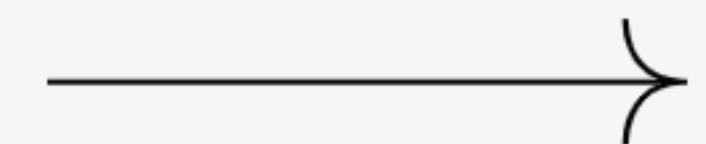


영화 리뷰 웹 사이트

프로젝트 ↗

MOVIE24
김은하



1:INDEX

프로젝트
개요

- 프로젝트 주제
- 프로젝트 목적
- 개발 환경 및 사용 언어
- 참고 사이트

프로젝트
설계 구조

- 프로젝트 구조
- H2 데이터베이스
- DAO 및 Controller
- JSP (뷰)

프로젝트
상세 기능

- 메인 화면 (영화 리스트)
- 영화 상세 정보 화면
- 리뷰 등록/삭제
- 공지사항, 커뮤니티 화면

프로젝트 개요

PROJECT OUTLINE

프로젝트 주제

영화에 대한 리뷰를 등록하고
사용자들끼리 소통할 수 있는 웹사이트
(가제: MOVIE 24)



영화 리뷰 웹 사이트 만들기

프로젝트 목적

Spring Framework, JSP에 대한 이해와
데이터베이스 연동하여
웹사이트 구현해보기

개발 환경 및 사용 언어

Spring Framework와
H2 데이터베이스를 기반으로 하여
JAVA, HTML, CSS 사용해 웹사이트 구현

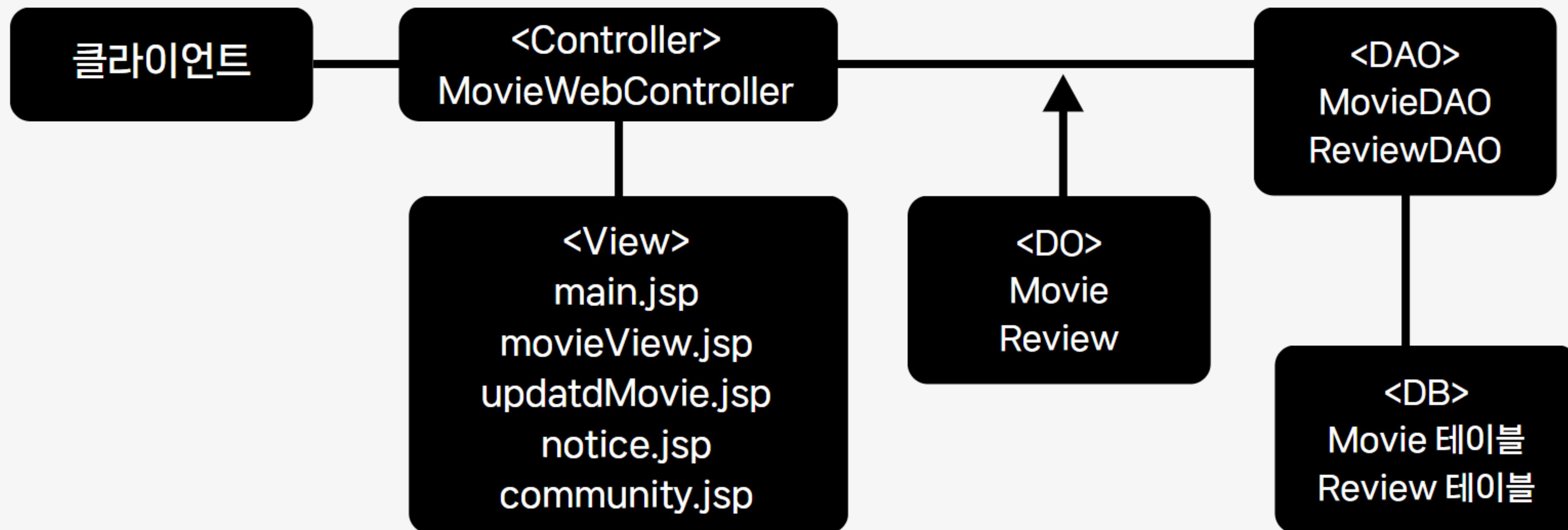
참고사이트

씨네21
[http://www.cine21.com/movie/lists/
playing](http://www.cine21.com/movie/lists/playing)

프로젝트 설계 구조

프로젝트 구조

PROJECTED STRUCTURE



프로젝트 설계 구조

H2 데이터베이스

H2 DATABASE

MOVIE TABLE	
ID	INT PRIMARY KEY AUTO_INCREMENT
NAME	VARCHAR
IMG	VARCHAR
RELEASE	TIMESTAMP
GENRE	VARCHAR
SHOWTIME	INT
AUDIENCE	INT
DIRECTOR	VARCHAR
APPEARANCE	VARCHAR
SUMMARY	TEXT

영화 정보가 들어있는 MOVIE 데이터베이스와
MOVIE 테이블의 ID 컬럼을 참조하여
리뷰 정보가 들어있는 REVIEW 데이터베이스 생성

REVIEW TABLE	
REVIEW_ID	INT PRIMARY KEY AUTO_INCREMENT
MOVIE_ID	INT
REVIEWER_NAME	VARCHAR
CONTENT	TEXT
CREATED_AT	TIMESTAMP DEFAULT CURRENT_TIMESTAMP

프로젝트 설계 구조

DAO 및 Controller

MovieDAO.java

```
@Component
public class MovieDAO {

    final String JDBC_DRIVER = "org.h2.Driver";
    final String JDBC_URL="jdbc:h2:tcp://localhost/~/jwbookdb";

    // DB 연결을 가져오는 메서드
    public Connection open() {
        Connection conn = null;

        try {
            Class.forName(JDBC_DRIVER);
            conn = DriverManager.getConnection(JDBC_URL, "jwbook", "1234");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return conn;
    }
}
```

MovieDAO.java

- 스프링 빈으로 등록하기 위해
@Component 애너테이션 사용
- DB 와의 연결을 위한 open() 메서드

DAO 및 Controller

MovieDAO.java

```
// 영화 목록 전체 가져오기 위한 메서드
public List<Movie> getAll() throws Exception {

    Connection conn = open();
    List<Movie> movieList = new ArrayList<Movie>();

    String sql = "SELECT * FROM movie order by release desc";

    PreparedStatement pstmt = conn.prepareStatement(sql);
    ResultSet rs = pstmt.executeQuery();

    try(conn; pstmt; rs) {
        while(rs.next()) {
            Movie m = new Movie();
            m.setId(rs.getInt("id"));
            m.setName(rs.getString("name"));
            m.setImg(rs.getString("img"));
            m.setRelease(rs.getDate("release"));
            m.setGenre(rs.getString("genre"));
            m.setShowtime(rs.getInt("showtime"));
            m.setAudience(rs.getInt("audience"));
            m.setDirector(rs.getString("director"));
            m.setAppearance(rs.getString("appearance"));
            m.setSummary(rs.getString("summary"));

            movieList.add(m);
        }
    }
    return movieList;
}
```

getAll()

- 영화 목록 전체를 가져오기 위한 메서드
- movieList 객체를 배열로 생성하여 저장
- Movie 객체를 생성하여 while 문을 통해 각각 값을 저장하여 movieList 객체에 저장
- DB는 release (개봉일) 기준 내림차순하여 정렬

DAO 및 Controller

MovieDAO.java

```
// 영화 정보 가져오기 위한 메서드 (링크 클릭했을 때)
public Movie getMovie(int id) throws Exception {

    Connection conn = open();
    Movie m = null;

    String sql = "SELECT * FROM movie WHERE id = ?";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setInt(1, id);
    ResultSet rs = pstmt.executeQuery();

    rs.next();

    try(conn; pstmt; rs) {
        m = new Movie();
        m.setId(rs.getInt("id"));
        m.setName(rs.getString("name"));
        m.setImg(rs.getString("img"));
        m.setRelease(rs.getDate("release"));
        m.setGenre(rs.getString("genre"));
        m.setShowtime(rs.getInt("showtime"));
        m.setAudience(rs.getInt("audience"));
        m.setDirector(rs.getString("director"));
        m.setAppearance(rs.getString("appearance"));
        m.setSummary(rs.getString("summary"));
    }
    rs.close();
    pstmt.close();
    conn.close();

    return m;
}
```

getMovie()

- 영화 링크를 클릭했을 시 상세정보를 띠워주는 메서드
id 값을 받아 해당하는 영화의 정보를 보여주도록 처리

프로젝트 설계 구조

DAO 및 Controller

MovieDAO.java

```
// 영화 추가 메서드
public void addMovie(Movie m) throws Exception {
    Connection conn = open();
    String sql = "INSERT INTO movie (name, img, release,"
        + " genre, showtime, audience, director, appearance, summary)";
    sql += " VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?)";
    PreparedStatement pstmt = conn.prepareStatement(sql);

    try(conn; pstmt) {
        pstmt.setString(1, m.getName());
        pstmt.setString(2, m.getImg());
        pstmt.setDate(3, m.getRelease());
        pstmt.setString(4, m.getGenre());
        pstmt.setInt(5, m.getShowtime());
        pstmt.setInt(6, m.getAudience());
        pstmt.setString(7, m.getDirector());
        pstmt.setString(8, m.getAppearance());
        pstmt.setString(9, m.getSummary());
        pstmt.executeUpdate();
    }
}

// 영화 삭제 메서드
public void delMovie(int id) throws SQLException {
    Connection conn = open();
    String sql = "DELETE FROM movie WHERE id = ?";
    PreparedStatement pstmt = conn.prepareStatement(sql);

    try(conn; pstmt) {
        pstmt.setInt(1, id);
        if(pstmt.executeUpdate() == 0) {
            throw new SQLException("데이터 삭제 에러");
        }
    }
}

// 영화 수정 메서드
public void updateMovie(Movie m) throws Exception {
    Connection conn = open();
    String sql = "UPDATE movie SET name = ?, img = ?, release = ?"
        + ", genre = ?, showtime = ?, audience = ?"
        + ", director = ?, appearance = ?, summary = ? WHERE id = ?";
    PreparedStatement pstmt = conn.prepareStatement(sql);

    try(conn; pstmt) {
        pstmt.setString(1, m.getName());
        pstmt.setString(2, m.getImg());
        pstmt.setDate(3, m.getRelease());
        pstmt.setString(4, m.getGenre());
        pstmt.setInt(5, m.getShowtime());
        pstmt.setInt(6, m.getAudience());
        pstmt.setString(7, m.getDirector());
        pstmt.setString(8, m.getAppearance());
        pstmt.setString(9, m.getSummary());
        pstmt.setInt(10, m.getId());
        pstmt.executeUpdate();
    }
}
```

addMovie(), delMovie(), updateMovie()

- 영화 추가, 삭제, 수정 메서드
- 영화 추가, 수정 메서드의 경우 Movie를 객체로 받아 처리
- 영화 삭제의 경우 id 값을 받아 해당하는 데이터만 삭제되게 처리

DAO 및 Controller

ReviewDAO.java

```
// 리뷰 등록 메서드
public void addReview(Review review) throws Exception {
    Connection conn = open();
    String sql = "insert into review (movie_id, reviewer_name"
        + ", content, created_at) values(?, ?, ?, CURRENT_TIMESTAMP())";
    PreparedStatement pstmt = conn.prepareStatement(sql);

    try(conn; pstmt) {
        pstmt.setInt(1, review.getMovieId());
        pstmt.setString(2, review.getReviewerName());
        pstmt.setString(3, review.getContent());
        pstmt.executeUpdate();
    }
}

// 리뷰 삭제 메서드
public void delReview(int reviewId) throws Exception {
    Connection conn = open();
    String sql = "delete from review where review_id = ?";
    PreparedStatement pstmt = conn.prepareStatement(sql);

    try(conn; pstmt) {
        pstmt.setInt(1, reviewId);
        if(pstmt.executeUpdate() == 0) {
            throw new Exception("데이터 삭제 에러");
        }
    }
}
```

```
// 리뷰 목록 조회 메서드
public List<Review> getReview(int movieId) throws Exception {

    Connection conn = open();
    List<Review> reviewList = new ArrayList<Review>();

    String sql = "SELECT review_id, movie_id, reviewer_name, content, "
        + "FORMATDATETIME(created_at, 'yyyy-MM-dd hh:mm:ss') AS cdate ";
    sql += "FROM review ";
    sql += "WHERE movie_id = ?";

    PreparedStatement pstmt = conn.prepareStatement(sql);

    pstmt.setInt(1, movieId);
    ResultSet rs = pstmt.executeQuery();

    try(conn; pstmt; rs) {
        while(rs.next()) {
            Review r = new Review();
            r.setReviewId(rs.getInt("review_id"));
            r.setMovieId(rs.getInt("movie_id"));
            r.setReviewerName(rs.getString("reviewer_name"));
            r.setContent(rs.getString("content"));
            r.setCreatedAt(rs.getString("cdate"));

            reviewList.add(r);
        }
    }
    return reviewList;
}
```

addReview(), delReview(), getReview()

- 리뷰 목록 조회, 추가, 삭제 메서드
- 리뷰 등록은 Review를 객체로 받아 처리
- 목록 조회는 해당 영화의 id를 값으로 받아 처리
- 삭제는 리뷰 id를 값으로 받아 처리

프로젝트 설계 구조

DAO 및 Controller

MovieWebController.java

```
@Controller  
@RequestMapping("/movie")  
public class MovieWebController {  
  
    final MovieDAO movieDAO;  
    final ReviewDAO reviewDAO;  
  
    private final Logger logger = LoggerFactory.getLogger(this.getClass());  
  
    @Value("${movie.img.dir}") // 프로퍼티 추가 완료  
    private String fdir;  
  
    @Autowired  
    public MovieWebController(MovieDAO movieDAO, ReviewDAO reviewDAO) {  
        this.movieDAO = movieDAO;  
        this.reviewDAO = reviewDAO;  
    }  
}
```

listMovie(), addMovie()

- @Controller 애너테이션을 붙여 컨트롤러로 지정
- @RequestMapping 을 사용해 "/movie" url과 연결
- @Value 를 사용해 이미지 경로 지정
- @Autowired 를 사용해 new 키워드를 사용하지 않고 인스턴스 생성
- listMovie 메서드는 GetMapping 을 사용해 "/movie/main" 과 연결
- addMovie 메서드는 PostMapping 을 사용, 데이터가 추가되면 main으로 연결

```
// 영화 목록 조회(메인화면)  
@GetMapping("/main")  
public String listMovie(Model m) {  
    List<Movie> list;  
  
    try {  
        list = movieDAO.getAll();  
        m.addAttribute("movielist", list);  
    } catch (Exception e) {  
        e.printStackTrace();  
        logger.error("영화 목록 생성 과정에서 문제 발생!!");  
        m.addAttribute("error", "영화 목록이 정상적으로 처리되지 않았습니다!!");  
    }  
    return "movie/main";  
}  
  
// 영화 등록  
@PostMapping("/add")  
public String addMovie(@ModelAttribute Movie movie, Model m, @RequestParam("file") MultipartFile file) {  
  
    try {  
        1. 저장 파일 객체 생성  
        String fileName = file.getOriginalFilename();  
  
        String sysName = System.currentTimeMillis() + fileName.substring(fileName.lastIndexOf("."));  
        File dest = new File(fdir + "/" + sysName);  
  
        file.transferTo(dest); // 서버 경로에 파일 쓰기  
  
        movie.setImg("/img/" + dest.getName());  
  
        movieDAO.addMovie(movie);  
    } catch (Exception e) {  
        e.printStackTrace();  
        logger.info("영화 추가(등록) 과정에서 문제 발생!!");  
        m.addAttribute("error", "영화가 정상적으로 등록 되지 않았습니다!!");  
    }  
    return "redirect:/movie/main";  
}
```

프로젝트 설계 구조

DAO 및 Controller

MovieWebController.java

```
// 영화 상세정보 조회
@GetMapping("/{id}")
public String getMovie(@PathVariable int id, Model mo) {
    Movie m;
    try {
        m = movieDAO.getMovie(id);
        List<Review> reviews = reviewDAO.getReview(id);
        mo.addAttribute("movie", m);
        mo.addAttribute("reviews", reviews);
    } catch (Exception e) {
        e.printStackTrace();
        logger.error("영화 정보를 가져오는 과정에서 문제 발생!!!");
        mo.addAttribute("error", "영화 정보를 정상적으로 가져오지 못했습니다!!!");
    }
    return "movie/movieView";
}

// 영화 삭제
@GetMapping("/delete/{id}")
public String delMovie(@PathVariable int id, Model m) {
    try {
        movieDAO.delMovie(id);
    } catch (Exception e) {
        e.printStackTrace();
        logger.info("영화 삭제 과정에서 문제 발생!!!");
        m.addAttribute("error", "영화가 정상적으로 삭제되지 않았습니다!!!");
    }
    return "redirect:/movie/main";
}
```

getMovie(), delMovie()

- getMovie 메서드는 GetMapping을 사용해 영화 id를 받아 영화 상세정보와 리뷰를 보여주는 메서드
- delMovie() 메서드도 영화 id를 받고, 삭제가 되면 main 화면으로 돌아가는 메서드

프로젝트 설계 구조

DAO 및 Controller

MovieWebController.java

```
// 영화 수정 화면
@GetMapping("/update/{id}")
public String updateForm(@PathVariable int id, @ModelAttribute Movie movie, Model m) {
    try {
        movie.setId(id);
        movieDAO.getMovie(id);

    } catch (Exception e) {
        e.printStackTrace();
        logger.info("영화 수정 화면 이동 문제 발생!!!");
        m.addAttribute("error", "영화 수정 화면 이동이 정상적으로 처리 되지 않았습니다!!!");
    }
    return "movie/updateMovie";
}

// 영화 수정
@PostMapping("/update/{id}")
public String updateMovie(@PathVariable int id, @ModelAttribute Movie movie, Model m
    , @RequestParam("file") MultipartFile file) {

    try {
        //1. 저장 파일 객체 생성
        String fileName = file.getOriginalFilename();

        String sysName = System.currentTimeMillis() +
            fileName.substring(fileName.lastIndexOf("."));
        File dest = new File(fdir + "/" + sysName);

        file.transferTo(dest); // 서버 경로에 파일 쓰기

        movie.setImg("/img/" + dest.getName());

        movie.setId(id);
        movieDAO.updateMovie(movie);

    } catch (Exception e) {
        e.printStackTrace();
        logger.info("영화 수정 과정에서 문제 발생!!!");
        m.addAttribute("error", "영화가 정상적으로 수정 되지 않았습니다!!!");
    }
    return "redirect:/movie/" + id;
}
```

updateForm(), updateMovie()

- updateForm 메서드는 링크를 클릭했을 때 수정 화면으로 이동하게 해주는 메서드
- updateMovie 메서드는 데이터를 입력받아 저장해준 후 다시 상세 정보화면으로 이동하게 해주는 메서드

프로젝트 설계 구조

JSP (뷰)

main.jsp

```
<h2>영화 목록</h2>
<hr />
<ul class="list-group">
    <c:forEach var="movie" items="${movielist}">
        <li class="list-group-item list-group-item-action d-flex justify-content-between align-items-center">
            <a href="/movie/${movie.id}" class="text-decoration-none">
                [ ${movie.release} ] ${movie.name}
            </a>

            <a href="/movie/delete/${movie.id}"><span class="badge bg-secondary">&times;</span></a>
        </li>
    </c:forEach>
</ul>
```

- JSTL 커스텀 태그를 사용하여 html 안에 java를 넣어서 반복문 처리

```
<button type="button" class="btn btn-outline-secondary" data-bs-toggle="collapse"
data-bs-target="#addForm" aria-expanded="false" aria-controls="addForm">영화 등록</button>

<div class="collapse" id="addForm">

    <div class="card card-body">
        <form method="post" action="/movie/add" enctype="multipart/form-data">
            <label class="form-label">영화 제목</label>
            <input class="form-control" type="text" name="name">
            <label class="form-label">이미지</label>
            <input class="form-control" type="file" name="file">
            <label class="form-label">개봉일</label>
            <input class="form-control" type="text" name="release">
            <label class="form-label">영화 장르</label>
            <input class="form-control" type="text" name="genre">
            <label class="form-label">상영 시간</label>
            <input class="form-control" type="text" name="showtime">
            <label class="form-label">누적 관객수</label>
            <input class="form-control" type="text" name="audience">
            <label class="form-label">감독</label>
            <input class="form-control" type="text" name="director">
            <label class="form-label">출연</label>
            <input class="form-control" type="text" name="appearance">
            <label class="form-label">줄거리</label>
            <textarea class="form-control" name="summary" cols="50" rows="5"></textarea>
            <button type="submit" class="btn btn-primary mt-3">저장</button>
        </form>
    </div>
</div>
```

- 부트 스트랩의 toggle="collapse"를 사용하여 클릭하면 열렸다 닫혔다 하는 기능 구현

프로젝트 설계 구조

JSP (뷰)

movieView.jsp

```
<button type="button" class="btn btn-outline-secondary" data-bs-toggle="collapse" data-bs-target="#reviewForm" aria-expanded="false" aria-controls="reviewForm">리뷰 남기기</button>

<div class="collapse" id="reviewForm">
  <div class="card card-body">
    <form method="post" action="${pageContext.request.contextPath}/movie/${movie.id}/addReview" enctype="multipart/form-data">
      <div class="mb-3">
        <label for="reviewerName" class="form-label">이름</label>
        <input class="form-control" type="text" id="reviewerName" name="reviewerName" required>
      </div>
      <div class="mb-3">
        <label for="content" class="form-label">리뷰 작성</label>
        <textarea class="form-control" id="content" name="content" cols="50" rows="5" required></textarea>
      </div>
      <button type="submit" class="btn btn-primary mt-3">저장</button>
    </form>
  </div>
</div>
```

- main 화면과 마찬가리로 JSTL 태그 사용해주었고,
action 값으로 pageContext 객체를 사용해 addReview와 연결

```
<a href="/movie/main" class="btn btn-outline-secondary"><< HOME</a>
<a href="/movie/update/${movie.id}" onclick="window.open(this.href, '_self', '영화 정보 수정'); return false;" class="btn btn-outline-danger">영화 정보 수정</a>
```

- main으로 돌아가는 HOME 버튼 생성
- onclick 속성 사용해 update 페이지로 가는 영화 정보 수정 버튼 생성

프로젝트 상태 기능

메인 화면 (영화 리스트)

MAIN

MOVIE 24

공지사항

커뮤니티

영화 목록

- [2024-07-31] 파일럿
- [2024-07-24] 데드풀과 올버린
- [2024-07-24] 파편들의 집
- [2024-07-17] 명탐정 코난: 100만 달러의 펜타그램
- [2024-07-17] [재개봉] 비포 선라이즈
- [2024-07-17] 그리고 바통은 넘겨졌다
- [2024-07-03] 탈주
- [2024-06-26] 핸섬가이즈
- [2024-06-12] 인사이드 아웃 2

영화 등록

- updateView 영화 등록 폼은 영화 수정 화면과 동일

영화 등록

영화 제목

이미지

파일 선택 선택된 파일 없음

개봉일

영화 장르

상영 시간

누적 관객수

감독

출연

줄거리

저장

프로젝트 상세 기능

영화 상세 정보 화면

MOVIEVIEW

명탐정 코난: 100만 달러의 펜타그램

A screenshot of a movie review submission form. At the top right is a thumbnail image of the movie poster. Below it is a large input field for a review title, with the placeholder "리뷰 작성" (Review Content). To the left of this field is a smaller input field for the user's name. At the bottom right is a blue button labeled "저장" (Save). A callout bubble at the bottom right contains the text "- 리뷰 남기기 버튼 클릭 시".



개봉일: 2024-07-17

장르: 애니메이션

상영시간: 111분

누적 관람객수: 530059명

감독: 나가오카 치카

출연: 괴도키드, 핫토리 헤이지, 토야마 카즈하, 에도카와 코난 외

줄거리: 훗카이도 하코다테에 있는 오노에 재벌 가의 창고에 괴도 키드의 예고장이 도착한다. 빅 주얼만을 노리는 키드가 이번에 노리는 것은 과거 신선조 귀신 부장 히지카타 토키조와 엉친 전설적인 검. 검술 대회에 참가하기 위해 하코다테에 방문한 핫토리 헤이지와 그를 응원하기 위해 온 코난 일행도 괴도 키드를 막기 위해 사건에 뛰어들게 된다. 한편, 가슴에 열 십 자(+) 모양의 자상을 입은 시신이 발견되고 죽음의 상인이라고 불리는 무기상이 용의자로 지목된다. 그 역시 괴도 키드가 찾는 검을 노리고 있었고, 그 검이 오노에 가문이 세대에 걸쳐 지킨 보물을 찾을 열쇠임이 밝혀진다. 검을 쫓는 키드에게 수수께끼의 검사가 습격해 오고, 절체절명의 위기가 닥쳐오는데...! 검에 숨겨진 진실이 어두운 밤을 베고 달빛 아래 드러난다!

리뷰

괴도키드

괴도키드가 제일인 듯 | 2024-07-28 01:32:46

핫토리짱

핫토리가 제일 멋있음 | 2024-07-28 01:33:01

리뷰 남기기

- x버튼으로 리뷰 삭제 가능

프로젝트 상태 기능

공지사항, 커뮤니티 화면

NOTICE, COMMUNITY

MOVIE 24		
번호	제목	등록일
★	[공지] 사이트 이용에 참고 부탁드립니다.	2024-07-28
1	공지사항입니다.	2024-07-28
2	공지사항입니다.	2024-07-28
3	공지사항입니다.	2024-07-28
4	공지사항입니다.	2024-07-28
5		
6		
7		
8		
9		
10		

공지사항

번호	제목	등록일
★	[공지] 사이트 이용에 참고 부탁드립니다.	2024-07-28
1	공지사항입니다.	2024-07-28
2	공지사항입니다.	2024-07-28
3	공지사항입니다.	2024-07-28
4	공지사항입니다.	2024-07-28
5		
6		
7		
8		
9		
10		

- 공지사항 화면

MOVIE 24		
번호	제목	등록일
1	재미있는 영화가 참 많네요	2024-07-28
2	다들 어떤 영화 좋아하시나요?	2024-07-28
3	제일 기대되는 영화는 어떤 영화예요?	2024-07-28
4		
5		
6		
7		
8		
9		
10		

- 커뮤니티 화면

감사합니다

