浙江工业大学

JavaEE 技术实验报告

实验名称：

实验六 MyBatis 基础应用

学　　院：　计算机学院

班　　级：　软工03

姓　　名：　周少武

学　　号：　202103151022

时　　间：　2023.12.22

一、实验主要步骤

1、基础实验——MyBatis 框架搭建

（1）运行结果截图；

测试类设计如下

```
        try {
            InputStream config= Resources.
                    getResourceAsStream("mybatis-config.xml");
            SqlSessionFactory ssf=
                    new SqlSessionFactoryBuilder().build(config);
            SqlSession ss=ssf.openSession();
//查询一个用户
            MyUser mu = ss.selectOne( s: "UserMapper.selectUserById",  o: 1);
            System.out.println(mu.getUname());
//添加一个用户
            MyUser addmu=new MyUser();
            addmu.setUname("张三");
            addmu.setUsex("男");
            ss.insert( s: "UserMapper.addUser",addmu);
//修改一个用户
            addmu.setUid(11);
            addmu.setUname("万元神");
            ss.update( s: "UserMapper.updateUser",addmu);
////删除一个用户
            ss.delete( s: "UserMapper.deleteUser", o: 2);
////查询所有用户
            List<MyUser> list = ss.selectList( s: "UserMapper.selectAllUser");
            for(int i = 0; i < list.size(); i++){
                MyUser myUser = list.get(i);
                System.out.println(myUser.getUname());
            }
            ss.commit();
            ss.close();
```

改前数据库设计如下：

开始事务 文本 筛选 排

uid	uname	usex
1	王元神	男
2	源深	女
3	万克里	男
4	完德	女

改后数据库内数据如下：

uid	uname	usex
1	王元神	男
3	万克里	男
4	完德	女
5	万元神	男

（2）简述 MyBatis 的工作原理；

MyBatis 是一种持久层框架，主要用于将数据库操作与 Java 程序的业务逻辑分离。其工作原理可以概括为以下几个步骤：

SqlMap 配置： 开发人员需要编写一个或多个 XML 文件，其中包含了数据库操作的 SQL 语句以及与之对应的映射关系。这些配置文件被称为 SqlMap。

SqlSessionFactory 的创建： MyBatis 根据 SqlMap 配置文件生成一个 SqlSessionFactory，它是 MyBatis 的主要入口，负责创建 SqlSession 对象。

SqlSession 的获取： SqlSession 是 MyBatis 的核心对象，它提供了操作数据库的方法。通过 SqlSessionFactory 创建 SqlSession。

SQL 语句的执行： 在业务代码中，通过 SqlSession 执行 SQL 语句，MyBatis 将 SQL 语句发送到数据库执行，并将结果映射为 Java 对象。

映射关系的处理： MyBatis 使用 resultMap 或 resultType 将数据库中的结果集映射为 Java 对象，实现了对象关系映射（ORM）。

事务管理： MyBatis 支持事务管理，开发人员可以通过编程式或声明式的方式管理事务。

总体而言，MyBatis 的工作原理是通过配置文件定义 SQL 语句和映射关系，然后在 Java 代码中通过 SqlSession 执行 SQL，将结果映射为 Java 对象。

（3）简述 MyBatis 和 Hibernate 的异同点和优缺点；

相同点：

ORM 框架： MyBatis 和 Hibernate 都是用于实现对象关系映射（ORM）的框架，将数据库表与 Java 对象进行映射。

支持事务： 两者都提供了事务管理机制，可以通过编程式或声明式的方式进行事务控制。

不同点：

SQL 控制：

MyBatis： 提供了较为灵活的 SQL 控制，开发人员可以自由编写 SQL 语句，并通过映射配置将结果映射到 Java 对象。

Hibernate： 更倾向于使用 HQL（Hibernate Query Language）进行数据库操作，它是一种面向对象的查询语言，不需要直接编写 SQL。

灵活性：

MyBatis： 更灵活，开发人员可以直接控制 SQL 语句，适合对 SQL 有较强掌控力的开发者。

Hibernate： 对于简单的 CRUD 操作，提供了更高的抽象，开发者无需编写具体的 SQL 语句，更注重面向对象的设计。

对象状态：

MyBatis： 对象的状态由开发者手动维护，需要手动编写 SQL。

Hibernate： 提供了对象状态的自动管理，通过 Session 对象跟踪对象状态并自动生成 SQL。

学习曲线：

MyBatis： 学习曲线相对较低，易于上手，特别适合对 SQL 较熟悉的开发者。

Hibernate： 学习曲线较陡峭，因为需要理解其更复杂的对象关系映射和查询语言。
优缺点：
MyBatis 的优缺点：
优点：
灵活性高，可以灵活控制 SQL。
相对较简单，易于上手。
对于已有 SQL 优化经验的开发者更友好。
缺点：
对象状态需要手动维护。
需要手写 SQL，不够面向对象。
Hibernate 的优缺点：
优点：
高度抽象，无需手动编写 SQL。
对象状态自动管理。
查询语言 HQL 更符合面向对象的思想。
缺点：
学习曲线相对较高。
灵活性相对较差，不如 MyBatis 定制化高。

二、提高实验——映射器

（1）运行结果截图；
测试类设计如下

```
SqlSession ss=ssf.openSession();
Map<String, Object> map=new HashMap<>();
map.put("uname", "陈");
map.put("usex", "男");
List<MyUser> list=ss.selectList
        ( s: "UserMapper.selectAllUser",map);
for(MyUser myUser : list){
    System.out.println(myUser.getUname());
}
```

结果如下

```
陈某
```

SelectUserParam实现：

```
SelectUserParam su = new SelectUserParam();
su.setU_name("陈");
su.setU_sex("男");
List<MyUser> list=ss.selectList
        ( s: "UserMapper.selectAllUser",su);
for(MyUser myUser : list){
    System.out.println(myUser.getUname());
}
```

采用map进行selectalluser

```
List<Map<String, Object>> list =ss.selectList
        ( s: "UserMapper.selectAllUser");
for(Map<String, Object> myUser : list){
    System.out.println(myUser);
}
```

SelectResultMap

```
List<MapUser> list =ss.selectList
        ( s: "UserMapper.selectResultMap");
for(MapUser myUser : list){
    System.out.println(myUser);
}
```

结果为

```
User [uid=1, uname=王元神, usex=男]
User [uid=3, uname=万克里, usex=男]
User [uid=4, uname=完德, usex=女]
User [uid=5, uname=万元神, usex=男]
User [uid=6, uname=陈某, usex=男]
```

（2）结合实验过程，总结 MyBatis 实现查询时返回的结果集中常见的存储方式；

单一值（Scalar）：

结果集只包含一个单一的值，通常用于返回聚合函数的结果或者单一字段的查询结果。

简单对象（POJO）：

将查询结果映射到一个简单的 Java 对象，对象的属性与查询结果的列一一对应。这是 MyBatis 最基本的映射方式，也是最常见的方式。

Map 集合：

将查询结果映射为一个 Map 集合，其中键为列名，值为对应的数据。这种方式适用于动态查询或者不确定列名的情况。

嵌套结果集（Nested Result Maps）：

将查询结果映射到包含其他映射的对象中，实现了对象之间的关联。这种方式适用于数据库表之间存在关联关系的情况。

嵌套查询（Association）：

类似于嵌套结果集，但是更注重于通过嵌套查询关联另外一个对象。在数据库查询时，可以通过关联查询获取相关联的数据。

关联集合（Collection）：

将查询结果映射为包含其他映射的集合，通常用于一对多的关联关系。例如，一个作者对应多个书籍的情况。

自动映射（Auto-Mapping）：

MyBatis 支持自动映射，可以通过列名和属性名的匹配自动将查询结果映射到对象的属性中。这种方式省略了映射配置的繁琐过程，适用于简单的查询。

混合映射（Mixed Mapping）：

结合上述多种映射方式，根据实际需求采用不同的映射方式。这种方式可以在不同的场景中选择最合适的映射方式。

三、扩展实验——级联查询

（1）运行结果截图；

测试类设计如下

```
public static void main(String[] args){
    try {
        InputStream config=Resources.
                getResourceAsStream("mybatis-config.xml");
        SqlSessionFactory ssf=
                new SqlSessionFactoryBuilder().build(config);
        SqlSession ss=ssf.openSession();
        PersonDao personDao = ss.getMapper(PersonDao.class);
        Person p1 = personDao.selectPersonById1(1);
        System.out.println(p1);
        System.out.println("=======================");
        Person p2 = personDao.selectPersonById2(1);
        System.out.println(p2);
        System.out.println("=======================");
        SelectPersonById p3 = personDao.selectPersonById3(1);
        System.out.println(p3);
        ss.commit();
        ss.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
```

输出结果如下：

```
po.Person_$$_jvstd8e_0@4a11eb84
=======================
po.Person@435fb7b5
=======================
Person [id=1,name=aaa, age=111,code=123]
```

二、实验分析及总结

总结：

在 MyBatis 中，实现查询时返回的结果集有多种常见的存储方式，开发人员可以根据实际需求选择最合适的方式。以下是常见的结果集存储方式：

单一值（Scalar）： 适用于返回单一值的情况，例如聚合函数的结果或者单一字段的查询结果。

简单对象（POJO）： 将查询结果映射到一个简单的 Java 对象，对象的属性与查询结果的列一一对应，是最基本也是最常见的映射方式。

Map 集合： 将查询结果映射为一个 Map 集合，键为列名，值为对应的数据，适用于动态查询或者不确定列名的情况。
嵌套结果集（Nested Result Maps）： 实现了对象之间的关联，适用于数据库表之间存在关联关系的情况。
嵌套查询（Association）： 通过嵌套查询关联另外一个对象，适用于需要关联查询的情况。
关联集合（Collection）： 将查询结果映射为包含其他映射的集合，用于一对多的关联关系，例如一个作者对应多个书籍的情况。
自动映射（Auto-Mapping）： MyBatis 支持自动映射，通过列名和属性名的匹配自动映射查询结果，省略了映射配置的繁琐过程，适用于简单的查询。
混合映射（Mixed Mapping）： 结合多种映射方式，根据实际需求选择不同的映射方式，以灵活应对各种复杂的查询场景。
这些映射方式提供了丰富的选择，使得开发者能够根据项目需求灵活地控制和定制查询结果的处理方式。