第7章 移动端开发-体检预约

# 第一章. 体检预约流程

用户可以通过如下操作流程进行体检预约：

1、在移动端首页点击体检预约，页面跳转到套餐列表页面

2、在套餐列表页面点击要预约的套餐，页面跳转到套餐详情页面

3、在套餐详情页面点击立即预约，页面跳转到预约页面

4、在预约页面录入体检人信息，包括手机号，点击发送验证码

5、在预约页面录入收到的手机短信验证码，点击提交预约，完成体检预约

# 第二章. 短信发送

## 短信服务介绍

目前市面上有很多第三方提供的短信服务，这些第三方短信服务会和各个运营商（移动、联通、电信）对接，我们只需要注册成为会员并且按照提供的开发文档进行调用就可以发送短信。需要说明的是这些短信服务都是收费的服务。

本项目短信发送我们选择的是阿里云提供的短信服务。

短信服务（Short Message Service）是阿里云为用户提供的一种通信服务的能力，支持快速发送短信验证码、短信通知等。 三网合一专属通道，与工信部携号转网平台实时互联。电信级运维保障，实时监控自动切换，到达率高达99%。短信服务API提供短信发送、发送状态查询、短信批量发送等能力，在短信服务控制台上添加签名、模板并通过审核之后，可以调用短信服务API完成短信发送等操作。

## 注册阿里云账号

阿里云官网：<https://www.aliyun.com/>

点击官网首页免费注册跳转到如下注册页面：



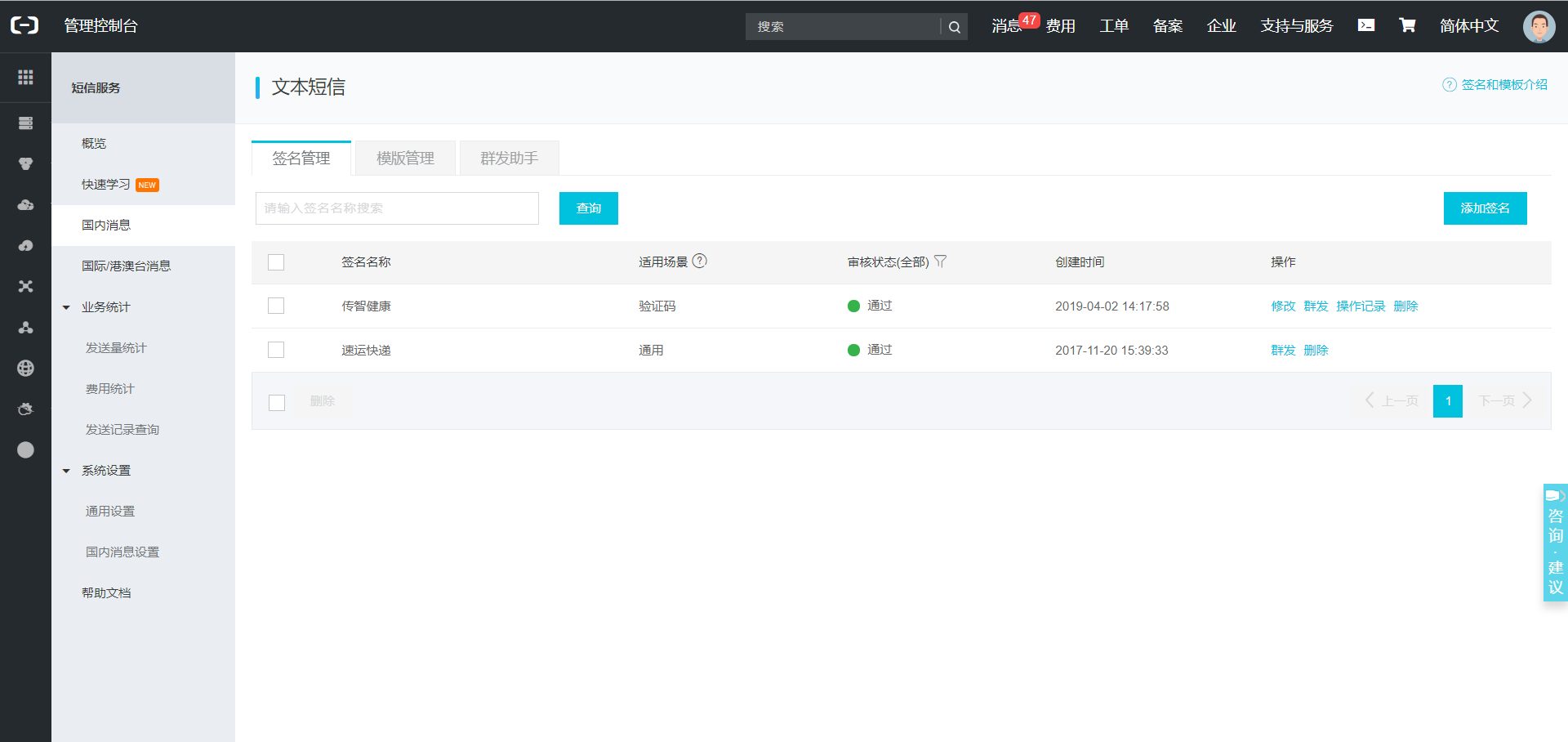
注册后，使用账号名登录



## 设置短信签名

注册成功后，点击登录按钮进行登录。登录后进入短信服务管理页面，选择国内消息菜单：

【签名】：



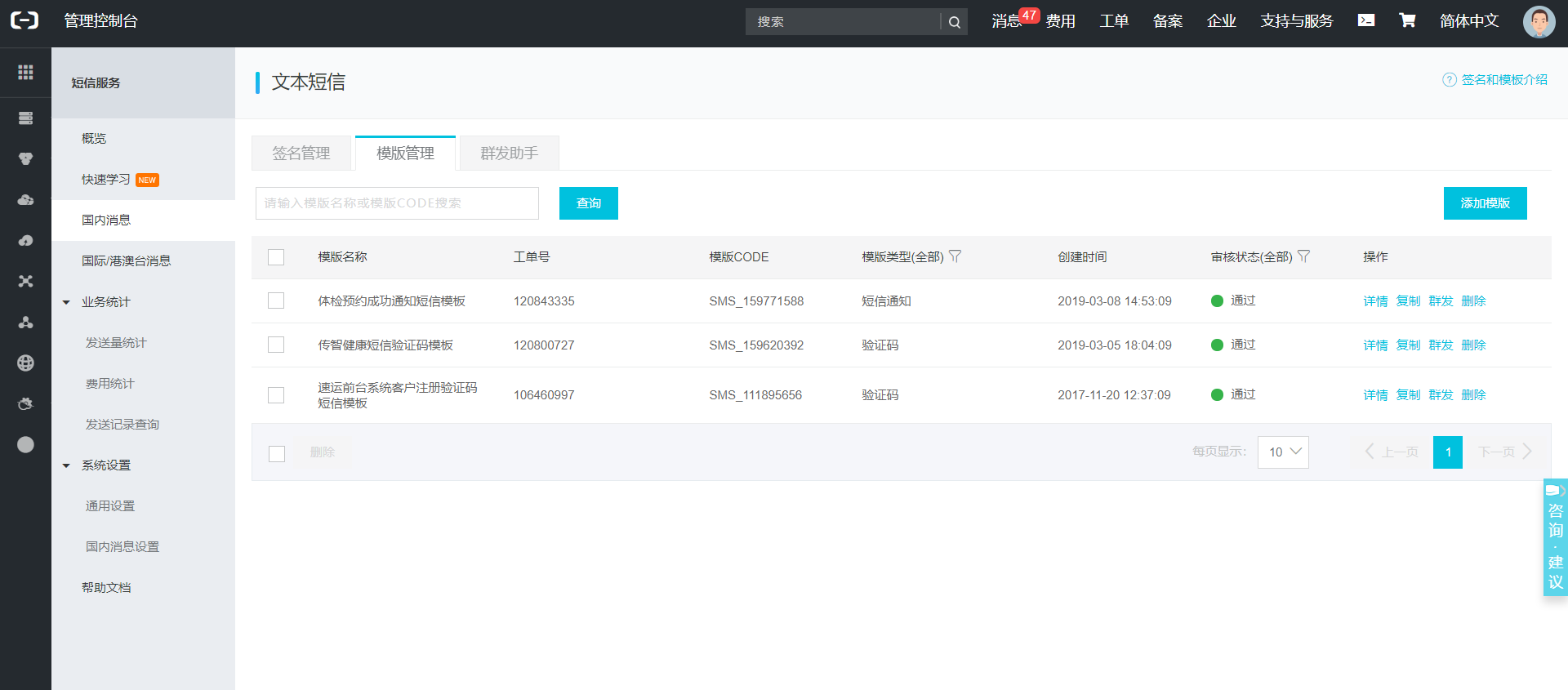
点击添加签名按钮：



目前个人用户只能申请适用场景为验证码的签名

## 设置短信模板

在国内消息菜单页面中，点击模板管理标签页：



点击添加模板按钮：

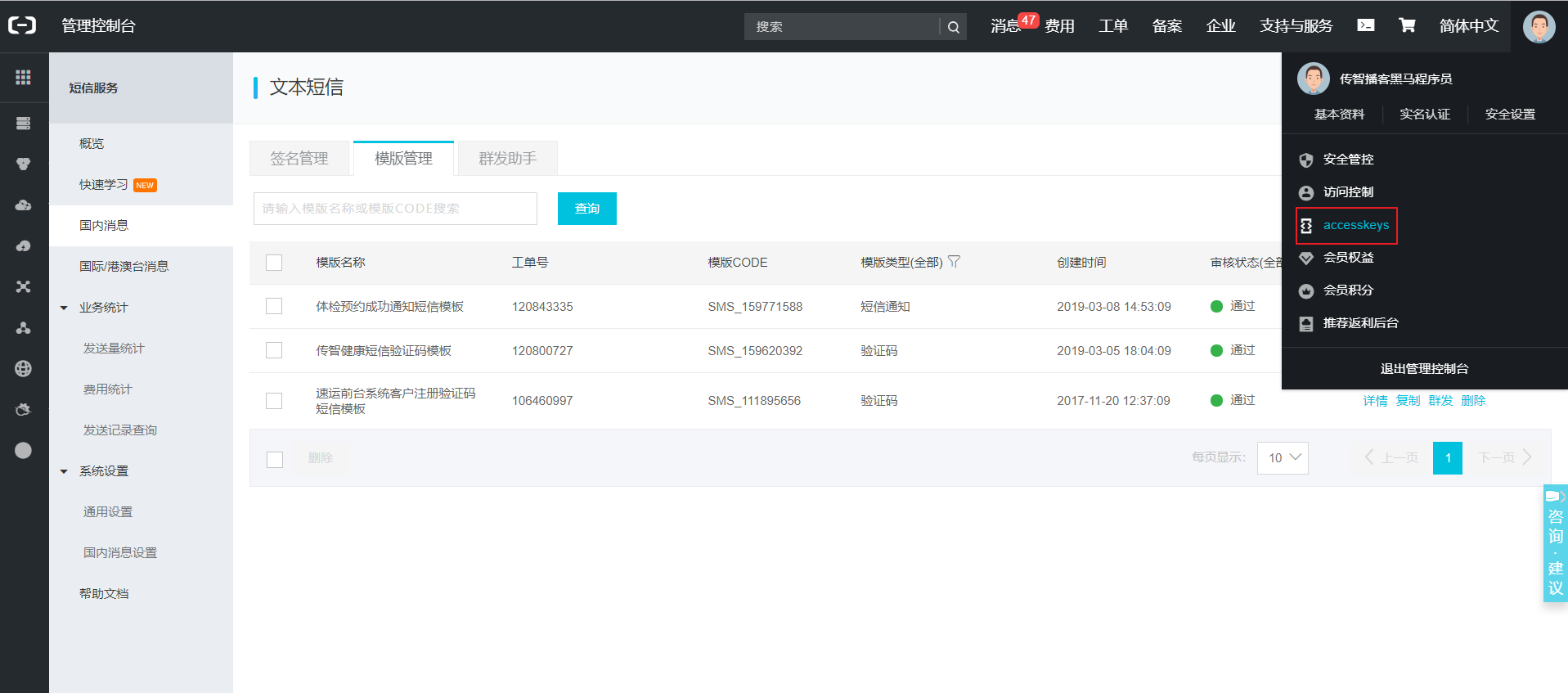


模板内容：传智健康 验证码${number}，您正进行传智健康系统的身份验证，打死不告诉别人！

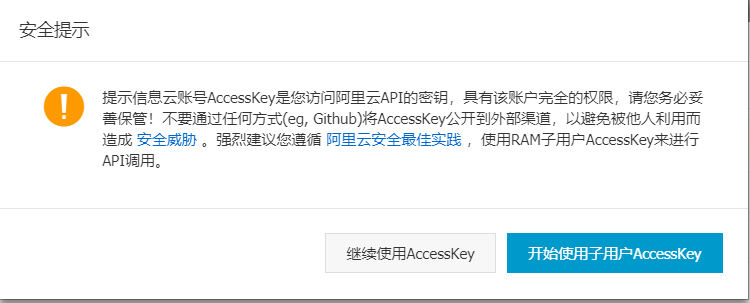
其中${number}为动态参数。

## 设置access keys

在发送短信时需要进行身份认证，只有认证通过才能发送短信。本小节就是要设置用于发送短信时进行身份认证的key和密钥。鼠标放在页面右上角当前用户头像上，会出现下拉菜单：



点击accesskeys：

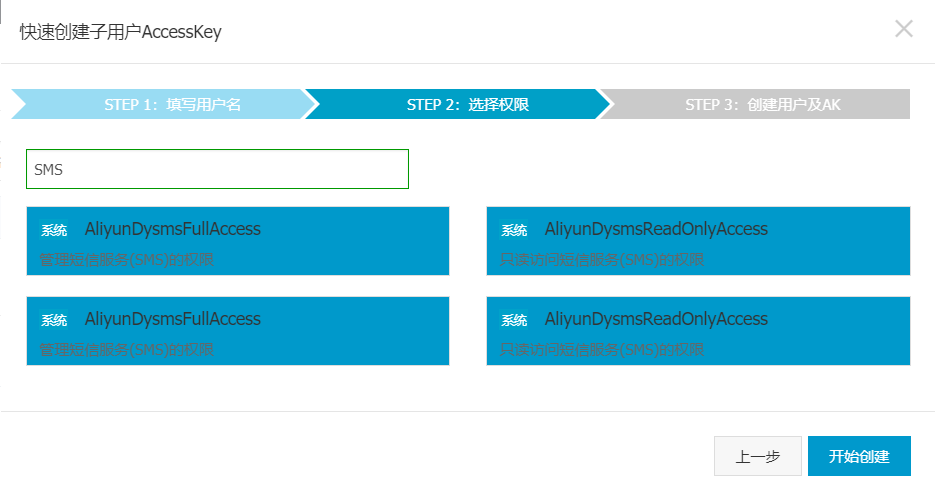


点击“开始使用子用户AccessKey”按钮：



用户名随便填写。

搜索“SMS”，表示短信服务，选择权限，点击“开始创建”



发送短信验证码

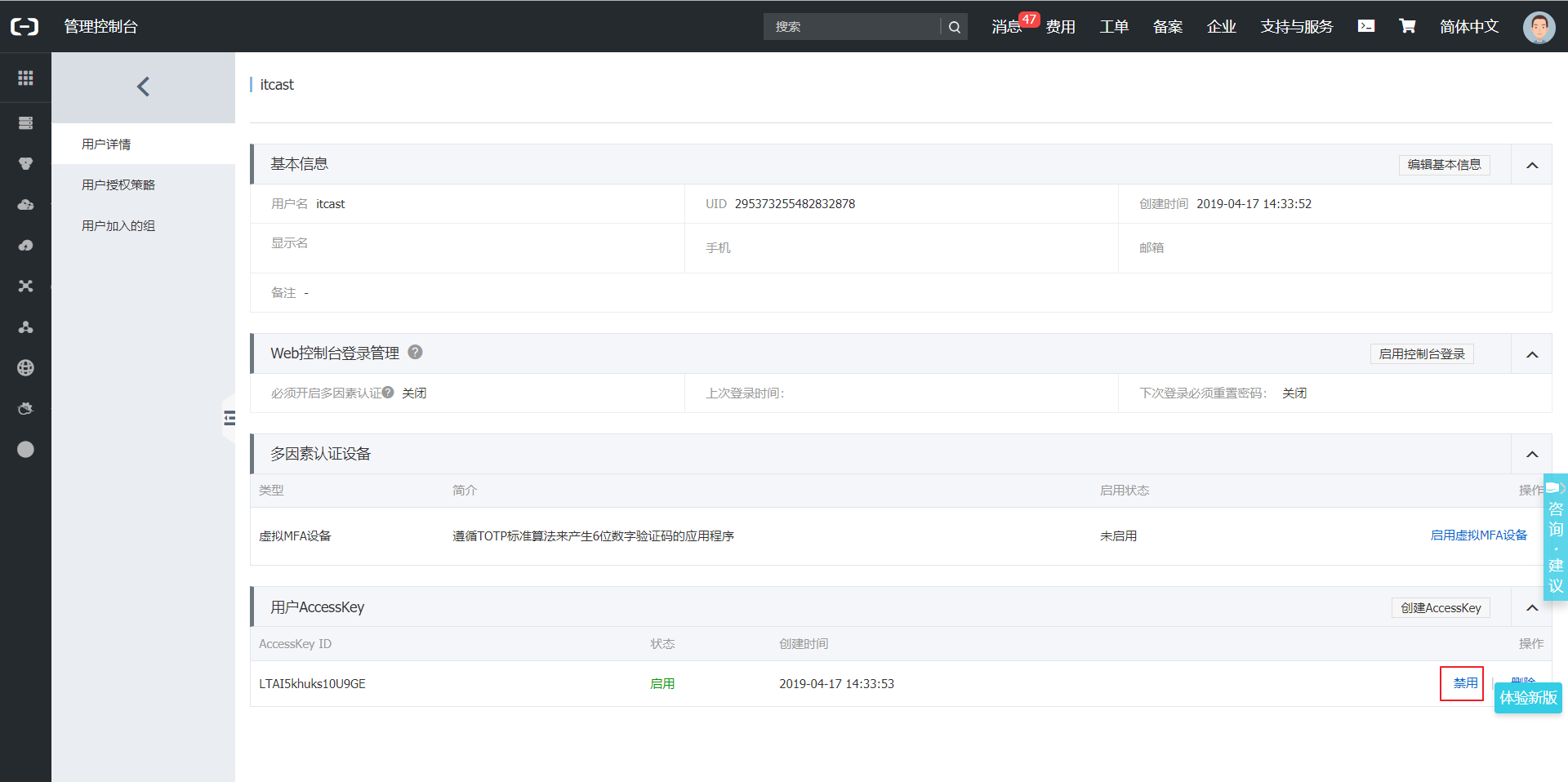




创建成功，其中AccessKeyID为访问短信服务时使用的ID，AccessKeySecret为密钥。

注意：需要马上保存AccessKeyID和AccessKeySecret，因为处于安全考虑，这个只显示1次，一旦退出页面就不再显示了。

点击“查看用户详情”，可以在用户详情页面下禁用刚刚创建的AccessKey：

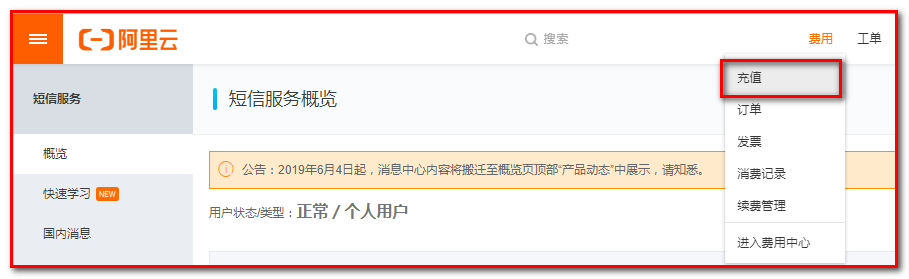


在短信服务中，点击“国内消息设置”。可以设置每日和每月短信发送上限：



由于短信服务是收费服务，所以还需要进行充值才能发送短信：

在费用中，点击“充值”





## 发送短信

### 导入maven坐标

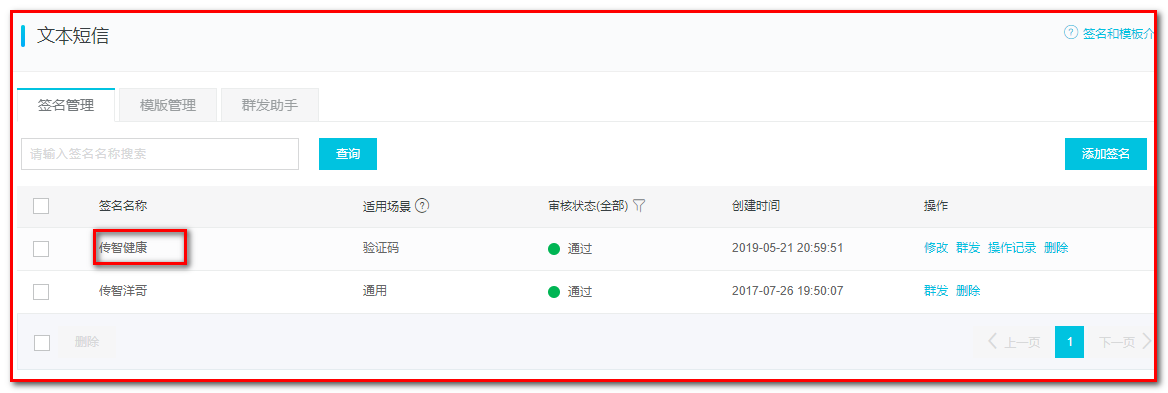
在health\_common中导入坐标

<dependency>  
  <groupId>com.aliyun</groupId>  
  <artifactId>aliyun-java-sdk-core</artifactId>  
  <version>3.3.1</version>  
</dependency>  
<dependency>  
  <groupId>com.aliyun</groupId>  
  <artifactId>aliyun-java-sdk-dysmsapi</artifactId>  
  <version>1.0.0</version>  
</dependency>

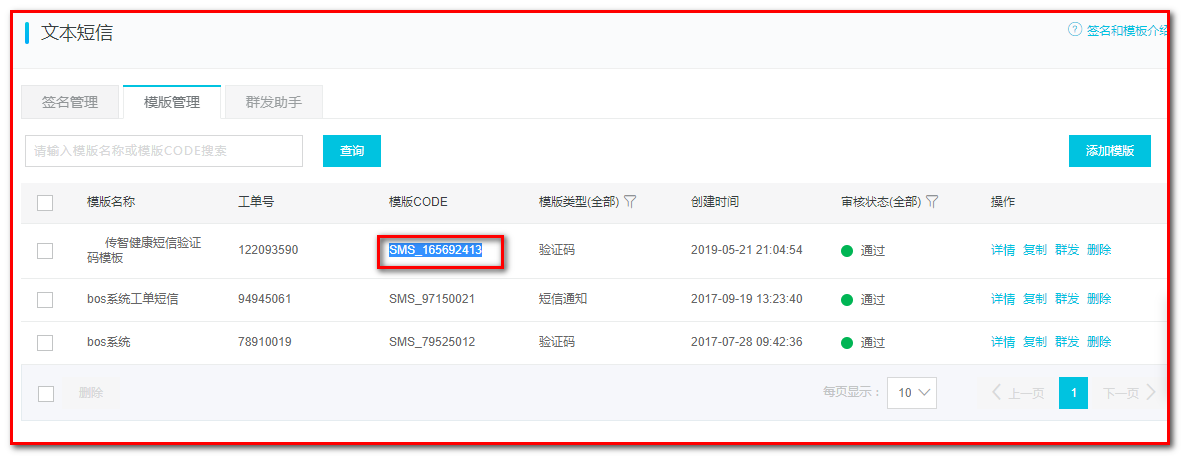
### 封装工具类

在health\_common中添加工具类

1：签名



2：模板code

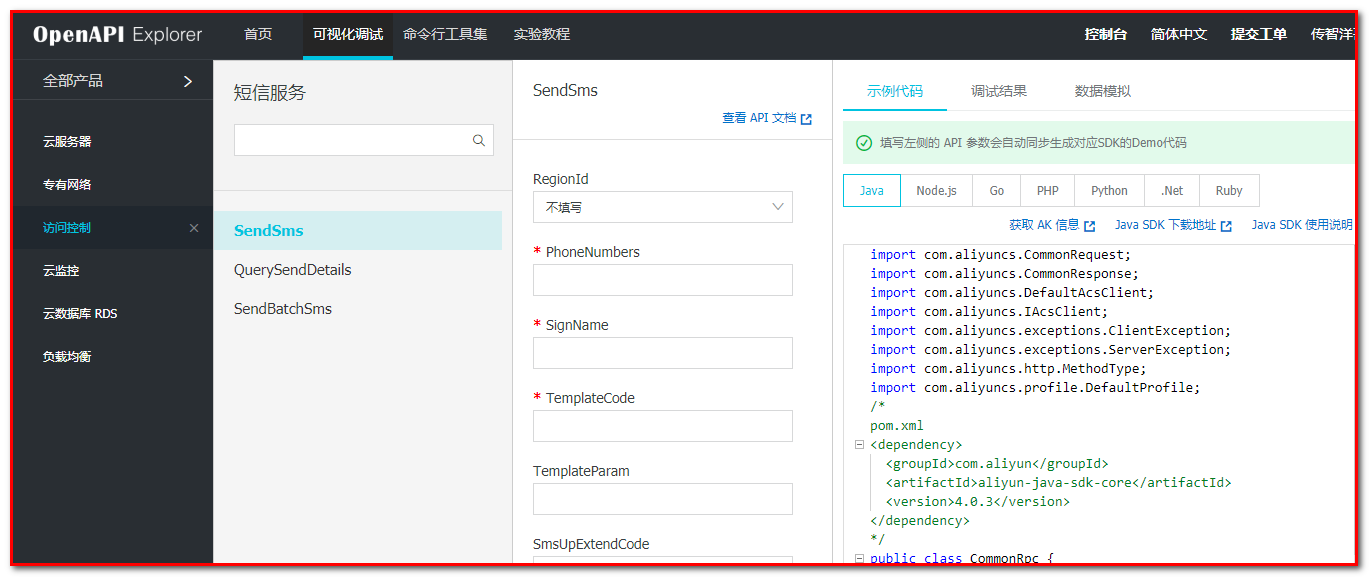


Api参考：



点击“试用api”

[https://api.aliyun.com/new?spm=a2c4g.11186623.2.13.314a19d9r0rEgr#/?product=Dysmsapi](https://api.aliyun.com/new?spm=a2c4g.11186623.2.13.314a19d9r0rEgr" \l "/?product=Dysmsapi)

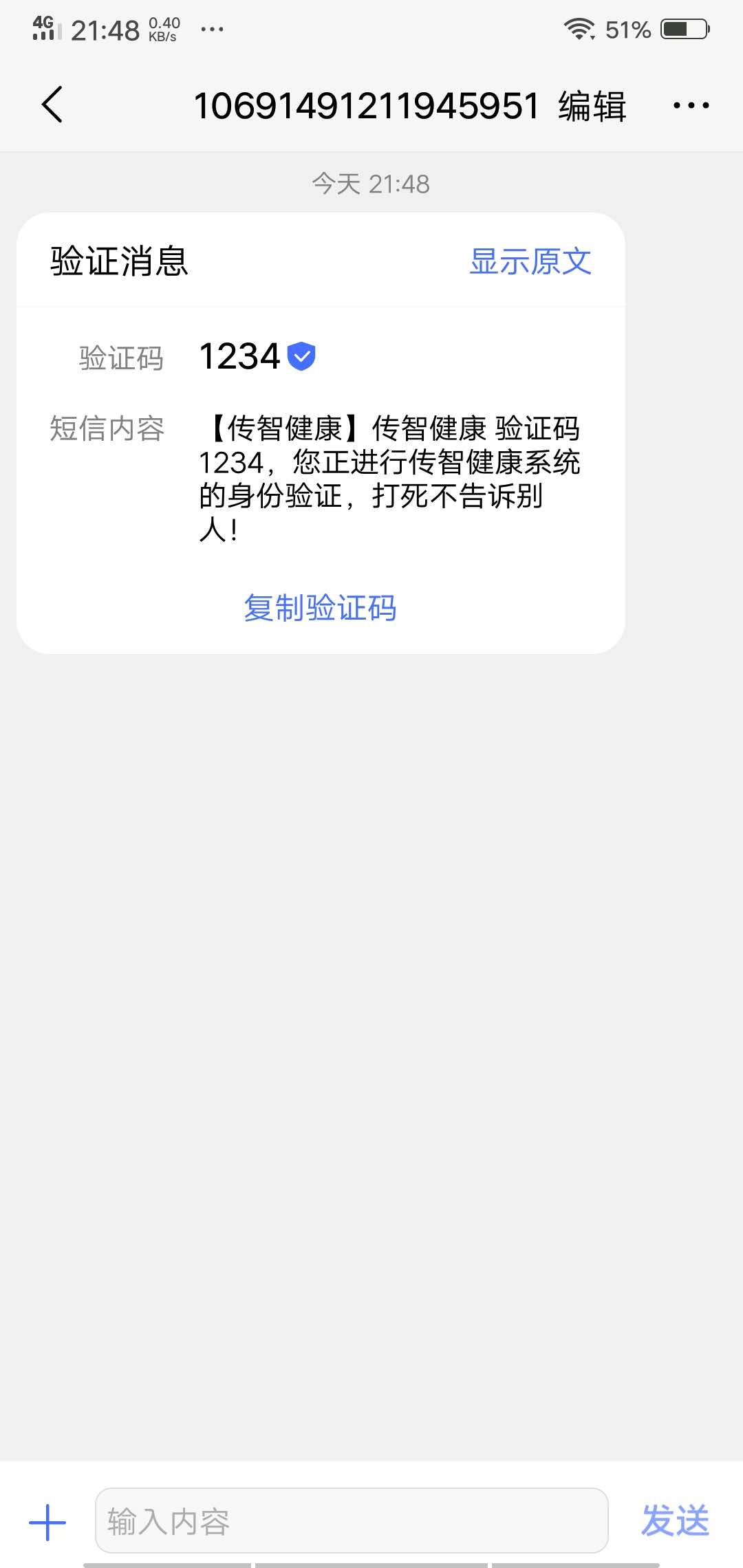


**package** com.itheima.health.utils;  
  
  
**import** com.aliyuncs.DefaultAcsClient;  
**import** com.aliyuncs.IAcsClient;  
**import** com.aliyuncs.dysmsapi.model.v20170525.SendSmsRequest;  
**import** com.aliyuncs.dysmsapi.model.v20170525.SendSmsResponse;  
**import** com.aliyuncs.exceptions.ClientException;  
**import** com.aliyuncs.http.MethodType;  
**import** com.aliyuncs.profile.DefaultProfile;  
**import** com.aliyuncs.profile.IClientProfile;  
  
*/\*\*  
 \* 短信发送工具类  
 \*/***public class** SMSUtils {  
 **public static final** String ***VALIDATE\_CODE*** = **"SMS\_159620392"**;*//发送短信验证码* **public static final** String ***ORDER\_NOTICE*** = **"SMS\_159771588"**;*//体检预约成功通知  
  
 /\*\*  
 \* 发送短信  
 \** ***@param phoneNumbers*** *\** ***@param param*** *\** ***@throws*** *ClientException  
 \*/* **public static void** sendShortMessage(String templateCode,String phoneNumbers,String param) **throws** ClientException{  
 *// 设置超时时间-可自行调整* System.*setProperty*(**"sun.net.client.defaultConnectTimeout"**, **"10000"**);  
 System.*setProperty*(**"sun.net.client.defaultReadTimeout"**, **"10000"**);  
 *// 初始化ascClient需要的几个参数* **final** String product = **"Dysmsapi"**;*// 短信API产品名称（短信产品名固定，无需修改）* **final** String domain = **"dysmsapi.aliyuncs.com"**;*// 短信API产品域名（接口地址固定，无需修改）  
 // 替换成你的AK* **final** String accessKeyId = **"accessKeyId"**;*// 你的accessKeyId,参考本文档步骤2* **final** String accessKeySecret = **"accessKeySecret"**;*// 你的accessKeySecret，参考本文档步骤2  
 // 初始化ascClient,暂时不支持多region（请勿修改）* IClientProfile profile = DefaultProfile.*getProfile*(**"cn-hangzhou"**, accessKeyId, accessKeySecret);  
 DefaultProfile.*addEndpoint*(**"cn-hangzhou"**, **"cn-hangzhou"**, product, domain);  
 IAcsClient acsClient = **new** DefaultAcsClient(profile);  
 *// 组装请求对象* SendSmsRequest request = **new** SendSmsRequest();  
 *// 使用post提交* request.setMethod(MethodType.***POST***);  
 *// 必填:待发送手机号。支持以逗号分隔的形式进行批量调用，批量上限为1000个手机号码,批量调用相对于单条调用及时性稍有延迟,验证码类型的短信推荐使用单条调用的方式* request.setPhoneNumbers(phoneNumbers);  
 *// 必填:短信签名-可在短信控制台中找到* request.setSignName(**"传智健康"**);  
 *// 必填:短信模板-可在短信控制台中找到* request.setTemplateCode(templateCode);  
 *// 可选:模板中的变量替换JSON串,如模板内容为"亲爱的${name},您的验证码为${code}"时,此处的值为  
 // 友情提示:如果JSON中需要带换行符,请参照标准的JSON协议对换行符的要求,比如短信内容中包含\r\n的情况在JSON中需要表示成\\r\\n,否则会导致JSON在服务端解析失败* request.setTemplateParam(**"{\"number\":\""**+param+**"\"}"**);  
 *// 可选-上行短信扩展码(扩展码字段控制在7位或以下，无特殊需求用户请忽略此字段)  
 // request.setSmsUpExtendCode("90997");  
 // 可选:outId为提供给业务方扩展字段,最终在短信回执消息中将此值带回给调用者  
 // request.setOutId("yourOutId");  
 // 请求失败这里会抛ClientException异常* SendSmsResponse sendSmsResponse = acsClient.getAcsResponse(request);  
 **if** (sendSmsResponse.getCode() != **null** && sendSmsResponse.getCode().equals(**"OK"**)) {  
 *// 请求成功* System.***out***.println(**"请求成功"**);  
 }  
 }  
}

### 测试短信发送

**package** com.itheima.health.main;  
  
**import** com.itheima.health.utils.SMSUtils;  
  
  
**public class** SMSMain {  
 **public static void** main(String[] args) **throws** Exception {  
 SMSUtils.*sendShortMessage*(**"SMS\_165692413"**,**"13269210655"**,**"1234"**);  
 }  
}

测试：查看手机



# 第三章. 体检预约

## 页面调整



1：在详情页面（/pages/setmeal\_detail.html）点击体检预约

<**div class="box-button"**>  
 <**a @click="toOrderInfo()" class="order-btn"**>立即预约</**a**>  
</**div**>

toOrderInfo()方法：

toOrderInfo(){  
 **window**.**location**.**href** = **"orderInfo.html?id="** + ***id***;  
}

2：在预约页面（/pages/orderInfo.html）进行调整

### 展示预约的套餐信息

第一步：从请求路径中获取当前套餐的id

<**script**>  
 **var *id*** = *getUrlParam*(**"id"**);  
</**script**>

第二步：定义模型数据setmeal，用于套餐数据展示

<**script**>  
 **var *vue*** = **new** *Vue*({  
 **el**:**'#app'**,  
 **data**:{

**imgUrl**:**null**,*//套餐对应的图片链接*  
 **setmeal**:{},*//套餐信息* **orderInfo**:{  
 **setmealId**:***id***,  
 **sex**:**'1'** }*//预约信息* }

});  
</**script**>

<**div class="card"**>  
 <**div class="project-img"**>  
 <**img :src="imgUrl" width="100%" height="100%"** />  
 </**div**>

<**div class="project-text"**>  
 <**h4 class="tit"**>{{setmeal.name}}</**h4**>  
 <**p class="subtit"**>{{setmeal.remark}}</**p**>  
 <**p class="keywords"**><**span**>女</**span**><**span**>20-70</**span**></**p**>  
 </**div**>  
 <**div class="project-know"**>  
 <**a href="orderNotice.html" class="link-page"**>  
 <**i class="icon-ask-circle"**><**span class="path1"**></**span**><**span class="path2"**></**span**></**i**>  
 <**span class="word"**>预约须知</**span**>  
 <**span class="arrow"**><**i class="icon-rit-arrow"**></**i**></**span**>  
 </**a**>  
 </**div**>  
</**div**>

第三步：在VUE的钩子函数中发送ajax请求，根据id查询套餐信息

mounted(){  
 **axios**.post(**"/setmeal/findById.do?id="** + ***id***).then((response) => {  
 **this**.**setmeal** = response.**data**.**data**;

**this**.**imgUrl** = **'http://pspmuf4c3.bkt.clouddn.com/'** + **this**.**setmeal**.**img**;  
 });  
}

### 手机号校验

第一步：在orderInfo.html页面导入的healthmobile.js文件中已经定义了校验手机号的方法

<**script src="../plugins/healthmobile.js"**></**script**>

healthmobile.js：

*/\*\*  
 \* 手机号校验  
 1--以1为开头；  
 2--第二位可为3,4,5,7,8,中的任意一位；  
 3--最后以0-9的9个整数结尾。  
 \*/***function** *checkTelephone*(telephone) {  
 **var** reg=/^[1][3,4,5,7,8][0-9]{9}$/;  
 **if** (!reg.test(telephone)) {  
 **return false**;  
 } **else** {  
 **return true**;  
 }  
}

第二步：为发送验证码按钮绑定事件sendValidateCode

<**div class="input-row"**>  
 <**label**>手机号</**label**>  
 <**input v-model="orderInfo.telephone" type="text" class="input-clear" placeholder="请输入手机号"**>  
 <**input style="font-size**: **x-small**;**" id="validateCodeButton" @click="sendValidateCode()" type="button" value="发送验证码"**>  
</**div**>  
<**div class="input-row"**>  
 <**label**>验证码</**label**>  
 <**input v-model="orderInfo.validateCode" type="text" class="input-clear" placeholder="请输入验证码"**>  
</**div**>

sendValidateCode()方法：

对手机号进行校验

*//发送验证码*sendValidateCode(){  
 *//获取用户输入的手机号* **var** telephone = **this**.**orderInfo**.telephone;  
 *//校验手机号输入是否正确* **if** (!*checkTelephone*(telephone)) {  
 **this**.**$message**.error(**'请输入正确的手机号'**);  
 **return false**;  
 }  
},

### 30秒倒计时效果

前面在sendValidateCode方法中进行了手机号校验，如果校验通过，需要显示30秒倒计时效果

*//发送验证码*sendValidateCode(){  
 *//获取用户输入的手机号* **var** telephone = **this**.**orderInfo**.telephone;  
 *//校验手机号输入是否正确* **if** (!*checkTelephone*(telephone)) {  
 **this**.**$message**.error(**'请输入正确的手机号'**);  
 **return false**;  
 }  
 ***validateCodeButton*** = *$*(**"#validateCodeButton"**)[0];  
 ***clock*** = **window**.setInterval(*doLoop*, 1000); *//一秒执行一次*  
},

其中，validateCodeButton和clock是在healthmobile.js文件中定义的属性，doLoop是在healthmobile.js文件中定义的方法

**var *clock*** = **''**;*//定时器对象，用于页面30秒倒计时效果***var *nums*** = 30;  
**var *validateCodeButton***;  
*//基于定时器实现30秒倒计时效果***function** *doLoop*() {  
 ***validateCodeButton***.**disabled** = **true**;*//将按钮置为不可点击* ***nums***--;  
 **if** (***nums*** > 0) {  
 ***validateCodeButton***.**value** = ***nums*** + **'秒后重新获取'**;  
 } **else** {  
 clearInterval(***clock***); *//清除js定时器* ***validateCodeButton***.**disabled** = **false**;  
 ***validateCodeButton***.**value** = **'重新获取验证码'**;  
 ***nums*** = 30; *//重置时间* }  
}

### 发送ajax请求

*//发送验证码*sendValidateCode(){  
 *//获取用户输入的手机号* **var** telephone = **this**.**orderInfo**.telephone;  
 *//校验手机号输入是否正确* **if** (!*checkTelephone*(telephone)) {  
 **this**.**$message**.error(**'请输入正确的手机号'**);  
 **return false**;  
 }  
 ***validateCodeButton*** = *$*(**"#validateCodeButton"**)[0];  
 ***clock*** = **window**.setInterval(*doLoop*, 1000); *//一秒执行一次* **axios**.post(**"/validateCode/send4Order.do?telephone="** + telephone).then((response) => {  
 **if**(!response.**data**.**flag**){  
 *//验证码发送失败* **this**.**$message**.error(**'验证码发送失败，请检查手机号输入是否正确'**);  
 }  
 });  
},

创建ValidateCodeController，提供方法发送短信验证码，并将验证码保存到redis

**package** com.itheima.health.controller;  
  
**import** com.aliyuncs.exceptions.ClientException;  
**import** com.itheima.health.constant.MessageConstant;  
**import** com.itheima.health.constant.RedisMessageConstant;  
**import** com.itheima.health.entity.Result;  
**import** com.itheima.health.utils.SMSUtils;  
**import** com.itheima.health.utils.ValidateCodeUtils;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
**import** redis.clients.jedis.JedisPool;  
  
*/\*\*  
 \* 短信验证码  
 \*/*@RestController  
@RequestMapping(**"/validateCode"**)  
**public class** ValidateCodeController {  
 @Autowired  
 **private** JedisPool **jedisPool**;  
  
 *//体检预约时发送手机验证码* @RequestMapping(**"/send4Order"**)  
 **public** Result send4Order(String telephone){  
 Integer code = ValidateCodeUtils.*generateValidateCode*(4);*//生成4位数字验证码* **try** {  
 *//发送短信* SMSUtils.*sendShortMessage*(SMSUtils.***VALIDATE\_CODE***,telephone,code.toString());  
 } **catch** (ClientException e) {  
 e.printStackTrace();  
 *//验证码发送失败* **return new** Result(**false**, MessageConstant.***SEND\_VALIDATECODE\_FAIL***);  
 }  
 System.***out***.println(**"发送的手机验证码为："** + code);  
 *//将生成的验证码缓存到redis* **jedisPool**.getResource().setex(  
 telephone + RedisMessageConstant.***SENDTYPE\_ORDER***,5 \* 60,code.toString());  
 *//验证码发送成功* **return new** Result(**true**,MessageConstant.***SEND\_VALIDATECODE\_SUCCESS***);  
 }  
}

### 日历展示

页面中使用DatePicker控件来展示日历。根据需求，最多可以提前一个月进行体检预约，所以日历控件只展示未来一个月的日期

第一步：引入dataPicker.js

<**script src="../plugins/datapicker/datePicker.js"**></**script**>

第二步：定义体检日期

<**div class="date"**>  
 <**label**>体检日期</**label**>  
 <**i class="icon-date" class="picktime"**></**i**>  
 <**input v-model="orderInfo.orderDate" type="text" class="picktime" readonly**>  
</**div**>

第三步：定义日期控件

<**script**>  
 *//日期控件* **var *calendar*** = **new datePicker**();  
 ***calendar***.init({  
 **'trigger'**: **'.picktime'**,*/\*按钮选择器，用于触发弹出插件\*/* **'type'**: **'date'**,*/\*模式：date日期；datetime日期时间；time时间；ym年月；\*/* **'minDate'**: *getSpecifiedDate*(**new** Date(),1),*/\*最小日期\*/* **'maxDate'**: *getSpecifiedDate*(**new** Date(),30),*/\*最大日期\*/* **'onSubmit'**: **function**() { */\*确认时触发事件\*/  
 //var theSelectData = calendar.value;* },  
 **'onClose'**: **function**() { */\*取消时触发事件\*/* }  
 });  
</**script**>

其中getSpecifiedDate方法定义在healthmobile.js文件中

*//获得指定日期后指定天数的日期***function** *getSpecifiedDate*(date,days) {  
 date.setDate(date.getDate() + days);*//获取指定天之后的日期* **var** year = date.getFullYear();  
 **var** month = date.getMonth() + 1;  
 **var** day = date.getDate();  
 **return** (year + **"-"** + month + **"-"** + day);  
}



### 提交预约请求

为提交预约按钮绑定事件

第一步：定义“体检预约”

<**div class="box-button"**>  
 <**button @click="submitOrder()" type="button" class="btn order-btn"**>提交预约</**button**>  
</**div**>

第二步：submitOrder()方法

*//提交预约*submitOrder(){  
 *//校验身份证号格式* **if**(!*checkIdCard*(**this**.**orderInfo**.idCard)){  
 **this**.**$message**.error(**'身份证号码输入错误，请重新输入'**);  
 **return** ;  
 }  
 **axios**.post(**"/order/submit.do"**,**this**.**orderInfo**).then((response) => {  
 **if**(response.**data**.**flag**){  
 *//预约成功，跳转到预约成功页面* **window**.**location**.**href**=**"orderSuccess.html?orderId="** + response.**data**.**data**.**id**;  
 }**else**{  
 *//预约失败，提示预约失败信息* **this**.**$message**.error(response.**data**.**message**);  
 }  
 });  
}

其中checkIdCard方法是在healthmobile.js文件中定义的

*/\*\*  
 \* 身份证号码校验  
 \* 身份证号码为15位或者18位，15位时全为数字，18位前17位为数字，最后一位是校验位，可能为数字或字符X  
 \*/***function** *checkIdCard*(idCard){  
 **var** reg = /(^\d{15}$)|(^\d{18}$)|(^\d{17}(\d|X|x)$)/;  
 **if**(reg.test(idCard)){  
 **return true**;  
 }**else**{  
 **return false**;  
 }  
}

## 后台代码

### Controller

在healthmobile\_web工程中创建OrderController并提供submitOrder方法

**package** com.itheima.health.controller;  
  
**import** com.alibaba.dubbo.config.annotation.Reference;  
**import** com.aliyuncs.exceptions.ClientException;  
**import** com.itheima.health.constant.MessageConstant;  
**import** com.itheima.health.constant.RedisMessageConstant;  
**import** com.itheima.health.entity.Result;  
**import** com.itheima.health.pojo.Order;  
**import** com.itheima.health.utils.SMSUtils;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.web.bind.annotation.RequestBody;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
**import** redis.clients.jedis.JedisPool;  
  
**import** java.util.Map;  
  
*/\*\*  
 \* 体检预约  
 \*/*@RestController  
@RequestMapping(**"/order"**)  
**public class** OrderController {  
 @Reference  
 **private** OrderService **orderService**;  
 @Autowired  
 **private** JedisPool **jedisPool**;  
  
 */\*\*  
 \* 体检预约  
 \** ***@param map*** *\** ***@return*** *\*/* @RequestMapping(**"/submit"**)  
 **public** Result submitOrder(@RequestBody Map map){  
 String telephone = (String) map.get(**"telephone"**);  
 *//从Redis中获取缓存的验证码，key为手机号+RedisConstant.SENDTYPE\_ORDER* String codeInRedis = **jedisPool**.getResource().get(  
 telephone + RedisMessageConstant.***SENDTYPE\_ORDER***);  
 String validateCode = (String) map.get(**"validateCode"**);  
 *//校验手机验证码* **if**(codeInRedis == **null** || !codeInRedis.equals(validateCode)){  
 **return new** Result(**false**, MessageConstant.***VALIDATECODE\_ERROR***);  
 }  
 Result result =**null**;  
 *//调用体检预约服务* **try**{  
 map.put(**"orderType"**, Order.***ORDERTYPE\_WEIXIN***);  
 result = **orderService**.order(map);  
 }**catch** (Exception e){  
 e.printStackTrace();  
 *//预约失败* **return** result;  
 }  
 **if**(result.isFlag()){  
 *//预约成功，发送短信通知* String orderDate = (String) map.get(**"orderDate"**);  
 **try** {  
 SMSUtils.*sendShortMessage*(SMSUtils.***ORDER\_NOTICE***,telephone,orderDate);  
 } **catch** (ClientException e) {  
 e.printStackTrace();  
 }  
 }  
 **return** result;  
 }  
}

### 服务接口

在health\_interface工程中创建体检预约服务接口OrderService并提供预约方法

**package** com.itheima.health.service;  
  
**import** com.itheima.health.entity.Result;  
  
**import** java.util.Map;  
  
*/\*\*  
 \* 体检预约服务接口  
 \*/***public interface** OrderService {  
 Result order(Map map) **throws** Exception;  
}

### 3.2.3 服务实现类

在health\_service工程中创建体检预约服务实现类OrderServiceImpl并实现体检预约方法。

体检预约方法处理逻辑比较复杂，需要进行如下业务处理：

1、检查用户所选择的预约日期是否已经提前进行了预约设置，如果没有设置则无法进行预约

2、检查用户所选择的预约日期是否已经约满，如果已经约满则无法预约

3、检查用户是否重复预约（同一个用户在同一天预约了同一个套餐），如果是重复预约则无法完成再次预约

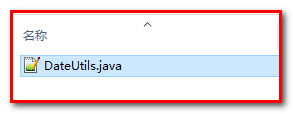
4、检查当前用户是否为会员，如果是会员则直接完成预约，如果不是会员则自动完成注册并进行预约

5、预约成功，更新当日的已预约人数

实现代码如下：

**package** com.itheima.health.service;  
  
**import** com.alibaba.dubbo.config.annotation.Service;  
**import** com.itheima.health.constant.MessageConstant;  
**import** com.itheima.health.dao.MemberDao;  
**import** com.itheima.health.dao.OrderDao;  
**import** com.itheima.health.dao.OrderSettingDao;  
**import** com.itheima.health.entity.Result;  
**import** com.itheima.health.pojo.Member;  
**import** com.itheima.health.pojo.Order;  
**import** com.itheima.health.pojo.OrderSetting;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.transaction.annotation.Transactional;  
  
**import** java.util.Date;  
**import** java.util.List;  
**import** java.util.Map;  
  
*/\*\*  
 \* 体检预约服务  
 \*/*@Service(interfaceClass = OrderService.**class**)  
@Transactional  
**public class** OrderServiceImpl **implements** OrderService {  
  
 @Autowired  
 **private** OrderSettingDao **orderSettingDao**;  
 @Autowired  
 **private** MemberDao **memberDao**;  
 @Autowired  
 **private** OrderDao **orderDao**;  
  
 *//体检预约* **public** Result order(Map map) **throws** Exception {  
 *//检查当前日期是否进行了预约设置* String orderDate = (String) map.get(**"orderDate"**);  
 Date date = DateUtils.parseString2Date(orderDate);  
 OrderSetting orderSetting = **orderSettingDao**.findByOrderDate(date);  
 **if** (orderSetting == **null**) {  
 **return new** Result(**false**, MessageConstant.***SELECTED\_DATE\_CANNOT\_ORDER***);  
 }  
  
 *//检查预约日期是否预约已满* **int** number = orderSetting.getNumber();*//可预约人数* **int** reservations = orderSetting.getReservations();*//已预约人数* **if** (reservations >= number) {  
 *//预约已满，不能预约* **return new** Result(**false**, MessageConstant.***ORDER\_FULL***);  
 }  
  
 *//检查当前用户是否为会员，根据手机号判断* String telephone = (String) map.get(**"telephone"**);  
 Member member = **memberDao**.findByTelephone(telephone);  
 *//防止重复预约* **if** (member != **null**) {  
 Integer memberId = member.getId();  
 **int** setmealId = Integer.*parseInt*((String) map.get(**"setmealId"**));  
 Order order = **new** Order(memberId, date, **null**, **null**, setmealId);  
 List<Order> list = **orderDao**.findByCondition(order);  
 **if** (list != **null** && list.size() > 0) {  
 *//已经完成了预约，不能重复预约* **return new** Result(**false**, MessageConstant.***HAS\_ORDERED***);  
 }  
 }  
  
 *//可以预约，设置预约人数加一* orderSetting.setReservations(orderSetting.getReservations() + 1);  
 **orderSettingDao**.editReservationsByOrderDate(orderSetting);  
  
 **if** (member == **null**) {  
 *//当前用户不是会员，需要添加到会员表* member = **new** Member();  
 member.setName((String) map.get(**"name"**));  
 member.setPhoneNumber(telephone);  
 member.setIdCard((String) map.get(**"idCard"**));  
 member.setSex((String) map.get(**"sex"**));  
 member.setRegTime(**new** Date());  
 **memberDao**.add(member);  
 }  
  
 *//保存预约信息到预约表* Order order = **new** Order(member.getId(),  
 date,  
 (String) map.get(**"orderType"**),  
 Order.***ORDERSTATUS\_NO***,  
 Integer.*parseInt*((String) map.get(**"setmealId"**)));  
 **orderDao**.add(order);  
  
 **return new** Result(**true**, MessageConstant.***ORDER\_SUCCESS***, order);  
 }  
}

导入DateUtils放置到health\_common中



### Dao接口

#### OrderSettingDao.java

**package** com.itheima.health.dao;  
  
**import** com.itheima.health.pojo.OrderSetting;  
**import** org.springframework.stereotype.Repository;  
  
**import** java.util.Date;  
**import** java.util.List;  
**import** java.util.Map;  
  
*/\*\*  
 \* 持久层Dao接口  
 \*/*@Repository  
**public interface** OrderSettingDao {  
  
  
 **long** findCountByOrderDate(Date orderDate);  
  
 **void** editNumberByOrderDate(OrderSetting orderSetting);  
  
 **void** add(OrderSetting orderSetting);  
  
 List<OrderSetting> getOrderSettingByMonth(Map map);  
 //根据预约日期查询预约设置信息  
 OrderSetting findByOrderDate(Date date);  
 //更新已预约人数  
 **void** editReservationsByOrderDate(OrderSetting orderSetting);  
}

#### MemberDao.java

**package** com.itheima.health.dao;  
  
**import** com.github.pagehelper.Page;  
**import** com.itheima.health.pojo.Member;  
**import** org.springframework.stereotype.Repository;  
  
**import** java.util.List;  
  
*/\*\*  
 \* 持久层Dao接口  
 \*/*@Repository  
**public interface** MemberDao {  
  
  
 **public** List<Member> findAll();  
 **public** Page<Member> selectByCondition(String queryString);  
 **public void** add(Member member);  
 **public void** deleteById(Integer id);  
 **public** Member findById(Integer id);  
 **public** Member findByTelephone(String telephone);  
 **public void** edit(Member member);  
 **public** Integer findMemberCountBeforeDate(String date);  
 **public** Integer findMemberCountByDate(String date);  
 **public** Integer findMemberCountAfterDate(String date);  
 **public** Integer findMemberTotalCount();  
}

#### OrderDao.java

**package** com.itheima.health.dao;  
  
**import** com.itheima.health.pojo.Order;  
**import** org.springframework.stereotype.Repository;  
  
**import** java.util.List;  
  
*/\*\*  
 \* 持久层Dao接口  
 \*/*@Repository  
**public interface** OrderDao {  
  
  
 List<Order> findByCondition(Order order);  
  
 **void** add(Order order);  
}

### Mapper映射文件

#### OrderSettingDao.xml

*<!--根据日期查询预约设置信息-->*<**select id="findByOrderDate" parameterType="date" resultType="com.itheima.health.pojo.OrderSetting"**>  
 select *\** from t\_ordersetting where orderDate = #{orderDate}  
</**select**>  
*<!--更新已预约人数-->*<**update id="editReservationsByOrderDate" parameterType="com.itheima.health.pojo.OrderSetting"**>  
 update t\_ordersetting set reservations = #{reservations} where orderDate = #{orderDate}  
</**update**>

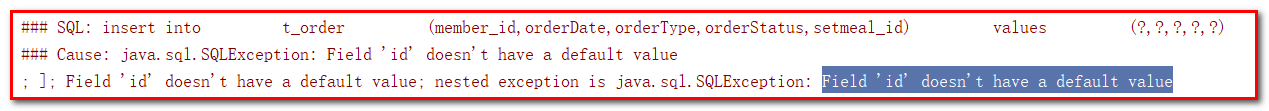
#### MemberDao.xml

*<?***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.itheima.health.dao.MemberDao"**>  
 <**select id="findAll" resultType="com.itheima.health.pojo.Member"**>  
 select *\** from t\_member  
 </**select**>  
  
 *<!--根据条件查询-->* <**select id="selectByCondition" parameterType="string" resultType="com.itheima.health.pojo.Member"**>  
 select *\** from t\_member  
 <**if test="value != null and value.length > 0"**>  
 where fileNumber = #{value} or phoneNumber = #{value} or name = #{value}  
 </**if**>  
 </**select**>  
  
 *<!--新增会员-->* <**insert id="add" parameterType="com.itheima.health.pojo.Member"**>  
 <**selectKey resultType="java.lang.Integer" order="AFTER" keyProperty="id"**>  
 SELECT LAST\_INSERT\_ID()  
 </**selectKey**>  
 insert into  
 t\_member  
 (fileNumber,name,sex,idCard,phoneNumber,  
 regTime,password,email,birthday,remark)  
 values  
 (#{fileNumber},#{name},#{sex},#{idCard},#{phoneNumber},  
 #{regTime},#{password},#{email},#{birthday},#{remark})  
 </**insert**>  
  
 *<!--删除会员-->* <**delete id="deleteById" parameterType="int"**>  
 delete from t\_member where id = #{id}  
 </**delete**>  
  
 *<!--根据id查询会员-->* <**select id="findById" parameterType="int" resultType="com.itheima.health.pojo.Member"**>  
 select *\** from t\_member where id = #{id}  
 </**select**>  
  
 *<!--根据id查询会员-->* <**select id="findByTelephone" parameterType="string" resultType="com.itheima.health.pojo.Member"**>  
 select *\** from t\_member where phoneNumber = #{phoneNumber}  
 </**select**>  
  
 *<!--编辑会员-->* <**update id="edit" parameterType="com.itheima.health.pojo.Member"**>  
 update t\_member  
 <**set**>  
 <**if test="fileNumber != null"**>  
 fileNumber = #{fileNumber},  
 </**if**>  
 <**if test="name != null"**>  
 name = #{name},  
 </**if**>  
 <**if test="sex != null"**>  
 sex = #{sex},  
 </**if**>  
 <**if test="idCard != null"**>  
 idCard = #{idCard},  
 </**if**>  
 <**if test="phoneNumber != null"**>  
 phoneNumber = #{phoneNumber},  
 </**if**>  
 <**if test="regTime != null"**>  
 regTime = #{regTime},  
 </**if**>  
 <**if test="password != null"**>  
 password = #{password},  
 </**if**>  
 <**if test="email != null"**>  
 email = #{email},  
 </**if**>  
 <**if test="birthday != null"**>  
 birthday = #{birthday},  
 </**if**>  
 <**if test="remark != null"**>  
 remark = #{remark},  
 </**if**>  
 </**set**>  
 where id = #{id}  
 </**update**>  
  
 *<!--根据日期统计会员数，统计指定日期之前的会员数-->* <**select id="findMemberCountBeforeDate" parameterType="string" resultType="int"**>  
 select *count*(id) from t\_member where regTime **&lt;**= #{value}  
 </**select**>  
  
 *<!--根据日期统计会员数-->* <**select id="findMemberCountByDate" parameterType="string" resultType="int"**>  
 select *count*(id) from t\_member where regTime = #{value}  
 </**select**>  
  
 *<!--根据日期统计会员数，统计指定日期之后的会员数-->* <**select id="findMemberCountAfterDate" parameterType="string" resultType="int"**>  
 select *count*(id) from t\_member where regTime **&gt;**= #{value}  
 </**select**>  
  
 *<!--总会员数-->* <**select id="findMemberTotalCount" resultType="int"**>  
 select *count*(id) from t\_member  
 </**select**>  
</**mapper**>

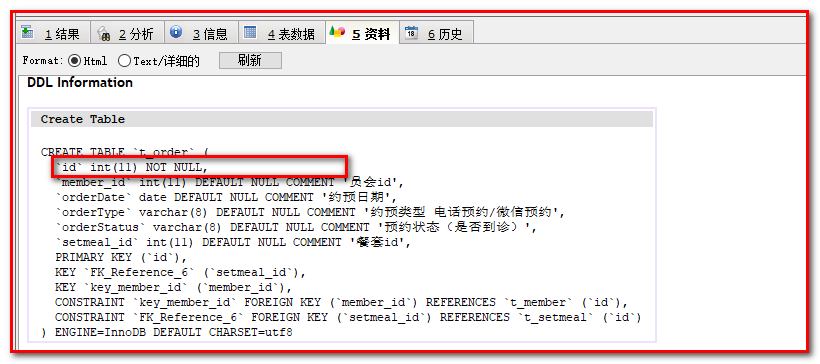
#### OrderDao.xml

*<?***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.itheima.health.dao.OrderDao"**>  
 <**resultMap id="orderResultMap" type="com.itheima.health.pojo.Order"**>  
 <**id column="id" property="id"**/>  
 <**result column="member\_id" property="memberId"**/>  
 <**result column="orderDate" property="orderDate"**/>  
 <**result column="orderType" property="orderType"**/>  
 <**result column="orderStatus" property="orderStatus"**/>  
 <**result column="setmeal\_id" property="setmealId"**/>  
 </**resultMap**>  
 *<!--新增-->* <**insert id="add" parameterType="com.itheima.health.pojo.Order"**>  
 <**selectKey resultType="java.lang.Integer" order="AFTER" keyProperty="id"**>  
 SELECT LAST\_INSERT\_ID()  
 </**selectKey**>  
 insert into  
 t\_order  
 (member\_id,orderDate,orderType,orderStatus,setmeal\_id)  
 values  
 (#{memberId},#{orderDate},#{orderType},#{orderStatus},#{setmealId})  
 </**insert**>  
  
 *<!--动态条件查询-->* <**select id="findByCondition" parameterType="com.itheima.health.pojo.Order"  
 resultMap="orderResultMap"**>  
 select *\** from t\_order  
 <**where**>  
 <**if test="id != null"**>  
 and id = #{id}  
 </**if**>  
 <**if test="memberId != null"**>  
 and member\_id = #{memberId}  
 </**if**>  
 <**if test="orderDate != null"**>  
 and orderDate = #{orderDate}  
 </**if**>  
 <**if test="orderType != null"**>  
 and orderType = #{orderType}  
 </**if**>  
 <**if test="orderStatus != null"**>  
 and orderStatus = #{orderStatus}  
 </**if**>  
 <**if test="setmealId != null"**>  
 and setmeal\_id = #{setmealId}  
 </**if**>  
 </**where**>  
 </**select**>  
</**mapper**>

测试：如果出现异常



说明t\_order表的id没有设置自增长，这是开发很常见的异常，可以设置：



创建表时设置主键自增长（主键必须是整型才可以自增长）：  
CREATE TABLE `t\_order` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`member\_id` int(11) DEFAULT NULL COMMENT '员会id',

`orderDate` date DEFAULT NULL COMMENT '约预日期',

`orderType` varchar(8) DEFAULT NULL COMMENT '约预类型 电话预约/微信预约',

`orderStatus` varchar(8) DEFAULT NULL COMMENT '预约状态（是否到诊）',

`setmeal\_id` int(11) DEFAULT NULL COMMENT '餐套id',

PRIMARY KEY (`id`),

KEY `key\_member\_id` (`member\_id`),

KEY `key\_setmeal\_id` (`setmeal\_id`),

CONSTRAINT `key\_member\_id` FOREIGN KEY (`member\_id`) REFERENCES `t\_member` (`id`),

CONSTRAINT `key\_setmeal\_id` FOREIGN KEY (`setmeal\_id`) REFERENCES `t\_setmeal` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=18 DEFAULT CHARSET=utf8;

修改表时设置主键自增长：  
ALTER TABLE t\_order CHANGE id id INT AUTO\_INCREMENT;

修改表时删除主键自增长：  
ALTER TABLE t\_order CHANGE id id INT;

# 第四章. 预约成功页面展示

前面已经完成了体检预约，预约成功后页面会跳转到成功提示页面（orderSuccess.html）并展示预约的相关信息（体检人、体检套餐、体检时间等）。

跳转到orderSuccess.html，传递预约成功的订单id。

**axios**.post(**"/order/submit.do"**,**this**.**orderInfo**).then((response) => {  
 **if**(response.**data**.**flag**){  
 *//预约成功，跳转到预约成功页面* **window**.**location**.**href**=**"orderSuccess.html?orderId="** + response.**data**.**data**.**id**;  
 }**else**{  
 *//预约失败，提示预约失败信息* **this**.**$message**.error(response.**data**.**message**);  
 }  
});

## 页面调整

提供orderSuccess.html页面，展示预约成功后相关信息

第一步：页面输出订单相关人的信息

<**div class="info-title"**>  
 <**span class="name"**>体检预约成功</**span**>  
</**div**>  
<**div class="notice-item"**>  
 <**div class="item-title"**>预约信息</**div**>  
 <**div class="item-content"**>  
 <**p**>体检人：{{orderInfo.member}}</**p**>  
 <**p**>体检套餐：{{orderInfo.setmeal}}</**p**>  
 <**p**>体检日期：{{orderInfo.orderDate}}</**p**>  
 <**p**>预约类型：{{orderInfo.orderType}}</**p**>  
 </**div**>  
</**div**>

第二步：使用订单id，查询订单详细信息。存放到orderInfo的变量中。

<**script**>  
 **var *vue*** = **new** *Vue*({  
 **el**:**'#app'**,  
 **data**:{  
 **orderInfo**:{}  
 },  
 mounted(){  
 **axios**.post(**"/order/findById.do?id="** + ***id***).then((response) => {  
 **this**.**orderInfo** = response.**data**.**data**;  
 });  
 }  
 });  
</**script**>

## 后台代码

### Controller

在OrderController中提供findById方法，根据预约id查询预约相关信息

*/\*\*  
 \* 根据id查询预约信息，包括套餐信息和会员信息  
 \** ***@param id*** *\** ***@return*** *\*/*@RequestMapping(**"/findById"**)  
**public** Result findById(Integer id){  
 Result result =**null**;  
 **try**{  
 result = **orderService**.findById4Detail(id);  
 *//查询预约信息成功* **return new** Result(**true**,MessageConstant.***QUERY\_ORDER\_SUCCESS***,result.getData());  
 }**catch** (Exception e){  
 e.printStackTrace();  
 *//查询预约信息失败* **return new** Result(**false**,MessageConstant.***QUERY\_ORDER\_FAIL***);  
 }  
}

### 服务接口

在OrderService服务接口中扩展findById4Detail方法

*//根据id查询预约信息，包括体检人信息、套餐信息*Result findById4Detail(Integer id);

### 服务实现类

在OrderServiceImpl服务实现类中实现findById4Detail方法

*//根据id查询预约信息，包括体检人信息、套餐信息***public** Result findById4Detail(Integer id) **throws** Exception {  
 Map map = **orderDao**.findById4Detail(id);  
 **if**(map != **null**){  
 *//处理日期格式* Date orderDate = (Date) map.get(**"orderDate"**);  
 map.put(**"orderDate"**,DateUtils.*parseDate2String*(orderDate));  
 **return new** Result(**true**,MessageConstant.***QUERY\_ORDER\_SUCCESS***,map);  
 }  
 **return new** Result(**false**,MessageConstant.***QUERY\_ORDER\_FAIL***);  
}

### Dao接口

在OrderDao接口中扩展findById4Detail方法

Map findById4Detail(Integer id);

### Mapper映射文件

在OrderDao.xml映射文件中提供SQL语句

*<!--根据预约id查询预约信息，包括体检人信息、套餐信息-->*<**select id="findById4Detail" parameterType="int" resultType="map"**>  
 select m.name member ,s.name setmeal,o.orderDate orderDate,o.orderType orderType  
 from  
 t\_order o,  
 t\_member m,  
 t\_setmeal s  
 where o.member\_id=m.id and o.setmeal\_id=s.id and o.id=#{id}  
</**select**>

m.name member ,s.name setmeal,o.orderDate orderDate,o.orderType orderType

对应：

页面内容

<**p**>体检人：{{orderInfo.member}}</**p**>  
<**p**>体检套餐：{{orderInfo.setmeal}}</**p**>  
<**p**>体检日期：{{orderInfo.orderDate}}</**p**>  
<**p**>预约类型：{{orderInfo.orderType}}</**p**>