实验04 Mybatis框架：注解方法

**【实验目的及要求】**

1. 理解注解的基本概念；
2. 熟悉注解的基本用法
3. 熟悉关联查询中的注解用法；
4. 要求所有回答的文本格式：**五号，宋体、1.5倍行距，保留段单元格背景。**

**【实验步骤】**

**1、单表查询（注解方式）**

（1）在IntelliJ IDEA中新建一个Maven项目，取名为ex04\_annotation。

（2）在pom.xml文件中声明3个依赖：mybatis，mysql-connector-java和junit。

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <groupId>org.example</groupId>  <artifactId>ex04\_annotation</artifactId>  <version>1.0-SNAPSHOT</version>   <dependencies>  *<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->* <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.5.9</version>  </dependency>  *<!-- https://mvnrepository.com/artifact/junit/junit -->* <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.13.2</version>  <scope>test</scope>  </dependency>  *<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->* <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.28</version>  </dependency>  </dependencies> </project> |

（3）在src\main\resources目录下新建config目录，在该目录中，新建配置文件db.properties和mybatis-config.xml，内容如下：

|  |
| --- |
| mysql.driver=com.mysql.cj.jdbc.Driver mysql.url=jdbc:mysql://localhost:3306/student?serverTimezone=UTC& characterEncoding=utf8&useUnicode=true&useSSL=false mysql.username=root mysql.password=root |
| *<?*xml version="1.0" encoding="UTF-8" *?>* <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd"*>* <configuration>  <properties resource="config/db.properties"/>  <environments default="mysql\_student">  <environment id="mysql\_student">  <transactionManager type="jdbc" />  <dataSource type="POOLED">  <property name="driver" value="${mysql.driver}" />  <property name="url" value="${mysql.url}" />  <property name="username" value="${mysql.username}" />  <property name="password" value="${mysql.password}" />  </dataSource>  </environment>  </environments> </configuration> |

（4）在src/main/java下，创建utils.StudentDBUtil，代码如下：

|  |
| --- |
| package utils;  import org.apache.ibatis.io.Resources; import org.apache.ibatis.session.SqlSession; import org.apache.ibatis.session.SqlSessionFactory; import org.apache.ibatis.session.SqlSessionFactoryBuilder;  import java.io.Reader;  public class StudentDBUtil {  private static SqlSessionFactory *sqlSessionFactory* = null;   static {  try {  Reader reader = Resources.*getResourceAsReader*("config/mybatis-config.xml");  *sqlSessionFactory* = new SqlSessionFactoryBuilder().build(reader);  } catch (Exception e) {  e.printStackTrace();  }  }  public static SqlSession getSession() {  return *sqlSessionFactory*.openSession();  } } |

（5）在src/main/java中新建包com.gduf.ex04.po，在该包中新建持久类Student，代码如下：

|  |
| --- |
| package com.gduf.ex04.po;  public class Student {  private String sno;  private String sname;  private String ssex;  private String snative;  private int mno;  // 省略了getter、setter和toString方法 } |

（6）在com.gduf.ex04.dao包下，新建接口StudentDao，并编写每个接口方法的注解，内容如下：

|  |
| --- |
| package com.gduf.ex04.dao;  import com.gduf.ex04.po.Student; import org.apache.ibatis.annotations.Delete; import org.apache.ibatis.annotations.Insert; import org.apache.ibatis.annotations.Select; import org.apache.ibatis.annotations.Update;  import java.util.List;  public interface StudentDao {  @Select("select \* from stu where sno=#{sno};")  Student findStudentBySno(String sno);  @Select("select \* from stu;")  List<Student> findStudentAll();  @Select("select \* from stu where sname like '%${sname}%';")  List<Student> findStudentBySname(String sname);  @Insert("insert into stu(sno,sname,ssex,snative,mno) values(#{sno},#{sname},#{ssex},#{snative},#{mno});")  int insertStudent(Student student);  @Update({"update stu set sno=#{sno},sname=#{sname},ssex=#{ssex},snative=#{snative},mno=#{mno}"," where sno=#{sno};"})  int updateStudent(Student student);  @Delete("delete from stu where " + "sno=#{sno};")  int deleteStudentBySno(String sno); } |

（7）在mybatis-config.xml文件中，添加<mappers>…</mappers>标签，加载StudentDao接口，内容如下：

|  |
| --- |
| <mappers>  <mapper class="com.gduf.ex04.dao.StudentDao"/> </mappers> |

（8）在src/test/java中，为StudentDao接口创建测试类com.gduf.ex04.StudentDaoTest，并为每个接口方法创建测试方法，代码如下：

|  |
| --- |
| package com.gduf.ex04;  import com.gduf.ex04.dao.StudentDao; import com.gduf.ex04.po.Student; import org.apache.ibatis.annotations.Delete; import org.apache.ibatis.annotations.Insert; import org.apache.ibatis.annotations.Select; import org.apache.ibatis.annotations.Update; import org.apache.ibatis.session.SqlSession; import org.junit.Test; import utils.StudentDBUtil;  import java.util.List;  public class StudentDaoTest {  @Test  public void findStudentBySnoTest(){  System.*out*.println("findStudentBySnoTest:");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentDao studentDao = sqlSession.getMapper(StudentDao.class);   Student student = studentDao.findStudentBySno("100000001");  System.*out*.println(student);  sqlSession.close();  }   @Test  public void findStudentAllTest(){  System.*out*.println("findStudentAllTest:");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentDao studentDao = sqlSession.getMapper(StudentDao.class);   List<Student> studentList = studentDao.findStudentAll();  System.*out*.println(studentList);  sqlSession.close();  }   @Test  public void findStudentBySnameTest(){  System.*out*.println("findStudentBySnameTest:");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentDao studentDao = sqlSession.getMapper(StudentDao.class);   List<Student> studentList = studentDao.findStudentBySname("小");  System.*out*.println(studentList);  sqlSession.close();  }   @Test  public void insertStudentTest(){  System.*out*.println("insertStudentTest:");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentDao studentDao = sqlSession.getMapper(StudentDao.class);   Student student = new Student();  student.setSno("888888888");  student.setSname("张思源");  student.setSsex("男");  student.setSnative("广东广州");  student.setMno(1);  int lines = studentDao.insertStudent(student);  sqlSession.commit();  sqlSession.close();  System.*out*.println("成功插入了" + lines + "条记录");  }   @Test  public void updateStudentTest(){  System.*out*.println("updateStudentTest:");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentDao studentDao = sqlSession.getMapper(StudentDao.class);   Student student = new Student();  student.setSno("888888888");  student.setSname("张思霞");  student.setSsex("女");  student.setSnative("广东佛山");  student.setMno(1);  int lines = studentDao.insertStudent(student);  sqlSession.commit();  sqlSession.close();  System.*out*.println("成功更新了" + lines + "条记录");  }   @Test  public void deleteStudentBySnoTest(){  System.*out*.println("deleteStudentBySnoTest:");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentDao studentDao = sqlSession.getMapper(StudentDao.class);   int lines = studentDao.deleteStudentBySno("888888888");  sqlSession.commit();  sqlSession.close();  System.*out*.println("成功删除了" + lines + "条记录");  } } |

（9）测试项目，分析和确定上述程序的有效性。

（10）参考第（5）~（8）步，实现对book表的查询、插入、更新和删除操作。

|  |
| --- |
| // 持久类设计 |
| // 接口设计 |
| // <mappers/>标记 |
| // 测试类（每个接口对应一个方法） |
| // 测试输出（数据部分） |

（11）参考第（5）~（8）步，实现对teacher表的查询、插入、更新和删除操作。

|  |
| --- |
| // 持久类设计 |
| // 接口设计 |
| // <mappers/>标记 |
| // 测试类（每个接口对应一个方法） |
| // 测试输出（数据部分） |

**2、关联映射（一对一，注解方式）**

（1）接第1个实验，在com.gduf.ex04.po包中，创建Course持久类（如果没有创建Book类，需要先创建Book类），代码如下：

|  |
| --- |
| package com.gduf.ex04.po;  public class Book {  private int id;  private String bname;  private float price;  private String pub;  // 省略了getter、setter和toString方法  } |
| package com.gduf.ex04.po;  public class Course {  private int cno;  private String cname;  private int period;  private Book book;  // 省略了getter、setter和toString方法 } |

（2）在com.gduf.ex04.dao包中，创建接口BookDao（如果已经创建，可忽略这一步），代码如下：

|  |
| --- |
| package com.gduf.ex04.dao;  import com.gduf.ex04.po.Book; import org.apache.ibatis.annotations.Select;  public interface BookDao {  @Select("select \* from book where id=#{id};")  Book findBookById(int id); } |

（3）在com.gduf.ex04.dao包中，创建接口CourseBookDao（也可以将该接口中的方法并入到CourseDao中，从而减少接口的数量），代码如下：

|  |
| --- |
| package com.gduf.ex04.dao;  import com.gduf.ex04.po.Course; import org.apache.ibatis.annotations.One; import org.apache.ibatis.annotations.Result; import org.apache.ibatis.annotations.Results; import org.apache.ibatis.annotations.Select;  public interface CourseBookDao {  @Select("select \* from course where cno=#{cno};")  @Results({@Result(column = "id",property = "book",one=@One(select = "com.gduf.ex04.dao.BookDao.findBookById"))})  Course findCourseBookByCno(int cno); } |

（4）在mybatis-config.xml中的<mappers/>标签中，加入如下配置信息：

|  |
| --- |
| <mapper class="com.gduf.ex04.dao.BookDao"/> <mapper class="com.gduf.ex04.dao.CourseBookDao"/> |

（5）在src/test/java中，为CourseBookDao接口创建测试类com.gduf.ex04.CourseBookDaoTest，并为接口方法创建测试方法，代码如下：

|  |
| --- |
| package com.gduf.ex04;  import com.gduf.ex04.dao.CourseBookDao; import com.gduf.ex04.dao.StudentDao; import com.gduf.ex04.po.Course; import org.apache.ibatis.session.SqlSession; import org.junit.Test; import utils.StudentDBUtil;  public class CourseBookDaoTest {  @Test  public void findCourseBookByCnoTest(){  System.*out*.println("findCourseBookByCnoTest:");  SqlSession sqlSession = StudentDBUtil.*getSession*();  CourseBookDao courseBookDao = sqlSession.getMapper(CourseBookDao.class);   Course course = courseBookDao.findCourseBookByCno(1);  System.*out*.println(course);  *sqlSession.close();*  } } |

（6）在student数据库创建一个idcard表，与stu表建立主/外键联系。

|  |
| --- |
| create table idcard(  id int primary key auto\_increment,  card\_no varchar(30) not null,  card\_name varchar(30) not null  ) ;  alter table stu add card\_id int;  alter table stu add constraint stu\_idcard\_id foreign key(card\_id) references idcard(id);  insert into idcard(card\_no,card\_name) values('120154199908162247','idcard');  insert into idcard(card\_no,card\_name) values('121154199911162222','idcard');  update stu set card\_id=1 where sno='100000001';  update stu set card\_id=2 where sno='100000002';  /\*  alter table stu drop card\_id;  alter table stu drop stu\_idcard\_id;  drop table idcard;  \*/ |

（7）仿照步骤（1）~（5），从数据库中查询学生及身份证信息。

|  |
| --- |
| // Idcard持久类 |
| // Student持久类（需要添加一个成员属性） |
| // 查询身份证信息的接口代码 |
| // 查询学生及身份证信息的接口代码 |
| // 在mybatis-config.xml文件中添加的配置信息 |
| // 测试代码 |
| // 测试输出（数据部分） |

**3、关联映射（一对多，注解方式）**

（1）创建持久类Sc，代码如下：

|  |
| --- |
| package com.gduf.ex04.po;  public class Sc {  private String sno;  private int cno;  private int tno;  private float participation;  private float sfinal;  private float total;  // 省略了getter、setter和toString方法 } |

（2）在持久类Student中添加一个成员属性List<Sc> scList;，添加其setter和getter，重新生成toString方法。

（3）添加ScDao接口，代码如下：

|  |
| --- |
| package com.gduf.ex04.dao;  import com.gduf.ex04.po.Sc; import org.apache.ibatis.annotations.Select;  public interface ScDao {  @Select("select \* from sc where sno=#{sno};")  List<Sc> findScBySno(String sno); } |

（4）添加StudentScDao接口，代码如下：

|  |
| --- |
| package com.gduf.ex04.dao;  import com.gduf.ex04.po.Student; import org.apache.ibatis.annotations.Many; import org.apache.ibatis.annotations.Result; import org.apache.ibatis.annotations.Results; import org.apache.ibatis.annotations.Select;  public interface StudentScDao {  @Select("select \* from stu where sno=#{sno};")  @Results({@Result(column = "sno", property = "scList",   many = @Many(select = "com.gduf.ex04.dao.ScDao.findScBySno"))})  Student findStudentScBySno(String sno); } |

（5）在mybatis-config.xml文件中添加配置信息，内容如下：

|  |
| --- |
| <mapper class="com.gduf.ex04.dao.ScDao"/> <mapper class="com.gduf.ex04.dao.StudentScDao"/> |

（6）在src/test/java中，为StudentScDao接口创建测试类com.gduf.ex04. StudentScDaoTest，并为接口方法创建测试方法，代码如下：

|  |
| --- |
| package com.gduf.ex04;  import com.gduf.ex04.dao.StudentScDao; import com.gduf.ex04.po.Student; import org.apache.ibatis.session.SqlSession; import org.junit.Test; import utils.StudentDBUtil;  public class StudentScDaoTest {  @Test  public void findStudentScBySnoTest(){  System.*out*.println("findStudentScBySnoTest:");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentScDao studentScDao = sqlSession.getMapper(StudentScDao.class);   Student student = studentScDao.findStudentScBySno("100000001");  System.*out*.println(student);  sqlSession.close();  } } |

（7）参照上述步骤（1）~（6），从数据库中查询某一门课程的所有成绩信息，并显示出来。

|  |
| --- |
| // Course类修改后的代码 |
| // 接口代码 |
| // mybatis-config.xml文件中添加的配置信息 |
| // 测试代码 |
| // 测试输出（数据部分） |

**4、关联映射（多对多，注解方式）**

（1）在Student持久类中添加属性List<Course> courseList;，添加其setter和getter，重新生成toString方法。

（2）如果没有创建持久类Course，在需要创建持久类Course。

（3）创建CourseDao接口（如CourseDao存在，则在其中添加findCourseBySno方法），内容如下：

|  |
| --- |
| package com.gduf.ex04.dao;  import com.gduf.ex04.po.Course; import org.apache.ibatis.annotations.Select;  import java.util.List;  public interface CourseDao {  @Select("select \* from course where cno in (select cno from sc where sno=#{sno});")  List<Course> findCourseBySno(String sno); } |

（4）创建StudentCourseDao接口（也可以将其方法并入到StudentDao中），内容如下：

|  |
| --- |
| package com.gduf.ex04.dao;  import com.gduf.ex04.po.Student; import org.apache.ibatis.annotations.Many; import org.apache.ibatis.annotations.Result; import org.apache.ibatis.annotations.Results; import org.apache.ibatis.annotations.Select;  import java.util.List;  public interface StudentCourseDao {  @Select("select \* from stu where sno=#{sno};")  @Results({@Result(column = "sno", property = "courseList", many = @Many(select = "com.gduf.ex04.dao.CourseDao.findCourseBySno"))})  Student findStudentCourseBySno(String sno); } |

（5）在mybatis-config.xml文件中添加配置信息，内容如下：

|  |
| --- |
| <mapper class="com.gduf.ex04.dao.CourseDao"/> <mapper class="com.gduf.ex04.dao.StudentCourseDao"/> |

（6）在src/test/java中，为StudentCourseDao接口创建测试类com.gduf.ex04. StudentCourseDaoTest，并为接口方法创建测试方法，代码如下：

|  |
| --- |
| package com.gduf.ex04;  import com.gduf.ex04.dao.StudentCourseDao; import com.gduf.ex04.po.Student; import org.apache.ibatis.session.SqlSession; import org.junit.Test; import utils.StudentDBUtil;  public class StudentCourseDaoTest {  @Test  public void findStudentCourseBySnoTest(){  System.*out*.println("findStudentCourseBySnoTest:");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentCourseDao studentCourseDao = sqlSession.getMapper(StudentCourseDao.class);   Student student = studentCourseDao.findStudentCourseBySno("100000001");  System.*out*.println(student);  sqlSession.close();  } } |

（7）参照上述步骤（1）~（6），从数据库中查询某一门课程的所有学生信息，并显示出来。

|  |
| --- |
| // Course类修改后的代码 |
| // 接口代码 |
| // mybatis-config.xml文件中添加的配置信息 |
| // 测试代码 |
| // 测试输出（数据部分） |