

Quiz后端-3、实现插入新用户接口

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；

1、前情回顾：

- 1、Java访问数据库：MyBatis框架；
- 2、浏览器访问Java：Controller；

2、实现用户注册与添加：

Quiz的用户数据表设计：

```
1 DROP TABLE IF EXISTS user;
2
3 create table user
4 (
5     id          bigint auto_increment primary key,
6     userName    varchar(256)                null comment '用户名',
7     userPassword varchar(512)                null comment '密码',
8     updateTime  datetime default CURRENT_TIMESTAMP null,
9     createTime  datetime default CURRENT_TIMESTAMP null,
10    isDelete     tinyint                      null,
11    userRole     int          default 0        null comment '表示用户角
色, 0 普通用户, 1 管理员'
12 )
13 comment '用户表';
```

字段解释：

- isDelete是逻辑删除标志位，所有的删除数据不真正将数据从数据库删除，而是置位逻辑符；
- userRole是用户角色，用于区别管理员与普通用户；

步骤1：创建数据库

步骤2：创建数据库对应的实体类

对应上述的user数据库表，如下是对应的user实体类：

```
1 @Data
2 public class User {
3     /** 用户ID */
4     private Long id;
5
6     /** 用户名 */
7     private String userName;
```

```

8
9      /* 密码*/
10     private String userPassword;
11
12     /* 是否删除 */
13     private Integer isDelete;
14
15     /* 用户角色: 0-普通用户, 1-管理员 */
16     private Integer userRole;
17
18     /* 创建时间 */
19     private Date createTime;
20
21     /*更新时间 */
22     private Date updateTime;
23 }

```

步骤3: 创建MyBatis配置文件

步骤4: 编写添加用户的mapper

UserMapper类原来的代码:

```

1  package com.tfzhang.quiz.mapper;
2
3  import com.tfzhang.quiz.model.User;
4  import org.apache.ibatis.annotations.Mapper;
5  import org.apache.ibatis.annotations.Select;
6
7  import java.util.List;
8
9  @Mapper //运行时, 框架会自动实现实现类对象, 并将对象实例交由IoC容器管理;
10 public interface UserMapper{
11
12     //查询全部用户;
13     @Select("select * from user")
14     public List<User> list();
15 }

```

现在要新增用户, 对应的sql语句如下:

```

1  insert into user(userName,
2      userPassword,isDelete,userRole,createTime,updateTime)
      values('tom', '123456', 0, 0, now(), now(),);

```

对应的Java代码:

```

1      @Insert("insert into user(userName,
2          userPassword,isDelete,userRole,createTime, updateTime)"+
3          "values(#{userName}, #{userPassword}, #{isDelete}, #{userRole}, #
4          {createTime}, #{updateTime})")
5      @Options(useGeneratedKeys = true, keyProperty = "id")
6      public int saveUser(User user);

```

步骤5: 对添加用户的mapper进行测试

同样地，在test.java.com.tfzhang.quiz.QuizApplicationTests中添加测试代码：

```
1 package com.tfzhang.quiz;
2
3 import com.tfzhang.quiz.mapper.UserMapper;
4 import org.junit.jupiter.api.Test;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.context.SpringBootTest;
7 import com.tfzhang.quiz.model.User;
8
9 import java.util.Date;
10 import java.util.List;
11
12 @SpringBootTest
13 class QuizApplicationTests {
14
15     @Autowired
16     private UserMapper userMapper;
17
18     @Test
19     public void testSaveUser(){
20         User user = new User();
21         user.setUserName("testuser");
22         user.setUserPassword("password123");
23
24         user.setUserRole(0);
25         user.setIsDelete(0);
26
27         Date now = new Date();
28         user.setCreateTime(now);
29         user.setUpdateTime(now);
30
31         // 执行插入操作,插入后返回插入成功的行数,这里默认就应该是1;
32         int result = -1;
33         result = userMapper.saveUser(user);
34         System.out.println(result);
35         //生成的id会默认回填到user对象中;
36         System.out.println(user.getId());
37         return result;
38     }
39 }
```

可以根据result的返回值，或者user.getId()的值来判断是否插入数据成功。

步骤6：编写添加用户的Controller

The image shows a registration form with the title "Register" in orange. It contains three input fields: "Username" with placeholder text "Choose a username", "Password" with placeholder text "Choose a password", and "Confirm Password" with placeholder text "Confirm your password". Below these fields is an orange "Register" button. At the bottom right, there is a link that says "Already have an account? Login".

根据上图Quiz用户前端的注册新用户页面，添加新用户时，浏览器端会发送3个参数：

- username
- password
- confirmpassword

在controller目录下新建：UserController.java

```
1  @RestController
2  public class UserController {
3
4      @RequestMapping("/register")
5      public Result addUser(String username, String password, String
6      checkpassword){
7          //代码逻辑步骤：
8          //1、用户输入的账户和密码不能为空；
9          //2、校验用户的账户、密码是否符合要求：
10         // - 账户字符不能少于4个；
11         // - 密码不能小于8位；
12         // - 密码和确认密码要一致；
13         // - 账户不能与已有的重复；
14         // - 账户不能包含特殊字符；
15         // - 密码和校验密码相同；
16         //3、对密码进行加密；保证后端工作人员不能看到用户密码；
17         //4、向数据库插入数据；
18     }
```

上述的很多检查可以在前端页面进行检查，但你不能保证所有的用户都是规矩的用户，所以有些检查，我们还是要在后端实现：

```

1 //代码逻辑步骤;
2 //1、用户输入的账户和密码不能为空;
3 //2、校验用户的账户、密码是否符合要求:
4 //    - 账户不能包含特殊字符;
5 //    - 密码和确认密码要一致;
6 //    - 账户不能与已有的重复;
7 //3、对密码进行加密; 保证后端工作人员不能看到用户密码; 密码不要用明文;
8 //4、向数据库插入数据;

```

1、用户输入的账户和密码不能为空;

```

1 if(StringUtils.isAnyBlank(username, password, checkpassword)){
2     return Result.error("用户名或密码为空");
3 }

```

此处的 StringUtils 库来自 common lang 库; 访问 maven 官网, 添加该库的配置文件到 pom.xml

```

1 https://mvnrepository.com/

```

2、判断 password 和 checkpassword 两者是否一致;

```

1 if (!password.equals(checkpassword)) {
2     return Result.error("两次输入的密码不一致");
3 }

```

3、校验用户的账户、密码是否符合要求:

```

1 String regex = "[a-zA-Z0-9]+$";
2 Pattern pattern = Pattern.compile(regex);
3 Matcher matcher = pattern.matcher(username);
4 if (!matcher.matches()) {
5     return Result.error("用户名包含特殊字符");
6 }

```

4、查询数据库, 查看用户名 username 是否已经存在:

```

1 //涉及到数据库查询; 暂时不实现;
2 //to do...

```

5、对密码进行加密; 保证后端开发人员不能看到用户密码;

```

1 final String SALT = "com.quiz";
2 String encryptedPassword =
    DigestUtils.md5DigestAsHex((SALT+userpassword).getBytes());

```

6、创建新用户, 并且向数据库插入新数据;

```

1 public Result addUser(String username, String password, String
checkpassword) {
2     //此处的逻辑代码;
3     if (StringUtils.isAnyBlank(username, password, checkpassword)) {
4         return Result.error("用户名或密码为空");

```

```

5         }
6
7         if (!password.equals(checkpassword)) {
8             return Result.error("两次输入的密码不一致");
9         }
10
11         String regex = "[a-zA-Z0-9]+$";
12         Pattern pattern = Pattern.compile(regex);
13         Matcher matcher = pattern.matcher(username);
14         if (!matcher.matches()) {
15             return Result.error("用户名包含特殊字符");
16         }
17
18         //查询数据库，确定是否已经存在用户名；
19         //to add...
20
21         //对密码进行加密；
22         final String SALT = "com.quiz";
23         String encrptedPassword = DigestUtils.md5DigestAsHex((SALT +
password).getBytes());
24
25         User user = new User();
26         user.setUserName(username);
27         user.setUserPassword(encrptedPassword);
28         /**
29          * 注册默认是普通用户，所以userRole设置为0；
30          */
31         user.setUserRole(0);
32         user.setIsDelete(0);
33
34         Date now = new Date();
35         user.setCreateTime(now);
36         user.setUpdateTime(now);
37
38         //4.插入到数据库；
39         int result = userMapper.saveUser(user);
40
41         if (result > 0)
42             return Result.success("新增用户成功");
43         else
44             return Result.error("注册用户失败");
45
46     }

```

3、三层架构的改造

参考资料：[Day05-09. 分层解耦-三层架构哔哩哔哩bilibili](#)

根据三层架构，当前的controller的问题：

- 业务逻辑混杂，没有独立出来；

步骤1：创建Service目录及文件

创建Service目录，集中放置业务逻辑代码；Service采用接口与实现分离的方式来实现；

```

1 //UserService.java:
2 public interface UserService{
3
4     //保存新用户;
5     public Result addUser(String username, String password, String
6     checkpassword);
7 }

```

```

1 //UserServiceImpl.java
2
3 @Service
4 public class UserServiceImpl implements UserService {
5
6     @Autowired
7     private UserMapper userMapper;
8
9     public Result addUser(String username, String password, String
10    checkpassword){
11        //此处的逻辑代码;
12        if (StringUtils.isBlank(username, password, checkpassword)) {
13            return Result.error("用户名或密码为空");
14        }
15
16        if (!password.equals(checkpassword)) {
17            return Result.error("两次输入的密码不一致");
18        }
19
20        String regex = "^[a-zA-Z0-9]+$";
21        Pattern pattern = Pattern.compile(regex);
22        Matcher matcher = pattern.matcher(username);
23        if (!matcher.matches()) {
24            return Result.error("用户名包含特殊字符");
25        }
26
27        //查询数据库，确定是否已经存在用户名;
28        //to add...
29
30        //对密码进行加密;
31        final String SALT = "com.quiz";
32        String encryptedPassword = DigestUtils.md5DigestAsHex((SALT +
33        password).getBytes());
34
35        User user = new User();
36        user.setUsername(username);
37        user.setUserPassword(encryptedPassword);
38        /**
39         * 注册默认是普通用户，所以userRole设置为0;
40         */
41        user.setUserRole(0);
42        user.setIsDelete(0);
43
44        Date now = new Date();
45        user.setCreateTime(now);
46        user.setUpdateTime(now);
47
48        //4.插入到数据库;

```

```

47         int result = userMapper.saveUser(user);
48
49         if (result > 0)
50             return Result.success("新增用户成功");
51         else
52             return Result.error("注册用户失败");
53     }
54 }

```

步骤2：改造其他代码：

比如删减Controller组件中的代码；

4、作业：

完成Service中如下的代码：

```

1         String regex = "[a-zA-Z0-9]+$";
2         Pattern pattern = Pattern.compile(regex);
3         Matcher matcher = pattern.matcher(username);
4         if (!matcher.matches()) {
5             return Result.error("用户名包含特殊字符");
6         }
7
8         //查询数据库，确定是否已经存在用户名；
9         //to add...
10
11        //对密码进行加密；
12        final String SALT = "com.quiz";
13        String encriptedPassword = DigestUtils.md5DigestAsHex((SALT +
password).getBytes());

```

要求：

- 完成MyBatis端的以用户名查询数据库；
- 再完成UserServiceImpl中，to add处的代码；