　上一篇博文[MyBatis学习总结(一)——MyBatis快速入门](http://www.cnblogs.com/xdp-gacl/p/4261895.html)中我们讲了如何使用Mybatis查询users表中的数据，算是对MyBatis有一个初步的入门了，今天讲解一下如何使用MyBatis对users表执行CRUD操作。本文中使用到的测试环境是上一篇博文中的测试环境。

**一、使用MyBatis对表执行CRUD操作——基于XML的实现**

1、定义sql映射xml文件

　　userMapper.xml文件的内容如下：

[复制代码](javascript:void(0);)

1 <?xml version="1.0" encoding="UTF-8" ?>

2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

3 <!-- 为这个mapper指定一个唯一的namespace，namespace的值习惯上设置成包名+sql映射文件名，这样就能够保证namespace的值是唯一的

4 例如namespace="me.gacl.mapping.userMapper"就是me.gacl.mapping(包名)+userMapper(userMapper.xml文件去除后缀)

5 -->

6 <mapper namespace="me.gacl.mapping.userMapper">

7 <!-- 在select标签中编写查询的SQL语句， 设置select标签的id属性为getUser，id属性值必须是唯一的，不能够重复

8 使用parameterType属性指明查询时使用的参数类型，resultType属性指明查询返回的结果集类型

9 resultType="me.gacl.domain.User"就表示将查询结果封装成一个User类的对象返回

10 User类就是users表所对应的实体类

11 -->

12 <!--

13 根据id查询得到一个user对象

14 -->

15 <select id="getUser" parameterType="int"

16 resultType="me.gacl.domain.User">

17 select \* from users where id=#{id}

18 </select>

19

20 <!-- 创建用户(Create) -->

21 <insert id="addUser" parameterType="me.gacl.domain.User">

22 insert into users(name,age) values(#{name},#{age})

23 </insert>

24

25 <!-- 删除用户(Remove) -->

26 <delete id="deleteUser" parameterType="int">

27 delete from users where id=#{id}

28 </delete>

29

30 <!-- 修改用户(Update) -->

31 <update id="updateUser" parameterType="me.gacl.domain.User">

32 update users set name=#{name},age=#{age} where id=#{id}

33 </update>

34

35 <!-- 查询全部用户-->

36 <select id="getAllUsers" resultType="me.gacl.domain.User">

37 select \* from users

38 </select>

39

40 </mapper>

[复制代码](javascript:void(0);)

　　单元测试类代码如下：

[复制代码](javascript:void(0);)

1 package me.gacl.test;

2

3 import java.util.List;

4 import me.gacl.domain.User;

5 import me.gacl.util.MyBatisUtil;

6 import org.apache.ibatis.session.SqlSession;

7 import org.junit.Test;

8

9 public class TestCRUDByXmlMapper {

10

11 @Test

12 public void testAdd(){

13 //SqlSession sqlSession = MyBatisUtil.getSqlSession(false);

14 SqlSession sqlSession = MyBatisUtil.getSqlSession(true);

15 /\*\*

16 \* 映射sql的标识字符串，

17 \* me.gacl.mapping.userMapper是userMapper.xml文件中mapper标签的namespace属性的值，

18 \* addUser是insert标签的id属性值，通过insert标签的id属性值就可以找到要执行的SQL

19 \*/

20 String statement = "me.gacl.mapping.userMapper.addUser";//映射sql的标识字符串

21 User user = new User();

22 user.setName("用户孤傲苍狼");

23 user.setAge(20);

24 //执行插入操作

25 int retResult = sqlSession.insert(statement,user);

26 //手动提交事务

27 //sqlSession.commit();

28 //使用SqlSession执行完SQL之后需要关闭SqlSession

29 sqlSession.close();

30 System.out.println(retResult);

31 }

32

33 @Test

34 public void testUpdate(){

35 SqlSession sqlSession = MyBatisUtil.getSqlSession(true);

36 /\*\*

37 \* 映射sql的标识字符串，

38 \* me.gacl.mapping.userMapper是userMapper.xml文件中mapper标签的namespace属性的值，

39 \* updateUser是update标签的id属性值，通过update标签的id属性值就可以找到要执行的SQL

40 \*/

41 String statement = "me.gacl.mapping.userMapper.updateUser";//映射sql的标识字符串

42 User user = new User();

43 user.setId(3);

44 user.setName("孤傲苍狼");

45 user.setAge(25);

46 //执行修改操作

47 int retResult = sqlSession.update(statement,user);

48 //使用SqlSession执行完SQL之后需要关闭SqlSession

49 sqlSession.close();

50 System.out.println(retResult);

51 }

52

53 @Test

54 public void testDelete(){

55 SqlSession sqlSession = MyBatisUtil.getSqlSession(true);

56 /\*\*

57 \* 映射sql的标识字符串，

58 \* me.gacl.mapping.userMapper是userMapper.xml文件中mapper标签的namespace属性的值，

59 \* deleteUser是delete标签的id属性值，通过delete标签的id属性值就可以找到要执行的SQL

60 \*/

61 String statement = "me.gacl.mapping.userMapper.deleteUser";//映射sql的标识字符串

62 //执行删除操作

63 int retResult = sqlSession.delete(statement,5);

64 //使用SqlSession执行完SQL之后需要关闭SqlSession

65 sqlSession.close();

66 System.out.println(retResult);

67 }

68

69 @Test

70 public void testGetAll(){

71 SqlSession sqlSession = MyBatisUtil.getSqlSession();

72 /\*\*

73 \* 映射sql的标识字符串，

74 \* me.gacl.mapping.userMapper是userMapper.xml文件中mapper标签的namespace属性的值，

75 \* getAllUsers是select标签的id属性值，通过select标签的id属性值就可以找到要执行的SQL

76 \*/

77 String statement = "me.gacl.mapping.userMapper.getAllUsers";//映射sql的标识字符串

78 //执行查询操作，将查询结果自动封装成List<User>返回

79 List<User> lstUsers = sqlSession.selectList(statement);

80 //使用SqlSession执行完SQL之后需要关闭SqlSession

81 sqlSession.close();

82 System.out.println(lstUsers);

83 }

84 }

[复制代码](javascript:void(0);)

**二、使用MyBatis对表执行CRUD操作——基于注解的实现**

　　1、定义sql映射的接口

　　UserMapperI接口的代码如下：

[复制代码](javascript:void(0);)

1 package me.gacl.mapping;

2

3 import java.util.List;

4 import me.gacl.domain.User;

5 import org.apache.ibatis.annotations.Delete;

6 import org.apache.ibatis.annotations.Insert;

7 import org.apache.ibatis.annotations.Select;

8 import org.apache.ibatis.annotations.Update;

9

10 /\*\*

11 \* @author gacl

12 \* 定义sql映射的接口，使用注解指明方法要执行的SQL

13 \*/

14 public interface UserMapperI {

15

16 //使用@Insert注解指明add方法要执行的SQL

17 @Insert("insert into users(name, age) values(#{name}, #{age})")

18 public int add(User user);

19

20 //使用@Delete注解指明deleteById方法要执行的SQL

21 @Delete("delete from users where id=#{id}")

22 public int deleteById(int id);

23

24 //使用@Update注解指明update方法要执行的SQL

25 @Update("update users set name=#{name},age=#{age} where id=#{id}")

26 public int update(User user);

27

28 //使用@Select注解指明getById方法要执行的SQL

29 @Select("select \* from users where id=#{id}")

30 public User getById(int id);

31

32 //使用@Select注解指明getAll方法要执行的SQL

33 @Select("select \* from users")

34 public List<User> getAll();

35 }

[复制代码](javascript:void(0);)

　　需要说明的是，我们不需要针对UserMapperI接口去编写具体的实现类代码，这个具体的实现类由MyBatis帮我们动态构建出来，我们只需要直接拿来使用即可。

　　2、在conf.xml文件中注册这个映射接口

[复制代码](javascript:void(0);)

1 <?xml version="1.0" encoding="UTF-8"?>

2 <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">

3 <configuration>

4 <environments default="development">

5 <environment id="development">

6 <transactionManager type="JDBC" />

7 <!-- 配置数据库连接信息 -->

8 <dataSource type="POOLED">

9 <property name="driver" value="com.mysql.jdbc.Driver" />

10 <property name="url" value="jdbc:mysql://localhost:3306/mybatis" />

11 <property name="username" value="root" />

12 <property name="password" value="XDP" />

13 </dataSource>

14 </environment>

15 </environments>

16

17 <mappers>

18 <!-- 注册userMapper.xml文件，

19 userMapper.xml位于me.gacl.mapping这个包下，所以resource写成me/gacl/mapping/userMapper.xml-->

20 <mapper resource="me/gacl/mapping/userMapper.xml"/>

21 <!-- 注册UserMapper映射接口-->

22 <mapper class="me.gacl.mapping.UserMapperI"/>

23 </mappers>

24

25 </configuration>

[复制代码](javascript:void(0);)

　　单元测试类的代码如下：

[复制代码](javascript:void(0);)

1 package me.gacl.test;

2

3 import java.util.List;

4 import me.gacl.domain.User;

5 import me.gacl.mapping.UserMapperI;

6 import me.gacl.util.MyBatisUtil;

7 import org.apache.ibatis.session.SqlSession;

8 import org.junit.Test;

9

10 public class TestCRUDByAnnotationMapper {

11

12 @Test

13 public void testAdd(){

14 SqlSession sqlSession = MyBatisUtil.getSqlSession(true);

15 //得到UserMapperI接口的实现类对象，UserMapperI接口的实现类对象由sqlSession.getMapper(UserMapperI.class)动态构建出来

16 UserMapperI mapper = sqlSession.getMapper(UserMapperI.class);

17 User user = new User();

18 user.setName("用户xdp");

19 user.setAge(20);

20 int add = mapper.add(user);

21 //使用SqlSession执行完SQL之后需要关闭SqlSession

22 sqlSession.close();

23 System.out.println(add);

24 }

25

26 @Test

27 public void testUpdate(){

28 SqlSession sqlSession = MyBatisUtil.getSqlSession(true);

29 //得到UserMapperI接口的实现类对象，UserMapperI接口的实现类对象由sqlSession.getMapper(UserMapperI.class)动态构建出来

30 UserMapperI mapper = sqlSession.getMapper(UserMapperI.class);

31 User user = new User();

32 user.setId(3);

33 user.setName("孤傲苍狼\_xdp");

34 user.setAge(26);

35 //执行修改操作

36 int retResult = mapper.update(user);

37 //使用SqlSession执行完SQL之后需要关闭SqlSession

38 sqlSession.close();

39 System.out.println(retResult);

40 }

41

42 @Test

43 public void testDelete(){

44 SqlSession sqlSession = MyBatisUtil.getSqlSession(true);

45 //得到UserMapperI接口的实现类对象，UserMapperI接口的实现类对象由sqlSession.getMapper(UserMapperI.class)动态构建出来

46 UserMapperI mapper = sqlSession.getMapper(UserMapperI.class);

47 //执行删除操作

48 int retResult = mapper.deleteById(7);

49 //使用SqlSession执行完SQL之后需要关闭SqlSession

50 sqlSession.close();

51 System.out.println(retResult);

52 }

53

54 @Test

55 public void testGetUser(){

56 SqlSession sqlSession = MyBatisUtil.getSqlSession();

57 //得到UserMapperI接口的实现类对象，UserMapperI接口的实现类对象由sqlSession.getMapper(UserMapperI.class)动态构建出来

58 UserMapperI mapper = sqlSession.getMapper(UserMapperI.class);

59 //执行查询操作，将查询结果自动封装成User返回

60 User user = mapper.getById(8);

61 //使用SqlSession执行完SQL之后需要关闭SqlSession

62 sqlSession.close();

63 System.out.println(user);

64 }

65

66 @Test

67 public void testGetAll(){

68 SqlSession sqlSession = MyBatisUtil.getSqlSession();

69 //得到UserMapperI接口的实现类对象，UserMapperI接口的实现类对象由sqlSession.getMapper(UserMapperI.class)动态构建出来

70 UserMapperI mapper = sqlSession.getMapper(UserMapperI.class);

71 //执行查询操作，将查询结果自动封装成List<User>返回

72 List<User> lstUsers = mapper.getAll();

73 //使用SqlSession执行完SQL之后需要关闭SqlSession

74 sqlSession.close();

75 System.out.println(lstUsers);

76 }

77 }

[复制代码](javascript:void(0);)

　　用到的MyBatisUtil工具类代码如下：

[复制代码](javascript:void(0);)

1 package me.gacl.util;

2

3 import java.io.InputStream;

4

5 import org.apache.ibatis.session.SqlSession;

6 import org.apache.ibatis.session.SqlSessionFactory;

7 import org.apache.ibatis.session.SqlSessionFactoryBuilder;

8

9 public class MyBatisUtil {

10

11 /\*\*

12 \* 获取SqlSessionFactory

13 \* @return SqlSessionFactory

14 \*/

15 public static SqlSessionFactory getSqlSessionFactory() {

16 String resource = "conf.xml";

17 InputStream is = MyBatisUtil.class.getClassLoader().getResourceAsStream(resource);

18 SqlSessionFactory factory = new SqlSessionFactoryBuilder().build(is);

19 return factory;

20 }

21

22 /\*\*

23 \* 获取SqlSession

24 \* @return SqlSession

25 \*/

26 public static SqlSession getSqlSession() {

27 return getSqlSessionFactory().openSession();

28 }

29

30 /\*\*

31 \* 获取SqlSession

32 \* @param isAutoCommit

33 \* true 表示创建的SqlSession对象在执行完SQL之后会自动提交事务

34 \* false 表示创建的SqlSession对象在执行完SQL之后不会自动提交事务，这时就需要我们手动调用sqlSession.commit()提交事务

35 \* @return SqlSession

36 \*/

37 public static SqlSession getSqlSession(boolean isAutoCommit) {

38 return getSqlSessionFactory().openSession(isAutoCommit);

39 }

40 }

[复制代码](javascript:void(0);)

　　 以上的相关代码是全部测试通过的，关于使用MyBatis对表执行CRUD操作的内容就这么多。