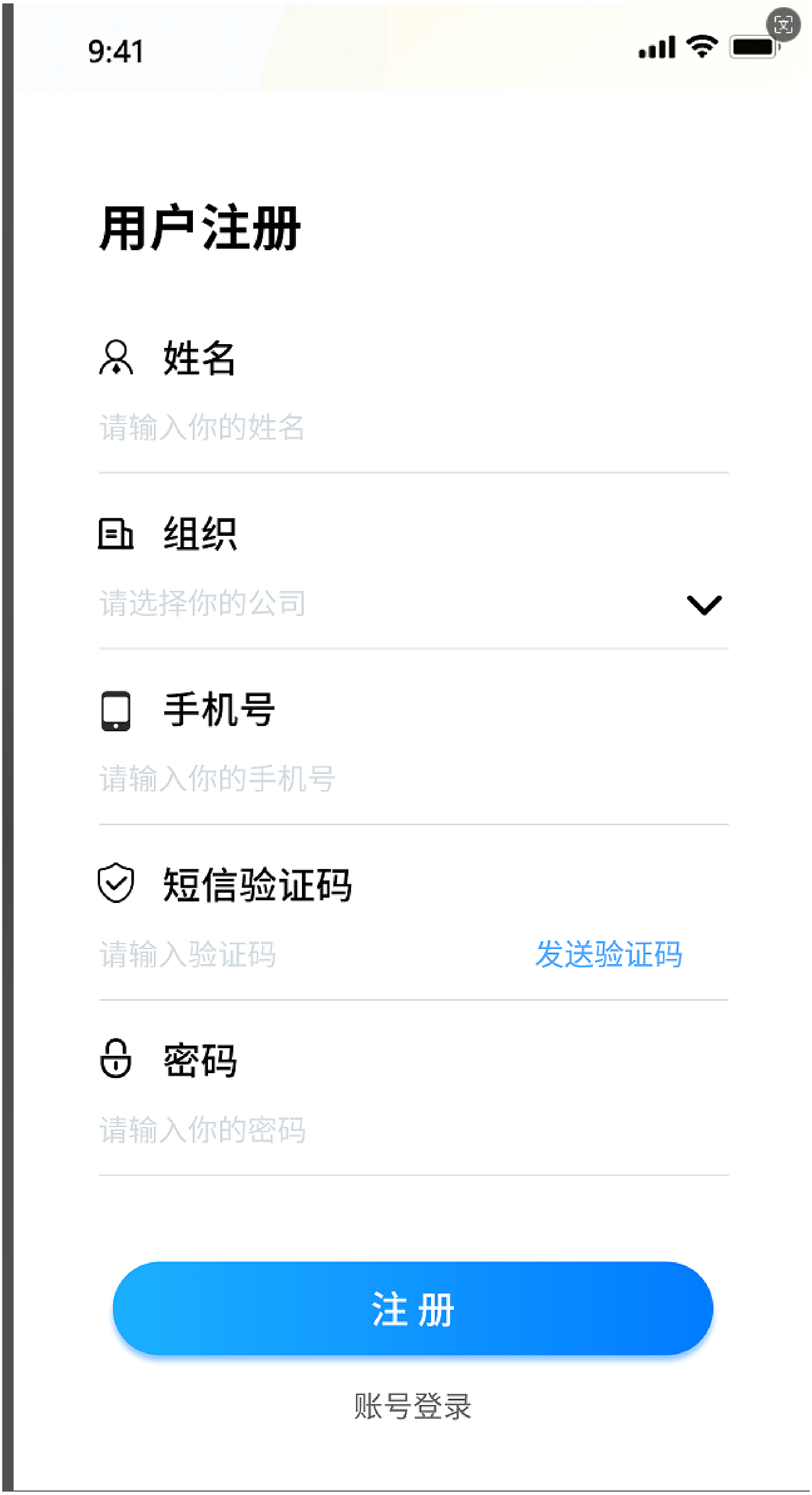
day3

注册接口实现



用户注册

姓名

请输入你的姓名

组织

请选择你的公司

手机号

请输入你的手机号

短信验证码

请输入验证码

密码

请输入你的密码

注册代码

usercontroller

```
package com.zlt.app.controller;

import com.zlt.app.dto.UserDTO;
import com.zlt.app.service.UserService;
import com.zlt.app.util.StringUtil;
import com.zlt.app.vo.Result;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.concurrent.TimeUnit;

@RestController//标记当前是控制层
@RequestMapping("user")
public class UserController {

    @GetMapping("test") //标记下面的方法是get请求
    public String test(){
        return "hello world";
    }
    @Autowired
    private RedisTemplate redisTemplate;
    @Autowired
    private UserService userService;

    /**
     * 发送验证码
     * @param phone
     * @return
     */

    @GetMapping("send")
    public ResponseEntity<Result>send(String phone){
        //生成一个四位数的随机字符串
        String code = StringUtil.getRandomNumber(4);
        System.out.println(code);
        //生成一个redis里面的key
         String key=StringUtil.uuid()+phone;
        System.out.println(key);
         //将生成的验证码放入redis
        redisTemplate.opsForValue().set(key,code,1, TimeUnit.MINUTES);


        return ResponseEntity.status(200).header("SMS_SEND",key).body(Result.success("发送成功"));
    }


    @PostMapping("reg")
    public ResponseEntity<Result>userReg(@RequestBody UserDTO userDTO,@RequestHeader("SMS_SEND")String key){
        //判断验证码是否正确
        String code =(String)redisTemplate.opsForValue().get(key);
        if (code==null){
            return ResponseEntity.status(200).body(Result.fail("验证码过期"));
        }
        if (!code.equals(userDTO.getCode())){
            return ResponseEntity.status(200).body(Result.fail("验证码错误"));
        }
        boolean success = userService.userReg(userDTO);
        return success ? ResponseEntity.status(200).body(Result.success("注册成功")):ResponseEntity.status(200).body(Result.fail("注册失败"));

    }

}
```

userService

```
package com.zlt.app.service;

import com.zlt.app.dto.UserDTO;

public interface UserService {
    boolean userReg(UserDTO userDTO);
}
```

UserServiceImpl

```
package com.zlt.app.service.impl;

import com.zlt.app.dto.UserDTO;
import com.zlt.app.entity.User;
import com.zlt.app.mapper.UserMapper;
import com.zlt.app.service.UserService;
import com.zlt.app.util.StringUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service //标记当前为业务层
public class UserServiceImpl implements UserService {
    @Autowired
    private UserMapper userMapper;

    @Override
    public boolean userReg(UserDTO userDTO) {
        //检查手机号是否被注册
        //取出用户输入的手机号
        String phone = userDTO.getPhone();
        User user = userMapper.findUserByPhone(phone);
        //根据手机号查找的对象是否为空，来判断手机号是否被注册
        if (user!=null){
            throw  new RuntimeException("手机号已经被注册");
        }
        String password = userDTO.getPassword();
        //对用户的输入的密码进行加密
        //生成一个盐值
        String salt = StringUtil.getRandomNumber(4);
        userDTO.setSalt(salt);
        String newPassword = StringUtil.md5Password(password, salt, 10);
        //将加密之后的password 赋值给UserDto 将之前用户输入的密码覆盖掉
        userDTO.setPassword(newPassword);
        userDTO.setState("0");
        int result = userMapper.insertUser(userDTO);

        return result>0;
    }
}
```

UserMapper

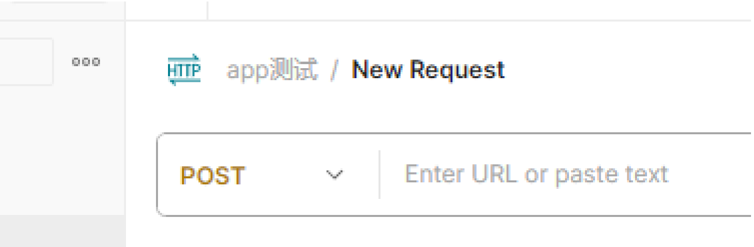
```
package com.zlt.app.mapper;

import com.zlt.app.dto.UserDTO;
import com.zlt.app.entity.User;
import org.apache.ibatis.annotations.Mapper;

@Mapper
public interface UserMapper {
    User findUserByPhone(String phone);

    int insertUser(UserDTO userDTO);
}
```

UserMapper.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.zlt.app.mapper.UserMapper" >
<select id="findUserByPhone" resultType="com.zlt.app.entity.User">
    select uid,user_name,did,phone,password,salt,state from user_info where
phone = #{phone}
</select>

    <insert id="insertUser" parameterType="com.zlt.app.dto.UserDTO">
        insert into user_info (user_name,did,phone,password,salt,state) values
(#{userName},#{did},#{phone},#{password},#{salt},#{state})
    </insert>

</mapper>
```

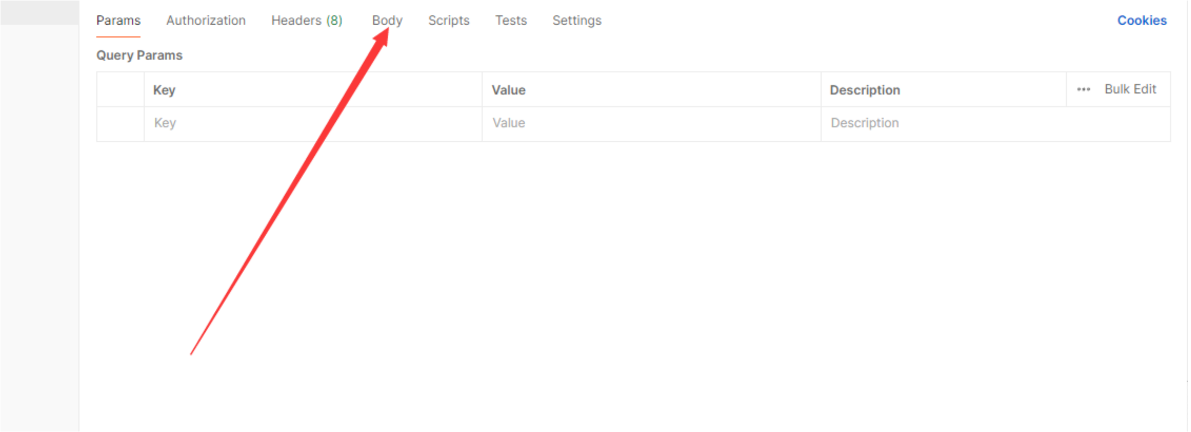
测试：

1切换请求为post


app测试 / New Request

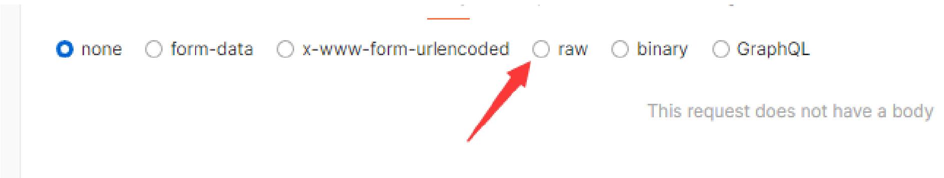
POST

2点击 body



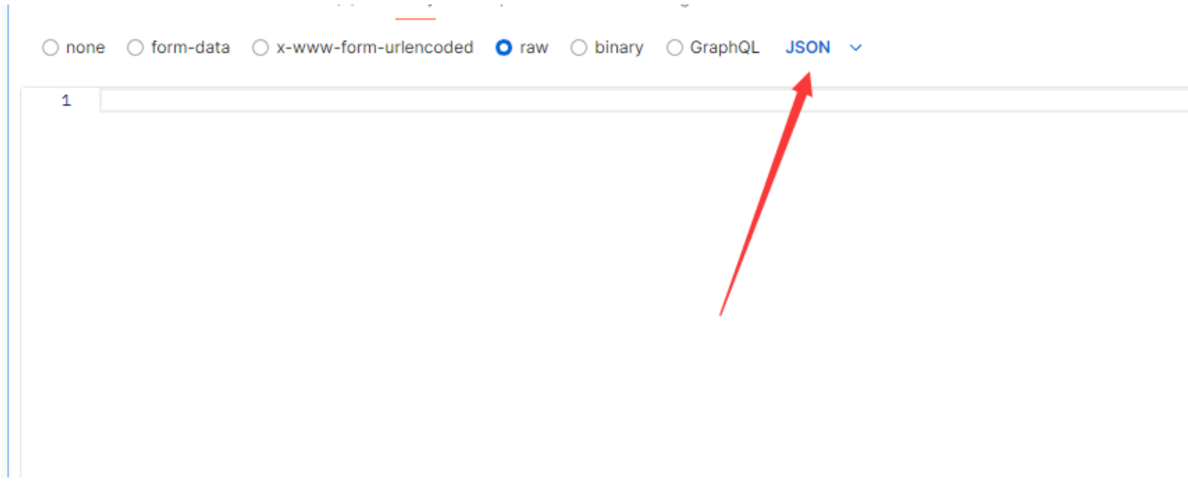
Key	Value	Description
Key	Value	Description

3点击raw

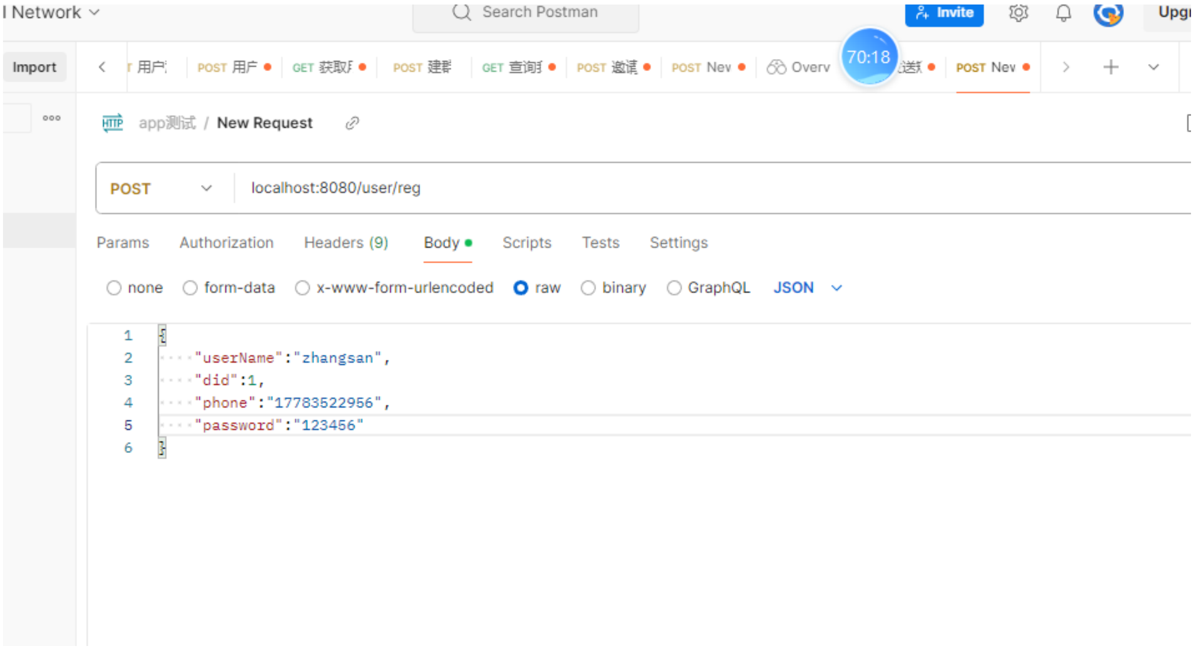

● none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

This request does not have a body

4选择json


○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

5传参



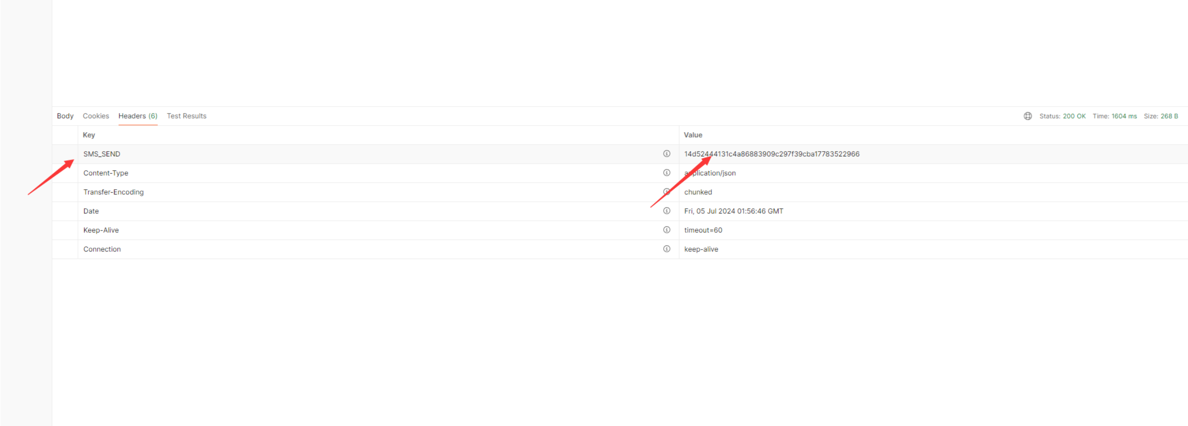
app测试 / New Request

POST localhost:8080/user/reg

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "userName":"zhangsan",
    "did":1,
    "phone":"17783522956",
    "password":"123456"
}
```

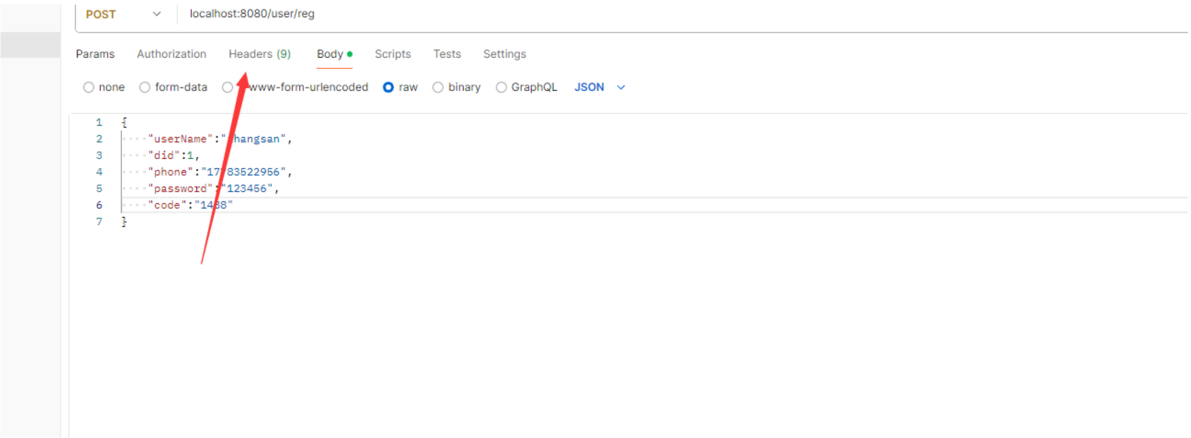
6 在发送验证码的响应体里面找到 SMS_SEND



Status: 200 OK Time: 1604 ms Size: 268 B

Key	Value
SMS_SEND	14d52444131c4a86883909c297f39cba17783522966
Content-Type	application/json
Transfer-Encoding	chunked
Date	Fri, 05 Jul 2024 01:56:46 GMT
Keep-Alive	timeout=60
Connection	keep-alive

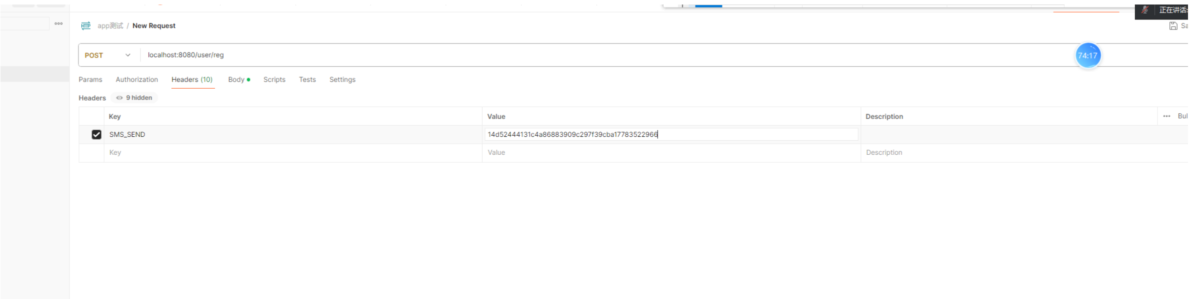
7将他们的值带到注册的请求头里面去



POST localhost:8080/user/reg

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "userName":"zhangsan",
    "did":1,
    "phone":"17783522956",
    "password":"123456",
    "code":"1488"
}
```



app测试 / New Request

POST localhost:8080/user/reg

Headers 9 hidden

	Key	Value	Description
☑	SMS_SEND	14d52444131c4a86883909c297f39cba17783522966	
	Key	Value	Description

8点击发送请求

登录代码

UserController

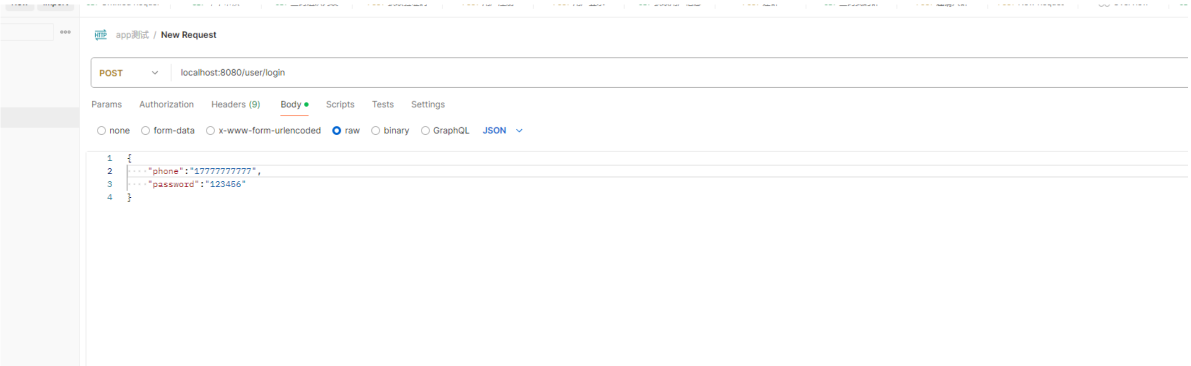
```
    /**
     * 登录
     */
    @PostMapping("login")
    public ResponseEntity<Result>login(@RequestBody UserDTO userDTO){
      try {
          String token = userService.login(userDTO);
          return
ResponseEntity.status(200).header("token",token).body(Result.success("登陆成功"));
      }catch (RuntimeException e){
         return ResponseEntity.status(200).body(Result.fail("账号或密码错误"));
      }
    }
```

UserService

```
String login(UserDTO userDTO);
```

UserServiceImpl

```
    @Override
    public String login(UserDTO userDTO) {
        //判断账号是否输入正确
        User user = userMapper.findUserByPhone(userDTO.getPhone());
        if (user==null){
            throw new RuntimeException("账号或密码错误");
        }
        //比对密码，将用户输入的密码，进行加密，比对加密之后的
        String password =
StringUtil.md5Password(userDTO.getPassword(),user.getSalt(),10);
        if (!password.equals(user.getPassword())){
            throw new RuntimeException("账号或密码错误");
        }
        //设置登录token
        String token = JWTUtil.sign(user.getPhone(), user.getPassword());
        redisTemplate.opsForValue().set("TOKEN_USER"+token,user,12,
TimeUnit.HOURS);
        return token;
    }
```



POST localhost:8080/user/login

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "phone":"17777777777",
    "password":"123456"
}
```