实验02 Mybatis框架：基本配置及应用

**【实验目的及要求】**

1. 了解Mybatis框架的基本工作原理；
2. 熟悉Mybatis框架的核心对象；
3. 掌握在Mybatis框架中，数据库单个表的访问方法；
4. 要求所有回答的文本格式：**五号，宋体、1.5倍行距，保留原来的单元格背景。**

**【实验步骤】**

**1、启动MySQL数据库服务器，执行下列脚本，创建student数据库，做好数据库访问前的准备工作。**

|  |
| --- |
| -- student数据库创建示例  create database student1 character set gbk;  use student1;  -- 设计客户端默认字符集  SET @saved\_cs\_client = @@character\_set\_client;  SET @@character\_set\_client = gbk;  -- 含指定引擎和默认字符集  create table stu  (  sno char(9) primary key,  sname varchar(30) not null,  ssex char(2) not null,  snative varchar(30),  mno int  )ENGINE=InnoDB DEFAULT CHARSET= gbk;  set names gbk;  insert into stu values('100000001','尚小云','女','广东广州',1);  insert into stu values('100000002','廖时飞','男','广东梅州',1);  insert into stu values('100000003','宋凌枫','男','湖南郴州',2);  insert into stu values('100000004','刘小纳', '女','广东佛山',2);  create table course  (  cno int primary key,  cname varchar(50) not null,  period int  )ENGINE=InnoDB DEFAULT CHARSET= gbk;  set names gbk;  insert into course values(1,'高等数学',80);  insert into course values(2,'大学英语',70);  insert into course values(3,'数据结构',70);  insert into course values(4,'数据库原理与应用',70);  create table teacher  (  tno int primary key,  tname varchar(24) not null,  tsex char(2) not null,  tel varchar(20)  )ENGINE=InnoDB DEFAULT CHARSET= gbk;  set names gbk;  insert into teacher values(1,'袁怀斌','男','13590909090');  insert into teacher values(2,'杨文远','男','13588888888');  insert into teacher values(3,'周云霞','女','13511118888');  create table sc  (  sno char(9),  cno int,  tno int,  participation float,  final float,  total float,  constraint fk\_sc\_stu foreign key(sno) references stu(sno),  constraint fk\_sc\_course foreign key(cno) references course(cno),  constraint fk\_sc\_teacher foreign key(tno) references teacher(tno),  constraint pk\_sc primary key(sno,cno,tno)  )ENGINE=InnoDB DEFAULT CHARSET= gbk;  set names gbk;  insert into sc values('100000001',1,1,80,87,85);  insert into sc values('100000001',2,2,85,75,81);  insert into sc values('100000002',1,1,78,82,80);  -- 恢复客户端默认字符集  SET @@character\_set\_client = @saved\_cs\_client; |

**2、访问student数据库**

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

（2）在pom.xml文件中声明3个依赖，mybatis（MyBatis框架）、mysql-connector-java（MySQL驱动）和junit（单元测试）。若想获得更多的依赖信息或采用不同的版本，可以在<https://mvnrepository.com>站点中，通过检索关键字获得。

|  |
| --- |
| <?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>com.ex02</groupId>  <artifactId>student</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）右击pom.xml文件，选择Maven\Reimport菜单项，启动构件下载，将上述3个构件从远程仓库中下载至本地仓库。

（4）在src\main\java目录中，新建StudentDBUtil类，用于构造会话对象。代码如下：

|  |
| --- |
| package com.ex02.student.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("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目录中新建Student类（即Java Bean），代码如下（请保持Java Bean的属性和数据表stu中字段间的一致）：

|  |
| --- |
| package com.ex02.student.po;  public class Student {  private String sno;  private String sname;  private String ssex;  private String snative;  private int mno;    // 省略了getter和setter方法  @Override  public String toString() {  return "Student{" +  "sno='" + sno + '\'' +  ", sname='" + sname + '\'' +  ", ssex='" + ssex + '\'' +  ", snative='" + snative + '\'' +  ", mno=" + mno +  '}';  }  } |

（6）在src\main\resources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="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>  <mappers>  <mapper resource="mapper/StudentMapper.xml"/>  </mappers>  </configuration> |

（7）在src\main\resources\mapper目录中，新建StudentMapper.xml文件，在文件中编写对stu表进行查询、插入、删除及更新等操作的配置信息。内容如下：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="com.ex02.student.po.Student">  <select id="selectStudentBySno" parameterType="String" resultType="com.ex02.student.po.Student">  select \* from stu where sno=#{sno};  </select>  <select id="selectStudentBySname" parameterType="String" resultType="com.ex02.student.po.Student">  select \* from stu where sname like concat('%',#{sname},'%');  </select>  <insert id="insertStudent" parameterType="com.ex02.student.po.Student">  insert into stu(sno,sname,ssex,snative,mno) values(#{sno},#{sname},#{ssex},#{snative},#{mno});  </insert>  <delete id="deleteStudent" parameterType="String">  delete from stu where sno=#{sno}  </delete>  <update id="updateStudent" parameterType="com.ex02.student.po.Student">  update stu set sname=#{sname},ssex=#{ssex},snative=#{snative},mno=#{mno} where sno=#{sno}  </update>  </mapper> |

（8）在src\test\java中，新建StudentTest类，在studentTest方法中编写代码，测试对stu表的查询、插入、删除和更新等操作。

|  |
| --- |
| package com.ex02.student;  import com.ex02.student.po.Student;  import com.ex02.student.utils.StudentDBUtil;  import org.apache.ibatis.session.SqlSession;  import org.junit.Test;  import java.util.List;  public class StudentTest {  @Test  public void studentTest(){  SqlSession sqlSession = StudentDBUtil.getSession();  int count = -1;  // 精确查询测试  Student student = sqlSession.selectOne("com.ex02.student.po.Student.selectStudentBySno", "100000001");  System.out.println(student);  // 模糊查询测试  List<Student> list = sqlSession.selectList("com.ex02.student.po.Student.selectStudentBySname", "小");  for (Student stu : list)  System.out.println(stu);  // 插入记录测试  student = new Student();  student.setSno("100000008");  student.setSname("刘伟");  student.setSsex("男");  student.setSnative("广东汕头");  student.setMno(2);  count = sqlSession.insert("com.ex02.student.po.Student.insertStudent", student);  sqlSession.commit();  System.out.println("成功插入了" + count + "条记录。");  /\*  // 删除记录测试  count = sqlSession.delete("com.ex02.student.po.Student.deleteStudent", "100000004");  sqlSession.commit();  System.out.println("成功删除了" + count + "条记录。");  // 更新记录测试  student.setSno("100000002");  student.setSname("廖凡");  student.setSsex("男");  student.setSnative("广东汕头");  student.setMno(2);  count = sqlSession.update("com.ex02.student.po.Student.updateStudent", student);  sqlSession.commit();  System.out.println("成功更新了" + count + "条记录。");  \*/  sqlSession.close();  }  } |

（9）在src\main\java目录中，定义接口StudentMapper，接口的方法与StudentMapper.xml文件中的各查询标签的id值对应。部分代码如下：

|  |
| --- |
| package com.ex02.student.mapper;  import com.ex02.student.po.Student;  public interface StudentMapper {  Student selectStudentBySno(String sno);  **// 请根据上例补全其他方法**  // ……  } |

（10）修改StudentMapper.xml映射文件中的mapper标签中namespace的值，即<mapper namespace="com.ex02.student.mapper.StudentMapper">。

（11）修改studentTest方法中的代码，通过由Mapper接口生成的代理对象访问数据库。代码如下：

|  |
| --- |
| package com.ex02.student;  import com.ex02.student.mapper.StudentMapper;  import com.ex02.student.po.Student;  import com.ex02.student.utils.StudentDBUtil;  import org.apache.ibatis.session.SqlSession;  import org.junit.Test;  import java.util.List;  public class StudentTest {  @Test  public void studentTest(){  SqlSession sqlSession = StudentDBUtil.getSession();  int count = -1;  // 生成Mapper的代理对象studentMapper  StudentMapper studentMapper = sqlSession.getMapper(StudentMapper.class);  // 精确查询测试  Student student = studentMapper.selectStudentBySno("100000001");  System.out.println(student);  // **请补全其他代码，完成对stu表的其他操作**  // ……  sqlSession.close();  }  } |

（12）重复步骤（5）~（11），依次完成数据表course、sc和teacher的访问（要求采用接口方式）。

|  |
| --- |
| course表的访问代码及映射文件： |
| 1、Course类 |
| 2、CourseMapper映射文件 |
| 3、CourseMapper接口 |
| 4、courseTest方法中的测试代码 |

|  |
| --- |
| sc表的访问代码及映射文件： |
| 1、Sc类 |
| 2、ScMapper映射文件 |
| 3、ScMapper接口 |
| 4、scTest方法中的测试代码 |

|  |
| --- |
| teacher表的访问代码及映射文件： |
| 1、Teacher类 |
| 2、TeacherMapper映射文件 |
| 3、TeacherMapper接口 |
| 4、teacherTest方法中的测试代码 |

（12）分析上述项目中的源码，请说明项目是如何将mybatis-config.xml、映射文件、接口及持久类关联在一起的。

|  |
| --- |
| 答： |