**前端**

**方法**

showBooks(books) {

this.bookList = books

this.bookVis = true

}

**变量**

bookList:[]

bookVis:false

**标签**

<el-dialog title="用户拥有的图书列表" v-model="bookVis" width="30%">

<el-table :data="bookList" stripe border>

<el-table-column prop="id" label="ID"></el-table-column>

<el-table-column prop="name" label="名称"></el-table-column>

<el-table-column prop="price" label="价格"></el-table-column>

</el-table>

</el-dialog>

**后端**

**操作器**

public Result<?> findPage(@RequestParam(defaultValue = "1") Integer pageNum,

@RequestParam(defaultValue = "10") Integer pageSize,

@RequestParam(defaultValue = "") String search){

//pageNum是当前页;pageSize是一页有多少条数据;search是查询的关键字(defaultValue是设置默认值)

LambdaQueryWrapper<User> wrapper=Wrappers.<User>lambdaQuery();//sql条件的执行对象

if (StrUtil.isNotBlank(search)){//StrUtil.isNotBlank(search)是hutool依赖用来帮助判断参数是否为空的

wrapper.like(User::getNickName,search);

}

Page<User> userPage= userMapper.findPage(new Page<>(pageNum,pageSize));

//selectPage需要Page对象和sql执行对象

//new Page<>(pageNum,pageSize)//分页对象

//Wrappers.<User>lambdaQuery().like(User::getNickName,search)//sql执行对象

// User::getNickName是User.getNickName的变形

return Result.success(userPage);//Result.success(变量名)//用返回变量

}

**实体类**

package com.example.demo.entity;

import com.baomidou.mybatisplus.annotation.IdType;

import com.baomidou.mybatisplus.annotation.TableField;

import com.baomidou.mybatisplus.annotation.TableId;

import com.baomidou.mybatisplus.annotation.TableName;

import lombok.Data;

import java.util.List;

@TableName("user")//用mybatisplus创建对象要与数据库的表名相对应

@Data//lombok中的帮助生成get和set方法并隐藏

public class User {

@TableId(value = "id",type = IdType.AUTO)//确认id为主键且自动生成

private Integer id;

private String username;

private String password;

private String nickName;

private Integer age;

private String sex;

private String address;

private Integer role;

@TableField(exist = false)//@TableField是表示booklist不在MySQL的表里

private List<Book> bookList;

}

**userMapper类**

package com.example.demo.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import com.baomidou.mybatisplus.extension.plugins.pagination.Page;

import com.example.demo.entity.User;

public interface UserMapper extends BaseMapper<User> {//UserMapper继承BaseMapper，BaseMapper自带增删改查

//一对多查询

Page<User> findPage(Page<User> page);

}

**userMapper的实现xml**

package com.example.demo.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import com.baomidou.mybatisplus.extension.plugins.pagination.Page;

import com.example.demo.entity.User;

public interface UserMapper extends BaseMapper<User> {//UserMapper继承BaseMapper，BaseMapper自带增删改查

//一对多查询

Page<User> findPage(Page<User> page);

}

**bookMapper类**

package com.example.demo.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import com.example.demo.entity.Book;

public interface BookMapper extends BaseMapper<Book> {//BookMapper继承BaseMapper，BaseMapper自带增删改查,(方括号内是实体类)

}

**bookMapper的实现xml**

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

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.example.demo.mapper.BookMapper"><!--mapper这是连接对象的mapper-->

<select id="findByUserId" resultType="com.example.demo.entity.Book"><!--写查询语句的位置,要表明返回对象-->

select `book`.\*, `user`.nick\_name

from `book`

left join `user` on `book`.user\_id = `user`.id

where `book`.user\_id = #{userId}

</select>

</mapper>、