스파르타코딩클럽 8기 4주차

🔥 **매 주차 강의자료 시작에 PDF파일과 영상 링크를 올려두었어요!**

▼ PDF 강의자료 다운받기

> https://s3-us-west-2.amazonaws.com/secure.notion-static.com/9c5b031c-f784-4b8f-8745-fa2749928781/scc-8-1.pdf

▼ 영상강의 참고하기

- 1주차 복습용 영상강의 링크

[수업 목표]

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

전반 3시간

[시작] : 체크인 & 출석체크

🔥 **튜터님은 체크인과 함께 출석 체크(링크)를 진행해주세요!**

▼ "15초 체크인"을 진행합니다.

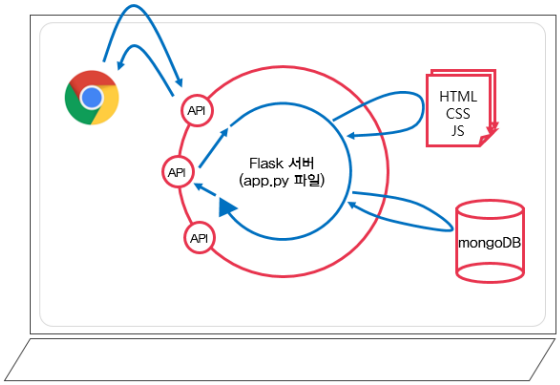
- 튜터님은 타이머를 띄워주세요! (링크)
- 본인의 감정상태와 오늘 있었던 소소한 일을 공유하는 시간입니다.

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

이번 주 완성본 1. 모두의책리뷰 → 결과물 링크

이번 주 완성본 2. 나홀로메모장 → 결과물 링크

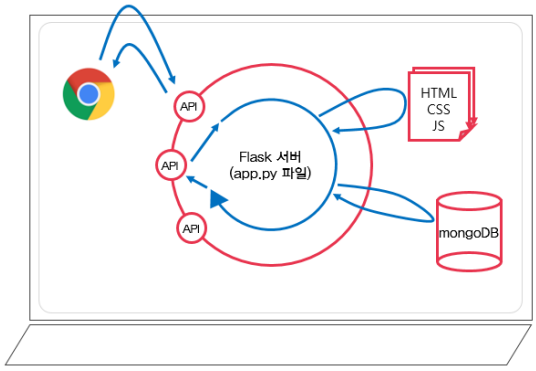
> 👉 오늘은 HTML과 mongoDB까지 연동해서 서버를 만들어봅니다!



x 2개 더!

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

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



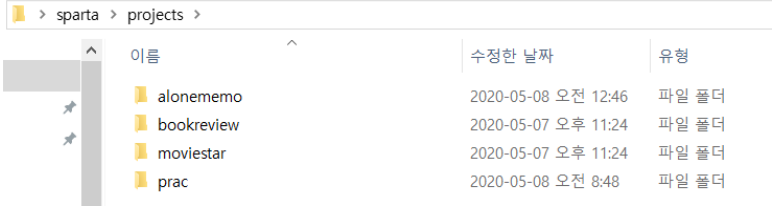
[5분] : 시작하기 전에 - 코드가 꼬이지 않도록

▼ 1) 폴더 네 개 만들고 시작하기

👉 웹개발의 꽃, 백엔드-프론트엔드를 연결하는 일이 익숙해지도록,
연습 → 모두의책리뷰 → 나홀로메모장 → 마이페이보릿무비스타

총 4번에 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 4개



- alonememo : "나홀로메모장" 관련 코드를 작성합니다. (오늘)
- bookreview : "모두의책리뷰" 관련 코드를 작성합니다. (오늘)
- moviestar : "마이페이보릿무비스타" 관련 코드를 작성합니다. (다음주)
- prac : flask기초를 연습 할 폴더 (오늘)

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

> 🔥 **VS Code 파일→폴더열기를 클릭해서,**
> **sparta > project > prac 폴더를 열고 시작!**

▼ 2) 새 프로젝트니까 - 가상환경 다시 잡기

> 👉 프로젝트마다 공구함이 필요하다고 했던 말, 기억하시나요!

- **터미널 → 새 터미널**을 열어서, 아래를 입력!
  (눈치 챘나요? 맨 마지막 myenv는 공구함 이름이라, 마음대로 정해도 무방!)

```
python -m venv myenv
```

- 다시, **터미널 → 새 터미널**을 열면, 공구함 리로딩 완료!
- 공구함 만든 김에, 필요한 패키지를 미리 설치해둡시다!

```
pip install flask
```

▼ 3) Flask 기초: 기본 실행

▼ 4) Flask 기초: URL 나눠보기

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

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

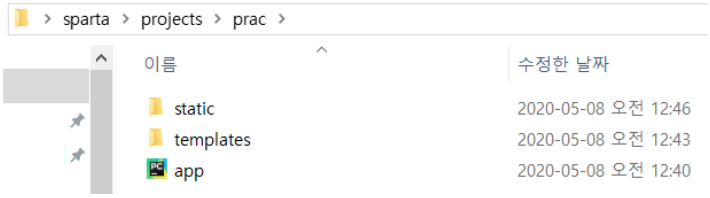
```
from flask import Flask
app = Flask(__name__)

@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)
```

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

👉 Flask 서버를 만들 때, 항상,

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

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



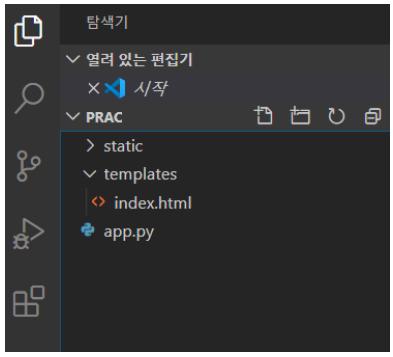
▼ 6) Flask 기초: HTML 파일 불러오기

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

1. 간단한 index.html 파일을 templates 안에 만들기

▼ 아래 코드를 붙여넣어보세요!

```
<!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,
```

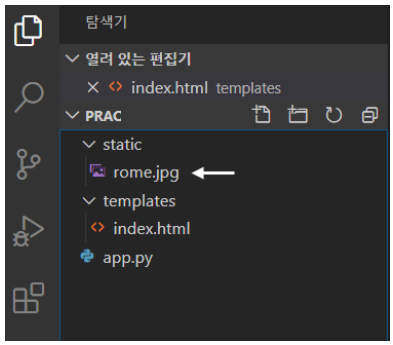
```
## route('/') 등의 주소가 같으면 안됩니다.

@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 아래 이미지를 넣습니다.

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



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

▼ 8) 들어가기 전에: GET, POST 요청타입 - 리마인드

- 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.

👉 통상적으로 **서버 데이터를 가져오기만 하면 GET**
(예 : 학생 중 3학년 학생의 이름을 모두 가져와라)

서버 데이터 수정이 있으면 POST를 사용합니다.
(예: 학생을 한 명 추가해라)
(예: 학생의 중간고사 점수를 업데이트 해라)

- 그 외엔 무엇이 있나요?

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

▼ 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?

```
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.

  👉 index.html 파일에서 개발자도구를 열어 아래를 붙여놓고 이해해볼게요!

  GET 요청 테스트

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

  POST 요청 테스트

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

[1시간] : 모두의책리뷰 완성하기

🔥 **VS Code 파일→폴더열기를 클릭해서,**
sparta > project > bookreview 폴더를 열고 시작!

▼ 10) 완성작부터 보기!

▼ 11) 새 프로젝트니까 - 가상환경 다시 잡기

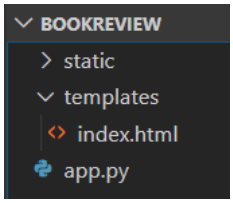
- **터미널 → 새 터미널**을 열어서, 아래를 입력!

```
python -m venv myenv
```

- 다시, **터미널 → 새 터미널**을 열면, 공구함 리로딩 완료!
- 공구함 만든 김에, 필요한 패키지를 미리 설치해둡시다!

```
pip install flask pymongo
```

▼ 12) 프로젝트 폴더 구조 만들기



▼ 13) app.py 준비하기

```
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)
```

▼ 14) index.html 준비하기

👉 HTML 코드는 미리 준비해두었어요.
templates 폴더 아래 index.html 에 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 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>
```

▼ 15) 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']);
                window.location.reload();
            }
        }
    })
}
```

▼ 4) 완성 확인하기

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

▼ 16) 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');
            }
        }
    })
}


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);
}
```

▼ 4) 완성 확인하기

새로고침했을 때 DB에 저장된 리뷰가 화면에 올바르게 나타나는지 확인합니다.

▼ 17) 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']);
                        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>
```

후반 3시간

[월수/화목반] : 체크인 & 출석체크

> 🔥 **튜터님은 체크인과 함께 출석 체크(링크)를 진행해주세요!**

▼ "15초 체크인"을 진행합니다.

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

[1.5시간] : 나홀로메모장 완성하기

> 🔥 **VS Code 파일→폴더열기를 클릭해서,**
> **sparta > project > alonememo 폴더를 열고 시작!**

▼ 18) 완성작부터 보기!

▼ 19) 새 프로젝트니까 - 가상환경 다시 잡기

- **터미널 → 새 터미널**을 열어서, 아래를 입력!

```
python -m venv myenv
```

- 다시, **터미널 → 새 터미널**을 열면, 공구함 세팅 완료!
- 공구함 만든 김에, 필요한 패키지를 미리 설치해둡시다!

```
pip install flask bs4 requests pymongo
```

▼ 20) 만들어야 할 API 두 개의 스펙을 생각해보기

▼ 21) 앗. meta태그 크롤링 하는 것을 안해봤다구요?

> 👉 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?

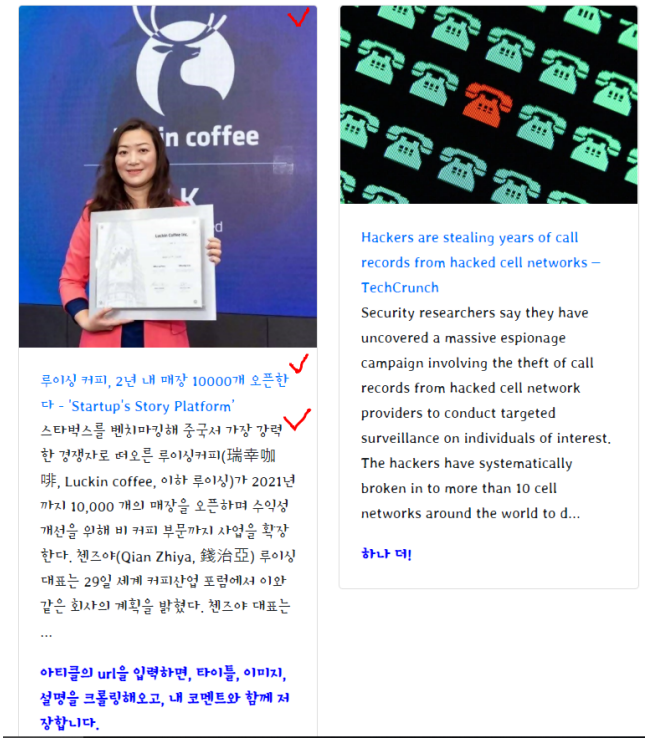
▼ 어떤 부분에 스크래핑이 필요한가요?

- 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?



▼ 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/7
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.

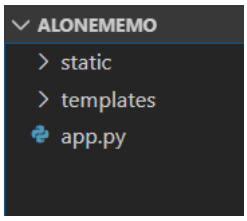
```
og_image = soup.select_one('meta[property="og:image"]')
og_title = soup.select_one('meta[property="og:title"]')
og_description = soup.select_one('meta[property="og:description"]')

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

- 가져온 meta tag의 content를 가져와봅시다.

> 👉 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)
```

▼ 22) index.html, app.py 준비하기



▼ index.html (2주차에 만들었던 나홀로메모장 html과 동일)

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

<head>

  <!-- Webpage Title -->
  <title>Hello, world!</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">

  <script>
    function openclose() {
      if ($('#posting-box').css('display') == 'block') {
        $('#posting-box').hide();
        $('#btn-posting-box').text('포스팅 박스 열기')
      } else {
        $('#posting-box').show();
        $('#btn-posting-box').text('포스팅 박스 닫기')
      }
    }

    $(document).ready(function () {
```

```
      $('#cards-box').html('');
      listing();
    });

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

    function make_card(comment, desc, image, title, url) {

    }

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

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

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

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

    .form-post {
      max-width: 500px;
      padding: 2rem;
      margin: 2rem auto;
      border-color: #e9ecef;
      border-radius: 0.3rem;
      border: solid;
      display: block;
    }

    #posting-box {
      display: none;
    }
  </style>

</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">
        <a id="btn-posting-box" onclick="openclose()" class="btn btn-primary btn-lg" href="#" role="button">포스팅박스 열기
      </p>
    </div>
    <div class="form-post" id="posting-box">
      <div>
        <div class="form-group">
          <label for="exampleFormControlInput1">아티클 URL</label>
          <input id="posting-url" class="form-control" placeholder="">
        </div>
        <div class="form-group">
          <label for="exampleFormControlTextarea1">간단 코멘트</label>
          <textarea id="posting-comment" class="form-control" rows="2"></textarea>
```

```
        </div>
        <button onclick="posting()" class="btn btn-primary">기사저장</button>
      </div>
    </div>
    <div class="card-columns" id="cards-box">
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          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 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          alt="Card image cap">
        <div class="card-body">
          <h5 class="card-title">여기 기사 제목이 들어가죠</h5>
          <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          alt="Card image cap">
        <div class="card-body">
          <h5 class="card-title">여기 기사 제목이 들어가죠</h5>
          <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          alt="Card image cap">
        <div class="card-body">
          <h5 class="card-title">여기 기사 제목이 들어가죠</h5>
          <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          alt="Card image cap">
        <div class="card-body">
          <h5 class="card-title">여기 기사 제목이 들어가죠</h5>
          <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          alt="Card image cap">
        <div class="card-body">
          <h5 class="card-title">여기 기사 제목이 들어가죠</h5>
          <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
    </div>
  </div>
</body>

</html>
```

▼ app.py

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

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.

## 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)
```

▼ 23) POST 연습: 메모하기

✅ **우리가 만들 API 두 가지**

1) 메모하기: 클라이언트에서 받은 url, comment를 이용해서 영화 정보를 찾고 저장하기
2) 메모 불러오기: 전체 메모 정보를 전달하기

▼ 1) 클라이언트와 서버 연결 확인하기

아래 클라이언트/서버 시작 코드는 쌍을 이루고 있습니다.

[서버 시작 코드]

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

[클라이언트 시작 코드]

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

👉 '기사저장'을 클릭했을 때, 'POST 연결되었습니다!' alert가 뜨면 연결에 성공한 것입니다!

▼ 2) 서버부터 만들기

API = 약속이라고 했습니다! 이 약속을 서버에서부터 시작해보겠습니다.

메모를 작성하기 위해 서버가 전달받아야 하는 정보는 다음과 같습니다.
1. url (url_receive) : meta tag를 가져올 url
2. comment (comment_receive) : 유저가 입력한 코멘트

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 saving():
    # 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
    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_image = og_image['content']
    url_title = og_title['content']
    url_description = og_description['content']

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

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

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

▼ 3) 클라이언트 만들기

API = 약속이라고 했습니다! 이제는 클라이언트를 만들어보겠습니다.

메모를 작성하기 위해 클라이언트가 전달해야 하는 정보는 다음과 같습니다.
1. url (url_give) : meta tag를 가져올 url
2. comment (comment_give) : 유저가 입력한 코멘트

따라서 클라이언트 로직은 다음 단계로 구성되어야 합니다.
1. 유저가 입력한 데이터를 #posting-url과 #posting-comment에서 가져오기
2. /memo에 POST 방식으로 메모 생성 요청하기
3. 성공 시 페이지 새로고침하기

```
function posting() {
  // 1. 유저가 입력한 데이터를 #posting-url과 #posting-comment에서 가져오기
  let url = $('#posting-url').val();
  let comment = $('#posting-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

- https://platum.kr/archives/129737 ← 이 url을 이용하여 기사저장을 눌렀을 때, '포스팅 성공!' alert이 뜨는지 확인합니다.

> 💡 메모 내용이 나오지 않아도 당황하지 마세요. 아직 GET 부분을 만들지 않았기 때문이지요. '포스팅 성공!' 메시지가 뜬다면 POST는 정상적으로 작성된 것이 맞습니다.

▼ 24) GET 연습: 메모 불러오기

> ✅ **우리가 만들 API 두 가지**
>
> 1) 메모하기: 클라이언트에서 받은 url, comment를 이용해서 영화 정보를 찾고 저장하기
> **2) 메모 불러오기: 전체 메모 정보를 전달하기**

▼ 1) 클라이언트와 서버 연결 확인하기

아래 클라이언트/서버 시작 코드는 쌍을 이루고 있습니다.

[서버 시작 코드]

```
@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!'})
```

[클라이언트 시작 코드]

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

function make_card(comment, desc, image, title, url) {

}
```

> 💡 새로고침했을 때, 'GET 연결되었습니다!' alert가 뜨면 연결에 성공한 것입니다!

▼ 2) 서버부터 만들기

👉 **API = 약속**이라고 했습니다! 이 약속을 서버에서부터 시작해보겠습니다.

모든 메모를 불러오기 위해 서버가 전달받아야 하는 정보는 없습니다.

따라서 서버 로직은 다음 단계로 구성되어야 합니다.
1. 모든 document 찾기 & _id 값은 출력에서 제외하기
2. articles라는 키 값으로 영화정보 내려주기

```
@app.route('/memo', methods=['GET'])
def listing():
    # 모든 document 찾기 & _id 값은 출력에서 제외하기
    result = list(db.articles.find({},{'_id':0}))
    # articles라는 키 값으로 영화정보 내려주기
    return jsonify({'result':'success', 'articles':result})
```

▼ 3) 클라이언트 만들기

👉 **API = 약속**이라고 했습니다! 이제는 클라이언트를 만들어보겠습니다.

모든 메모를 불러오기 위해 서버가 전달받아야 하는 정보는 없습니다.

따라서 클라이언트 로직은 다음 단계로 구성되어야 합니다.
1. /memo에 GET 방식으로 메모 정보 요청하기
2. 메모 정보를 articles로 받고, make_card 함수를 이용해서 카드 HTML 붙이기
(→ 2주차에 OpenAPI를 이용해 했던 Ajax 연습과 같습니다!)

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

function make_card(comment, desc, image, title, url) {
    let temp_html = `<div class="card">
                        <img class="card-img-top" src="${image}" alt="Card image cap">
                        <div class="card-body">
                        <a href="${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

새로고침했을 때, POST 단계에서 작성한 메모가 보이면 성공입니다.

▼ 25) 전체 완성 코드

▼ index.html

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

<head>

  <!-- Webpage Title -->
  <title>Hello, world!</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">

<script>
  function openclose() {
    if ($('#posting-box').css('display') == 'block') {
      $('#posting-box').hide();
      $('#btn-posting-box').text('포스팅 박스 열기')
    } else {
      $('#posting-box').show();
      $('#btn-posting-box').text('포스팅 박스 닫기')
    }
  }

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

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

  function make_card(comment, desc, image, title, url) {
        let temp_html = `<div class="card">
                            <img class="card-img-top" src="${image}" alt="Card image cap">
                            <div class="card-body">
                            <a href="${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);
    }

  function posting() {
    // 읽기
    let url = $('#posting-url').val();
    let comment = $('#posting-comment').val();

    // 우리가 크롬 콘솔창에서 썼던 그 코드!
    $.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!');
          window.location.reload();
        } else {
          alert('서버 오류!')
        }
      }
    })
  }
</script>

<!-- style -->
<style type="text/css">
```

```
    * {
      font-family: 'Stylish', sans-serif;
    }

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

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

    .form-post {
      max-width: 500px;
      padding: 2rem;
      margin: 2rem auto;
      border-color: #e9ecef;
      border-radius: 0.3rem;
      border: solid;
      display: block;
    }

    #posting-box {
      display: none;
    }
  </style>

</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">
        <a id="btn-posting-box" onclick="openclose()" class="btn btn-primary btn-lg" href="#" role="button">포스팅박스 열기
      </p>
    </div>
    <div class="form-post" id="posting-box">
      <div>
        <div class="form-group">
          <label for="exampleFormControlInput1">아티클 URL</label>
          <input id="posting-url" class="form-control" placeholder="">
        </div>
        <div class="form-group">
          <label for="exampleFormControlTextarea1">간단 코멘트</label>
          <textarea id="posting-comment" class="form-control" rows="2"></textarea>
        </div>
        <button onclick="posting()" class="btn btn-primary">기사저장</button>
      </div>
    </div>
    <div class="card-columns" id="cards-box">
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          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 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          alt="Card image cap">
        <div class="card-body">
          <h5 class="card-title">여기 기사 제목이 들어가죠</h5>
          <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          alt="Card image cap">
        <div class="card-body">
          <h5 class="card-title">여기 기사 제목이 들어가죠</h5>
          <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
      <div class="card">
        <img class="card-img-top"
```

```
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          alt="Card image cap">
        <div class="card-body">
          <h5 class="card-title">여기 기사 제목이 들어가죠</h5>
          <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          alt="Card image cap">
        <div class="card-body">
          <h5 class="card-title">여기 기사 제목이 들어가죠</h5>
          <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
      <div class="card">
        <img class="card-img-top"
          src="https://www.eurail.com/content/dam/images/eurail/Italy%20OCP%20Promo%20Block.adaptive.767.1535627244182.
          alt="Card image cap">
        <div class="card-body">
          <h5 class="card-title">여기 기사 제목이 들어가죠</h5>
          <p class="card-text">기사의 요약 내용이 들어갑니다. 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라만세 무궁화 삼천리 화려강산.
          <p class="card-text comment">여기에 코멘트가 들어갑니다.</p>
        </div>
      </div>
    </div>
  </div>
</body>

</html>
```

▼ app.py

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

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.

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

@app.route('/memo', methods=['GET'])
def listing():
    # 모든 document 찾기 & _id 값은 출력에서 제외하기
    result = list(db.articles.find({},{'_id':0}))
    # articles라는 키 값으로 영화정보 내려주기
    return jsonify({'result':'success', 'articles':result})

## API 역할을 하는 부분
@app.route('/memo', methods=['POST'])
def saving():
    # 클라이언트로부터 데이터를 받는 부분
    url_receive = request.form['url_give']
    comment_receive = request.form['comment_give']

    # meta tag를 스크래핑 하는 부분
    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_image = og_image['content']
    url_title = og_title['content']
    url_description = og_description['content']

    # mongoDB에 넣는 부분
    article = {'url': url_receive, 'comment': comment_receive, 'image': url_image,
               'title': url_title, 'desc': url_description}
```

```
    db.articles.insert_one(article)

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

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

[1.5시간]: 1~4주차 소화타임

▼ 오늘은 소화 타임이 30분 늘었네요?

- 소화하실게 많을거 같아 늘려두었습니다. (오늘 거의 무한리필이었죠?)
- 4주 동안 숨가쁘게 달려오느라 수고하셨습니다. 이제 A-Z를 한번 다 봤어요.👏
- 다음 주에는 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 <모두의책리뷰>를 처음부터 혼자 완성해보세요. 숙제와 아주 유사하답니다.

[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 슬랙에 공유하기

- 1주차에 완성한 쇼핑몰을 완성해주세요!

> 👉 페이지가 로딩되면, 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!

[숙제2] - 6~8주차 프로젝트 생김새 완성하기!

- "프로젝트 생김새"를 그려와주세요! (아이디어는 바뀌어도 무방!)

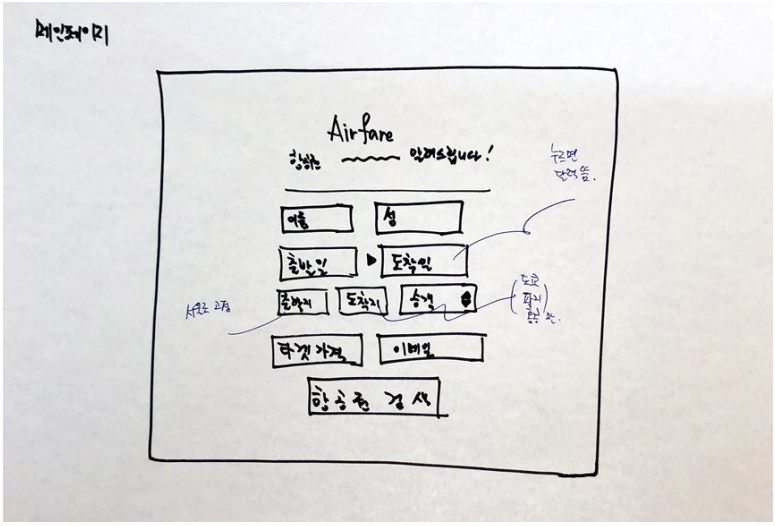
> 👉 생김새가 완성되어야 API를 구체적으로 생각해볼 수 있고,
> 그래야 **프로젝트 완성 가능성**이 무척 올라간답니다!

> 👉 기 수강생 작품 참고: (링크)

> 👉 꼭 같이 하고 싶은 분이 있나요? → 개별 프로젝트를 권장합니다만, 같이 하고 싶은 분이 있으면 튜터님과 상의해서 팀으로 진행해도 무방합니다!

- 대략 이런 느낌!

> 👉 챗봇 등 프론트페이지가 없는 경우,
> 발표를 위해 간단한 로직과 기획을 글로 작성해와주세요



[설치] - 다음 시간을 위해 미리 설치해와야 할 것들

- 없음!