手机登录

## 一、前言

本打算实现手机验证码登录，但考虑到阿里云的手机短信服务现在已经不支持个人申请，出于学习目的，直接跳过验证码阶段。

无注册界面，第一次登录根据手机号判断系统是否存在，如果不存在则自动注册。

## 二、登录

### 1、在service模块中创建service-user模块

### 2、添加配置文件application.properties

1. # 服务端口
2. server.port=8203
3. # 服务名
4. spring.application.name=service-user
6. # 环境设置：dev、test、prod
7. spring.profiles.active=dev
9. # mysql数据库连接
10. spring.datasource.driver-class-name=com.mysql.jdbc.Driver
11. spring.datasource.url=jdbc:mysql://localhost:3306/yygh\_user?characterEncoding=utf-8&useSSL=false
12. spring.datasource.username=root
13. spring.datasource.password=root
15. #返回json的全局时间格式
16. spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
17. spring.jackson.time-zone=GMT+8
19. # nacos服务地址
20. spring.cloud.nacos.discovery.server-addr=localhost:8848
22. #配置mapper xml文件的路径
23. mybatis-plus.mapper-locations=classpath:com/xlh/yygh/user/mapper/xml/\*.xml

### 3、添加启动类

1. **package** com.xlh.yygh.user;
3. **import** org.springframework.boot.SpringApplication;
4. **import** org.springframework.boot.autoconfigure.SpringBootApplication;
5. **import** org.springframework.cloud.client.discovery.EnableDiscoveryClient;
6. **import** org.springframework.cloud.openfeign.EnableFeignClients;
7. **import** org.springframework.context.annotation.ComponentScan;
9. @SpringBootApplication
10. @ComponentScan(basePackages = "com.xlh")
11. @EnableDiscoveryClient
12. @EnableFeignClients(basePackages = "com.xlh")
13. **public** **class** ServiceUserApplication {
14. **public** **static** **void** main(String[] args) {
15. SpringApplication.run(ServiceUserApplication.**class**, args);
16. }
17. }

### 4、配置gateway网关

1. #设置路由id
2. spring.cloud.gateway.routes[2].id=service-user
3. #设置路由的uri
4. spring.cloud.gateway.routes[2].uri=lb://service-user
5. #设置路由断言,代理servicerId为auth-service的/auth/路径
6. spring.cloud.gateway.routes[2].predicates= Path=/\*/user/\*\*

### 5、添加用户基础类

### 6、添加UserInfoMapper和UserInfoMapper.xml

后者用于复杂sql语句

### 7、添加service和controller

## 三、登录前端

### 1、创建相关api，并安装cookie

npm install js-cookie

**说明：登录成功用户信息记录cookie**

其他细节详见代码。

## 四、用户认证与网关整合

**思路：**

1.所有请求都会经过服务网关，服务网关对外暴露服务，在网关进行统一用户认证；

2.既然要在网关进行用户认证，网关得知道对哪些url进行认证，所以我们得对ur制定规则

3.Api接口异步请求的，我们采取url规则匹配，如：/api/\*\*/auth/\*\*，如凡是满足该规则的都必须用户认证

### 1、调整server-gateway模块

#### 1.1 在服务网关添加fillter

代码见下方；

#### 1.2 在服务网关中判断用户登录状态

在网关中如何获取用户信息：

我们统一从header头信息中获取

如何判断用户信息合法：

登录时我们返回用户token，在服务网关中获取到token后，我在到redis中去查看用户id，如何用户id存在，则token合法，否则不合法

1. **package** com.xlh.yygh.gateway.filter;
3. **import** com.alibaba.fastjson.JSONObject;
4. **import** com.xlh.yygh.common.helper.JwtHelper;
5. **import** com.xlh.yygh.common.result.Result;
6. **import** com.xlh.yygh.common.result.ResultCodeEnum;
7. **import** org.springframework.cloud.gateway.filter.GatewayFilterChain;
8. **import** org.springframework.cloud.gateway.filter.GlobalFilter;
9. **import** org.springframework.core.Ordered;
10. **import** org.springframework.core.io.buffer.DataBuffer;
11. **import** org.springframework.http.server.reactive.ServerHttpRequest;
12. **import** org.springframework.http.server.reactive.ServerHttpResponse;
13. **import** org.springframework.stereotype.Component;
14. **import** org.springframework.util.AntPathMatcher;
15. **import** org.springframework.util.StringUtils;
16. **import** org.springframework.web.server.ServerWebExchange;
17. **import** reactor.core.publisher.Mono;
19. **import** java.nio.charset.StandardCharsets;
20. **import** java.util.List;
22. /\*\*
23. \* <p>
24. \* 全局Filter，统一处理会员登录与外部不允许访问的服务
25. \* </p>
26. \*
27. \* @author qy
28. \* @since 2019-11-21
29. \*/
30. @Component
31. **public** **class** AuthGlobalFilter **implements** GlobalFilter, Ordered {
33. **private** AntPathMatcher antPathMatcher = **new** AntPathMatcher();
35. @Override
36. **public** Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
37. ServerHttpRequest request = exchange.getRequest();
38. String path = request.getURI().getPath();
39. System.out.println("==="+path);
41. //内部服务接口，不允许外部访问
42. **if**(antPathMatcher.match("/\*\*/inner/\*\*", path)) {
43. ServerHttpResponse response = exchange.getResponse();
44. **return** out(response, ResultCodeEnum.PERMISSION);
45. }
47. Long userId = **this**.getUserId(request);
48. //api接口，异步请求，校验用户必须登录
49. **if**(antPathMatcher.match("/api/\*\*/auth/\*\*", path)) {
50. **if**(StringUtils.isEmpty(userId)) {
51. ServerHttpResponse response = exchange.getResponse();
52. **return** out(response, ResultCodeEnum.LOGIN\_AUTH);
53. }
54. }
55. **return** chain.filter(exchange);
56. }
58. @Override
59. **public** **int** getOrder() {
60. **return** 0;
61. }
63. /\*\*
64. \* api接口鉴权失败返回数据
65. \* @param response
66. \* @return
67. \*/
68. **private** Mono<Void> out(ServerHttpResponse response, ResultCodeEnum resultCodeEnum) {
69. Result result = Result.build(**null**, resultCodeEnum);
70. **byte**[] bits = JSONObject.toJSONString(result).getBytes(StandardCharsets.UTF\_8);
71. DataBuffer buffer = response.bufferFactory().wrap(bits);
72. //指定编码，否则在浏览器中会中文乱码
73. response.getHeaders().add("Content-Type", "application/json;charset=UTF-8");
74. **return** response.writeWith(Mono.just(buffer));
75. }
77. /\*\*
78. \* 获取当前登录用户id
79. \* @param request
80. \* @return
81. \*/
82. **private** Long getUserId(ServerHttpRequest request) {
83. String token = "";
84. List<String> tokenList = request.getHeaders().get("token");
85. **if**(**null**  != tokenList) {
86. token = tokenList.get(0);
87. }
88. **if**(!StringUtils.isEmpty(token)) {
89. **return** JwtHelper.getUserId(token);
90. }
91. **return** **null**;
92. }
93. }

### 2、调整前端yygh-site

请求服务器端接口时我们默认带上token，需要登录的接口如果token没有或者token过期，服务器端会返回208状态，然后发送登录事件打开登录弹出层登录。修改utils/request.js文件。