스파르타코딩클럽 7기 5주차 - 학습자료

[수업 목표]

1. Flask 프레임워크를 활용해서 API를 만들 수 있다.
2. '나홀로메모장'을 완성한다.
3. '마이 페이보릿 무비스타'를 완성한다.
4. 내 프로젝트의 와이어프레임을 완성하고, 코딩을 시작한다.

📢: 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.

[1.5시간] : 나홀로메모장 완성하기

▼ 1) 만들어야 할 API의 스펙을 생각해보기

- 이론 되짚어보기

  https://s3-us-west-2.amazonaws.com/secure.notion-static.com/36b28fde-9a6b-4b05-8e8b-06bbee08844f/_5__.pdf

- 포스팅API: 카드 생성하기
  - 요청 타입 → POST
  - 요청할 URL → /memo
  - 요청할 때 함께 줄 데이터 → URL, 코멘트
  - 서버 내부에서 수행 할 기능
    1. url의 meta태그 정보를 바탕으로 제목, 설명, 이미지URL 크롤링
    2. (제목, 설명, URL, 이미지URL, 코멘트) 정보를 모두 DB에 저장
  - 요청 결과 값 → 잘 됐는지 아닌지만 알려주면 됨
- 리스팅API: 카드 조회하기(Read)
  - 요청 타입 → GET
  - 요청할 URL → /memo
  - 요청할 때 함께 줄 데이터 → 없음
  - 서버 내부에서 수행 할 기능
    1. DB에 저장돼있는 모든 (제목, 설명, URL, 이미지URL, 코멘트) 정보를 가져오기

- 요청 결과 값 → 포스트들의 정보(제목, 설명, URL, 이미지URL, 코멘트)를 주기

▼ 2) 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?

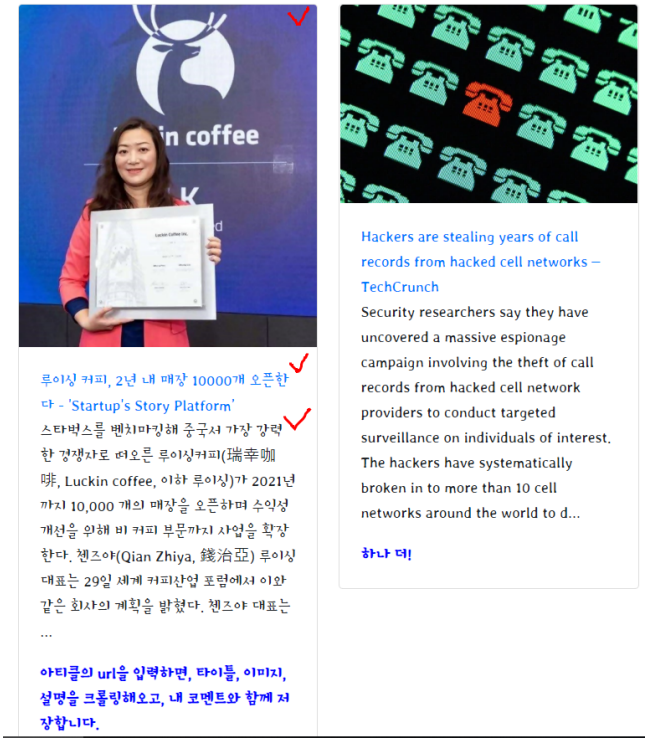
▼ 어떤 부분에 스크래핑이 필요할까요?

- 우리는 기사URL만 입력했는데, 자동으로 불러와지는 부분들이 있습니다.

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

- 바로 '기사 제목', '썸네일 이미지', '내용' 입니다.

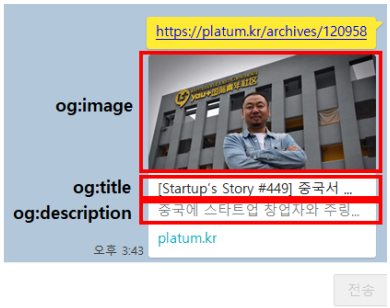
> 👉 이 부분은, 'meta'태그를 크롤링 함으로써 공통적으로 얻을 수 있습니다.
>
> meta태그가 무엇이고, 어떻게 스크래핑 하는지, 함께 살펴볼까요?



▼ 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.

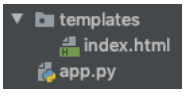
```
og_image = soup.select_one('meta[property="og:image"]')
og_title = soup.select_one('meta[property="og:title"]')
og_description = soup.select_one('meta[property="og:description"]')

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)
```

▼ 3) index.html, app.py 준비하기



▼ 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) {
        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)
```

▼ 4) 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 = 약속**이라고 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 = 약속**이라고 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.

▼ 5) 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

```
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
       }
    }
  })
}
```

▼ 4) 완성 확인하기

새로고침했을 때, POST 단계에서 작성한 메모가 보이면 성공입니다.

▼ 6) 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 + '" 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] : 마이 페이보릿 무비스타 만들어보기

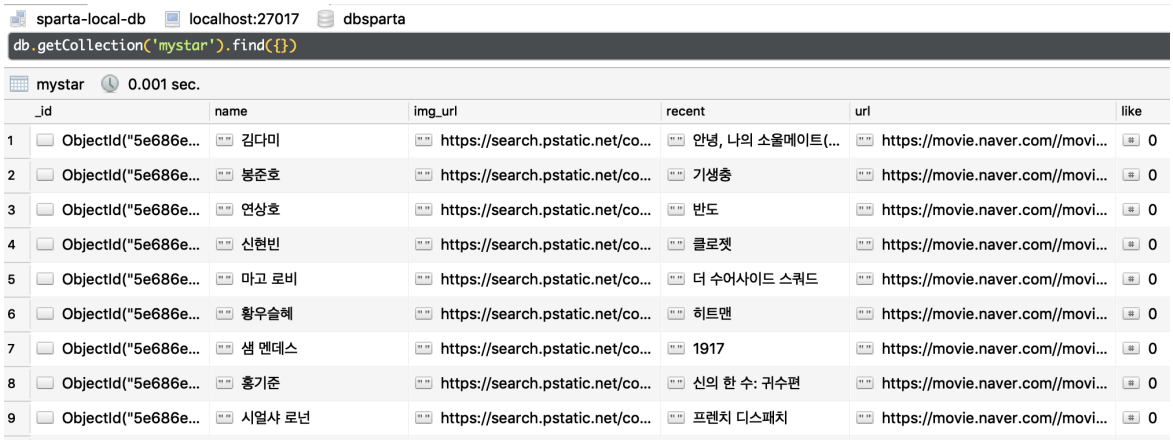
▼ 7) 완성작 보기: 함께 만들어봐요!

> 👉 아래 링크를 클릭해서 완성된 모습을 확인해보세요!
> http://spartacodingclub.shop/moviestar

▼ 8) 복붙으로 사전 세팅! - 크롤링으로 DB 쌓기

> 👉 **API를 구상하고 직접 만드는 것에만 집중**하실 수 있게, 데이터를 크롤링해 저장하는 과정은 저희가 미리 작성해두었습니다.

mystar 콜렉션에 세팅 완료된 모습은 다음과 같습니다.


sparta-local-db localhost:27017 dbsparta

```
db.getCollection('mystar').find({})
```

mystar 0.001 sec.

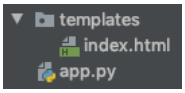
	_id	name	img_url	recent	url	like
1	ObjectId("5e686e...	김다미	https://search.pstatic.net/co...	안녕, 나의 소울메이트(...	https://movie.naver.com//movi...	0
2	ObjectId("5e686e...	봉준호	https://search.pstatic.net/co...	기생충	https://movie.naver.com//movi...	0
3	ObjectId("5e686e...	연상호	https://search.pstatic.net/co...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

아래 코드를 복사-붙여넣기 후 실행하시면 mystar 콜렉션에 영화인들의 정보가 저장됩니다.

```
import requests
from bs4 import BeautifulSoup

from pymongo import MongoClient

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

# DB에 저장할 영화인들의 출처 url을 가져옵니다.
def get_urls():
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3
    data = requests.get('https://movie.naver.com/movie/sdb/rank/rpeople.nhn', headers=headers)

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

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

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

    return urls

# 출처 url로부터 영화인들의 사진, 이름, 최근작 정보를 가져오고 mystar 콜렉션에 저장합니다.
def insert_star(url):
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86
    data = requests.get(url, headers=headers)

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

    name = soup.select_one('#content > div.article > div.mv_info_area > div.mv_info.character > h3 > a').text
```

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

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

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

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


### 실행하기
insert_all()
```

▼ 9) index.html, app.py 준비하기



위와 같은 모습으로 app.py 파일과 ndex.html 파일을 만들고, 다음 내용을 복사-붙여넣기 합니다.

▼ index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>프론트-백엔드 연결 마지막 예제!</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bulma@0.8.0/css/bulma.min.css"
    />
    <script
      defer
      src="https://use.fontawesome.com/releases/v5.3.1/js/all.js"
    ></script>
    <style>
      .make-center {
        text-align: center;
      }
      .star-list {
        width: 500px;
        margin: 20px auto 0 auto;
      }
      .star-name {
          display: inline-block;
      }
      .star-name:hover {
          text-decoration: underline;
      }
      .card {
          margin-bottom: 15px;
      }
    </style>
    <script>
      $(document).ready(function() {
        // index.html 로드가 완료되면 자동으로 show_star() 함수를 호출합니다.
        show_star();
      });

      function show_star(){
        $.ajax({
            type: 'GET',
            url:  '/api/list',
            data: {},
            success: function (response) {
```

```
              if (response['result'] == 'success') {
                let msg = response['msg'];
                alert(msg);
              }
            }
          });
      }

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

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

    </script>
  </head>
  <body>
    <section class="hero is-warning">
      <div class="hero-body">
        <div class="container make-center">
          <h1 class="title">
            마이 페이보릿 무비스타😉
          </h1>
          <h2 class="subtitle">
            순위를 매겨봅시다
          </h2>
        </div>
      </div>
    </section>
    <div class="star-list" id="star-box">
      <div class="card">
        <div class="card-content">
          <div class="media">
            <div class="media-left">
              <figure class="image is-48x48">
                <img
                  src="https://search.pstatic.net/common/?src=https%3A%2F%2Fssl.pstatic.net%2Fsstatic%2Fpeople%2Fportra
                  alt="Placeholder image"
                />
              </figure>
            </div>
            <div class="media-content">
              <a href="#" target="_blank" class="star-name title is-4">김다미 (좋아요: 3)</a>
              <p class="subtitle is-6">안녕, 나의 소울메이트(가제)</p>
            </div>
          </div>
        </div>
        <footer class="card-footer">
          <a href="#" onclick="like_star('김다미')" class="card-footer-item has-text-info">
            위로!
            <span class="icon">
              <i class="fas fa-thumbs-up"></i>
            </span>
          </a>
          <a href="#" onclick="delete_star('김다미')" class="card-footer-item has-text-danger">
            삭제
            <span class="icon">
              <i class="fas fa-ban"></i>
            </span>
          </a>
        </footer>
      </div>
    </div>
  </body>
</html>
```

▼ app.py

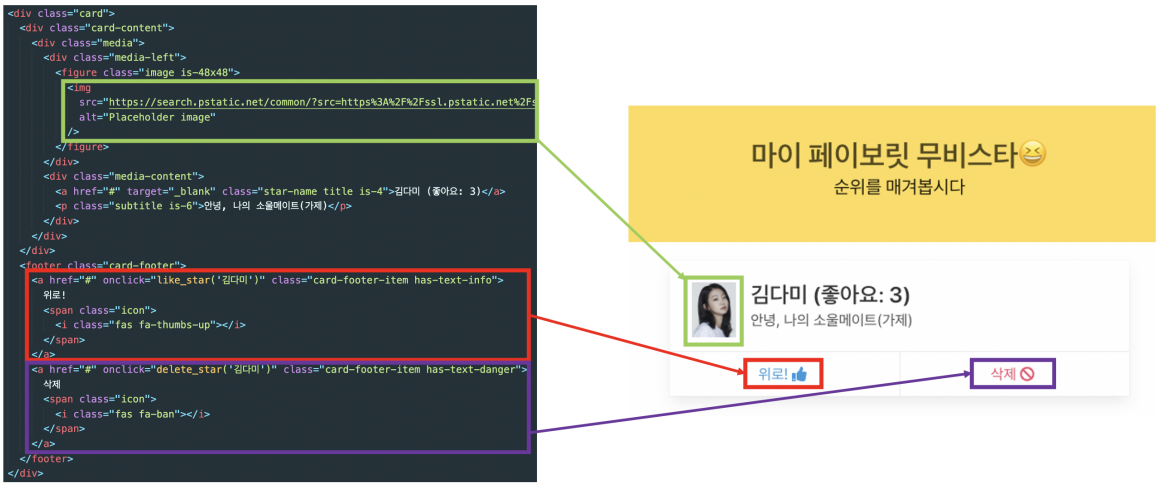
```
from pymongo import MongoClient

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

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


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


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


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

▼ 10) 영화인 카드의 HTML 살펴보기

VS Code를 통해 index.html만을 브라우저에서 실행시켜보세요.

다음을 중점적으로 살펴본 다음에 **영화인 카드를 어떻게 만들 수 있을지** 생각해봅시다.

* 이름이 들어가는 부분
* 이미지 src 속성이 들어가는 부분
* 좋아요 개수와 최근작 내용이 들어가는 부분
* 좋아요 기능을 담당하는 like_star 함수의 onclick 호출 부분
* 삭제 기능을 담당하는 delete_star 함수의 onclick 호출 부분



서버가 넘겨준 정보를 이용하여 영화인 정보 카드를 만들 때는 다음 코드를 참고합니다.

```
<!-- 다음 코드가 하나의 카드를 이루는 div 입니다. -->
<div class="card">
  <div class="card-content">
    <div class="media">
      <div class="media-left">
        <figure class="image is-48x48">
          <img
            src="https://search.pstatic.net/common/?src=https%3A%2F%2Fssl.pstatic.net%2Fsstatic%2Fpeople%2Fportrait%2F20180
            alt="Placeholder image"
          />
        </figure>
      </div>
      <div class="media-content">
        <a href="#" target="_blank" class="star-name title is-4">김다미 (좋아요: 3)</a>
        <p class="subtitle is-6">안녕, 나의 소울메이트(가제)</p>
      </div>
    </div>
  </div>
  <footer class="card-footer">
    <a href="#" onclick="like_star('김다미')" class="card-footer-item has-text-info">
      위로!
      <span class="icon">
        <i class="fas fa-thumbs-up"></i>
      </span>
    </a>
    <a href="#" onclick="delete_star('김다미')" class="card-footer-item has-text-danger">
      삭제
      <span class="icon">
        <i class="fas fa-ban"></i>
      </span>
    </a>
  </footer>
</div>
```

▼ 11) GET 연습: 영화인 조회하기

우리가 만들 API 세 가지

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

▼ 1) 클라이언트와 서버 연결 확인하기

아래 클라이언트/서버 시작 코드는 쌍을 이루고 있습니다.

[서버 시작 코드]

```
@app.route('/api/list', methods=['GET'])
def stars_list():
```

```
    # 1. mystar 목록 전체를 검색합니다. ID는 제외하고 like 가 많은 순으로 정렬합니다.
    # 참고) find({},{'_id':False}), sort()를 활용하면 굿!
    # 2. 성공하면 success 메시지와 함께 stars_list 목록을 클라이언트에 전달합니다.
    return jsonify({'result': 'success','msg':'list 연결되었습니다!'})
```

[클라이언트 시작 코드]

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

👉 새로고침했을 때, 'list 연결되었습니다!' 내용의 alert가 뜨면 연결에 성공한 것입니다!

▼ 2) 서버부터 만들기

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

영화인 정보 전체를 조회하기 위해 서버가 받을 정보는 없습니다.

따라서 서버 로직은 다음 단계로 구성되어야 합니다.
1. mystar 목록 전체를 검색합니다. ID는 제외하고 like 가 많은 순으로 정렬합니다.
2. 성공하면 success 메시지와 함께 stars_list 목록을 클라이언트에 전달합니다.

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

▼ 3) 클라이언트 만들기

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

영화인 정보 전체를 조회하기 위해 클라이언트가 전달할 정보는 없습니다.

따라서 클라이언트 로직은 다음 단계로 구성되어야 합니다.
1. #star_box의 내부 html 태그를 모두 삭제합니다.
2. 서버에 1) GET 방식으로, 2) /api/list 라는 주소로 stars_list를 요청합니다.
3. 서버가 돌려준 stars_list를 stars라는 변수에 저장합니다.
4. for 문을 활용하여 stars 배열의 요소를 차례대로 조회합니다.
5. stars[i] 요소의 name, url, img_url, recent, like 키 값을 활용하여 값을 조회합니다.
6. 영화인 카드를 만듭니다.
7. #star-box에 temp_html을 붙입니다.

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

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

              // 6. 영화인 카드를 만듭니다.
              let temp_html = '<div class="card">\
                              <div class="card-content">\
                              <div class="media">\
                                  <div class="media-left">\
                                  <figure class="image is-48x48">\
                                      <img src="'+img_url+'" alt="Placeholder image"/>\
                                  </figure>\
                                  </div>\
                                  <div class="media-content">\
                                  <a href="'+url+'" target="_blank" class="star-name title is-4">'+name+' (좋아요: '+like
                                  <p class="subtitle is-6">'+recent+'</p>\
                                  </div>\
                              </div>\
                              </div>\
                              <footer class="card-footer">\
                              <a href="#" onclick="like_star(\''+name+'\'); return false;" class="card-footer-item has-
                                  위로!\
                                  <span class="icon">\
                                  <i class="fas fa-thumbs-up"></i>\
                                  </span>\
                              </a>\
                              <a href="#" onclick="delete_star(\''+name+'\'); return false;" class="card-footer-item ha
                                  삭제\
                                  <span class="icon">\
                                  <i class="fas fa-ban"></i>\
                                  </span>\
                              </a>\
                              </footer>\
                          </div>'
              // 7. #star-box에 temp_html을 붙입니다.
              $('#star-box').append(temp_html)
          }
      }
  });
}
```

▼ 4) 완성 확인하기

index.html을 새로고침 했을 때 영화인 정보가 조회되는지 확인합니다.

후반 3시간

[1시간] : 마이 페이보릿 무비스타 완성하기

▼ 12) POST 연습: 좋아요 +1

✅ **우리가 만들 API 세 가지**

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

▼ 1) 클라이언트와 서버 연결 확인하기

아래 클라이언트/서버 시작 코드는 쌍을 이루고 있습니다.

[서버 시작 코드]

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

[클라이언트 시작 코드]

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

👉 '위로' 버튼을 눌렀을 때, 'like 연결되었습니다!' 내용의 alert가 뜨면 연결에 성공한 것입니다!

▼ 2) 서버부터 만들기

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

좋아요 수를 증가시키기 위해 서버가 필요한 정보는 다음과 같습니다.
1. 영화인의 이름 (name_give라고 클라이언트가 전달)

따라서 서버 로직은 다음 단계로 구성되어야 합니다.
1. 클라이언트가 전달한 name_give를 name_receive 변수에 넣습니다.
2. mystar 목록에서 find_one으로 name이 name_receive와 일치하는 star를 찾습니다.
3. star의 like 에 1을 더해준 new_like 변수를 만듭니다.
4. mystar 목록에서 name이 name_receive인 문서의 like 를 new_like로 변경합니다.

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

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

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

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

▼ 3) 클라이언트 만들기

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

좋아요 수를 증가시키기 위해 클라이언트가 전달할 정보는 다음과 같습니다.
1. 영화인의 이름 (name_give라고 클라이언트가 전달)

따라서 클라이언트 로직은 다음 단계로 구성되어야 합니다.
1. 서버에 1) POST 방식으로, 2) /api/like 라는 url에, 3) name_give라는 이름으로 name을 전달합니다. (참고)
POST 방식이므로 data: {'name_give': name} 과 같은 양식이 되어야합니다!
2. '좋아요 완료!' 얼럿을 띄웁니다.
3. 변경된 정보를 반영하기 위해 새로고침합니다.

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

▼ 4) 완성 확인하기

'위로' 버튼을 눌렀을 때 좋아요가 증가하고 영화인 카드의 순위가 변경되는지 확인합니다.

▼ 13) POST 연습: 영화인 삭제하기

✅ **우리가 만들 API 세 가지**

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

▼ 1) 클라이언트와 서버 연결 확인하기

아래 클라이언트/서버 시작 코드는 쌍을 이루고 있습니다.

[서버 시작 코드]

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

[클라이언트 시작 코드]

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

```
            let msg = response['msg'];
            alert(msg);
        }
    }
  });
}
```

👉 '삭제' 버튼을 눌렀을 때, 'delete 연결되었습니다!'라는 alert가 뜨면 연결에 성공한 것입니다!

▼ 2) 서버부터 만들기

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

영화인 카드를 삭제하기 위해 필요한 정보는 다음과 같습니다.
1. 영화인의 이름 (name_give라고 클라이언트가 전달)

따라서 서버 로직은 다음 단계로 구성되어야 합니다.
1. 클라이언트가 전달한 name_give를 name_receive 변수에 넣습니다.
2. mystar 에서 delete_one으로 name이 name_receive와 일치하는 star를 제거합니다.
3. 성공하면 success 메시지를 반환합니다.

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

▼ 3) 클라이언트 만들기

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

영화인 카드를 삭제하기 위해 필요한 정보는 다음과 같습니다.
1. 영화인의 이름 (name_give라고 클라이언트가 전달)

따라서 클라이언트 로직은 다음 단계로 구성되어야 합니다.
1. 서버에 1) POST 방식으로, 2) /api/delete 라는 url에, 3) name_give라는 이름으로 name을 전달합니다. (참고) POST 방식이므로 data: {'name_give': name} 과 같은 양식이 되어야합니다!
2. '삭제 완료! 안녕!' 얼럿을 띄웁니다.
3. 변경된 정보를 반영하기 위해 새로고침합니다.

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

▼ 4) 완성 확인하기

삭제 버튼을 눌렀을 때 영화인 카드가 삭제되는지 확인합니다.

▼ 14) 전체 완성 코드

▼ index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>프론트-백엔드 연결 마지막 예제!</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bulma@0.8.0/css/bulma.min.css"
    />
    <script
      defer
      src="https://use.fontawesome.com/releases/v5.3.1/js/all.js"
    ></script>
    <style>
      .make-center {
        text-align: center;
      }
      .star-list {
        width: 500px;
        margin: 20px auto 0 auto;
      }
      .star-name {
          display: inline-block;
      }
      .star-name:hover {
          text-decoration: underline;
      }
      .card {
          margin-bottom: 15px;
      }
    </style>
    <script>
      $(document).ready(function() {
        // index.html 로드가 완료되면 자동으로 show_star() 함수를 호출합니다.
        show_star();
      });

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

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

                    // 6. 영화인 카드를 만듭니다. (이미 완성되어 있습니다.)
                    let temp_html = '<div class="card">\
                                    <div class="card-content">\
                                    <div class="media">\
                                        <div class="media-left">\
                                        <figure class="image is-48x48">\
                                            <img src="'+img_url+'" alt="Placeholder image"/>\
                                        </figure>\
                                        </div>\
                                        <div class="media-content">\
                                        <a href="'+url+'" target="_blank" class="star-name title is-4">'+name+' (좋아요:
                                        <p class="subtitle is-6">'+recent+'</p>\
                                        </div>\
                                    </div>\
                                    </div>\
                                    <footer class="card-footer">\
```

```
                                <a href="#" onclick="like_star(\''+name+'\'); return false;" class="card-footer-ite
                                    위로!\
                                    <span class="icon">\
                                    <i class="fas fa-thumbs-up"></i>\
                                    </span>\
                                </a>\
                                <a href="#" onclick="delete_star(\''+name+'\'); return false;" class="card-footer-i
                                    삭제\
                                    <span class="icon">\
                                    <i class="fas fa-ban"></i>\
                                    </span>\
                                </a>\
                                </footer>\
                            </div>'
                    // 7. #star-box에 temp_html을 붙입니다.
                    $('#star-box').append(temp_html)
                }
            }
        });
    }

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

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

  </script>
</head>
<body>
  <section class="hero is-warning">
    <div class="hero-body">
      <div class="container make-center">
        <h1 class="title">
          마이 페이보릿 무비스타😉
        </h1>
        <h2 class="subtitle">
          순위를 매겨봅시다
        </h2>
      </div>
    </div>
  </section>
  <div class="star-list" id="star-box">
    <div class="card">
      <div class="card-content">
        <div class="media">
          <div class="media-left">
            <figure class="image is-48x48">
              <img
                src="https://search.pstatic.net/common/?src=https%3A%2F%2Fssl.pstatic.net%2Fsstatic%2Fpeople%2Fportra
                alt="Placeholder image"
              />
            </figure>
          </div>
          <div class="media-content">
            <a href="#" target="_blank" class="star-name title is-4">김다미 (좋아요: 3)</a>
            <p class="subtitle is-6">안녕, 나의 소울메이트(가제)</p>
```

```
            </div>
          </div>
        </div>
        <footer class="card-footer">
          <a href="#" onclick="like_star('김다미')" class="card-footer-item has-text-info">
            위로!
            <span class="icon">
              <i class="fas fa-thumbs-up"></i>
            </span>
          </a>
          <a href="#" onclick="delete_star('김다미')" class="card-footer-item has-text-danger">
            삭제
            <span class="icon">
              <i class="fas fa-ban"></i>
            </span>
          </a>
        </footer>
      </div>
    </div>
  </body>
</html>
```

▼ app.py

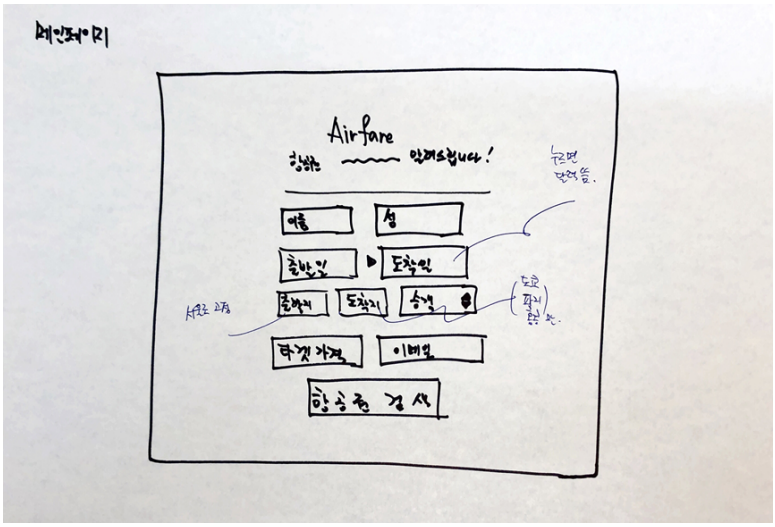
```
from pymongo import MongoClient

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

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


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

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

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

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


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


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

[2시간]: 프로젝트 개요 & 목표 공유

▼ 15) 프로젝트의 "생김새"를 그려봅니다.

- '종이'에 그리는 것을 추천합니다만, 파워포인트도 무방합니다.
- 대략 이런 느낌!

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



▼ 16) 첫 번째 개발일지를 작성해보기

> 👉 일단 말하고 나면, 이루어진답니다! :-)

- 공유 가능한 블로그(Medium을 추천!)에 사진+글로 정리하고, 톡방에 공유합니다!
  - 가입: https://medium.com → get started 클릭
  - 3기 수강생 분의 개발일지 예시: (링크1), (링크2)
  - 6~8주차에도 계속해서 개발일지를 작성 할 예정입니다.
  - 본인이 익숙한, 톡방에 공개 가능한 블로그 등이 있으면 사용하셔도 무방합니다.

[끝]

▼ "15초 체크아웃"을 진행합니다.

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

- 아래 숙제 & 설치해와야 할 것들을 설명합니다.

숙제 & 설치

[숙제 1] - Github Repository를 하나 새로 만들고, 작업을 시작합니다!

- public repository로 만들어 튜터도 볼 수 있게 해주세요.
- 그럼, 앞으로 3주동안, 파이팅!

[숙제 2] - 첫 개발일지를 작성해오기!

- Medium에 아래 내용이 포함된 개발일지를 작성해옵니다.

👉 개발일지는, 본인을 위해 작성하는 것입니다! 꼭 일주일에 한 편씩만 작성하지 않아도 되며, 사실은 개발을 할 때마다 매일 작성하는 것이 가장 좋은 방법입니다.

- 6기 수강생 분의 개발일지 예시: (스파르타코딩클럽 개발일지 - 1) 프로젝트 설명)

참고) TIL (today I learned)를 검색하면, 많은 분들의 개발일지를 볼 수 있습니다.

1. 한 주 동안의 회고
2. 한 주 동안의 배운 것들
3. 이번주의 목표

[설치] - 다음 시간을 위해 미리 설치해와야 할 것들

1. AWS 가입하기 (승인까지 최대 24시간이 걸리니, 미리 해주세요!)
   - 가입: https://console.aws.amazon.com/console/home

   ▼ 결제 관련 알아두기!

   - AWS는 개인에게 클라우드 환경의 가상서버를 제공합니다. 기본 사양의 서버(EC2)를 1년 동안 무료로 사용할 수 있으며, 이후 월 1만 원정도의 금액이 결제될 수 있습니다.
   - 가입 시 결제된 금액은 다시 반환됩니다. (일종의 결제 테스트 목적)

   ▼ 가입 후 아래와 같은 화면에 접속 하면 성공!

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