第三章习题答案

1.填空题

（1）一对一、一对多、多对多

（2）一

（3）column

（4）<id>

（5）一对多

2.选择题

（1）D

（2）C

（3）A

（4）B

（5）B

3.思考题

（1）

相比于一对一，一对多的表关系在开发中更为常见，处理过程也略微复杂。通常情况下，如果使用MyBatis处理一对多的关系，需要在MyBatis的映射文件中添加<collection>元素。

（2）

通常情况下，多对多表关系要转化为一对多的形式进行处理，这种转化是通过一张中间表来实现的。在使用MyBatis处理多对多关系时，需要先将多对多关系转化为一对多关系，然后使用<collection>元素完成映射。

4.编程题

Ⅰ.创建Goods实体类

1. package com.qfedu.pojo;
2. import java.util.Date;
3. public class Goods {
4. private Integer id;
5. private String name;
6. private double price;
7. private Date pubdate;
8. private String typeName;
9. private String intro;
10. private String picture;
11. private Integer flag;
12. private Integer star;
13. public Goods() {
14. super();
15. // TODO Auto-generated constructor stub
16. }
17. public Goods(Integer id, String name, double price, Date pubdate, String typeName, String intro, String picture,
18. Integer flag, Integer star) {
19. super();
20. this.id = id;
21. this.name = name;
22. this.price = price;
23. this.pubdate = pubdate;
24. this.typeName = typeName;
25. this.intro = intro;
26. this.picture = picture;
27. this.flag = flag;
28. this.star = star;
29. }
30. public Integer getId() {
31. return id;
32. }
33. public void setId(Integer id) {
34. this.id = id;
35. }
36. public String getName() {
37. return name;
38. }
39. public void setName(String name) {
40. this.name = name;
41. }
42. public double getPrice() {
43. return price;
44. }
45. public void setPrice(double price) {
46. this.price = price;
47. }
48. public Date getPubdate() {
49. return pubdate;
50. }
51. public void setPubdate(Date pubdate) {
52. this.pubdate = pubdate;
53. }
54. public String getTypeName() {
55. return typeName;
56. }
57. public void setTypeName(String typeName) {
58. this.typeName = typeName;
59. }
60. public String getIntro() {
61. return intro;
62. }
63. public void setIntro(String intro) {
64. this.intro = intro;
65. }
66. public String getPicture() {
67. return picture;
68. }
69. public void setPicture(String picture) {
70. this.picture = picture;
71. }
72. public Integer getFlag() {
73. return flag;
74. }
75. public void setFlag(Integer flag) {
76. this.flag = flag;
77. }
78. public Integer getStar() {
79. return star;
80. }
81. public void setStar(Integer star) {
82. this.star = star;
83. }
84. @Override
85. public String toString() {
86. return "Goods [id=" + id + ", name=" + name + ", price=" + price + ", pubdate=" + pubdate + ", typeName="
87. + typeName + ", intro=" + intro + ", picture=" + picture + ", flag=" + flag + ", star=" + star + "]";
88. }
89. }

Ⅱ. OrderGoods.java

1. package com.qfedu.pojo;
2. public class OrderGoods {
3. private Integer id;
4. private Integer orderId;
5. private Integer goodsId;
6. public OrderGoods() {
7. super();
8. // TODO Auto-generated constructor stub
9. }
10. public OrderGoods(Integer id, Integer orderId, Integer goodsId) {
11. super();
12. this.id = id;
13. this.orderId = orderId;
14. this.goodsId = goodsId;
15. }
16. public Integer getId() {
17. return id;
18. }
19. public void setId(Integer id) {
20. this.id = id;
21. }
22. public Integer getOrderId() {
23. return orderId;
24. }
25. public void setOrderId(Integer orderId) {
26. this.orderId = orderId;
27. }
28. public Integer getGoodsId() {
29. return goodsId;
30. }
31. public void setGoodsId(Integer goodsId) {
32. this.goodsId = goodsId;
33. }
34. @Override
35. public String toString() {
36. return "OrderGoods [id=" + id + ", orderId=" + orderId + ", goodsId=" + goodsId + "]";
37. }
38. }

Ⅲ. OrderGoodsDao.java

1. package com.qfedu.dao;
2. import java.util.List;
3. import com.qfedu.pojo.Goods;
4. public interface OrderGoodsDao {
5. List<Goods> selectByOid(Integer oid);
6. }

Ⅳ. OrderGoodsMapper.xml

1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "../mybatis-3-mapper.dtd" >
3. <mapper namespace="com.qfedu.dao.OrderGoodsDao">
4. <select id="selectByOid" resultType="com.qfedu.pojo.Goods">
5. select t2.\* from (select o1.\*,og1.goodsId,og1.orderId og from t\_order o1 left join order\_goods og1 on o1.id = og1.orderid where o1.id =#{oid}) t1
6. left join(select og2.orderId,og2.goodsId ,g.\* from order\_goods og2 left join t\_goods g on og2.id = g.id) t2 on t1.id = t2.orderId and t1.goodsId = t2.goodsId
7. </select>
8. </mapper>

Ⅴ.application.xml

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xmlns:aop="http://www.springframework.org/schema/aop"
6. xmlns:tx="http://www.springframework.org/schema/tx"
7. xsi:schemaLocation="http://www.springframework.org/schema/beans
8. http://www.springframework.org/schema/beans/spring-beans.xsd
9. http://www.springframework.org/schema/context
10. http://www.springframework.org/schema/context/spring-context.xsd
11. http://www.springframework.org/schema/aop
12. http://www.springframework.org/schema/aop/spring-aop.xsd
13. http://www.springframework.org/schema/tx
14. http://www.springframework.org/schema/tx/spring-tx.xsd">
15. <context:component-scan base-package="com.qfedu.\*" />
16. <!-- 引入外部 properties 文件 -->
17. <context:property-placeholder location="classpath:jdbc.properties"/>
18. <!-- 注册数据源 -->
19. <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
20. <property name="driverClass" value="com.mysql.jdbc.Driver"/>
21. <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/chapter03?characterEncoding=UTF-8"/>
22. <property name="user" value="root"/>
23. <property name="password" value="root"/>
24. </bean>
25. <!--3 会话工厂bean sqlSessionFactoryBean -->
26. <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
27. <!-- 数据源 -->
28. <property name="dataSource" ref="dataSource"></property>
29. <!-- <property name="configLocation" value="classpath:mybatis-config.xml"></property> -->
30. <!-- sql映射文件路径 -->
31. <property name="mapperLocations" value="classpath\*:com/qfedu/dao/impl/\*Mapper.xml"></property>
32. </bean>
33. <!-- 自动扫描对象关系映射 -->
34. <!--创建dao-->
35. <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
36. <!--指定会话工厂，如果当前上下文中只定义了一个则该属性可省去 -->
37. <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"></property>
38. <!-- 指定要自动扫描接口的基础包，实现接口 -->
39. <property name="basePackage" value="com.qfedu.dao"></property>
40. </bean>
41. <bean id="txManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
42. <property name="dataSource" ref="dataSource"/>
43. </bean>
44. <tx:annotation-driven transaction-manager="txManager"/>
45. </beans>

Ⅵ.测试类

1. package com.qfedu.test;
2. import java.util.List;
3. import org.junit.Test;
4. import org.junit.runner.RunWith;
5. import org.springframework.beans.factory.annotation.Autowired;
6. import org.springframework.test.context.ContextConfiguration;
7. import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
8. import com.qfedu.dao.OrderGoodsDao;
9. import com.qfedu.pojo.Goods;
10. @RunWith(SpringJUnit4ClassRunner.class)
11. @ContextConfiguration(locations = { "classpath:application.xml" })
12. public class TestUtil {
13. @Autowired
14. private OrderGoodsDao od;
15. @Test
16. public void test1() {
17. List<Goods> list = od.selectByOid(1);
18. for (Goods goods : list) {
19. System.out.println(goods);
20. System.out.println("======================");
21. }
22. }
23. }