

2020-2  
웹 시스템 설계  
학습과제

*Week 11*

Express with Mongoose(2) & Ajax

**Composed by:**  
**WISE Research Lab Ajou University**



AJOU UNIVERSITY

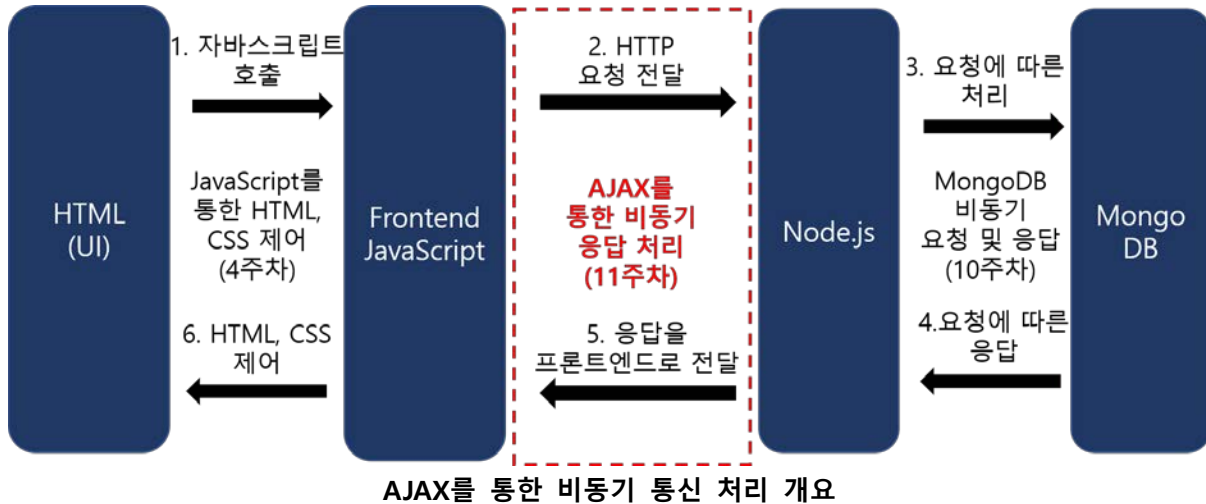
WISE

## 학습목표

1. Express Framework 와 MongoDB 를 연동하여 웹 어플리케이션을 구축할 수 있다.
2. Ajax 를 활용하여 프론트 엔드 측에서 백엔드 측으로 비동기 요청을 수행할 수 있다.

# AJAX(Asynchronous JavaScript And XML)

## 1. <sup>1</sup>AJAX 란?



- AJAX란 비동기 자바스크립트와 XML을 말한다.
- JSON, XML, HTML 및 일반 텍스트 형식 등을 포함한 다양한 포맷을 주고 받을 수 있다.
- AJAX의 주요 두가지 특징은 다음과 같다.
  - ✓ 페이지 새로고침 없이 서버에 요청
  - ✓ 서버로부터 데이터를 받고 작업을 수행
- XMLHttpRequest API의 프로퍼티 정의

프로퍼티 명	설명
<b>XMLHttpRequest.readyState</b>	<p>XMLHttpRequest 요청의 상태를 반환한다.</p> <p>readyState는 다음과 같은 5가지 상태를 정의한다.</p> <ol style="list-style-type: none"><li>0. Unsent 상태, open()이 호출되지 않은 상태</li><li>1. OPENED 상태, open()이 호출된 상태</li><li>2. HEADER_RECEIVED 상태, send()가 호출되어 header와 status가 available한 상태</li><li>3. LOADING : responseText가 부분데이터를 가지고 있는 상태</li><li>4. DONE : 연산이 종료된 상태</li></ol>

<sup>1</sup> [https://developer.mozilla.org/ko/docs/Web/Guide/AJAX/Getting\\_Started](https://developer.mozilla.org/ko/docs/Web/Guide/AJAX/Getting_Started) , <https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>

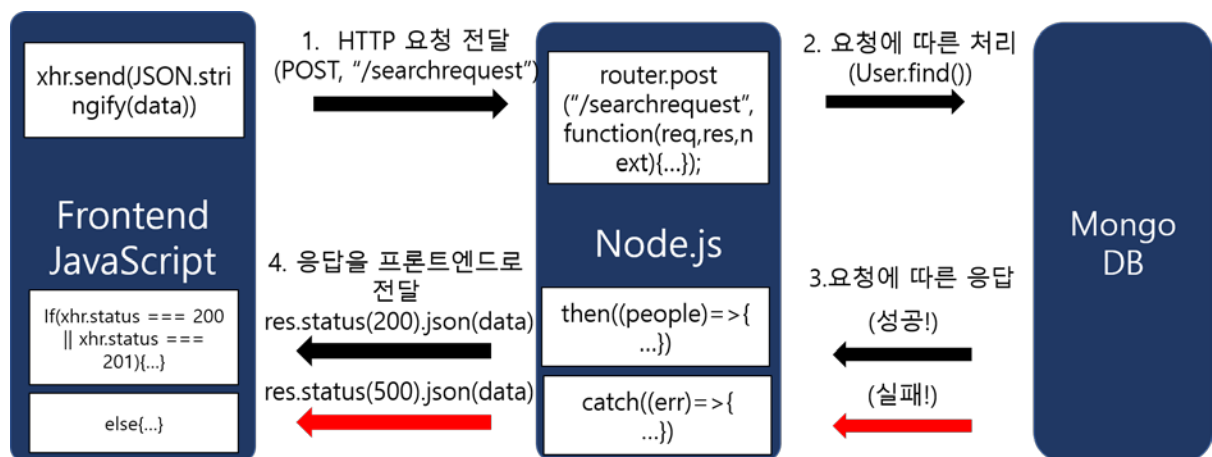
<b>XMLHttpRequest.status</b>	<p>HTTP status 코드를 반환한다.</p> <p>HTTP 의 status 코드는 아래와 같이 정의된다.</p> <ol style="list-style-type: none"> <li>1. Informational responses (100–199)</li> <li>2. Successful responses (200–299)</li> <li>3. Redirects (300–399)</li> <li>4. Client errors (400–499)</li> <li>5. Server errors (500–599)</li> </ol>
<b>XMLHttpRequest.response</b>	<p>요청에 대한 응답을 XMLHttpRequest.responseType 의 값에 따라 ArrayBuffer, Blob, Document JavaScript 객체 또는 DOMString 을 반환한다.</p>
<b>XMLHttpRequest.responseText</b>	<p>요청에 대한 응답을 텍스트로 갖는 DOMString(UTF-16 문자열)을 반환한다. 요청이 성공하지 못했거나, 전송되지 않았으면, Null 값을 반환한다.</p>

- XMLHttpRequest API 의 메소드 정의

메소드명	매개변수	설명
<b>XMLHttpRequest</b>	없음	XMLHttpRequest()의 생성자
<b>onload</b>		백엔드로부터 응답을 받을 때 호출하게 되는 함수를 정의한다. XMLHttpRequest 객체의 status 속성 값을 검사해 서버의 응답이 정상인지 확인한다.
<b>onreadystatechange</b>		요청 상태(readyState)가 변경될 때 마다 호출되는 EventHandler. 요청상태와 요청이 완료된 경우(DONE, 4 번상태)인 경우, status 속성 값을 받아 응답의 상태코드를 확인할 수 있다. 즉, onload 의 기능 또한 수행할 수 있다.
<b>open()</b>	HTTP 메소드 명, URL	요청을 초기화한다.
<b>setRequestHeader</b>	정의할 헤더 이름, 헤더의 값	HTTP 요청 헤더의 값을 설정 open()후 send() 전에 setRequestHeader() 로 헤더 정보를 정의해야함.
<b>send</b>	요청과 함께 전달할 추가정보	open() 및 setRequestHeader()를 통해 정의된 요청을 추가정보와 함께 전달한다.

## 2. 활용 예제

- 이번 학습과제에서는 가장 기본적인 구현방법인 XMLHttpRequest API 를 통해 구현한 코드를 통해 예제 및 실습을 진행한다.
- XMLHttpRequest API 를 활용한 AJAX 처리 예제(기초 사용 패턴)
  - ✓ Frontend 와 Backend 간의 비동기 통신 예시의 오버뷰



✓ Frontend 코드

```
function ajaxFindByName(){
    //XMLHttpRequest 객체 생성
    var xhr = new XMLHttpRequest();
    var name = document.getElementById("name");

    //백엔드 측으로 전달할 입력데이터 생성
    var data = {
        'name' : name.value
    }

    //요청의 상태가 DONE 일 경우 프론트엔드 측에서 요청을 처리하는 방식을 정의
    xhr.onload = function(){
        alert("검색");
        if(xhr.status === 200 || xhr.status === 201){
            //string으로 받은 json 타입의 데이터를 JSON 객체화
            let response = JSON.parse(xhr.response);
            // response JSON 타입 객체 내 people 데이터에 접근
            let people = response.people;
            console.log(response)
            //이후, 응답 값을 통해 HTML 문서 내 정의된 정보를 변경가능

        }else {
            alert("검색 실패");
        }
    }
}
```

```

//생성된 XMLHttpRequest 객체의 초기화
xhr.open("POST", "/searchrequest");
//요청의 헤더 설정(json 타입을 통신에 활용할 것이므로, 컨텐츠 타입을 json 으
로 선언)
xhr.setRequestHeader("Content-Type", 'application/json');
//서버측으로 전달할 json 타입의 데이터를 string 화 한 데이터와 함께 요청을 전
달.
xhr.send(JSON.stringify(data));
}

```

✓ Backend 코드

```

router.post('/searchrequest', function(req, res, next){
  User.find(req.body).then((people)=>{
    let data = {
      success : true,
      people : people
    }
    res.status(200).json(data);
  }).catch((err)=>{
    let data = {
      success : false
    }
    res.status(500).json(data);
  });
});
});

```

# 학습과제

---

## 1. 과제 개요

- NoSQL 기반인 MongoDB 의 collection 와 document 개념을 익힌다.
- Mongoose 를 사용하여 Node.js backend 에서 MongoDB 의 데이터를 조작하는 방법에 대해 학습한다.
- MongoDB 와 연동된 Node.js 백엔드 및 Pug(Jade) 템플릿 기반 프론트 엔드를 구성하여 영화 정보 관리 웹 어플리케이션을 구성한다.
- 제출된 학습보고서 3~4 를 통해 AJAX 의 구현방법 중 하나인 XMLHttpRequest API 의 기본적인 활용법에 대해 익힌다.

## 2. 학습과제 파일 구성 및 과제 제출 양식

- 학습과제는 express-generator 를 기반으로 한 스켈레톤을 기반으로 구성되며, 추가되는 파일 구성은 아래와 같다.

상위 폴더 명	추가 파일 이름	설명
models	<a href="#">movie.js</a>	영화 정보를 저장하는 스키마 및 모델을 정의
public/images	poster_[1,2,3,4,5].jpg	포스터 이미지 파일
public/javascripts	<a href="#">main.js</a>	프론트엔드 단의 자바스크립트 파일
public/stylesheets	<a href="#">style.css</a>	프론트엔드 단의 CSS 파일
routes	<a href="#">index.js</a>	기본적인 웹페이지 요청을 처리하는 모듈
	<a href="#">movie.js</a>	Movie 정보에 대한 CRUD 요청을 처리하는 모듈
views	<a href="#">admin.pug</a>	관리자 웹페이지를 정의하는 템플릿
	<a href="#">editmovie.pug</a>	관리자 웹페이지에서 영화 별 편집 요청 시 이동되는 영화 편집 페이지
	<a href="#">index.pug</a>	영화 정보 웹 어플리케이션의 시작 웹 페이지

	<a href="#">layout.pug</a>	Navbar 및 CSS, JavaScript 로드를 위한 레이아웃 템플릿
	<a href="#">newmovie.pug</a>	신규 영화 등록을 웹페이지

- 이외 express-generator 를 통해 생성되는 파일들에 대해 수정이 필요한 경우, 수정을 수행한다.

### 3. 제출방법 및 주의사항

- 압축 파일의 이름을 "Lab11\_자신의학번.zip"으로 지정하여 폴더(프로젝트)를 압축한다.
- 설치된 node package 들을 제외한 프로젝트 폴더를 압축해서 제출
  - ✓ 즉, node\_module 폴더는 압축파일에 포함시키지 않는다.
- 압축된 프로젝트 폴더와 함께, 제출한 학습보고서 1~4를 작성하여 함께 제출한다.
  - ✓ 제출된 학습보고서 3~4의 경우 학습과제에서 제시된 기능중 하나를 XMLHttpRequest API를 활용하여 AJAX로 바꾸는 과정을 수행해야함(참고 요망)
- 지각 제출 시, 0점으로 처리한다.
- 채점 시 완성도의 평가는 Google의 [Chrome 브라우저](#)에서 렌더링 된 화면을 기준으로 함

※ "4. 프로그램 설계"를 참고하여 프로그램이 올바르게 동작하도록 구현한다.

## 4. 프로그램 설계

### 4.1 개요

- frontend는 Pug 템플릿 기반으로 백엔드 측에서 렌더링이 진행된다.
- backend는 영화 정보에 대한 CRUD(생성, 검색, 갱신, 삭제) 요청 및 웹페이지에 대한 요청을 처리한다.
- 예시들의 서버 포트 번호는 3000번으로 가정한다.

### 4.2 템플릿 및 프론트엔드 단 JavaScript 및 CSS

- 아래의 템플릿은 layout.pug를 상속받으며, layout.pug의 content 블록을 각각 정의한다.

#### 4.2.1 layout.pug

- layout.pug는 본 웹 어플리케이션의 템플릿의 기반이 되는 템플릿으로써 이후 설명되는



템플릿은 layout.pug 를 확장한다. 이 템플릿은 다음과 같은 역할을 수행한다.

1. 외부 CSS 정의 로드
2. Javascript 파일 로드
3. 메뉴 이동을 위한 Navbar 정의(Home, New Movie, Admin)
  - Home : 서버로 ["/", Get 메소드](#) 요청
  - New Movie : 서버로 ["/newmovie", GET 메소드](#) 요청
  - Admin : 서버로 ["/admin" GET 메소드](#) 요청

#### 4.2.2. index.pug

- 사용자로부터 ["/" 경로로 Get 메소드 요청](#)이 올 때, 렌더링 되는 템플릿이다.
- 템플릿은 백엔드 측에서 영화 정보 리스트 및 인기 영화 리스트를 얻어와 두가지 과정을 수행한다.
  1. 영화 정보(영화 이름, 영화 개봉년도, 영화 포스터 이미지) 리스트 출력
  2. 인기 상영 영화 리스트(영화 이름, 영화 개봉년도, 영화 포스터 이미지) 출력

#### 4.2.3. newmovie.pug

- 사용자로부터 ["/newmovie" 경로로 GET 요청](#)이 올 때 렌더링되는 템플릿이다.
- 이 템플릿은 새로운 영화 등록을 위한 템플릿을 제공하며, 영화 등록 UI 구성 및 영화등록 요청을 백엔드에 전달하기 위해 다음과 같은 형태의 form element를 구성한다.

Form 의 메소드	POST
Form 의 Action	<a href="#">/routes/movie/create</a>
Label element	Text content 가 Title 인 element
Input element	name attribute 가 title 인 element
Label element	Text content 가 Year 인 element
Input element	name element 가 year 인 element
Label element	Text content 가 Image Path 인 element
Input element	name element 가 url 인 element

#### 4.2.4 admin.pug

- 사용자로부터 ["/admin" 경로로 GET 요청](#)이 올 때 렌더링되는 템플릿이다.
- 이 템플릿은 백엔드 측에서 렌더링과정에서 영화정보 리스트를 가져와 아래의 과정을 수

행한다.

- 각 영화정보(이미지, 영화제목, 영화개봉년도) 를 편집버튼, 삭제 버튼, 인기 영화 지정/비지정 버튼 과 함께 렌더링
  - ✓ 편집버튼은 [프론트엔드 측 자바스크립트를 통해 백엔드 측으로 /routes/movie/read/:id 에 GET 요청](#)을 한다.
  - ✓ 삭제버튼은 [프론트엔드 측 자바스크립트를 통해 백엔드 측으로 /routes/movie/delete/:id 에 POST 요청](#)을 한다.
  - ✓ 인기 영화 지정/비지정 버튼은 [프론트엔드 측 자바스크립트를 통해 백엔드측으로 인기 영화 여부 정보와 함께 /routes/movie/update/:id 에 POST 요청](#)을 한다.

#### 4.2.5. editfile.pug

- 사용자로부터 ["/routes/movie/read/:id"\(:id는 현재 선택된 영화 정보 document의 ID\) 경로로 GET 요청](#)이 올 때 렌더링되는 템플릿이다.
- 이 템플릿은 기존 영화 정보의 수정을 위한 템플릿이며, 선택된 영화 정보를 얻어와 다음과 같은 영화 수정을 위한 form element 를 구성한다.

Form 의 메소드	POST
Form 의 Action	<a href="#">/routes/movie/update/:id</a>
Label element	Text content 가 Title 인 element
Input element	name attribute 가 title 인 element (기본 value 값은 현재 영화 정보의 title)
Label element	Text content 가 Year 인 element
Input element	name element 가 year 인 element (기본 value 값은 현재 영화 정보의 year)
Label element	Text content 가 Image Path 인 element
Input element	name element 가 url 인 element (기본 value 값은 현재 영화 정보의 url)

#### 4.2.5. 프론트엔드 측 자바스크립트(/public/javascripts/main.js)

- 프론트 엔드 측 자바스크립트는 HTML 엘리먼트로 구성되지 않은 form 요청을 정의하기 위해 활용된다.
- 자바스크립트를 통해 정의되는 form 요청 함수는 아래와 같다. (구현에 따라, 함수의 개수와 정의는 변경될 수 있다.)

영화 정보 편집 페이지 이동 함수	<a href="/routes/movie/read/:id">/routes/movie/read/:id</a> 로 GET 요청을 하는 form 구성 후 form submit
삭제 요청 함수	<a href="/routes/movie/delete/:id">/routes/movie/delete/:id</a> 로 POST 요청을 하는 form 구성 후, form submit
인기 영화 지정 함수	<a href="/routes/movie/update/:id">/routes/movie/update/:id</a> 로 POST 요청을 하는 form 구성 후, form submit 이때, hidden type 의 name 이 trending 인 input 태그의 값을 true 로 구성한 필드를 추가하여 인기영화로 정보를 변경하도록함.
인기 영화 비지정 함수	<a href="/routes/movie/update/:id">/routes/movie/update/:id</a> 로 POST 요청을 하는 form 구성 후, form submit 이때, hidden type 의 name 이 trending 인 input 태그의 값을 false 로 구성한 필드를 추가하여 인기영화로 정보를 변경하도록함.

#### 4.2.5. CSS(/public/stylesheets/style.css)

- 과제의 구현 난이도로 인해 별도의 CSS 요구사항은 존재하지 않으나, 직관성 있는 CSS를 구성할 경우 소폭 추가 점수 부여 예정.

### 4.3. 백엔드단 요청 처리

#### 4.3.1 MongoDB 설정(/app.js)

- **Mongoose** 로 연결하는 MongoDB 의 주소는 각자 로컬에 설치한 서버 주소와 기본 포트를 사용한다
- 데이터를 저장하는 **Database** 의 이름은 **“Lab11-자신의 학번”** 으로 지정한다.
- 따라서 해당 데이터베이스에 접속하는 **url** 은 다음과 같다.  
( <mongodb://localhost:27017/Lab11-학번> )
- **Mongoose-auto-increment** 패키지를 설치하고 **Import** 하여 **id** 의 중복이 발생하지 않도록 한다.

```
const mongoose = require('mongoose');
const mongooseAutoInc = require('mongoose-auto-increment');

mongoose.connect('mongodb://localhost:27017/Lab11-201724525', {
  useFindAndModify: false,
  useNewUrlParser: true,
  useUnifiedTopology: true,
});
mongooseAutoInc.initialize(mongoose.connection);
```

#### 4.3.2. 영화 스키마 및 모델 정의(models/movie.js)

- Id의 자동 증가를 위해 mongoose-auto-increment를 Import 한다.

```
const mongoose = require('mongoose');
const mongooseAutoInc = require('mongoose-auto-increment');

const Schema = mongoose.Schema;

const movieSchema = new Schema({
  title: {
    type: String,
    required: true,
  },
  year: {
    type: Number,
    required: true,
  },
  url: {
    type: String,
    required: true,
  },
  trending: {
    type: Boolean,
    required: true,
    default: false,
  },
});

movieSchema.plugin(mongooseAutoInc.plugin, 'movie');

module.exports = mongoose.model('movie', movieSchema);
```

#### 4.3.3 index.js

- Index.js 의 경우, 영화 정보 어플리케이션의 주요 웹페이지의 렌더링을 처리하는 모듈을 정의함.
- 요청 실패 상황은 고려하지 않는다.

Method	URI	동작
GET	/	<ul style="list-style-type: none"><li>✓ MongoDB 내에 저장된 <u>모든 영화의 목록을 배열 형태로 반환한다.</u></li><li>✓ 반환된 영화 목록을 활용하여 유행영화 목록 생성</li></ul>

		✓ 영화 목록 및 유행 영화 목록을 렌더링을 위한 local variable로 활용하여 index.pug를 렌더링한다..
GET	/newmovie	✓ MongoDB에 새로운 영화 등록을 위한 newmovie.pug 템플릿을 렌더링한다..
GET	/admin	✓ MongoDB 내에 저장된 <u>모든 영화의 목록을 배열 형태로 반환한다.</u> ✓ 반환된 영화 목록을 렌더링을 위한 local variable로 활용하여 admin.pug를 렌더링한다. ✓ 이 때 영화가 유행하는 영화 일 경우, 별표 표시를 한다.

#### 4.3.4 movie.js

- movie.js 의 경우, 영화 정보의 CRUD를 담당한다.
- 요청 실패 상황은 고려하지 않는다.

POST	/routes/movie/create	✓ 클라이언트가 전송한 영화 정보를 <b>MongoDB</b> 에 저장한다. ✓ 서버에 저장되는 영화 정보의 데이터 구조는 다음과 같다. <div style="text-align: center;">{ _id, title, year, url, trending }</div> → <b>_id</b> : mongoose-auto-increment 에 의해 자동 생성된 index → <b>title</b> : 클라이언트로부터 전송 받은 <u>영화 제목</u> → <b>year</b> : 클라이언트로부터 전송 받은 <u>영화의 개봉년도</u> → <b>url</b> : 클라이언트로부터 전송 받은 <u>영화 포스터의 url</u> → <b>trending</b> : 인기 영화를 분류하기 위한 <b>Boolean</b> 값으로, <u>영화 생성시 false 값</u> 이 저장되어야 한다. ✓ 데이터 저장에 성공하면, <a href="http://localhost:3000/">http://localhost:3000/</a> 로 redirect한다.
------	----------------------	---

GET	/routes/movie/read/:id	✓ <u>id 번 영화의 정보를 반환한다.</u> ✓ 해당 id 에 영화가 존재한다면, <a href="http://localhost:3000/editmovie">http://localhost:3000/editmovie</a> 를 영화 정보와 함께 렌더링한다.
POST	/routes/movie/update/:id	✓ <u>id 번 영화의 정보를 수정한다.</u> ✓ 클라이언트는 수정 요청을 할 때, <u>영화의 정보 수정{ title, year, url }</u> 또는 <u>인기 영화 지정{ trending }</u> 두 가지 형태의 데이터를 전송하게 되며, 서버는 전송 받은 값을 <u>id 번 영화에 반영한다.</u> ✓ 해당 id 에 영화가 존재한다면, 영화의 정보를 수정한 후에 <a href="http://localhost:3000/admin">http://localhost:3000/admin</a> 으로 redirect 한다.
POST	/routes/movie/delete/:id	✓ <u>id 번 영화의 정보를 삭제한다.</u> ✓ 해당 id 영화가 존재할 경우 <u>영화 정보를 삭제</u> 한 후에 <a href="http://localhost:3000/admin">http://localhost:3000/admin</a> 으로 redirect 한다.




## 5. 실행 예시

- 인기영화 예시에서는 Begin Again의 인기영화 지정 후, Trending Movies에 인기영화로 추가됨을 보여줌.
- 실행 예시에서는 노란 별이 인기영화 표시(요구사항 아님)
- 예시 설명
  - ✓ **1번 예시** : "/"에 접속하면 홈 화면이 보인다. 홈 화면에는 영화 리스트와 인기영화 리스트가 출력된다.
  - ✓ **2번 예시** : 네비게이션 바의 New Movie를 출력하면 새 영화를 입력할 수 있는 창으로 이동되며, 새 영화를 입력하면, 홈화면과 Admin화면에서 새 영화가 추가된 것을 확인할 수 있다.
  - ✓ **3번 예시** : 네비게이션 바의 Admin을 클릭하면 관리자페이지로 이동하며, 관리자 페이지에서는 각 영화의 수정, 삭제, 인기 영화 지정/비지정을 수행할 수 있다.
  - ✓ **4번 예시** : 관리자페이지에서 각 영화의 인기영화 지정/비 지정 버튼을 클릭하면 각

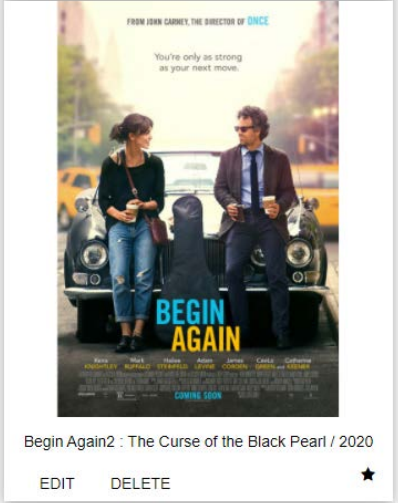

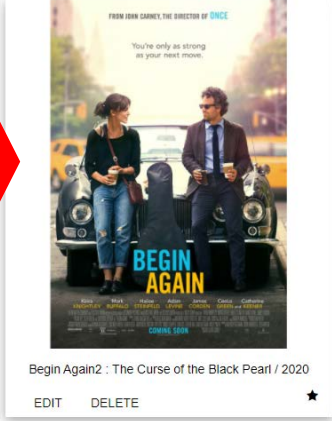
영화의 인기영화 여부를 지정할 수 있으며, 인기영화 지정 여부의 결과는 홈 화면의 인기영화리스트와 관리자 화면의 인기영화 지정/비지정 버튼 정보로 확인된다. (예시에서는 검은별 -> 노란별 변경)

- ✓ **5번 예시** : 관리자 화면에서 각 화면의 EDIT 버튼을 클릭하면 수정페이지로 이동하여, 영화정보를 수정할 수 있다.
- ✓ **6번 예시** : 인기영화 비지정버튼을 누르면, 영화가 인기영화에서 제외되며, 이는 홈 화면 인기영화 리스트에서의 제외 및 관리자화면의 인기영화 비지정 버튼이 지정 버튼으로 변화되는것으로 확인할 수 있다.(예시에서는 노란별 -> 검은별 변경)
- ✓ **7번 예시** : 관리자 화면에서 영화를 삭제하면 홈화면과 관리자 화면의 영화리스트에서 영화가 제외된다.

	<p style="text-align: center;">Movies</p>
<p>1. <a href="http://localhost:3000/">http://localhost:3000/</a> 접속</p>	<p>2 NavBar 의 New Movie 선택 ➡ 새로운 영화 등록 및 확인</p>

		
<p>3 네비게이션 바의 Admin 선택</p>	<p>4 인기영화 지정(실행 예시에서는 노란 별이 인기영화 표시)</p>	<p>5 영화 별 정보 수정 및 수정 정보 전달</p>



	<div data-bbox="655 230 756 264">Movies</div> <div data-bbox="667 277 983 734">  </div> <div data-bbox="1018 338 1377 371"> <a href="#">Home</a> <a href="#">New Movie</a> <a href="#">Admin</a> </div> <div data-bbox="1018 387 1118 421">Movies</div> <div data-bbox="1029 434 1362 853">  </div>
<p>6 인기영화 비지정(비지정시 Home 의 Trending Movies 에서 제외)</p>	<p>7 영화 삭제</p>