**测试mybatis的crud操作xml文件**

| | | | |
|---|---|---|---|
| **笔记本:** | 编程练习学习 | | |
| **创建时间:** | 2020/2/12 17:43 | **更新时间:** | 2020/2/12 18:25 |
| **作者:** | 1003269315@qq.com | | |
| **URL:** | about:blank | | |

```
测试mybatis的crud操作xml文件
<!-- 查询所有 -->
<select id="findAll" resultMap="userMap">
<!--select id as userId,username as userName,address as
userAddress,sex as userSex,birthday as userBirthday from user;-
->
select * from user;
</select>

<!-- 保存用户 -->
<insert id="saveUser" parameterType="user">
<!-- 配置插入操作后，获取插入数据的id -->
<selectKey keyProperty="userId" keyColumn="id" resultType="int"
order="AFTER">
select last_insert_id();
</selectKey>
insert into user(username,address,sex,birthday)values(#
{userName},#{userAddress},#{userSex},#{userBirthday});
</insert>

<!-- 更新用户 -->
<update id="updateUser" parameterType="USER">
update user set username=#{userName},address=#
{userAddress},sex=#{userAex},birthday=#{userBirthday} where
id=#{userId}
</update>

<!-- 删除用户-->
<delete id="deleteUser" parameterType="java.lang.Integer">
delete from user where id = #{uid}
</delete>
<!-- 根据id查询用户 -->
<select id="findById" parameterType="INT" resultMap="userMap">
select * from user where id = #{uid}
</select>

<!-- 根据名称模糊查询 -->
```

```
<select id="findByName" parameterType="string"
resultMap="userMap">
select * from user where username like #{name}
 //prepatedstatement占位符
<!-- select * from user where username like '%${value}%'-->  //
字符串拼接sql
</select>

<!-- 获取用户的总记录条数 -->
<select id="findTotal" resultType="int">
select count(id) from user;
</select>

<!-- 根据queryVo的条件查询用户 -->
<select id="findUserByVo" parameterType="传入数据类型"
resultMap="userMap">
select * from user where username like #{user.username}
</select>
```

**测试mybatis的crud操作主文件**

| | | | |
|---|---|---|---|
| **笔记本:** | 编程练习学习 | | |
| **创建时间:** | 2020/2/12 17:29 | **更新时间:** | 2020/2/12 18:25 |
| **作者:** | 1003269315@qq.com | | |
| **URL:** | about:blank | | |

```
测试mybatis的crud操作主文件
public class MybatisTest {

private InputStream in;
private SqlSession sqlSession;
private IUserDao userDao;
//
@Before      //用于在测试方法执行之前执行（连接生成Dao接口代理）
public void init()throws Exception{
//1.读取配置文件，生成字节输入流
in = Resources.getResourceAsStream("SqlMapConfig.xml");
//2.获取SqlSessionFactory
SqlSessionFactory factory = new
SqlSessionFactoryBuilder().build(in);
//3.获取SqlSession对象
sqlSession = factory.openSession();
//4.获取dao的代理对象
userDao = sqlSession.getMapper(IUserDao.class);
}
//中间执行@Test语句（示例）
@After//用于在测试方法执行之后执行(保存和关闭的操作)
public void destroy()throws Exception{
//提交事务
sqlSession.commit();
//6.释放资源
sqlSession.close();
in.close();
}

/**
* 测试查询所有
*/
@Test
public void testFindAll(){
//5.执行查询所有方法
List<User> users = userDao.findAll();
for(User user : users){
System.out.println(user);
}

}

/**
```

```
* 测试保存操作
*/
@Test
public void testSave(){
User user = new User();
user.setUserName("modify User property");
user.setUserAddress("北京");
user.setUserSex("男");
user.setUserBirthday(new Date());
System.out.println("保存操作之前: "+user);
//5.执行保存方法
userDao.saveUser(user);

System.out.println("保存操作之后: "+user);
}

/**
* 测试更新操作
*/
@Test
public void testUpdate(){
User user = new User();
user.setUserId(50);
user.setUserName("mybastis update user");
user.setUserAddress("南京");
user.setUserSex("女");
user.setUserBirthday(new Date());

//5.执行保存方法
userDao.updateUser(user);
}

/**
* 测试删除操作
*/
@Test
public void testDelete(){
//5.执行删除方法
userDao.deleteUser(1001);
}

/**
* 测试删除操作
*/
@Test
public void testFindOne(){
//5.执行查询一个方法
User user = userDao.findById(50);
System.out.println(user);
}
```

```
/**
* 测试模糊查询操作
*/
@Test
public void testFindByName(){
//5.执行查询一个方法
List<User> users = userDao.findByName("%李%");
// List<User> users = userDao.findByName("王");
for(User user : users){
System.out.println(user);
}
}

/**
* 测试查询总记录条数
*/
@Test
public void testFindTotal(){
//5.执行查询一个方法
int count = userDao.findTotal();
System.out.println(count);
}


/**
* 测试使用QueryVo作为查询条件
*/
@Test
public void testFindByVo(){
QueryVo vo = new QueryVo();
User user = new User();
user.setUserName("%李%");
vo.setUser(user);
//5.执行查询一个方法
List<User> users = userDao.findUserByVo(vo);
for(User u : users){
System.out.println(u);
}
}
}
```

# Mybatis配置流程分析

**笔记本：** 编程练习学习

**创建时间：** 2020/2/12 17:24 **更新时间：** 2020/2/12 17:26

**作者：** 1003269315@qq.com

---

Mybatis配置流程分析

```
public class MybatisTest {
public static void main(String[] args)throws Exception {
//1.读取配置文件
InputStream in = Resources.getResourceAsStream("SqlMapConfig.xml");
//2.创建 SqlSessionFactory 的构建者对象
SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
//3.使用构建者创建工厂对象
SqlSessionFactory   SqlSessionFactory factory = builder.build(in);
  //4.使用 SqlSessionFactory 生产 SqlSession 对象
  SqlSession session = factory.openSession();
  //5.使用 SqlSession 创建 dao 接口的代理对象
  IUserDao userDao = session.getMapper(IUserDao.class);
  //6.使用代理对象执行查询所有方法
  List<User> users = userDao.findAll();
  for(User user : users) {    System.out.println(user);   }
  //7.释放资源
  session.close();   in.close();  }
}
```