# 스파르타코딩클럽 7기 4주차 - 학습자료

**[수업 목표]**

1. pymongo를 통해 mongoDB를 제어할 수 있다.
2. Flask 프레임워크를 활용해서 API를 만들 수 있다.

📢: 실습 시간

* 강의 상황에 따라, 시간은 유동적일 수 있습니다.

*전반 3시간*

**[시작]**

▼ "15초 체크인"을 진행합니다.

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

▼ 간단 복습하기 & 오늘 배울 것 이야기

- 튜터님이 5분 동안 지난번에 배웠던 키워드들과, 오늘 배울것을 이야기합니다.

**[1시간]: pymongo 사용하기 + Robo 3T로 확인하기**

▼ 1) DB 설치 확인

먼저, 각자 설치해온 DB가 잘 작동하는지 확인합니다. 크롬 창에 localhost:27017 이라고 쳤을 때, 아래와 같은 화면이 나오면 mongoDB가 돌아가고 있는 것입니다.


▼ 2) 들어가기 전에 (1) : DB의 두 가지 종류

- Database에는, 크게 두 가지 종류가 있습니다.

- RDBMS(SQL): 행/열의 생김새가 정해진 엑셀에 데이터를 저장하는 것과 유사합니다.
  데이터 50만 개가 적재된 상태에서, 갑자기 중간에 열을 하나 더하기는 어려울 것입니다. 그러나, 정형화되어 있는 만큼, 데이터의 일관성이나 / 분석에 용이할 수 있습니다.
  ex) MS-SQL, My-SQL 등
- No-SQL: 딕셔너리 형태로 데이터를 저장해두는 DB입니다. 고로 데이터 하나 하나 마다 같은 값들을 가질 필요가 없게 됩니다. 자유로운 형태의 데이터 적재에 유리한 대신, 일관성이 부족할 수 있습니다.
  ex) MongoDB

▼ 3) 들어가기 전에(2) : mongoDB의 구조/데이터 적재 방식

- mongoDB는, db 아래에 collection이 있는 구조입니다.
- db는 엑셀파일의 이름. collection은 시트에 해당한다고 보셔도 무방합니다.
  ex) "baemin" db아래에, "user"와, "restaurants"라는 collection이 있을 수 있음

▼ 4) pymongo로 조작하기

- 패키지 설치하기

```
pymongo
```

- DB연결하기 & 데이터 넣기

```
from pymongo import MongoClient           # pymongo를 임포트 하기(패키지 인스톨 먼저 해야겠죠?)
client = MongoClient('localhost', 27017)  # mongoDB는 27017 포트로 돌아갑니다.
db = client.dbsparta                      # 'dbsparta'라는 이름의 db를 만듭니다.

# MongoDB에 insert 하기

# 'users'라는 collection에 {'name':'bobby','age':21}를 넣습니다.
db.users.insert_one({'name':'bobby','age':21})
db.users.insert_one({'name':'kay','age':27})
db.users.insert_one({'name':'john','age':30})
```

- 모든 결과 값을 보기

```
# MongoDB에서 데이터 모두 보기
all_users = list(db.users.find())

# 참고) MongoDB에서 특정 조건의 데이터 모두 보기
same_ages = list(db.users.find({'age':21}))

print(all_users[0])         # 0번째 결과값을 보기
print(all_users[0]['name']) # 0번째 결과값의 'name'을 보기

for user in all_users:      # 반복문을 돌며 모든 결과값을 보기
    print(user)
```

- 특정 결과 값을 뽑아 보기

```
user = db.users.find_one({'name':'bobby'})
print (user)

# 그 중 특정 키 값을 빼고 보기
user = db.users.find_one({'name':'bobby'},{'_id':0})
print (user)
```

- 수정하기

```
# 생김새
db.people.update_many(찾을조건,{ '$set': 어떻게바꿀지 })

db.users.update_one({'name':'bobby'},{'$set':{'age':19}})
```

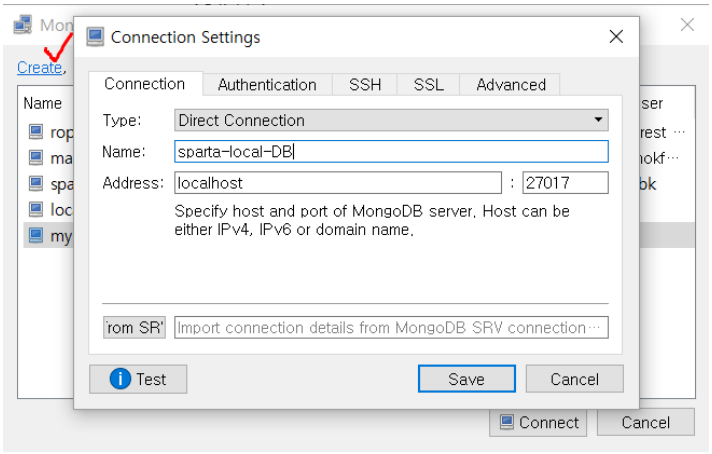
```
user = db.users.find_one({'name':'bobby'})
print (user)
```

- 삭제하기 (거의 안 씀)

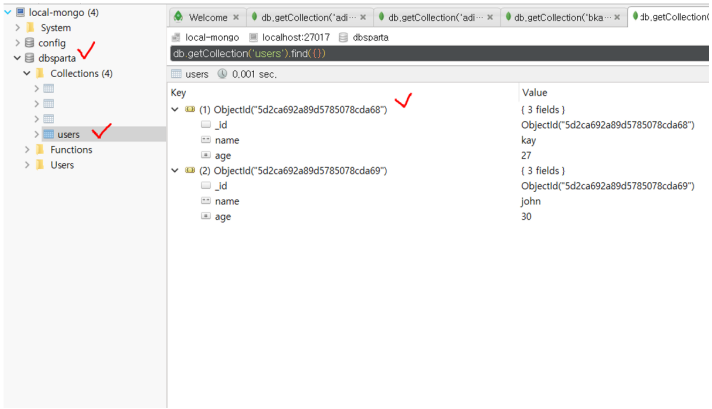
```
db.users.delete_one({'name':'bobby'})

user = db.users.find_one({'name':'bobby'})
print (user)
```

▼ 5) robo 3T로 확인하기

- mongoDB 내의 데이터를 좀 더 편하게 확인하는 방법
  (sparta-local-DB는 아무 이름이나 입력해도 됩니다)




- db, collection, documents(각 데이터들을 지칭)를 확인 가능합니다.




**[1시간]: 스크래핑 한 결과를 mongoDB에 저장하기**

▼ 6) insert 연습하기 - 웹스크래핑 결과를 DB에 저장하기

- 이 코드에서 시작해봅시다! (복사+붙여넣기)

```
import requests
from bs4 import BeautifulSoup

# URL을 읽어서 HTML를 받아오고,
headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.
data = requests.get('https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=pnt&date=20200303',headers=headers)

# HTML을 BeautifulSoup이라는 라이브러리를 활용해 검색하기 용이한 상태로 만듦
soup = BeautifulSoup(data.text, 'html.parser')
```

```
# select를 이용해서, tr들을 불러오기
movies = soup.select('#old_content > table > tbody > tr')

# movies (tr들) 의 반복문을 돌리기
rank = 1
for movie in movies:
    # movie 안에 a 가 있으면,
    a_tag = movie.select_one('td.title > div > a')
    if a_tag is not None:
        title = a_tag.text
        star = movie.select_one('td.point').text
        print(rank,title,star)
        rank += 1
```

- pymongo 기본 세팅

```
import requests
from bs4 import BeautifulSoup

from pymongo import MongoClient           # pymongo를 임포트 하기(패키지 인스톨 먼저 해야겠죠?)
client = MongoClient('localhost', 27017)  # mongoDB는 27017 포트로 돌아갑니다.
db = client.dbsparta                      # 'dbsparta'라는 이름의 db를 만듭니다.

# URL을 읽어서 HTML를 받아오고,
headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.
data = requests.get('https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=pnt&date=20200303',headers=headers)

# HTML을 BeautifulSoup이라는 라이브러리를 활용해 검색하기 용이한 상태로 만듦
soup = BeautifulSoup(data.text, 'html.parser')

# select를 이용해서, tr들을 불러오기
movies = soup.select('#old_content > table > tbody > tr')

# movies (tr들) 의 반복문을 돌리기
rank = 1
for movie in movies:
    # movie 안에 a 가 있으면,
    a_tag = movie.select_one('td.title > div > a')
    if a_tag is not None:
        title = a_tag.text
        star = movie.select_one('td.point').text
        print(rank,title,star)
        rank += 1
```

- 도큐먼트 만들어 하나씩 insert 하기

```
import requests
from bs4 import BeautifulSoup

from pymongo import MongoClient           # pymongo를 임포트 하기(패키지 인스톨 먼저 해야겠죠?)
client = MongoClient('localhost', 27017)  # mongoDB는 27017 포트로 돌아갑니다.
db = client.dbsparta                      # 'dbsparta'라는 이름의 db를 만듭니다.

# URL을 읽어서 HTML를 받아오고,
headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.
data = requests.get('https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=pnt&date=20200303',headers=headers)

# HTML을 BeautifulSoup이라는 라이브러리를 활용해 검색하기 용이한 상태로 만듦
soup = BeautifulSoup(data.text, 'html.parser')

# select를 이용해서, tr들을 불러오기
movies = soup.select('#old_content > table > tbody > tr')

# movies (tr들) 의 반복문을 돌리기
rank = 1
for movie in movies:
    # movie 안에 a 가 있으면,
    a_tag = movie.select_one('td.title > div > a')
    if a_tag is not None:
        title = a_tag.text
        star = movie.select_one('td.point').text

        doc = {
            'rank' : rank,
            'title' : title,
            'star' : star
```

```
        }
        db.movies.insert_one(doc)
        rank += 1
```

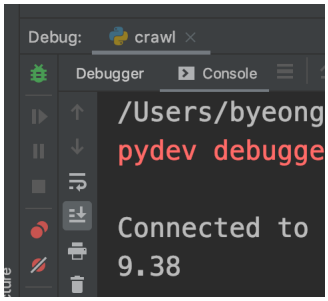
▼ 7 ) find, update 연습하기 (delete는 연습 안할게요!)

- 파이썬 파일을 새로 하나 만들어 연습해봅니다.

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

## 코딩 할 준비 ##
```

- 📢(1) '어벤져스: 엔드게임'의 평점을 가져오기

  ▼ 예시 - 이렇게 되면 완성

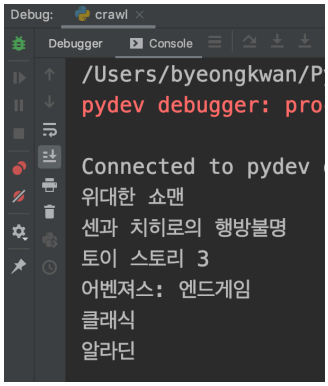



  ▼ 코드

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

## 코딩 할 준비 ##

target_movie = db.movies.find_one({'title':'어벤져스: 엔드게임'})
print (target_movie['star'])
```

- 📢(2) '어벤져스: 엔드게임'의 평점과 같은 평점의 영화 제목들을 가져오기

  ▼ 예시 - 이렇게 되면 완성




  ▼ 코드

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

## 코딩 할 준비 ##
```

```
target_movie = db.movies.find_one({'title':'어벤져스: 엔드게임'})
target_star = target_movie['star']

movies = list(db.movies.find({'star':target_star}))

for movie in movies:
    print(movie['title'])
```

- 📢(3) '어벤져스: 엔드게임'의 평점과 같은 평점의 영화 제목들의 평점을 0으로 만들기

  ▼ 예시 - 이렇게 되면 완성

  (robo3T로 봤을 때)

| | | | | |
|---|---|---|---|---|
| 12 | ObjectId("5... | 12 | 월-E | 9.40 |
| 13 | ObjectId("5... | 13 | 알라딘 | 0 |
| 14 | ObjectId("5... | 14 | 나 홀로 집에 | 0 |
| 15 | ObjectId("5... | 15 | 매트릭스 | 0 |
| 16 | ObjectId("5... | 16 | 인생은 아름다워 | 0 |
| 17 | ObjectId("5... | 17 | 라이언 일병 구하기 | 0 |
| 18 | ObjectId("5... | 18 | 백 투 더 퓨쳐 | 0 |
| 19 | ObjectId("5... | 19 | 헬프 | 0 |
| 20 | ObjectId("5... | 20 | 포레스트 검프 | 0 |
| 21 | ObjectId("5... | 21 | 사운드 오브 뮤직 | 0 |
| 22 | ObjectId("5... | 22 | 글래디에이터 | 9.38 |
| 23 | ObjectId("5... | 23 | 살인의 추억 | 9.38 |

  ▼ 코드

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

## 코딩 할 준비 ##

target_movie = db.movies.find_one({'title':'어벤져스: 엔드게임'})
target_star = target_movie['star']

db.movies.update_many({'star':target_star},{'$set':{'star':0}})
```

**[1시간] : 드디어! 서버를 만든다! - Flask 기초 (HTML페이지를 주는 API)**

▼ 8) HTML을 주는 API: 기본 실행

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

> 👉 프레임워크를 쓰지 않으면 태양초를 담궈서 고추장을 만들어야 하는 격!
> 프레임워크는 3분 요리/소스 세트라고 생각하면 되겠습니다!

- Pycharm에서 Flask 패키지를 설치하고 아래 코드를 실행하면 서버 동작

```
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)
```

▼ 9) HTML을 주는 API: URL 나눠보기

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

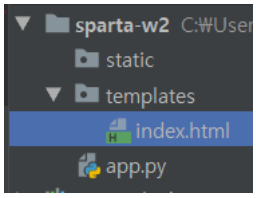
👉 url 별로 함수명이 같거나, route('/')내의 주소가 같으면 안됩니다.

```
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)
```

▼ 10) HTML을 주는 API: HTML 파일 불러오기

- 기본 폴더구조

👉 flask 프레임워크를 쓸 때의 기본 폴더구조입니다. 오타에 유의하세요!
반드시 templates 폴더내에 html 파일을 둬야 불러올 수 있습니다.




- html 파일 불러오기

👉 render_template를 import 에 추가하고, home()내에 써보았습니다.

```
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)
```

▼ 11) HTML을 주는 API: 앗. render_template를 했더니 css가 깨져요!

*후반 3시간*

**[1시간] : Flask - 본격 API 만들기 (JSON 데이터를 주는 API)**

▼ 12) 들어가기 전에: GET, POST 요청타입 - 리마인드

- '요청 타입'이란?

👉 같은 예금 창구도 입금 내역, 입금 하기 두 가지 기능을 모두 할 수 있는 것처럼
같은 URL 이더라도 타입을 통해 창구의 기능을 구분해둘 수 있습니다.

# 예금 창구

## 입금내역
Read

## 입금하기
Write

💡 마찬가지로 영화 서비스라면 영화 창구를 기준으로 목록보기, 영화추가 기능을 요청의 Method(GET/POST)로 구분합니다.

# /movie

## 목록보기
GET

## 영화추가
POST

- 데이터 전달 방식은 어떻게 다른가요?

  👉 서버에 데이터를 전달할  때에,

  GET은 → URL 뒤에 물음표를 붙여 변수를 전달하고 (ex: google.com?q=북극곰)
  POST는 → 보이지 않는 body에 key:value 형태로 적어서 보내옵니다.

  👉 눈치 채셨나요?

  우리가 평소에 브라우저에서 URL을 입력한 뒤 엔터를 누르는 것은, GET 요청이랍니다! "google.com?q=북극곰"의 예제를 보면 알 수 있죠?

- 그 외엔 무엇이 있나요?

> 👉 PUT, HEAD, DELETE ... 등 아주 많습니다. 그러나 우리는, 가장 많이 쓰이는 GET, POST 두개만 공부하겠습니다!

▼ 13) 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?

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

@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)
```

- ajax에서 테스트 요청은 이렇게 해보면 됩니다.

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

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

**[1시간] : Flask - 본격 API 만들기 (mongoDB를 엮어보자)**

▼ 14) 완성작부터 보기!

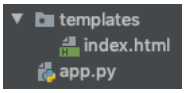
👉완성작 보러가기

▼ 15) pymongo를 임포트하기

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

from pymongo import MongoClient          # pymongo를 임포트 하기(패키지 인스톨 먼저 해야겠죠?)
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('/reviews', methods=['POST'])
def write_review():
  return jsonify({'result':'success', 'msg': '이 요청은 POST!'})


@app.route('/reviews', 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)
```

▼ 16) index.html 준비하기




templates 폴더 아래 index.html 파일을 만들고, 다음 내용을 복사 붙여넣기 합니다.

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

<head>
    <!-- Webpage Title -->
    <title>나홀로 책 리뷰 | 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/3.4.1/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>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
        integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
        crossorigin="anonymous"></script>

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

    <script type="text/javascript">

        $(document).ready(function () {
            $('#orders-box').html('');
            listing();
        });

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

        function listing() {
            // 1. 리뷰 목록을 서버에 요청하기
            // 2. 요청 성공 여부 확인하기
```

```
            // 3. 요청 성공했을 때 리뷰를 올바르게 화면에 나타내기
            $.ajax({
                type: "GET",
                url: "/reviews",
                data: {},
                success: function (response) {
                    if (response['result'] == 'success') {
                        alert(response['msg']);
                        // 2. 성공했을 때 리뷰를 올바르게 화면에 나타내기
                    } else {
                        alert('리뷰를 받아오지 못했습니다');
                    }
                }
            });
        }

        function make_card(title, author, review) {
            let temp_html = '<tr>\
                                <td>'+ title + '</td>\
                                <td>'+ author + '</td>\
                                <td>'+ review + '</td>\
                            </tr>';
            $('#orders-box').append(temp_html);
        }

        function is_long(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;
        }

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

        .img {
            background-image: url('https://previews.123rf.com/images/maxxyustas/maxxyustas1511/maxxyustas151100002/47858355
            background-size: cover;
            background-position: center;
            width: 500px;
            height: 300px;
        }

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

        .user-info {
            margin: 20px 5px auto 5px;
        }

        h1,
        h5 {
            display: inline;
        }

        .order {
            text-align: center;
        }

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

<body>
    <div class="wrap">
        <div class="img"></div>
        <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>
        <div class="info">
            <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" aria-describedby="basic-addon3">
            </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" aria-describedby="basic-addon3">
            </div>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">리뷰</span>
                </div>
                <textarea class="form-control" aria-describedby="basic-addon3" name="bookReview" id="review" cols="30"
                    rows="5" placeholder="140자까지 입력할 수 있습니다." onkeyup="is_long(this)"></textarea>
            </div>
            <div class="order">
                <button onclick="make_review()" type="button" class="btn btn-primary">리뷰 작성하기</button>
            </div>
        </div>
        <div class="orders">
            <table class="table">
                <thead>
                    <tr>
                        <th scope="col">제목</th>
                        <th scope="col">저자</th>
                        <th scope="col">리뷰</th>
                    </tr>
                </thead>
                <tbody id="orders-box">
                    <tr>
                        <td>Otto</td>
                        <td>@mdo</td>
                        <td>@mdo</td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
</body>

</html>
```

▼ 17) POST 연습: 제목, 저자, 리뷰 정보 삽입하기

▼ 1) 클라이언트와 서버 연결 확인하기

아래 클라이언트/서버 시작 코드는 쌍을 이루고 있습니다.

[ 서버 시작 코드 ]

```
## API 역할을 하는 부분
@app.route('/reviews', 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 시작 코드 ]

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

👉 '리뷰 시작하기' 버튼을 눌렀을 때, '리뷰가 성공적으로 작성되었습니다.' 라는 내용의 alert가 뜨면 연결에 성공한 것입니다!

▼ 2) 서버부터 만들기

💡 API = 약속이라고 했습니다! 이 약속을 서버에서부터 시작해보겠습니다.

리뷰를 작성하기 위해 필요한 정보는 다음 세 가지 입니다.
- 제목(title)
- 저자(author)
- 리뷰(review)

따라서 서버 로직은 다음 세 단계로 구성되어야 합니다.
1. 클라이언트가 준 title, author, review 가져오기.
2. DB에 정보 삽입하기
3. 성공 여부 & 성공 메시지 반환하기

```
@app.route('/reviews', 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 = 약속이라고 했습니다! 이제는 클라이언트를 만들어 보겠습니다.

리뷰를 작성하기 위해 필요한 정보는 다음 세 가지 입니다.
- 제목(title)
- 저자(author)
- 리뷰(review)

따라서 클라이언트 로직은 다음 세 단계로 구성되어야 합니다.
1. input에서 title, author, review 가져오기
2. 입력값이 하나라도 없을 때 alert 띄우기.
3. Ajax로 서버에 저장 요청하고 기존 입력값 비우기

```
function make_review() {
    // 1. 제목, 저자, 리뷰 내용을 가져옵니다.
    let title = $('#title').val();
    let author = $('#author').val();
    let review = $('#review').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');
        $('#review').focus();
        return;
    }

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

▼ 4) 완성 확인하기

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

▼ 18) GET 연습: 저장된 리뷰를 받아와서 화면에 보여주기

▼ 1) 클라이언트와 서버 연결 확인하기

아래 클라이언트/서버 시작 코드는 쌍을 이루고 있습니다.

[ 서버 시작 코드 ]

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

[ 클라이언트 시작 코드 ]

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

▼ 2) 서버부터 만들기



API = 약속이라고 했습니다! 이 약속을 서버에서부터 시작해보겠습니다.

서버 로직은 다음 단계로 구성되어야 합니다.
1. DB에서 리뷰 정보 모두 가져오기
2. 성공 여부 & 리뷰 목록 반환하기

```
@app.route('/reviews', 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 = 약속이라고 했습니다! 이제는 클라이언트를 만들어 보겠습니다.

클라이언트 로직은 다음 단계로 구성되어야 합니다.
1. 리뷰 목록을 서버에 요청하기
2. 요청 성공 여부 확인하기
3. 요청 성공했을 때 리뷰를 올바르게 화면에 나타내기

```
function listing() {
    // 1. 리뷰 목록을 서버에 요청하기
    $.ajax({
        type: "GET",
        url: "/reviews",
        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++) {
                    make_card(reviews[i]['title'], reviews[i]['author'], reviews[i]['review']);
                }
            } else {
                alert('리뷰를 받아오지 못했습니다');
            }
        }
    })
}
```

▼ 4) 완성 확인하기

새로고침했을 때 DB에 저장된 리뷰가 화면에 올바르게 나타나는지 확인합니다.

▼ 19) 전체 완성 코드!

[ 서버 ]

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

app = Flask(__name__)

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

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('/reviews', methods=['POST'])
def write_review():
    title_receive = request.form['title_give']
    author_receive = request.form['author_give']
    review_receive = request.form['review_give']

    review = {
       'title': title_receive,
       'author': author_receive,
       'review': review_receive
    }
```

```
    db.reviews.insert_one(review)
    return jsonify({'result': 'success', 'msg': '리뷰가 성공적으로 작성되었습니다.'})


@app.route('/reviews', methods=['GET'])
def read_reviews():
    reviews = list(db.reviews.find({},{'_id':0}))
    return jsonify({'result': 'success', 'reviews': reviews})


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

[ 클라이언트 ]

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

<head>
    <!-- Webpage Title -->
    <title>나홀로 책 리뷰 | 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/3.4.1/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>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
        integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
        crossorigin="anonymous"></script>

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

    <script type="text/javascript">
        function is_long(obj) {
            let content = $(obj).val();
            console.log(content);
            console.log(content.length);
            if (content.length > 140) {
                alert('리뷰는 140자까지 기록할 수 있습니다.');
                $(obj).val(content.substring(0, 140));
            }
        }

        function make_review() {
            let title = $('#title').val();
            let author = $('#author').val();
            let review = $('#review').val();

            if (title == '') {
                alert('제목을 입력해주세요');
                $('#title').focus();
                return;
            } else if (author == '') {
                alert('저자를 입력해주세요');
                $('#author').focus();
                return;
            } else if (review == '') {
                alert('리뷰를 입력해주세요');
                $('#review').focus();
                return;
            }

            $.ajax({
                type: "POST",
                url: "/reviews",
                data: { title_give: title, author_give: author, review_give: review },
                success: function (response) {
                    if (response['result'] == 'success') {
                        alert(response['msg']);
                        $('#title').val('');
                        $('#author').val('');
                        $('#review').val('');
                        window.location.reload();
```

```
                }
            }
        })
    }

    $(document).ready(function () {
        $('#orders-box').html('');
        listing();
    });

    function listing() {
        $.ajax({
            type: "GET",
            url: "/reviews",
            data: {},
            success: function (response) {
                if (response['result'] == 'success') {
                    let reviews = response['reviews'];
                    for (let i = 0; i < reviews.length; i++) {
                        make_card(reviews[i]['title'], reviews[i]['author'], reviews[i]['review']);
                    }
                } else {
                    alert('리뷰를 받아오지 못했습니다');
                }
            }
        })
    }

    function make_card(title, author, review) {
        let temp_html = '<tr>\
                            <td>'+ title + '</td>\
                            <td>'+ author + '</td>\
                            <td>'+ review + '</td>\
                        </tr>';
        $('#orders-box').append(temp_html);
    }
</script>

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

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

    .img {
        background-image: url('https://previews.123rf.com/images/maxxyustas/maxxyustas1511/maxxyustas151100002/47858355
        background-size: cover;
        background-position: center;
        width: 500px;
        height: 300px;
    }

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

    .user-info {
        margin: 20px 5px auto 5px;
    }

    h1,
    h5 {
        display: inline;
    }

    .order {
        text-align: center;
    }

    .orders {
        margin-top: 100px;
    }

    .meta_info {
        width: 20%;
    }
</style>
</head>

<body>
    <div class="wrap">
        <div class="img"></div>
```

```
        <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>
        <div class="info">
            <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" aria-describedby="basic-addon3">
            </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" aria-describedby="basic-addon3">
            </div>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text">리뷰</span>
                </div>
                <textarea class="form-control" aria-describedby="basic-addon3" name="bookReview" id="review" cols="30"
                    rows="5" placeholder="140자까지 입력할 수 있습니다." onkeyup="is_long(this)"></textarea>
            </div>
            <div class="order">
                <button onclick="make_review()" type="button" class="btn btn-primary">리뷰 작성하기</button>
            </div>
        </div>
        <div class="orders">
            <table class="table">
                <thead>
                    <tr>
                        <th class="meta_info" scope="col">제목</th>
                        <th class="meta_info" scope="col">저자</th>
                        <th scope="col">리뷰</th>
                    </tr>
                </thead>
                <tbody id="orders-box">
                    <tr>
                        <td>Otto</td>
                        <td>@mdo</td>
                        <td>@mdo</td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
</body>

</html>
```

**[1시간] : 소화 타임**

▼ 일단 오늘 배운 것을 복습하고, 본격적으로 숙제를 시작합니다.

- 주요 키워드들: pymongo로 DB 저장하기. flask 기초. render_template 활용하기, get/post 요청 만들기.
- 키워드들을 바탕으로 튜터가 수업 내용을 간단히 복습해줍니다. (5분)

**[끝]**

▼ "15초 체크아웃"을 진행합니다.

- 튜터는 타이머를 띄워주세요! (링크)
- 마찬가지로, 현재 본인의 감정상태와 수업후기에 관해 이야기합니다.

- 하단 숙제 & 설치해와야 할 것들을 설명합니다.

*숙제 & 설치*

**[숙제1] - 다음 수업 D-1 까지 자신의 github에 올리고, url을 카톡방에 공유하기**

- 1주차에 완성한 쇼핑몰을 완성해주세요!

> 👉 페이지가 로딩되면, 두 가지 기능을 수행해야 합니다.
>
> 1) 주문하기(POST): 주문 정보 입력 후 '주문하기' 클릭 시 주문 목록에 추가되어야 합니다.
> 2) 주문내역보기(GET): 하단 주문 목록이 자동으로 보여야 합니다.
>
> 아래 완성본을 참고해주세요!
> http://spartacodingclub.shop/homework

**[숙제2] - 6~8주차 프로젝트 구상해보기**

- 6~8주차에 어떤 프로젝트를 만들 것인지, 구상해보세요! 다음시간에 5분씩 발표 할 예정입니다.

  종이에 펜으로, "완성본의 생김새"를 그려오면 가장 좋습니다!

  > 👉 기 수강생 작품 참고: (링크)

- 프로젝트는 단독 또는 팀으로 진행됩니다. 일단 모든 분이 아이디어를 생각해와주시면, 5주차에 팀으로 하고픈 분들을 조사하여 튜터가 짝을 지어주겠습니다.

  > 👉 서로 미리 연락해서 팀을 만들어오는 것도 환영입니다!

**[설치] - 다음 시간을 위해 미리 설치해와야 할 것들**

- 없음!