**6-19 ｜ 登录注册流程前端对接**

后台部署以下服务：

* qiyu-live-id-generate-provider
* qiyu-live-user-provider
* qiyu-live-msg-provider
* qiyu-live-gateway
* qiyu-live-api

**思考：接口中直接返回token给到我们的调用方需要注意什么？**

**Cookie的domain作用域设置**

**什么是domain？**

给cookie设置domain，可以保证，只有访问指定域名下的服务器，才会携带cookie，同时只有制定domain的页面，才能收到server端会写的cookie。

**设置cookie可能失效的几个原因**

* 后台没有设置跨域
* 后台没有设置关键配置：response.setHeader("Access-Control-Allow-Credentials", "true");

其实可以通过后端的HttpServletResponse直接将token写回给到调用方的cookie中进行存储，不过要记得设置好domain选项，保证该cookie是制定域名下专属的cookie：

|  |
| --- |
| Java @Override public WebResponseVO login(String phone, Integer code, HttpServletResponse response) {  if (StringUtils.isEmpty(phone)) {  return WebResponseVO.errorParam("手机号不能为空");  }  if (!Pattern.matches(PHONE\_REG, phone)) {  return WebResponseVO.errorParam("手机号格式异常");  }  if (code == null || code < 100000) {  return WebResponseVO.errorParam("验证码格式异常");  }  MsgCheckDTO msgCheckDTO = smsRpc.checkLoginCode(phone, code);  if (!msgCheckDTO.isCheckStatus()) {  return WebResponseVO.bizError(msgCheckDTO.getDesc());  }  //验证码校验通过  UserLoginDTO userLoginDTO = userPhoneRPC.login(phone);  Cookie cookie = new Cookie("qytk", userLoginDTO.getToken());  //设置顶级域名，接着我们的二级域名中进行web页面的访问，后续请求就会携带上它了  cookie.setDomain("qiyu.live.com");  cookie.setPath("/");  cookie.setMaxAge(30 \* 24 \* 3600);  response.setHeader("Access-Control-Allow-Credentials", "true");  response.addCookie(cookie);  return WebResponseVO.success(ConvertBeanUtils.convert(userLoginDTO, UserLoginVO.class)); } |

修改我们的前端页面，在js中开发登录接口调用和发短信接口调用功能：

主要改动点如下：

|  |
| --- |
| Java mobileLogin: function () {  if (this.code == '') {  this.$message.error('请输入验证码');  return;  }  var checkStatus = this.checkPhone();  if(!checkStatus) {  return;  }  var that = this;  let data = new FormData();  data.append("phone",this.mobile);  data.append("code",this.code);  //请求登录接口  httpPost(loginUrl,data).then(resp=>{  //登录成功  if(resp.code==200) {  that.userId=resp.data.userId;  that.$message.success('登录成功');  that.hiddenLoginPopNow();  that.isLogin=true;  that.userId=resp.data.userId;  } else {  that.$message.error(resp.msg);  }  })  },   sendSmsCode: function () {  if (this.hasSendSms) {  return;  }  console.log(this.mobile);  var checkStatus = this.checkPhone();  if(!checkStatus) {  return;  }  //发送验证码按钮文字调整  var that = this;  let data = new FormData();  data.append("phone",this.mobile);  //请求短信发送接口  httpPost(sendSmsUrl,data).then(resp=>{  if(resp.code==200) {  that.hasSendSms = true;  that.$message.success('短信发送成功');  var interval = setInterval(function () {  that.loginCodeBtn = '发送中(' + that.lastTime + ')';  if (that.lastTime == 0) {  that.lastTime = 60;  that.loginCodeBtn = '验证码';  that.hasSendSms = false;  console.log('清理定时器');  clearInterval(interval);  return;  } else {  that.lastTime = that.lastTime - 1;  }  }, 1000);  } else {  that.$message.error(resp.msg);  }  })  }, |

修改的代码内容如下：

|  |
| --- |
| Java new Vue({  el: '#app',  data: {  userId: 0,  showLoginPop: false,  loginCodeBtn: '验证码',  lastTime: 60,  mobile: '',  code: '',  hasSendSms: false,  livingRoomList: {},  isLogin: false,  initInfo: "登录",  listType: 0,  startLivingRoomTab: false  },   //页面初始化的时候会调用下这里面的方法  mounted() {  this.initPage();  },   methods: {  initPage:function() {  },  showLoginPopNow: function () {  this.showLoginPop = true;  },  hiddenLoginPopNow: function () {  this.showLoginPop = false;  },   mobileLogin: function () {  if (this.code == '') {  this.$message.error('请输入验证码');  return;  }  var checkStatus = this.checkPhone();  if(!checkStatus) {  return;  }  var that = this;  let data = new FormData();  data.append("phone",this.mobile);  data.append("code",this.code);  //请求登录接口  httpPost(loginUrl,data).then(resp=>{  //登录成功  if(resp.code==200) {  that.userId=resp.data.userId;  that.$message.success('登录成功');  that.hiddenLoginPopNow();  that.isLogin=true;  that.userId=resp.data.userId;  } else {  that.$message.error(resp.msg);  }  })  },   sendSmsCode: function () {  if (this.hasSendSms) {  return;  }  console.log(this.mobile);  var checkStatus = this.checkPhone();  if(!checkStatus) {  return;  }  //发送验证码按钮文字调整  var that = this;  let data = new FormData();  data.append("phone",this.mobile);  //请求短信发送接口  httpPost(sendSmsUrl,data).then(resp=>{  if(resp.code==200) {  that.hasSendSms = true;  that.$message.success('短信发送成功');  var interval = setInterval(function () {  that.loginCodeBtn = '发送中(' + that.lastTime + ')';  if (that.lastTime == 0) {  that.lastTime = 60;  that.loginCodeBtn = '验证码';  that.hasSendSms = false;  console.log('清理定时器');  clearInterval(interval);  return;  } else {  that.lastTime = that.lastTime - 1;  }  }, 1000);  } else {  that.$message.error(resp.msg);  }  })  },   checkPhone: function(){  let phoneReg = /(^(13[0-9]|14[01456879]|15[0-35-9]|16[2567]|17[0-8]|18[0-9]|19[0-35-9])\d{8}$)/;  if (this.mobile == '' || !phoneReg.test(this.mobile)) {  this.$message.error('手机号格式有误');  return false;  }  return true;  },   }  }) |