>김르탄</td>
              <td>역시 왕초보 코딩교육의 명가답군요. 따라하다보니 눈 깜짝할 사이에 8주가 지났습니다.</td>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
  </body>
</html>

```

08. [모두의책리뷰] - POST 연습(리뷰 저장)

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

▼ 1. 클라이언트와 서버 확인하기

- 여기서는 적혀 있는 쌍으로 되어있는 서버-클라이언트 코드를 확인하고 같게요.
- 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 가지 입니다.

- 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 가지 입니다.

- 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 성공했다
고 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.

09. [모두의책리뷰] - GET 연습(리뷰 보여주기)

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

▼ 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 가지 입니다.

- 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-Gn5384xqQ1aowXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/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/dist/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 /review 에 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 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);
}

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>
    </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>김르탄</td>
        <td>역시 왕초보 코딩교육의 명가답군요. 따라하다보니 눈 깜짝할 사이에 8주가 지났습니다.</td>
      </tr>
    </tbody>
  </table>
</div>
</div>
</body>
</html>

```

10. [나홀로메모장] - 프로젝트 세팅



sparta → projects → alonememo 폴더를 열고 시작!

▼ 17) 문제 분석 - 완성작부터 보기!

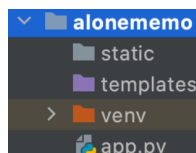
▼ [코드스니펫] - 11. 나홀로메모장 보러가기

<http://spartacodingclub.shop/>

▼ 18) 프로젝트 준비 - flask 폴더 구조 만들기



static, templates 폴더 + app.py 만들기! 이젠 너무 익숙하죠?



11. [나홀로메모장] - API 설계하기

▼ 19) 프로젝트 설계 - 만들 API 설계

✓ 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'

✓ 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

12. [나홀로메모장] - 조각 기능 구현해보기

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

👉 이렇게, API에서 수행해야하는 작업 중 익숙하지 않은 것들은, 따로 python 파일을 만들어 실행해보고, 잘 되면 코드를 붙여넣는 방식으로 하는 게 편합니다.

그럼, meta tag가 뭔지 공부해볼까요?

▼ 어떤 부분에 스크래핑이 필요한가요?

- 우리는 기사URL만 입력했는데, 자동으로 불러와지는 부분들이 있습니다.

👉 함께 확인해볼까요?
<http://spartacodingclub.shop/>

- 바로 '기사 제목', '썸네일 이미지', '내용' 입니다.

👉 이 부분은, 'meta'태그를 크롤링 함으로써 공통적으로 얻을 수 있습니다.
meta태그가 무엇이고, 어떻게 스크래핑 하는지, 함께 살펴볼까요?



그린 북

1962년 미국, 입담과 주먹만 믿고 살아가던 토니 발레루가(비고 모텐슨)는 교양과 우아함 그 자체인현재...

안녕하세요. 케이론입니다오늘 리뷰할 영화는그린 북입니다.지극히 개인적인 몇 줄 평<그린 북>은 2018 토포로국제영화제 관객상 수상과다가오는 골든 글로브 시상식에 서감독상, 남우주연상 등 5개 부문 노미네이트는 물론크리틱...



매트릭스

인간의 기억마저 AI에 의해 입력되고 삭제되는 세상.진짜보다 더 진짜 같은 가상 현실 '매트릭스'그 ...

<영화 매트릭스 줄거리와 그 의미> by goodhand (네이버 영화평) 이 글은 <매트릭스3>의 영화평 1위에 올라와 있는 글인데, 사실은 매트릭스 시리즈 전반에 대한 영화평이기 때문에 <매트릭스>의 감상평으로 다시 올...



동주

이름도, 언어도, 꿈도, 모든 것이 허락되지 않았던 일제강점기.한 집에서 태어나고 자란 동갑내기 사촌...

이 영화를 아직 보지 않은 분들께 말씀드립니다.이 영화는 반일 독립투사 영화가 아닙니다.민족 애국혼을 자극해 눈물을 강요하는 영화도 아닙니다.이 영화는 시인 윤동주와 그 친구 송몽규 선생의우정과 삶과 시그리고 죽음을 정말 담...

▼ meta 태그에 대해 알아보기

- (링크)에 접속한 뒤 크롬 개발자 도구를 이용해 HTML의 생김새를 살펴볼까요?
- 메타 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

- 연습을 위해 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/71.0.3578.98 Safari/537.36'}
data = requests.get(url,headers=headers)

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

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

- select_one을 이용해 meta tag를 먼저 가져와봅니다.

👉 select의 새로운 사용법! 이렇게 또 알아가네요~!

```
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)
```

- 가져온 meta tag의 content를 가져와봅시다.

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

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

13. [나홀로메모장] - 뼈대 준비하기

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

▼ [코드스니펫] - 12. 나홀로메모장-app.py

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

import requests
from bs4 import BeautifulSoup

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

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

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

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

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

▼ [코드스니펫] - 13. 나홀로메모장-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+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/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=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 열
기
      </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
무궁화 삼천리 화려강산...</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
무궁화 삼천리 화려강산...</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
무궁화 삼천리 화려강산...</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
무궁화 삼천리 화려강산...</p>
        <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
      </div>
    </div>
  </div>

```



```

무궁화 삼천리 화려강산...</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
무궁화 삼천리 화려강산...</p>
      <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
    </div>
  </div>
</div>
</div>
</body>
</html>

```

14. [나홀로메모장] - POST 연습(메모하기)

▼ 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) 리스팅API - 저장된 카드 보여주기 (Read)

▼ 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 가지 입니다.

- 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
    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 가지 입니다.

- URL (url_give) : meta tag를 가져올 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 있으니, meta tag 가 있는 사이트만 저장이 제대로 되겠죠?)

참고!

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

15. [나홀로메모장] - GET 연습(보여주기)

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

✅ 우리가 만들 API 두 가지

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

▼ 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 거니까요!

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+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/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=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 바뀌두기
          $("#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 바뀌두기
          $("#btn-post-box").text("포스팅 박스 닫기");
        }
      }
    </script>
  </html>
```

```

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"]
            )
            }
        }
    })
}

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 class="card-text comment">여기에 코멘트가 들어갑니다.</p>
                </div>
            </div>
        </div>
    </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 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 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 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 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 돌아옵니다.
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
    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

```

```

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)

```

16. 끝 & 숙제 설명



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!

아래 주문정보를 붙일 때에는, 보트스트랩 Table(링크)을 이용하세요!

▼ [코드스니펫] - 14. 나홀로쇼핑몰 보러가기

<http://spartacodingclub.shop/homework>

▼ [코드 app.py] - 뼈대 코드로 사용하세요!

▼ [코드스니펫] - 15. 원페이지쇼핑몰-app.py

```

from flask import Flask, render_template, jsonify, request

app = Flask(__name__)

from pymongo import MongoClient

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

```



```


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

# 주문하기(POST) API
@app.route('/order', methods=['POST'])
def save_order():
    return jsonify({'result': 'success', 'msg': '이 요청은 POST!'})

# 주문 목록보기(Read) API
@app.route('/order', methods=['GET'])
def view_orders():
    return jsonify({'result': 'success', 'msg': '이 요청은 GET!'})

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

```

▼ [ 코드 `index.html`] - 2주차 숙제가 없는 사람만!