Jsp 프로젝트

-ai 사용 기록-

학번 : 202207004, 이름 : 심동준

1. 데이터베이스 구성 중 트리거 사용 부분

```
CREATE OR REPLACE TRIGGER trg_update_avg_rating
FOR INSERT OR UPDATE OR DELETE ON reviews
COMPOUND TRIGGER

    TYPE movie_id_assoc_array_t IS TABLE OF BOOLEAN INDEX BY PLS_INTEGER;
    g_movie_ids movie_id_assoc_array_t;

    AFTER EACH ROW IS
    BEGIN
        IF INSERTING OR UPDATING THEN
            g_movie_ids(:NEW.movie_id) := TRUE;
        ELSIF DELETING THEN
            g_movie_ids(:OLD.movie_id) := TRUE;
        END IF;
    END AFTER EACH ROW;

    AFTER STATEMENT IS
        v_avg NUMBER(3,1);
        v_movie_id PLS_INTEGER;
    BEGIN
        v_movie_id := g_movie_ids.FIRST;

        WHILE v_movie_id IS NOT NULL
        LOOP
            SELECT NVL(ROUND(AVG(rating), 1), 0)
            INTO v_avg
            FROM reviews
            WHERE movie_id = v_movie_id;

            UPDATE movies
            SET avg_rating = v_avg
            WHERE movie_id = v_movie_id;

            v_movie_id := g_movie_ids.NEXT(v_movie_id);
        END LOOP;

    END AFTER STATEMENT;
```

2. DAO 구조 설계 중 싱글톤 패턴 사용 부분

```java
public class MovieDAO {
    // DAO 객체가 여러 개 생성되지 않도록 싱글톤 패턴으로 설계
    // 하나의 객체만 생성되어 자원을 효율적으로 관리 가능
    private static MovieDAO instance = new MovieDAO();
    private MovieDAO() {}
    public static MovieDAO getInstance() {
        return instance;
    }
```

3. 데이터베이스 연결방법인 DBCP 사용 부분

```java
public class DBManager {

    /**
     * 커넥션 풀에서 Connection 객체를 하나 빌려오는 메소드
     * @return 연결된 Connection 객체
     */
    public static Connection getConnection() {
        Connection conn = null;
        try {
            // 1. 탐색기(Context) 열기
            // JNDI(Java Naming and Directory Interface) 서비스를 시작합니다.
            // 서버 설정 파일에 있는 자원을 찾을 준비를 하는 것입니다.
            Context initContext = new InitialContext();

            // 2. 환경 설정 폴더로 이동
            // "java:/comp/env"는 톰캣이 리소스를 관리하는 표준 디렉터리 경로입니다.
            Context envContext  = (Context)initContext.lookup("java:/comp/env");

            // 3. 'jdbc/oracle'이라는 이름표를 가진 자원(DataSource) 찾기
            // context.xml 파일에 설정해둔 name="jdbc/oracle"을 찾는 것입니다.
            // 여기서 DataSource는 '커넥션 풀 관리자'입니다.
            DataSource ds = (DataSource)envContext.lookup("jdbc/oracle");

            // 4. 관리자에게 연결 하나 빌려오기
            // 풀에 미리 만들어져 있는 Connection 객체 중 놀고 있는 것을 하나 가져옵니다.
            conn = ds.getConnection();
        } catch (Exception e) {
            System.err.println("getConnection 메소드 오류");
            e.printStackTrace();
        }
        return conn;
    }
```

4. 서브 이미지 리스트 구성을 파일 처리로 구성하는 부분

```java
private List<String> getImagesFromFolder(HttpServletRequest request, long movieId) {
    List<String> imageList = new ArrayList<>();

    // 1. 영화의 이미지 경로
    String webPath = "/images/" + movieId;

    // 2. 실제 서버의 물리적 경로 찾기
    String realPath = request.getServletContext().getRealPath(webPath);

    // 3. 폴더 객체 생성
    File dir = new File(realPath);

    // 4. 폴더가 존재하고, 실제 디렉토리인지 확인
    if (dir.exists() && dir.isDirectory()) {
        // 폴더 내의 모든 파일 목록을 가져옴
        File[] files = dir.listFiles();

        if (files != null) {
            for (File file : files) {
                // 파일인 경우만 처리 (하위 폴더 제외)
                if (file.isFile()) {
                    String fileName = file.getName();

                    // 이미지 파일인지 확인 (jpg, png, gif 등) - 대소문자 무시
                    String lowerName = fileName.toLowerCase();
                    if (lowerName.endsWith(".jpg") || lowerName.endsWith(".png") ||
                        lowerName.endsWith(".jpeg") || lowerName.endsWith(".gif")) {

                        // "main.png" 메인 포스터 제외
                        if (!fileName.equals("main.png")) {
                            imageList.add(webPath + "/" + fileName);
                        }
                    }
                }
            }
        }
    }
    // 숫자 순서대로 정렬
    Collections.sort(imageList);
```