实验07 Spring框架：框架整合及事务管理

**【实验目的及要求】**

1. 掌握Spring和MyBatis框架整合思路；
2. 掌握Spring声明式事务管理；
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、访问student数据库**

（1）在IDEA中新建一个Maven项目，取名为ex09\_db。

（2）在项目中增加Spring框架核心依赖及MySQL数据库连接驱动依赖。

|  |
| --- |
| <dependencies> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>5.3.22</version> </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>5.3.14</version>  </dependency> <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.5.9</version>  </dependency>  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>2.0.3</version>  </dependency>  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>1.2.9</version>  </dependency>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.28</version>  </dependency>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.13.2</version>  <scope>test</scope>  </dependency> </dependencies> |

（3）在main/resources目录下新建文件夹config，在config中新建配置文件beans.xml，依次完成数据源、MyBatis会话工厂、映射器接口扫描及注解扫描配置。

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  **xmlns:context="http://www.springframework.org/schema/context"**  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.3.xsd   **http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-4.3.xsd**">  *<!-- 数据库连接信息： （1）数据库驱动类型：com.mysql.cj.jdbc.Driver （2）URL：jdbc:mysql://localhost/student?serverTimezone=UTC&   characterEncoding=utf8&useUnicode=true&useSSL=false （3）用户名：root （4）密码：888888 -->* |
| *// 配置数据源* |
| *// MyBatis会话工厂* |
| *//* 映射器接口扫描配置 |
| </beans> |

（4）在com.javaee.ex09\_db.po包中新建实体类Student，代码如下：

|  |
| --- |
| package com.javaee.ex09\_db.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 } |

（5）在com.javaee.ex09\_db.dao中新增接口StudentDao，代码如下（实际应用中，接口中方法的多少应视具体情况来定）：

|  |
| --- |
| package com.javaee.ex09\_db.dao; import com.javaee.ex09\_db.po.Student; import java.util.List;  public interface StudentDao {  Student selectStudentBySno(String sno);  List<Student> selectStudentByName(String sname);  int updateStudent(Student student);  int deleteStudentBySno(String sno);  //int deleteStudentBySname(String sname);  int insertStudent(Student student); } |

（6）在resources/com/javaee/ex09\_db/dao目录中，添加映射文件StudentDao.xml（注意映射文件的路径和映射器接口的包路径一致）。

|  |
| --- |
| // StudentDao映射器接口对应的映射文件内容 |

（7）在test/java目录中创建测试类com.javaee.ex09\_db.StudentDaoTest，给每个StudentDao接口方法编写一个测试方法。

|  |
| --- |
| package com.javaee.ex09\_db;  import com.javaee.ex09\_db.dao.StudentDao; import com.javaee.ex09\_db.dao.impl.StudentDaoImpl; import com.javaee.ex09\_db.po.Student; import org.junit.Before; import org.junit.Test; import org.springframework.context.ApplicationContext; import org.springframework.context.support.ClassPathXmlApplicationContext;  import java.util.List;  public class StudentDaoTest {  private ApplicationContext applicationContext;  private StudentDao studentDao;  @Before  public void ready(){  applicationContext =  new ClassPathXmlApplicationContext("config/beans.xml");  studentDao = applicationContext.getBean(StudentDao.class);  }  @Test  public void queryTest(){  insertStudentTest();  selectStudentBySnoTest();  updateStudentTest();  selectStudentByNameTest();  deleteStudentTest();  }  private void insertStudentTest(){  Student student = new Student();  student.setSno("18181818");  student.setSname("张晓晓");  student.setSsex("男");  student.setSnative("广东广州");  student.setMno(1);  int res = studentDao.insertStudent(student);  System.*out*.println("插入了" + res + "条记录");  }  private void selectStudentBySnoTest(){  Student student = studentDao.selectStudentBySno("18181818");  System.*out*.println(student);  }  private void selectStudentByNameTest(){  List<Student> studentList = studentDao.selectStudentBySname("小");  System.*out*.println(studentList);  }  private void updateStudentTest(){  Student student = new Student();  student.setSno("18181818");  student.setSname("张大大");  student.setSsex("女");  int res = studentDao.updateStudent(student);  System.*out*.println("更新了" + res + "条记录");  }  private void deleteStudentTest(){  int res = studentDao.deleteStudentBySno("18181818");  System.*out*.println("删除了" + res + "条记录");  } } |

（8）运行程序，给出程序的输出如图7.1所示。

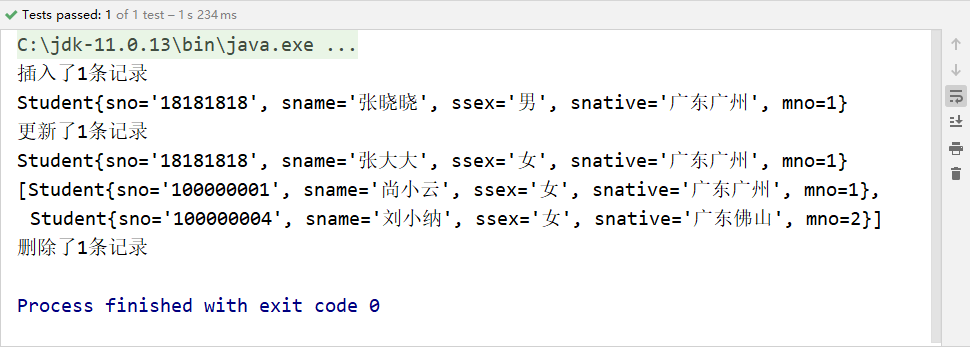


图7.1 queryTest方法的运行结果

（9）重复步骤（4）~（8），依次完成数据表course的插入、查询、更新及删除操作。

|  |
| --- |
| // 实体类 |
| // 数据库访问接口 |
| // 接口实现 |
| // 测试代码 |
| // 输出截图 |

**2、基于xml的声明式事务管理**

（1）接第1个实验，在StudentDao接口中添加int updateMnoBySno(String sno, int mno);方法，并在StudentDao.xml映射文件中添加对应的SQL代码，更新学号为sno的学生mno。

|  |
| --- |
| // 仅列出<update/>元素的代码 |

（2）在Student数据库中，执行SQL语句“create table history(id int primary key auto\_increment, info varchar(256) not null);”，创建一个数据表history，用于记录学生mno字段的修改。

（3）创建一个实体类com.javaee.ex09\_db.po.HistoryInfo，内容如下：

|  |
| --- |
| package com.javaee.ex09\_db.po;  public class HistoryInfo {  private int id;  private String info;   // 省略了getter、setter和toString方法  } |

（4）定义数据库访问接口com.javaee.ex09\_db.dao.HistoryInfoDao，内容如下：

|  |
| --- |
| package com.javaee.ex09\_db.dao; import com.javaee.ex09\_db.po.HistoryInfo; import java.util.List;  public interface HistoryInfoDao {  int insertHistoryInfo(String info); } |

（5）在resources/com/javaee/ex09\_db/dao目录中，创建HistoryInfoDao接口的映射文件HistoryInfoDao.xml，实现记录的插入操作。

|  |
| --- |
| // 仅列出<insert/>元素的代码 |

1. 创建业务类com.javaee.ex09\_db.service.StudentTransService，代码如下（注意，在业务前添加注解，在配置文件中的配置注解扫描）。

|  |
| --- |
| package com.javaee.ex09\_db.service; public class StudentTransService {  private HistoryInfoDao historyInfoDao;  private StudentDao studentDao;  public void trans(String sno, int newMno) {  String str = "学号为" + sno + "的同学转到了" + newMno + "专业";  historyInfoDao.insertHistoryInfo(str);  int i = 1 / 0;  studentDao.updateMnoBySno(sno, newMno);  } } |

（7）在test/java目录中，创建测试类com.javaee.ex09\_db. StudentTransServiceTest，内容如下：

|  |
| --- |
| package com.javaee.ex09\_db; // 省略了import语句  public class StudentTransServiceTest {  private StudentTransService studentTransService;  @Before  public void ready(){  ApplicationContext applicationContext =  new ClassPathXmlApplicationContext("config/beans.xml");  studentTransService =   applicationContext.getBean(StudentTransService.class);  }  @Test  public void queryTest(){  studentTransService.trans("100000001", 2);  } } |

（8）运行queryTest方法，给出测试输出结果。

|  |
| --- |
| // 测试输出结果 |

**说明**：由测试输出结果可知，程序执行出现异常，同学们查询stu和history表，会发现history表中多了一记录，而stu表中的mno未更新。显然，历史信息记录与mno修改出现了不一致。

（9）在pom.xml文件中，添加声明式事务配置相关的依赖信息。

|  |
| --- |
| <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-tx</artifactId>  <version>5.3.22</version> </dependency> <dependency>  <groupId>aopalliance</groupId>  <artifactId>aopalliance</artifactId>  <version>1.0</version> </dependency> <dependency>  <groupId>org.aspectj</groupId>  <artifactId>aspectjweaver</artifactId>  <version>1.9.9.1</version> </dependency> |

（10）在beans.xml文件中添加基于xml方式的事务配置信息，实现对StudentTransService 类中方法的事务增强。要求StudentTransService 类的trans()方法以事务方式运行，即要么两个查询都做，要么都不做。

|  |
| --- |
| // 仅列出事务配置相关的代码 |

（11）再次运行StudentTransServiceTest 类的queryTest()方法，给出运行结果的截图。

|  |
| --- |
|  |

**3、基于注解的声明式事务管理**

（1）接第2个实验，将基于XML配置文件方式改为注解方式。

（2）StudentTransService类中添加声明式事务管理的注解信息。

|  |
| --- |
|  |

（3）修改beans.xml配置文件，使其满足基于注解的声明式事务管理的要求。

|  |
| --- |
|  |

（4）运行StudentTransServiceTest类的queryTest()方法，给出运行结果的截图。

|  |
| --- |
|  |