实验03 Mybatis框架：扩展配置及应用

**【实验目的及要求】**

1. 熟悉动态SQL应用；
2. 掌握数据表的关联查询；
3. 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、动态SQL应用**

（1）在实验02创建的student项目基础上，打开映射文件StudentMapper.xml，在其中增加一个<select>…</select>标签，具体如下：

|  |
| --- |
| <select id=*"selectStudent"* parameterType=*"Student"* resultType=*"Student"*>  <bind name=*"search\_sname\_key"* value=*"'%'+sname+'%'"*/>  select \* from stu  <where>  <if test=*"sno!=null and sno!=''"*>  and sno=#{sno}  </if>  <if test=*"sname!=null and sname!=''"*>  and sname like #{search\_sname\_key}  </if>  </where>  </select> |

（2）在com.ex02.student.mapper.StudentMapper接口中添加一个接口方法List<Student> selectStudent(Student student);

（3）在com.ex02.student.StudentTest类中，添加一个测试方法，增加测试id为selectStudent的查询功能，要求sno和sname分别为null或""时的查询结果。

|  |
| --- |
| 参考测试代码：  @Test public void selectStudentTest(){  System.*out*.println("selectStudentTest");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentMapper studentMapper = sqlSession.getMapper(StudentMapper.class);  Student student = new Student();  student.setSno("");  student.setSname("小");  List<Student> list = studentMapper.selectStudent(student);  for (Student stu : list)  System.*out*.println(stu);  sqlSession.close(); } |

（4）修改上述查询功能，要求：若sno不为null或""，则以sno为精确查询条件在stu表中查询记录；若sno为null或""，则以sname为模糊条件在stu表中查询记录；否则，查询出snative不为null的所有记录。请给出StudentMapper.xml映射文件中，相关SQL的配置信息及程序运行结果。

|  |
| --- |
| 配置信息： |
| 运行结果（只包括查询结果）： |

（5）给出若干个姓名构成的集合（如{'尚小云','刘伟','宋凌枫'}），从stu表中查询出姓名在集合中的记录。SQL脚本如下：

|  |
| --- |
| <select id=*"selectStudentBySnames"* parameterType=*"List"* resultType=*"Student"*>  select \* from stu where sname in  <foreach item=*"sname"* index=*"index"* collection=*"list"* open=*"("* separator=*","* close=*")"*>  #{sname}  </foreach>  </select> |

（6）在com.ex02.student.mapper.StudentMapper接口中添加一个接口方法List<Student> selectStudentBySnames(List<String> list);

（7）在com.ex02.student.StudentTest类中，添加一个测试方法，增加测试id为selectStudentBySnames的查询功能。测试代码如下：

|  |
| --- |
| @Test public void selectStudentBySnamesTest(){  System.*out*.println("selectStudentBySnamesTest");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentMapper studentMapper = sqlSession.getMapper(StudentMapper.class);  List<String> list = new ArrayList<String>();  list.add("尚小云");  list.add("刘伟");  list.add("宋凌枫");  List<Student> stuList = studentMapper.selectStudentBySnames(list);  for (Student stu : stuList)  System.*out*.println(stu); } |

（8）按照第（5）~（7）的方法，从Course表中查询出课程名为“高等数学”、“大学英语”、“数据库原理与应用”等课程的记录。

|  |
| --- |
| SQL配置信息： |
| 测试方法： |
| 运行结果（只需含查询数据）： |

**2、关联映射（一对一映射）**

（1）接第1个实验，在student数据库中创建book表并与course表建立联系，代码如下：

|  |
| --- |
| /\* 创建book表 \*/  create table book(  id int primary key auto\_increment,  bname varchar(30) not null,  price float,  pub varchar(30)  ) ;  /\* 在course表中添加外键bookid \*/  alter table course add bookid int;  alter table course add constraint course\_book\_id foreign key(bookid) references book(id);  /\* 在book表中插入记录 \*/  insert into book(bname,price,pub) values('高等数学（上）',56.5,'高等教育出版社');  insert into book(bname,price,pub) values('大学英语教程',75,'上海外语出版社');  insert into book(bname,price,pub) values('数据结构教程（第二版）',49,'清华大学出版社');  insert into book(bname,price,pub) values('数据库原理与应用教程（第三版）',45.5,'清华大学出版社');  /\* 在course表中设置外键值 \*/  update course set bookid=1 where cno=1;  update course set bookid=2 where cno=2;  update course set bookid=3 where cno=3;  update course set bookid=4 where cno=4;  /\*  alter table course drop foreign key course\_book\_id;  alter table course drop column bookid;  drop table book;  \*/ |

（2）在com.ex02.student.po包中，建立两个Java Bean（即Book和Course），代码如下：

|  |
| --- |
| package com.ex02.student.po;  public class Book {  private int id;  private String bname;  private float price;  private String pub;   // 省略了getter和setter方法 … …   @Override  public String toString() {  return "Book{" +  "id=" + id +  ", bname='" + bname + '\'' +  ", price=" + price +  ", pub='" + pub + '\'' +  '}';  } } |
| package com.ex02.student.po;  public class Course {  private int cno;  private String cname;  private int period;  private Book book;   //省略了getter和setter方法 … …  @Override  public String toString() {  return "Course{" +  "cno=" + cno +  ", cname='" + cname + '\'' +  ", period=" + period +  ", book=" + book +  '}';  } } |

（3）在resources/mapper文件夹中，创建映射文件BookMapper.xml，实现对book表的查询。在com.ex02.student.mapper包中创建接口BookMapper，并添加Book selectBookById(int id);。配置信息及接口代码如下：

|  |
| --- |
| *<?*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.mapper.BookMapper">  <select id="selectBookById" parameterType="Integer" resultType="Book">  select *\** from book where id=#{id};  </select> </mapper> |
| package com.ex02.student.mapper; import com.ex02.student.po.Book; public interface BookMapper {  Book selectBookById(int id); } |

（4）在resources/mapper文件夹中，创建映射文件CourseMapper.xml，实现对course表的查询。配置信息如下：

|  |
| --- |
| *<?*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.mapper.CourseMapper">  <select id="selectCourseBookByCno" parameterType="Integer" resultMap="courseBookMap">  select *\** from course where cno=#{cno};  </select>  <resultMap type="Course" id="courseBookMap">  <id property="cno" column="cno"/>  <result property="cname" column="cname"/>  <result property="period" column="period"/>  <association property="book" column="bookid" javaType="Book" select=" com.ex02.student.mapper.BookMapper.selectBookById"/>  </resultMap> </mapper> |

（5）在testmybatis-config.xml文件结束位置，加载映射文件BookMapper.xml（如果CourseMapper.xml没有加载，也需要加载）。

（6）在test\java目录的com.ex02.student包中，创建CourseTest类，参考下列代码编写测试方法。

|  |
| --- |
| package com.ex02.student;  import com.ex02.student.mapper.CourseMapper; import com.ex02.student.po.Course; import com.ex02.student.utils.StudentDBUtil; import org.apache.ibatis.session.SqlSession; import org.junit.Test;  public class CourseTest {  @Test  public void courseBookTest(){  System.*out*.println("courseTest：");  SqlSession sqlSession = StudentDBUtil.*getSession*();  CourseMapper courseMapper = sqlSession.getMapper(CourseMapper.class);  Course course = courseMapper.selectCourseBookByCno(1);  System.*out*.println(course);  } } |

（7）在上述嵌套查询方法的基础上，修改代码，以嵌套结果方式实现相同的功能。要求给出修改或新增的代码。

|  |
| --- |
| 映射文件CourseMapper.xml中新增的配置信息： |
| 新增的接口方法： |
| 测试方法： |

（8）结合多表查询机制，不采用<association>…</association>标签，使用<resultMap>…</resultMap>标签，实现po类属性与查询记录集之间的映射。要求给出修改或新增的代码。

|  |
| --- |
| 映射文件CourseMapper.xml中新增的配置信息： |
| 新增的接口方法： |
| 测试方法： |

**3、关联映射（一对多映射）**

（1）接第2个实验，在com.ex02.student.po包中，新增一个Score类，并修改Student类。代码如下：

|  |
| --- |
| package com.ex02.student.po;  public class Score {  private String sno;  private int cno;  private int tno;  private float participation;  private float sfinal;  private float total;   // 省略了getter和setter方法  @Override  public String toString() {  return "Score{" +  "sno='" + sno + '\'' +  ", cno=" + cno +  ", tno=" + tno +  ", participation=" + participation +  ", sfinal=" + sfinal +  ", total=" + total +  '}';  } } |
| package com.ex02.student.po;  import java.util.List;  public class Student {  private String sno;  private String sname;  private String saex;  private String snative;  private int mno;  private List<Score> listScore;   // 省略了getter和setter方法  public String toString() {  return "Student{" +  "sno='" + sno + '\'' +  ", sname='" + sname + '\'' +  ", saex='" + saex + '\'' +  ", snative='" + snative + '\'' +  ", mno=" + mno +  ", listScore=" + listScore +  '}';  } } |

（2）在resources/mapper目录中创建映射文件ScoreMapper.xml，实现对sc表的查询。配置信息如下：

|  |
| --- |
| *<?*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.mapper.ScoreMapper">  <select id="selectScoreBySno" parameterType="String" resultMap="selectScoreBySnoMap">  select *\** from sc where sno=#{sno};  </select>  <resultMap type="Score" id="selectScoreBySnoMap">  <id property="cno" column="cno"/>  <id property="sno" column="sno"/>  <id property="tno" column="tno"/>  <result property="participation" column="participation"/>  <result property="sfinal" column="final"/>  <result property="total" column="total"/>  </resultMap> </mapper> |

（3）在映射文件StudentMapper.xml中，增加下列配置信息，以嵌套查询方式实现关联查询。在StudentMapper接口中添加方法。配置信息及接口代码如下：

|  |
| --- |
| <select id=*"selectStudentScoresBySno"* parameterType=*"String"* resultMap=*"studentScoreMap"*>  select \* from stu where sno=#{sno};  </select>  <resultMap type=*"Student"* id=*"studentScoreMap"*>  <id property=*"sno"* column=*"sno"*/>  <result property=*"sname"* column=*"sname"*/>  <result property=*"ssex"* column=*"ssex"*/>  <result property=*"snative"* column=*"snative"*/>  <result property=*"mno"* column=*"mno"*/>  <collection property=*"listScore"* column=*"sno"* ofType=*"Score"* select=*"com.ex02.student.mapper.ScoreMapper.selectScoreBySno"*/>  </resultMap> |
| package com.ex02.student.mapper;  import com.ex02.student.po.Student;  import java.util.List;  public interface StudentMapper {  Student selectStudentBySno(String sno);  List<Student> selectStudentBySname(String sname);  int insertStudent(Student student);  int deleteStudent(String sno);  int updateStudent(Student student);  List<Student> selectStudent(Student student);  List<Student> selectStudentBySnames(List<String> list);  Student selectStudentScoresBySno(String sno); } |

（4）在mybatis-config.xml文件结束位置，增加<mapper>…</mapper>，加载第（2）~（3）步的两个映射文件。

（5）在test\java目录下的com.ex02.student.StudentTest类中添加测试方法，编写如下测试代码。

|  |
| --- |
| @Test public void selectStudentScoresBySnoTest(){  System.*out*.println("selectStudentScoresBySnoTest");  SqlSession sqlSession = StudentDBUtil.*getSession*();  StudentMapper studentMapper = sqlSession.getMapper(StudentMapper.class);  Student student = studentMapper.selectStudentScoresBySno("100000001");  System.*out*.println(student); } |

（6）在上述嵌套查询方法的基础上，修改代码，以嵌套结果方式实现相同的功能。要求给出修改或新增的代码。

|  |
| --- |
| 修改或新增代码及配置信息： |
|  |
|  |
|  |

**4、关联映射（多对多映射）**

（1）接第3个实验，在com.ex02.student.po包中，修改Course和Student类。代码如下：

|  |
| --- |
| package com.ex02.student.po;  import java.util.List;  public class Student {  private String sno;  private String sname;  private String saex;  private String snative;  private int mno;  private List<Score> listScore;  private List<Course> courseList;   // 省略了getter和setter方法  @Override  public String toString() {  return "Student{" +  "sno='" + sno + '\'' +  ", sname='" + sname + '\'' +  ", saex='" + saex + '\'' +  ", snative='" + snative + '\'' +  ", mno=" + mno +  ", listScore=" + listScore +  ", courseList=" + courseList +  '}';  } } |
| package com.ex02.student.po;  import java.util.List;  public class Course {  private int cno;  private String cname;  private int period;  private Book book;  private List<Student> studentList;   // 省略了getter和setter方法  @Override  public String toString() {  return "Course{" +  "cno=" + cno +  ", cname='" + cname + '\'' +  ", period=" + period +  ", book=" + book +  ", studentList=" + studentList +  '}';  } } |

（2）在映射文件CourseMapper.xml中，增加如下配置信息，以嵌套查询方式实现关联查询。在CourseMapper中添加Course selectCourseStudentsByCno(int cno);方法。

|  |
| --- |
| <select id="selectCourseStudentsByCno" parameterType="Integer" resultMap="courseStudentMap">  select *\** from course where cno=#{cno}; </select>  <resultMap type="Course" id="courseStudentMap">  <id property="cno" column="cno"/>  <result property="cname" column="cname"/>  <result property="period" column="period"/>  <collection property="studentList" column="cno" ofType="Student" select="com.ex02.student.mapper.StudentMapper.selectStudentByCno"/> </resultMap> |
| package com.ex02.student.mapper; import com.ex02.student.po.Course; public interface CourseMapper {  Course selectCourseBookByCno(int cno);  Course selectCourseStudentsByCno(int cno); } |

（3）在映射文件StudentMapper.xml中，增加如下配置信息，以嵌套查询方式实现关联查询。

|  |
| --- |
| <select id=*"selectStudentByCno"* parameterType=*"Integer"* resultType=*"Student"*>  select \* from stu where sno in(  select sno from sc where cno=#{cno}  )  </select> |

（4）在test\java目录下的com.ex02.student.CourseTest类中添加测试方法，编写如下测试代码。

|  |
| --- |
| @Test public void courseStudentTest(){  System.*out*.println("courseStudentTest：");  SqlSession sqlSession = StudentDBUtil.*getSession*();  CourseMapper courseMapper = sqlSession.getMapper(CourseMapper.class);  Course course = courseMapper.selectCourseStudentsByCno(1);  System.*out*.println(course); } |

（5）执行courseStudentTest()测试方法，给出控制台的运行结果：

|  |
| --- |
| 控制台的运行结果： |

（6）在上述嵌套查询方法的基础上，修改代码，以嵌套结果方式实现相同的功能。要求给出修改或新增的代码。

|  |
| --- |
| 新增或修改的代码及配置信息： |
|  |
|  |
|  |