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：

```sql
DROP TABLE IF EXISTS user;

create table user
(
    id           bigint auto_increment primary key,
    userName     varchar(256)                       null comment '用户名',
    userPassword varchar(512)                       null comment '密码',
    updateTime   datetime default CURRENT_TIMESTAMP null,
    createTime   datetime default CURRENT_TIMESTAMP null,
    isDelete     tinyint                            null,
    userRole     int      default 0                 null comment '表示用户角色， 0 普通用户， 1 管理员'
)
    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：

```java
@Data
public class User {
    /** 用户ID */
    private Long id;

    /** 用户名 */
    private String userName;
```

```
    /* 密码*/
    private String userPassword;

    /* 是否删除 */
    private Integer isDelete;

    /* 用户角色：0-普通用户，1-管理员 */
    private Integer userRole;

    /* 创建时间 */
    private Date createTime;

    /*更新时间 */
    private Date updateTime;
}
```

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：

```
package com.tfzhang.quiz.mapper;

import com.tfzhang.quiz.model.User;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Select;

import java.util.List;

@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；
public interface UserMapper{

    //查询全部用户；
    @Select("select * from user")
    public List<User> list();
}
```

现在要新增用户，对应的sql语句如下：

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

对应的Java代码：

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

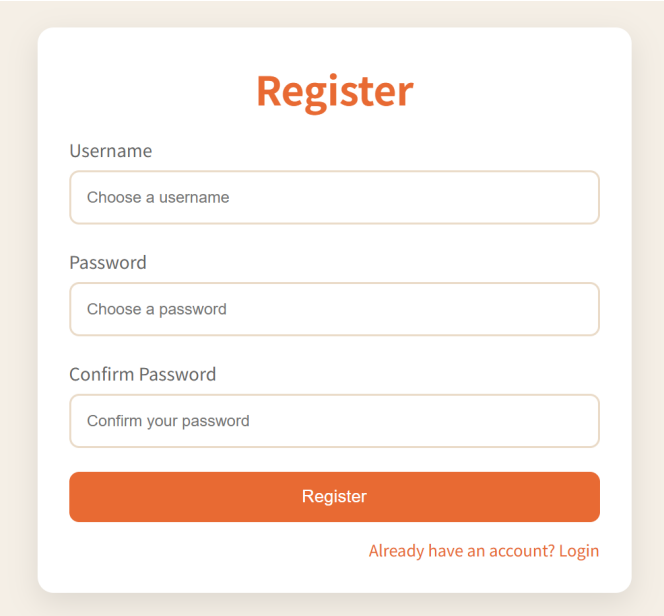
步骤5：对添加用户的mapper进行测试

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

```java
package com.tfzhang.quiz;

import com.tfzhang.quiz.mapper.UserMapper;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.tfzhang.quiz.model.User;

import java.util.Date;
import java.util.List;

@SpringBootTest
class QuizApplicationTests {

    @Autowired
    private UserMapper userMapper;

    @Test
    public void testSaveUser(){
        User user = new User();
        user.setUserName("testuser");
        user.setUserPassword("password123");

        user.setUserRole(0);
        user.setIsDelete(0);

        Date now = new Date();
        user.setCreateTime(now);
        user.setUpdateTime(now);

        // 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;
        int result = -1;
        result = userMapper.saveUser(user);
        System.out.println(result);
        //生成的id会默认回填到user对象中;
        System.out.println(user.getId());
        return result;
    }
}
```

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



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

```
@RestController
public class UserController {

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

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：

```
        //代码逻辑步骤；
        //1、用户输入的账户和密码不能为空；
        //2、校验用户的账户、密码是否符合要求：
           //    - 账户不能包含特殊字符；
           //    - 密码和确认密码要一致；
           //    - 账户不能与已有的重复；
        //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；
        //4、向数据库插入数据；
```

1、用户输入的账户和密码不能为空；

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

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

```
https://mvnrepository.com/
```

2、判断password和checkpassword两者是否一致；

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

3、校验用户的账户、密码是否符合要求：

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

4、查询数据库，查看用户名username是否已经存在：

```
//涉及到数据库查询；暂时不实现；
//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；

```
final String SALT = "com.quiz";
String encrptedPassword =
DigestUtils.md5DigestAsHex((SALT+userpassword).getBytes());
```

6、创建新用户，并且向数据库插入新数据；

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

```
        }

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

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

        //查询数据库，确定是否已经存在用户名;
        //to add...

        //对密码进行加密;
        final String SALT = "com.quiz";
        String encrptedPassword = DigestUtils.md5DigestAsHex((SALT +
password).getBytes());

        User user = new User();
        user.setUserName(username);
        user.setUserPassword(encrptedPassword);
        /**
         * 注册默认是普通用户，所以userRole设置为0;
         */
        user.setUserRole(0);
        user.setIsDelete(0);

        Date now = new Date();
        user.setCreateTime(now);
        user.setUpdateTime(now);

        //4.插入到数据库;
        int result = userMapper.saveUser(user);

        if (result > 0)
            return Result.success("新增用户成功");
        else
            return Result.error("注册用户失败");

    }
```

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；

```java
//UserService.java:
public interface UserService{

    //保存新用户;
    public Result addUser(String username, String password, String checkpassword);
}
```

```java
//UserServiceImpl.java

@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserMapper userMapper;

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

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

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

        //查询数据库，确定是否已经存在用户名;
        //to add...

        //对密码进行加密;
        final String SALT = "com.quiz";
        String encrptedPassword = DigestUtils.md5DigestAsHex((SALT + password).getBytes());

        User user = new User();
        user.setUserName(username);
        user.setUserPassword(encrptedPassword);
        /**
         * 注册默认是普通用户，所以userRole设置为0;
         */
        user.setUserRole(0);
        user.setIsDelete(0);

        Date now = new Date();
        user.setCreateTime(now);
        user.setUpdateTime(now);

        //4.插入到数据库;
```

```
        int result = userMapper.saveUser(user);

        if (result > 0)
            return Result.success("新增用户成功");
        else
            return Result.error("注册用户失败");
    }
}
```

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：

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

        //查询数据库，确定是否已经存在用户名;
        //to add...

        //对密码进行加密;
        final String SALT = "com.quiz";
        String encrptedPassword = 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；