

Quiz后端-4、实现查询展示用户

日期：2025.07.19 作者：tfzhang

后端主要提供用户与题目的注册、查询、删除等操作。使用到的开发工具：

- IDEA2024; 关注公众号”青椒工具”，发送”IDEA”，获取windows下的IDEA安装包
- mysql 5.7; 关注公众号”青椒工具”，发送”mysql”，获取windows下的mysql5.7安装包；

0、前情回顾：

- 1、Java访问数据库：MyBatis框架；
- 2、浏览器访问Java：Controller；
- 3、实现插入新用户接口；
- 4、实现删除用户；

1、查询展示用户的两个接口：

- 按照分页的方式展示用户；
- 按照用户名查询；



2、按照分页查询用户；

预备知识：mysql分页查询基础

a、在mysql中采用select后添加limit参数来实现分页查询：

```
1 select * from user limit 0, 5;
2 //limit后两个参数的含义0表示起始数据，5表示返回5条数据；
```

b、在mysql中要查询总的记录数：

```
1 select count(*) from user;
```

对比上述的前端查询页，在执行分页查询时，我们需要获取总的记录数和当前页的数据；

当前页的起始点计算：(当前页码-1)*每页条数；

c、前端要提供给后端的数据：当前页码， 每页展示数；

后端要提供给前端的数据：当前页码的数据list， 以及总记录数；

参考资料：[Day10-06. 案例-员工管理-分页查询-分析哔哩哔哩bilibili](#)

步骤1：在model新建PageBean对象用于controller返回数据

```
1  @Data
2  public class PageBean{
3      private int total;
4      private List<User> rows;
5  }
```

步骤2：usermapper中添加代码

```
1  @Select("SELECT COUNT(*) FROM user WHERE isDelete=0")
2  public int count();
3
4  @Select("SELECT * FROM user WHERE isDelete=0 limit #{start},#{pageSize}")
5  public List<User> page(Integer start, Integer pageSize);
```

步骤3：Controller中添加代码

```
1  @GetMapping("/users")
2  public Result getPage(@RequestParam(defaultValue="1")Integer page,
3      @RequestParam(defaultValue="5")Integer pageSize){
4      PageBean pageBean=userService.page(page, pageSize);
5      return Result.success(pageBean);
6  }
```

步骤4：Service层中添加代码

```
1  //接口：
2  PageBean page(Integer page, Integer pageSize);
3
4  //实现：
5  public PageBean page(Integer page, Integer pageSize){
6      //获取总的记录数：
7      Integer total=userMapper.count();
8
9      //获取分页查询结果列表：
10     Integer start = (page-1)*pageSize;
11     List<User> userList=userMapper.page(start, pageSize);
12
13     //封装PageBean对象：
14     PageBean pageBean = new PageBean();
15     pageBean.setTotal(total);
16     pageBean.setRows(userList);
17
18     return pageBean;
19 }
```

步骤5：分页查询测试；

如果出现错误，可以打开**MyBatis**的日志，输出错误信息到控制台：

[Day09-03. Mybatis-基础操作-删除\(预编译SQL\)哔哩哔哩bilibili](#)

3、按照用户名查询

数据库预备知识：

要在用户名中查询某个特定的字符串，可以使用如下的sql模糊查询语句：

```
1 SELECT * FROM user WHERE username like '%abc%';
```

对应的MyBatis的mapper层的代码：

```
1 @Mapper
2 public interface UserMapper {
3     @Select("SELECT * FROM user WHERE username LIKE CONCAT('%', #{keyword},
4         '%') AND isDelete=0")
5     List<User> findByName(String keyword);
6 }
```

Controller中的代码：

```
1 @GetMapping("/findUser")
2 public Result getUser(String keyword){
3     List<User> users=userService.findByName(keyword);
4     return Result.success(users);
5 }
```

Service层中的代码：

```
1 //接口;
2 public List<User> findByName(String keyword);
3
4 //实现:
5 public List<User> findByName(String keyword){
6     List<User> userList=userMapper.findByName(keyword);
7     for(User user:userList){
8         user.setUserPassword("*****");
9     }
10    return userList;
11 }
```

