

**2023-2024 学年 第 2 学期**

**框架技术开发 实验报告**

**专 业：软件工程**

**班 级：5班**

**学 号：2021105220250**

**姓 名：李艺杰**

**指导教师： 丁钰**

**地 点：11-206**

**日 期： 2024.3.25**

|  |
| --- |
| **一、实验题目**  实验四 数据表操作映射 MAPPER 文件的建立 |
| **二、实验目的**  1.掌握映射文件的结构及各元素的含义;  2.掌握数据表插入操作的映射文件的建立方法;  3.掌握数据表更新操作的映射文件的建立方法;  4.掌握数据表删除操作的映射文件的建立方法;  5.掌握数据表查询操作的映射文件的建立方法; |
| **三、实验内容**  1、namespace的命名必须跟某个接口同名;  2、接口中的方法与映射文件中SQL语句id -一对应;  3、完成各数据表的增、删、改(不只改一个字段 动态 SQL)的操作4、完成各数据表的查询(单表的、多表的联合动态SQL)的操作 |
| 1. **实验步骤**   **代码实现** 定义数据模型类 public class Movie {  private Integer id;  private String title;  private String director;  private String actors;  private Date releaseDate;  // getters and setters  } 2. 创建映射文件 在**MyBatis中，你需要为Movie类创建一个映射文件，例如MovieMapper.xml。**  **<mapper namespace="com.example.mapper.MovieMapper">**  **<resultMap id="MovieResultMap" type="com.example.model.Movie">**  **<id property="id" column="id" />**  **<result property="title" column="title" />**  **<!-- Other fields -->**  **</resultMap>**    **<insert id="insertMovie" parameterType="com.example.model.Movie">**  **INSERT INTO movie (title, director, actors, release\_date)**  **VALUES (#{title}, #{director}, #{actors}, #{releaseDate})**  **</insert>**    **<update id="updateMovie" parameterType="com.example.model.Movie">**  **UPDATE movie**  **SET**  **title = #{title},**  **director = #{director},**  **actors = #{actors},**  **release\_date = #{releaseDate}**  **WHERE id = #{id}**  **</update>**    **<delete id="deleteMovie" parameterType="java.lang.Integer">**  **DELETE FROM movie WHERE id = #{id}**  **</delete>**    **<select id="selectMovie" parameterType="java.lang.Integer" resultMap="MovieResultMap">**  **SELECT \* FROM movie WHERE id = #{id}**  **</select>**    **<!-- Add more dynamic queries as needed -->**  **</mapper>** **3. 定义Mapper接口** **package com.example.mapper;**  **public interface MovieMapper {**  **int insertMovie(Movie movie);**  **int updateMovie(Movie movie);**  **int deleteMovie(Integer id);**  **Movie selectMovie(Integer id);**  **// Add more methods for dynamic queries**  **}** **4. 使用MyBatis操作数据库** **在服务层或业务逻辑层，你可以使用MovieMapper接口来执行数据库操作。**  **@Service**  **public class MovieService {**  **private final MovieMapper movieMapper;**  **@Autowired**  **public MovieService(MovieMapper movieMapper) {**  **this.movieMapper = movieMapper;**  **}**  **public void addMovie(Movie movie) {**  **movieMapper.insertMovie(movie);**  **}**  **public void updateMovie(Movie movie) {**  **movieMapper.updateMovie(movie);**  **}**  **public void deleteMovie(Integer id) {**  **movieMapper.deleteMovie(id);**  **}**  **public Movie getMovieById(Integer id) {**  **return movieMapper.selectMovie(id);**  **}**  **// Add more methods for dynamic queries**  **}**  **public class User {**  **private Integer id;**  **private String username;**  **private String password;**  **private String email;**  **// getters and setters**  **}**  **<mapper namespace="com.example.mapper.UserMapper">**  **<resultMap id="UserResultMap" type="com.example.model.User">**  **<id property="id" column="id" />**  **<result property="username" column="username" />**  **<!-- Other fields -->**  **</resultMap>**      **<select id="selectUserByUsername" parameterType="java.lang.String" resultMap="UserResultMap">**  **SELECT \* FROM user WHERE username = #{username}**  **</select>**  **</mapper>**  **package com.example.mapper;**  **public interface UserMapper {**  **User selectUserByUsername(String username);**    **}**  **package com.example.mapper;**  **public interface UserMapper {**  **User selectUserByUsername(String username);**    **}**  **public class Actor {**  **private Integer id;**  **private String name;**  **private Date birthDate;**  **private String nationality;**  **// getters and setters**  **}**  **<mapper namespace="com.example.mapper.ActorMapper">**  **<resultMap id="ActorResultMap" type="com.example.model.Actor">**  **<id property="id" column="id" />**  **<result property="name" column="name" />**  **<!-- Other fields -->**  **</resultMap>**      **<select id="selectActorById" parameterType="java.lang.Integer" resultMap="ActorResultMap">**  **SELECT \* FROM actor WHERE id = #{id}**  **</select>**  **</mapper>**  **package com.example.mapper;**  **public interface ActorMapper {**  **Actor selectActorById(Integer id);**  **}**  **public class Category {**  **private Integer id;**  **private String name;**  **private String description;**  **// getters and setters**  **}**  **<mapper namespace="com.example.mapper.CategoryMapper">**  **<resultMap id="CategoryResultMap" type="com.example.model.Category">**  **<id property="id" column="id" />**  **<result property="name" column="name" />**  **<!-- Other fields -->**  **</resultMap>**        **<select id="selectCategoryById" parameterType="java.lang.Integer" resultMap="CategoryResultMap">**  **SELECT \* FROM category WHERE id = #{id}**  **</select>**  **</mapper>**  **package com.example.mapper;**  **public interface CategoryMapper {**  **Category selectCategoryById(Integer id);**    **}** |
| 1. **遇到的问题及处理记录**  1. 数据库连接问题 **问题**：无法连接到数据库。  **处理**：检查数据库服务是否正在运行，数据库的URL、用户名和密码是否正确。 3. 数据库权限问题 **问题**：数据库操作因为权限不足而失败。  **处理**：确保数据库用户具有足够的权限来执行所需的操作。 |
| **六、实验结果及分析**  1、namespace的命名必须跟某个接口同名;  2、接口中的方法与映射文件中SQL语句id -一对应;  3、完成各数据表的增、删、改(不只改一个字段 动态 SQL)的操作4、完成各数据表的查询(单表的、多表的联合动态SQL)的操作 |
| **评语：**  **分数： 教师签字： 日期： 年 月 日** |

备注：学生用黑色签字笔填写，于实验项目结束后按时完成。