# 移动端开发-旅游预约

学习目标：

1：了解旅游预约流程业务

2：能够基于阿里云短信服务实现短信发送

3：掌握旅游预约的实现过程

4：掌握预约成功页面展示的实现过程

# 第一章. 旅游预约流程需求

用户可以通过如下操作流程进行旅游预约：

1、在移动端首页点击旅游预约，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

4、在预约页面录入旅游人信息，包括手机号，点击发送验证

5、在预约页面录入收到的手机短信验证码，点击提交预约，完成旅游预约

效果如下图：

* 套餐列表页面：



* 套餐详情页面：

计算机生成了可选文字:
香港九龙尖沙咀商双飞3．8日自由行套餐
：杳港丿L丿它王臼氵J叾（Roseda》eHotelKOW℃on）亻立于亍旨中心
擎华闹市，地处九龙中心地带，毗邻多个购物、娱乐中
心；酒店提供班车来往旺角、少嘴和“圆方"购物区，方
便宾客出行。香港九龙丽酒店（Roseda|eHotel
Kow|oon）设计精巧、新颖独扌寺，拥有精致、高雅的各式
客房。酒店客房均配有LED智能电视、iPod/iPhone底
座，讠上您尽享便捷、舒适生活。
性别不眼18一60
套餐详情
项目名称
项目内容
越南芽庄6日5晚大阪海游诓亲@接触5日
自行休闲游日本奈良公匠小
鹿乱7日古都游大阝反
道顿堀小吃不停6日美
食游
立即预约
项乬解读
土艮冥10日游超6000人
选陉牛人亏线／五星航空
直飞丙双飞／持希耶／
红吕点齐打卡丿五星泗
店升1蹼曰洞穴氵酉店，分
级4持色萏／2顿中餐

* 预约页面：

计算机生成了可选文字:
@昵笥须知
《旅游人信息
旅游人
性别
满输入手机号
验证码
身份证号
旅游日
清输入身亻分证号
0

点击【提交预约】完成预约。

# 第二章. 短信发送【阿里短信】

### 【目标】

能够基于阿里云短信服务实现短信发送

### 【路径】

1. 短信服务介绍
2. 注册阿里云账号
3. 设置短信签名
4. 设置短信模板
5. 设置access keys
6. 短信服务API
7. 发送短信

### 【讲解】

## 2.1. 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（**S**hort **M**essage **S**ervice）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。

## 2.2. 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：

计算机生成了可选文字:
欢迎注册阿里云
设会名
适再次蝓．&你密
旖0住快》动到E边
同息条款并注册
囗《阿里云飕：》（法廑声地权，
已白珂丰应，还或1588陣号？快0登

注册后，使用账号名登录

计算机生成了可选文字:
忘记码忘记会员名兔薪注册

## 2.3. 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：

点击产品分类->云计算基础->云通信->短信服务



计算机生成了可选文字:
C+〕阝可里E中亠
最斤氵舌丞力
品《分类
短信服务
域名商际公到
0云服务器ECS
云服务器ECS《弹性公》P《物联网无线连接服务《SSL证书《云数据库MySQL版
企业应中心解决方案云市场支持与服务台亻乍伙伴与生态开发者
@）视频简介
短信服务（SMessageService)是阿里云为户提亻斯一种通信服务的能力
支拮国内和国际快速发送验证码、短信通知不隹广短信，服务范围覆盖全球2開多个国家和地区。
国内短信支}寺三双合一一专属通与工信部憮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％。完美支撑双11期「0亿短信发送，乙户氟木。
立即购买

【签名】：

选择签名管理

计算机生成了可选文字:
C+〕阿里E
AP长解决7
短信服务
快速学习0
国内消息
国际／港澳台消息
业务统计
《短信服务开通
产品名称：
开通说明：
Q搜索文档、控制台、
短信服务
产品免费开通，按量实时计费。查看计费说
囝我已阅读并同意《短信服务开通协议》
立即开通

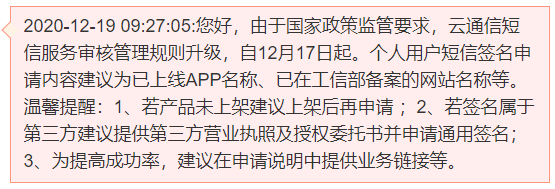
计算机生成了可选文字:
〕阿里云
0
《文本哣信
笞叾管理
0
适00E@
状0001
0
三咩与噁
019·0，寻1？0，951
2017．07一2015007
盒河0中皇
0

点击添加签名按钮：

计算机生成了可选文字:
C+〕阿里E
短信服务
快速学习0
匡内消息
匡示过压台息
业务统计
用统计
莩统没置
通用设詈
O是
国内消息讠殳詈
用
工单
企业
支持与服务
《添加签名
返回上层
@针灯网站、APP、小程序或公众号未上线的情况，干台层持发送验证码：如已上线，可窜适、通用·的适用场昱，可发送验证码、短信通到：
*适用场昱．
*筌名夹源．
*是否涉及箪三方权兰@、
0/12
若筌名/椏内容犯到箪三方权兰必须茌得箪三方真窭杈
．无须添加0、0、[]符号，筌名发送会目带0符号，避免重复
了解更多筌名哐板圭适规范
验证码@通用@
·不能便用个人姓名作为短信筌名
·个人用户可由适1个验证码筌名，通用场昱筌名一天支适1个
C)企事业单位的全称或简称
C)工信部网站的全称或简称
C)APP应甲的全称或简称
C)公众号或小程序的全称或简称

目前个人用户只能申请适用场景为验证码的签名，通用需要企业认证。

另注：



## 2.4. 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：

计算机生成了可选文字:
签名管理
快速学习0
国内消思．
国港澳台消息
业务统汁
模版管理@
工单号
群发助手
囗
囗
模版名称
美年国宾
125911491
t*fiCODE
SMS180947117
模版类型（全
验证码
创建时间
2019．12．2319：11：05
审陔状态（全部）
．审核中

点击添加模板按钮：



我的模板内容是：美年国宾 验证码${number}，您正进行美年国宾系统的身份验证，打死不告诉别人！

其中${number}为动态参数，需要我们后续在代码中控制。

## 2.5. 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：

计算机生成了可选文字:
基本资料
e安全管控
访问控制
§AccessKey管里
a唯荐源利后台
实名认证
退出登录
安全设置

点击accesskeys：

计算机生成了可选文字:
安全堤丁
O
于信0云账号A〔（sKey皇您访河阿里云I的密绢有泫账产完全的权。适您心妥
善保管！不要过任何方过0，Github〕樗齔Key公开到外部渠洹，以免地人利用而
适成童全胁。睚哪建议您遵貊阿里云安全哇跬使用&AM子用户A蕊张ey蔗行
API调用。
续使用A蕊sKey
开始使用子用户A〔（'

点击“开始使用子用户AccessKey”按钮，指定用户权限，而不是分配所有权限。

第一步：新建用户

输入登录名称和显示名称，点击【确认】

计算机生成了可选文字:
新建用户
*用户账号信息
atgwgu
+添加户
访问方式0
囗
控制台密码登录
编裎访问
显示名称
atgwgu
@1763833522788396℃n引iyun℃om
户厍账号码访问阿里云控制台
确定
源回

接收短信，防止信息泄露 ，输入验证码即可。

计算机生成了可选文字:
手忄佥证
您绑定的手机
133****5560更换手机
获取验证码
取消

新建用户成功，成功之后，发现 ak 没有授权

计算机生成了可选文字:
新建用户
若开通编访问》及时保存AccessKey信页面关闭后将无法再次获取信息。
用户信息
下载CSV文件
用户0录名称
囗
atguigu@178彐522788彐96℃niyunom
．成功
登录密码
无
AccessKey旧
无
AccessKeySecret
无
悍作
囗复制

第二步：授权

选择用户，添加权限

计算机生成了可选文字:
人员管理
用户组
SSO管理
权限管理
权裝策略管理
用户
RAM用户是一个身份实体，它通韋代表您的组织中需要访问云资源的人员或应用程序·
通常的操作步骤如下：
1．创建用户，并为用户设首录密码（用户0录控制台场吊〕ü@JÉAccessKey〔应用程序调用AP《场景）。
2舔加用户到用户组（需要先创建用户组芬完成对用户组刊授杈）
户录名称丷请输入
厍户0录名称／显示名称
atguigu@1763833522788396．on引iyun℃om
atguigu
Q
创建时间
2019年12月23日19：17：23
添加到用户
添加权限

搜索“SMS”，表示短信服务，选择权限，点击“开始创建”。

计算机生成了可选文字:
添加权限
被授权主体
atgt」igu@1763833522788396℃n引iYL」n.com
选择权限
系统权眼策略
权眼策略名称
AliyunDysmsFullAccess
SmS
备汪
管理短信服务(SMS)的杈眼
只读访问短信服务(SMS)的杈眼
0
@i畢
Q
已选择〔2〕
AliyunDysmsFullAccess
AliyunDysmsReadOnlyAccess
AliyunDysmsReadOnlyAccess
3
取消

第三步：创建AccessKeyID

点击创建的用户，进入到详情页面。

计算机生成了可选文字:
RAM户是一个身份实体，它诵常代表您的绢织中需要访问云资源的人员或应用程序。
通常的操作步摆如下：
10建户，并为户设甬登录密码（户登录控制台场景〕或创建AccessKey（应用程序调用API场景）。
2舔加用户到用户组（需要先创建用户组并完成对用户组的授权）。
新建用户
用户甾录名称v
厍户登录名称/显示名称
请输入
atguigu@1763833522788396．on引iyun℃om
atgwgu
0
点击
创建时间
2019年12月23日
19：21：40
操作
氵忝加到户组添加杈阝艮删除

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

计算机生成了可选文字:
〔刁阃里E
RAM访问控制
用户
SSO管三
权管
权略管
RAM色管理
0虹th应甲
二望
与
园
0
认证管理
控制台登冒理
控制台访闫
加入的组
叹眼管理
修改登录过
@JEAccessKey
须开启多
适及时《早存或发送AK启息至灯应员工彈到关将无法再次获取该启息，但您可以随时创津新的AK：
多因认
匚e“K封'》0
AccessKeySecret:
用户Ae
±下〔哎0亘制
创建新的
AccessK
LTA14tZAmy7v6WNn
0
0AccessKey创建成功
00还厶n的
snMIB38W0
201g年8用7E1012：24

注意：一定，一定，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。如果没有保存，那么就删除，重新创建

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：

计算机生成了可选文字:
〔刁阿里云
《国内消息设置
发送率设置
验证码防盗刷、监控
用
黑名单解除记录
工单
企业
短信服务
俣速学习0
国内消息
国示过台為皂
业务统计
发送量统讠十
用统计
统没兰
通用设首
亠
帮助文栏
为了预防账户嗡正码盗刷，平台为用户提供送量及成功的双重监控，如果发了防盗刷预善规则，将会为用户发送预短1瓦
发送总量或值@
日发送量：达
月发送量：达
5
10
景预：达
条预：达
10
50
条限额，暂停遼盲发送·
条限额，暂停遴信发送·
0
提示：达预善匾及限额值，统会发送逗信堤醒到手几号。达到限额值盯，统笞停逗信发送，宗要您手动更改。
0

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”，一般充 5 毛钱 ，就够用了。

计算机生成了可选文字:
C-〕阃里E
|短信服务概览
0公眚：2019年6月4三．盾皂甲心不将迁至甾览刘《俨昼甲粼．诂迅·
孚习0
用户状：正帛/个人用户
匡内可鬯
工望
订里
彝已衾
乏入用中心

## 2.6. 短信服务API

点击帮助文档

计算机生成了可选文字:
〔刁阿里E
短信服务
速学习0
国内消息
国港台河0
业务统计
用统计
椠统没置
通用设置
匡内油设置
帮助文栏

找到短信服务中的“短信发送API”



计算机生成了可选文字:
技术对接步摆
1：下载SDK工日包
SDK工具包中一共包含了2
package$éÜ者mvnd
SDK&DEMO(下载地址]
0
案例
2：编写样例程
可下载查看
一个aliyun-java-sdk-core@，另外一个Ealicom-dysms-api@，将这两个包执行mvn
叩令打包出相应的jar包，添加到工程类库中哝赖使用·
注：有备注无需0改的位置清勿改动。
1．
2，
土
4．
S．
//设起时时间．可自行讠周整
System·setProperty("sun·net，client·defaultConnectTimeout“
System．setProperty("Sunnet，client“defaultReadTimeout"
//初始化a“Client需要的几个磐数
finalStringproduct
finalStringdomain
"Dysmsapi"；//信I产品名称〔短信产品名亘定．无需修改〕
"dysmsapi.a1iyuncs.com/;//EIEAPIFR$.æ（接口地址亘定，无需修改）

将代码可以拷贝到工程中测试：

需要修改：

1：accessKeyId和accessKeySecret

final String accessKeyId = "LTAI4Fbws3CCEzKx3JPHqXQC";//你的accessKeyId,参考本文档步骤2  
final String accessKeySecret = "3XFb3eNq6IDZ5Ik3x10vo9YHHzMGBz";//你的accessKeySecret，参考本文档步骤2

2：手机号

request.setPhoneNumbers("1326921xxxx");

3：签名和模板

request.setSignName("美年国宾");  
//必填:短信模板-可在短信控制台中找到，发送国际/港澳台消息时，请使用国际/港澳台短信模版  
request.setTemplateCode("SMS\_180947117");

4：发送的验证码及参数number（根据模板短信内容）

String params = "111111";  
request.setTemplateParam("{\"number\":\""+params+"\"}");

## 2.7. 发送短信

### 2.7.1. 导入maven坐标

在 meinian\_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>

### 2.7.2. 封装工具类

在 meinian\_common中添加工具类

1：签名

计算机生成了可选文字:
文本短信
签名管理
模版管理
囗
[囗
签名名称
美年国宾
群发助丰
适用场@
验证码
审核状态（全部）
．通过
创建时间
2019．12．2319：08：41
@签名和模板介
添加签名
修改群发操作记录删除

2：模板code

计算机生成了可选文字:
签名管理
模版管理
工单号
群发助手
囗
囗
模版名称
忄莫fiCODE
SMS180947117
模版类瞓全
年国宾
125911491
创建时间
2019．12．2319：11℃5
审核状态（全部）
．通过
添加模版
详情复制群发删除

在 meinian\_common中，封装 SMSUtils.java

传递验证码和手机号

package com.atguigu.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;  
​  
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";//预约成功通知  
​  
 /\*\*  
 \* 发送短信  
 \* @param phoneNumbers  
 \* @param param  
 \* AccessKeyID:  
 \*         LTAI4Fbws3CCEzKx3JPHqXQC  
 \* AccessKeySecret:  
 \*         3XFb3eNq6IDZ5Ik3x10vo9YHHzMGBz  
 \*/  
 public static void sendShortMessage(String phoneNumbers,String param) throws Exception{  
 // 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 = "LTAI4Fbws3CCEzKx3JPHqXQC";// 你的accessKeyId,参考本文档步骤2  
 final String accessKeySecret = "3XFb3eNq6IDZ5Ik3x10vo9YHHzMGBz";// 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("美年国宾");  
 // 必填:短信模板-可在短信控制台中找到  
 request.setTemplateCode("SMS\_180947117");  
 // 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"+sendSmsResponse.getCode());  
 }  
 }  
}

### 2.7.3. 测试短信发送

package com.atguigu;  
​  
import com.atguigu.utils.SMSUtils;  
import com.atguigu.utils.ValidateCodeUtils;  
import org.junit.Test;  
​  
public class TestSms {​  
   @Test  
   public void test()  throws Exception{  
       Integer code = ValidateCodeUtils.generateValidateCode(4);  
       SMSUtils.sendShortMessage("133\*\*\*\*5560",String.valueOf(code));  
  }  
}

测试：查看手机



### 【小结】

阿里云使用步骤

1. 注册, 登录
2. 搜索 短信服务, 开通短信服务
3. 进入短信控制台
4. 申请 签名 , 模版
5. 充钱(5块左右), 查看access keys
6. 添加依赖, 拷贝工具类到项目
7. 测试

注意事项

工具类里面(需要改 模版code,签名, access keys，验证码code==>number)

## 2.8. 短信发送【山东鼎信】【扩展】

### 2.8.1 阿里云，云市场，购买短信API

通过阿里云购买第三方短信发送，免费的即可（5次）

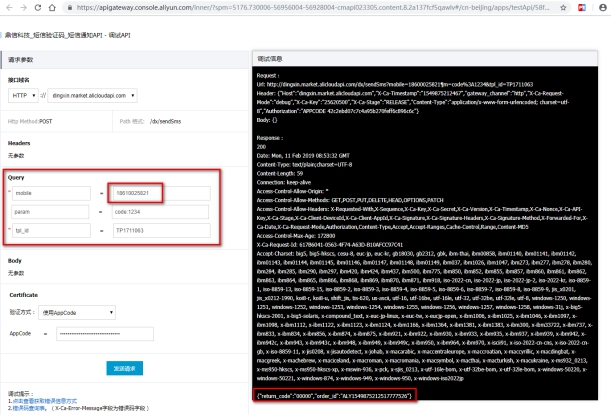


### 2. 8.2 通过示例测试

#### 2.8.2.1 通过【去调试】进行测试

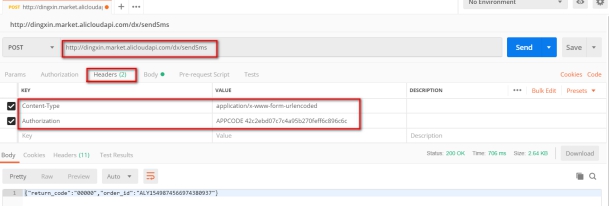


只需要填写手机号码即可

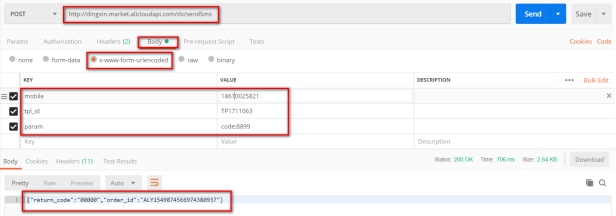


#### 2.8.2.2 通过postman测试

输入请求地址，请求头信息



输入请求参数，点击【Send】按钮，查看返回结果



#### 2.8.2.3 通过Java代码测试

##### 1 SmsTest

参考：<https://market.aliyun.com/products/56928004/cmapi023305.html?spm=5176.2020520132.101.3.4b697218KUgzH1#sku=yuncode1730500007>

package com.atguigu;

import java.util.HashMap;

import java.util.Map;

import org.apache.http.HttpResponse;

public class SmsTest {

   public static void main(String[] args) {

      String host = "<http://dingxin.market.alicloudapi.com>";

      String path = "/dx/sendSms";

      String method = "POST";

      String appcode = "62c2ebd07c7c4a95b270feff6c896c6c";

      Map<String, String> headers = new HashMap<String, String>();

      // 最后在header中的格式(中间是英文空格)为Authorization:APPCODE 83359fd73fe94948385f570e3c139105

      headers.put("Authorization", "APPCODE " + appcode);

      Map<String, String> querys = new HashMap<String, String>();

      querys.put("mobile", "18610025821");

      querys.put("param", "code:9966");

      querys.put("tpl\_id", "TP1711063");

      Map<String, String> bodys = new HashMap<String, String>();

      try {

            /\*\*

            \* 重要提示如下: HttpUtils请从<https://github.com/aliyun/api-gateway-demo-sign-java/blob/master/src/main/java/com/aliyun/api/gateway/demo/util/HttpUtils.java>

            \* 下载依赖: <https://github.com/aliyun/api-gateway-demo-sign-java/blob/master/pom.xml>

            \*/

            HttpResponse response = HttpUtils.doPost(host, path, method, headers, querys, bodys);

            System.out.println(response.toString());

            // 获取response的body

            // System.out.println(EntityUtils.toString(response.getEntity()));

            //{"return\_code":"00000","order\_id":"ALY1549881237643456814"}

      } catch (Exception e) {

            e.printStackTrace();

      }

   }

}

将测试类，封装成工具类SMSUtils.java，放在 meinian\_common中。

##### 2 HttpUtils.java

参考：<https://github.com/aliyun/api-gateway-demo-sign-java/blob/master/src/main/java/com/aliyun/api/gateway/demo/util/HttpUtils.java>

##### 3 common项目中配置pom.xml

参考：<https://github.com/aliyun/api-gateway-demo-sign-java/blob/master/pom.xml>

在 meinian\_common 中导入坐标

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.2.15</version>

</dependency>

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.2.1</version>

</dependency>

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpcore</artifactId>

<version>4.2.1</version>

</dependency>

<dependency>

<groupId>commons-lang</groupId>

<artifactId>commons-lang</artifactId>

<version>2.6</version>

</dependency>

<dependency>

<groupId>org.eclipse.jetty</groupId>

<artifactId>jetty-util</artifactId>

<version>9.3.7.v20160115</version>

</dependency>

#### 2.8.2.4 封装成工具类，存放到meinian\_common项目中

# 第三章. 旅游预约

### 【目标】

实现旅游预约

需求：

计算机生成了可选文字:
香港九龙尖沙咀商圈双飞3·8日自由行套餐
杳港九龙丽酒店（Roseda《eHotelKow|00n）位于市中心
擎华闹市，地处九龙中心哋带，毗邻多个购物、娱乐中
心；酒店提亻共班车来往旺角、少嘴和“圆方"购物区，方
便宾客出行。香港九龙珀丽酒店（Roseda《eHotel
Kowloon)设计精巧、新颖独特，拥有精致、高雅的各式
客房。酒店客房均配有LED智能电视、iPod/iPhone底
座，讠上您尽亨便捷、舒适生活。
性别不眼18．60
@预约须知
《旅游人信息
0
套餐洋肓
项目名称
旅游人
性别
几号
旅游日期
洁输入姓名
i古输入验证码
清輸入身份证号
项目内容
越南芽庄6日5晚大阪海游亲密接触5日
自行休闲游日本奈良公匠小
鹿乱撞7日古都游大阝反
道顿堀小吃不停6日美
食游
立即预约
项目解读
土艮其10日游超6000人
选择牛人亏线／五星航空
直飞丙葒双飞／些特希耶／
网红冯点齐打卡/五星酒
店升1晚自洞穴酒店/升
级4持色餐/2顿中餐

1. 在套餐详情页面(setmeal\_detail.html)点击立即预约，页面跳转到预约页面(orderInfo.html)
2. 在预约页面(orderInfo.html)录入旅游人信息，包括手机号，点击发送验证码
3. 在预约页面录入收到的手机短信验证码，点击提交预约，完成预约

### 【路径】

前台代码编写

在/pages/orderInfo.html

完成需求：  
1.展示预约套餐信息  
2.手机号校验  
3.30秒倒计时效果  
4.发送ajax请求，完成【发送验证码】功能  
5.日期中使用日历插件  
6.【提交预约】请求

后台代码编写

1.类OrderController.java

2.类OrderService.java

3.类OrderServiceImpl.java

4.类OrderDao.java

* OrderSettingDao.java
* MemberDao.java

5.配置文件OrderDao.xml

* OrderSettingDao.xml
* MemberDao.xml

完成需求：

后台代码1：短信验证码  
1. 填写 手机号码, 点击发送验证码 请求服务器(携带手机号码)  
2. 在后台  
  - 获得手机号码  
  - 生成验证码  
  - 使用阿里云服务发送验证码  
  - 把生成验证码存到redis里面(存5分钟)  
​  
​  
​  
提交预约  
后台代码2  
1：Controller  
​  
点击提交预约, 把用户信息 提交到服务器  
在Controller里面  
  - 获得用户信息  
  - 校验验证码(redis里面存的和用户输入的比较)  
  - 调用业务, 进行预约, 响应  
​  
2：Service  
​   
1. 判断当前的日期是否可以预约(根据orderDate查询t\_ordersetting, 能查询出来可以预约;查询不出来,不能预约)  
2. 判断当前日期是否预约已满(判断reservations（已经预约人数）是否等于number（最多预约人数）)  
3. 判断是否是会员(根据手机号码查询t\_member)  
 - 如果是会员(能够查询出来), 防止重复预约(根据member\_id,orderDate,setmeal\_id查询t\_order)  
 - 如果不是会员(不能够查询出来) ,自动注册为会员(直接向t\_member插入一条记录)  
4.进行预约  
  - 向t\_order表插入一条记录  
   - t\_ordersetting表里面预约的人数reservations+1

导入数据库

DROP TABLE IF EXISTS `t\_member`;  
CREATE TABLE `t\_member` (  
 `id` int(11) NOT NULL AUTO\_INCREMENT,  
 `fileNumber` varchar(32) DEFAULT NULL,  
 `name` varchar(32) DEFAULT NULL,  
 `sex` varchar(8) DEFAULT NULL,  
 `idCard` varchar(18) DEFAULT NULL,  
 `phoneNumber` varchar(11) DEFAULT NULL,  
 `regTime` date DEFAULT NULL,  
 `password` varchar(32) DEFAULT NULL,  
 `email` varchar(32) DEFAULT NULL,  
 `birthday` date DEFAULT NULL,  
 `remark` varchar(128) DEFAULT NULL,  
 PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO\_INCREMENT=93 DEFAULT CHARSET=utf8;  
​  
​​  
-- ----------------------------  
-- Records of t\_member  
-- ----------------------------  
INSERT INTO `t\_member` VALUES ('82', null, '小明', '1', '123456789000999999', '18544279942', '2020-02-08', null, null, null, null);  
INSERT INTO `t\_member` VALUES ('83', null, '美丽', '1', '132333333333333', '13412345678', '2020-02-11', null, null, null, null);  
INSERT INTO `t\_member` VALUES ('84', null, '小星星', null, null, '18343569942', '2020-02-13', null, null, null, null);  
INSERT INTO `t\_member` VALUES ('85', null, '小强', null, null, 18344249942, '2020-02-06', null, null, null, null);  
INSERT INTO `t\_member` VALUES ('86', null, '红康', null, null, 18544579942, '2020-02-04', null, null, null, null);  
INSERT INTO `t\_member` VALUES ('87', null, '大海', null, null, 18544274442, '2020-02-06', null, null, null, null);  
INSERT INTO `t\_member` VALUES ('88', null, '小婷', null, null, 18544269942, '2020-02-10', null, null, null, null);  
INSERT INTO `t\_member` VALUES ('89', null, '晓晓', null, null, 18544339942, '2020-02-01', null, null, null, null);  
INSERT INTO `t\_member` VALUES ('90', null, '纯生', null, null, 18544669942, '2020-02-02', null, null, null, null);  
INSERT INTO `t\_member` VALUES ('91', null, '闰土', null, null, 18544779942, '2020-02-01', null, null, null, null);  
INSERT INTO `t\_member` VALUES ('92', null, '波波', '2', '234234145432121345', '18019286521', '2020-02-19', null, null, null, null);  
​  
​  
​DROP TABLE IF EXISTS `t\_order`;  
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（是否旅游）',  
 `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;  
​  
​  
​-- ----------------------------  
-- Records of t\_order  
-- ----------------------------  
INSERT INTO `t\_order` VALUES (1, 84, '2020-02-01', '微信预约', '已出游', 3);  
INSERT INTO `t\_order` VALUES (2, 84, '2020-02-17', '微信预约', '未出游', 3);  
INSERT INTO `t\_order` VALUES (3, 84, '2020-02-01', '微信预约', '未出游', 3);  
INSERT INTO `t\_order` VALUES (4, 84, '2020-02-17', '微信预约', '已出游', 4);  
INSERT INTO `t\_order` VALUES (5, 84, '2020-02-09', '微信预约', '未出游', 4);  
INSERT INTO `t\_order` VALUES (6, 84, '2020-02-09', '微信预约', '已出游', 4);  
INSERT INTO `t\_order` VALUES (7, 84, '2020-02-18', '微信预约', '未出游', 5);  
INSERT INTO `t\_order` VALUES (8, 84, '2020-02-10', '微信预约', '未出游', 6);  
INSERT INTO `t\_order` VALUES (9, 84, '2020-02-10', '微信预约', '未出游', 6);  
INSERT INTO `t\_order` VALUES (10, 84, '2020-02-13', '微信预约', '未出游', 7);  
INSERT INTO `t\_order` VALUES (17, 84, '2020-02-06', '微信预约', '已出游', 12);

### 【讲解】

## 3.1. 前端代码

计算机生成了可选文字:
套餐详肯
项目名称
项目内容
越南芽庄6日5晚大阪海游诓亲@接触5日
自行休闲游日本奈良公匠小
鹿乱7日古都游大阝反
道顿堀小吃不停6日美
食游
立即预约
项乬解读
土艮冥10日游超6000人
选陉牛人亏线／五星航空
直飞丙双飞／持希耶／
红吕点齐打卡丿五星泗
店升1蹼曰洞穴氵酉店，分
级4持色萏／2顿中餐

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

<div class="box-button">

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

</div>

2：toOrderInfo()方法：

toOrderInfo(){

window.location.href = "orderInfo.html?id=" + id;

}

3：在预约页面（/pages/orderInfo.html）进行调整

### 3.1.1. 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, // 用于传递套餐id  
               sex:'1' // 用于默认显示性别男  
          }//预约信息  
      }  
  });  
</script>

第三步：显示套餐信息

<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>{{setmeal.sex == '0' ? '性别不限' : setmeal.sex == '1' ? '男':'女'}}</span>  
           <span>{{setmeal.age}}</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>

第四步：在VUE的钩子函数中发送ajax请求，根据id查询套餐信息，修改七牛云图片地址

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;  
  });  
}

第五步：修改七牛云图片地址



### 3.1.2. 手机号校验

回顾：正则表达式简单语法及常用正则表达式

<https://www.w3cschool.cn/jsref/jsref-obj-regexp.html>

基本符号：

^ 表示匹配字符串的开始位置 (例外 用在中括号中[ ] 时,可以理解为取反,表示不匹配括号中字符串)

$ 表示匹配字符串的结束位置

\* 表示匹配 零次到多次

+ 表示匹配 一次到多次 (至少有一次)

? 表示匹配零次或一次

. 表示匹配单个字符

| 表示为或者,两项中取一项

( ) 小括号表示匹配括号中全部字符

[ ] 中括号表示匹配括号中一个字符 范围描述 如[0-9 a-z A-Z]

{ } 大括号用于限定匹配次数 如 {n}表示匹配n个字符 {n,}表示至少匹配n个字符 {n,m}表示至少n,最多m

\ 转义字符 如上基本符号匹配都需要转义字符 如 \\* 表示匹配\*号

\w 表示英文字母和数字 \W 非字母和数字

\d 表示数字 \D 非数字

例子：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

</body>

<script type="text/javascript">

// 邮箱："^[a-zA-Z0-9-\_]+@[a-zA-Z0-9-\_]+([.][a-zA-Z0-9-\_]+)+$"

var reg = new RegExp("1[345678]\\d{9}")

var tel = "13123462789";

var result = reg.test(tel);

console.log(result)

</script>

</html>

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;

}

},

### 3.1.3. 30秒倒计时效果

第一步：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  
},

第二步：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  
  }  
}

### 3.1.4. 发送ajax请求

接口文档：

请求方式：get 请求

请求路径 ：

"/validateCode/send4Order.do?telephone=" + telephone

第一步：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');  
      }  
  });  
},

第二步：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.atguigu.controller;  
​  
import com.atguigu.constant.MessageConstant;  
import com.atguigu.constant.RedisConstant;  
import com.atguigu.constant.RedisMessageConstant;  
import com.atguigu.entity.Result;  
import com.atguigu.utils.SMSUtils;  
import com.atguigu.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;  
​  
@RestController  
@RequestMapping("/validateCode")  
public class ValidateCodeController {  
   
   @Autowired  
   private JedisPool jedisPool;  
​  
   //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(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);  
  }  
}

### 3.1.5. 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：定义日期

通过样式：.picktime，对应input组件中的class="picktime"

<div class="date">

<label>旅游日期</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

.picktime表示通过样式查找输入框。

<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);

}

计算机生成了可选文字:
2019
25
？6

### 3.1.6. 提交预约请求（身份证校验）

为提交预约按钮绑定事件

第一步：定义“旅游预约”

<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);

}

});

}

第三步：其中checkIdCard方法是在healthmobile.js文件中定义的，用来验证身份证的js

/\*\*

\* 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;

}

}

## 3.2. 后台代码

### 3.2.1. Controller

在 meinian\_mobile\_web 工程中创建 OrderMobileController 并提供 submitOrder 方法

package com.atguigu.controller;  
​  
import com.alibaba.dubbo.config.annotation.Reference;  
import com.atguigu.constant.MessageConstant;  
import com.atguigu.constant.RedisMessageConstant;  
import com.atguigu.entity.Result;  
import com.atguigu.pojo.Order;  
import com.atguigu.service.OrderService;  
import com.atguigu.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;  
​  
@RestController  
@RequestMapping("/order")  
public class OrderMobileController {  
   @Reference  
   private OrderService orderService;  
   @Autowired  
   private JedisPool jedisPool;  
​  
   /\*\*  
    \* 旅游预约  
    \* @param map  
    \* @return  
    \*/  
   @RequestMapping("/submit")  
   public Result submitOrder(@RequestBody Map map){  
       // ① 在页面获取手机号  
       String telephone = (String) map.get("telephone");  
        //② 在页面获取验证码  
       String validateCode = (String) map.get("validateCode");  
       //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);  
        
       //校验手机验证码  
       if(codeInRedis == null || !codeInRedis.equals(validateCode)){  
           return new Result(false, MessageConstant.VALIDATECODE\_ERROR);  
      }  
       Result result =null;  
       //调用旅游预约服务  
       try{  
           map.put("orderType", Order.ORDERTYPE\_WEIXIN);  
           // 断点调试，查看map里面封装了哪些数据  
           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，短信通知内容可以是“预约时间”，“预约人”，“预约地点”，“预约事项”等信息。  
           String orderDate = (String) map.get("orderDate");  
           try {  
               SMSUtils.sendShortMessage(telephone,orderDate);  
          } catch (ClientException e) {  
               e.printStackTrace();  
          }  
      }  
       return result;  
  }  
}

### 3.2.2. 服务接口

在 meinian\_interface 工程中创建旅游预约服务接口 OrderService 并提供预约方法

package com.atguigu.service;  
​  
import com.atguigu.entity.Result;  
import java.util.Map;  
​  
public interface OrderService {  
   Result order(Map map) throws Exception;  
}

### 3.2.3. 服务实现类

在 meinian\_service 工程中创建旅游预约服务实现类 OrderServiceImpl 并实现旅游预约方法。

旅游预约方法处理逻辑比较复杂，需要进行如下业务处理：

【路径】

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.atguigu.service.impl;  
​  
import com.alibaba.dubbo.config.annotation.Service;  
import com.atguigu.constant.MessageConstant;  
import com.atguigu.dao.MemberDao;  
import com.atguigu.dao.OrderDao;  
import com.atguigu.dao.OrdersettingDao;  
import com.atguigu.entity.Result;  
import com.atguigu.pojo.Member;  
import com.atguigu.pojo.Order;  
import com.atguigu.pojo.OrderSetting;  
import com.atguigu.service.CheckItemService;  
import com.atguigu.service.OrderService;  
import com.atguigu.utils.DateUtils;  
import org.apache.poi.ss.usermodel.DateUtil;  
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;  
​  
@Service(interfaceClass = OrderService.class)  
@Transactional  
public class OrderServiceImpl implements OrderService {  
​  
   @Autowired  
   private OrderSettingDao orderSettingDao;  
   @Autowired  
   private MemberDao memberDao;  
   @Autowired  
   private OrderDao orderDao;  
​  
   //旅游预约  
   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");  
       // 因为数据库预约设置表里面的时间是date类型，http协议传递的是字符串类型，所以需要转换  
       Date date = DateUtils.parseString2Date(orderDate);  
       // 使用预约时间查询预约设置表，看看是否可以 进行预约  
        //（1）使用预约时间，查询预约设置表，判断是否有该记录  
       OrderSetting orderSetting = orderSettingDao.findByOrderDate(date);  
       // 如果预约设置表等于null，说明不能进行预约，压根就没有开团  
       if (orderSetting == null) {  
           // 如果没有说明预约设置表没有进行设置，此时不能预约  
           return new Result(false,MessageConstant.SELECTED\_DATE\_CANNOT\_ORDER);  
      }else{  
              //如果有，说明预约可以进行预约，//可预约人数  
               int number = orderSetting.getNumber();  
              //已预约人数  
               int reservations = orderSetting.getReservations();  
           //如果预约人数大于等于最大预约数，此时不能预约，提示“预约已满”  
               if(reservations>=number){  
                   return new Result(false, MessageConstant.ORDER\_FULL);  
              }  
        }  
​  
       //获取手机号  
       String telephone = (String) map.get("telephone");  
      //（2）使用手机号，查询会员表，判断当前预约人是否是会员  
       // 根据手机号，查询会员表，判断当前预约人是否是会员  
       Member member = memberDao.findByTelephone(telephone);  
       //如果是会员，防止重复预约（一个会员、一个时间、一个套餐不能重复，否则是重复预约）  
       if (member != null) {  
           Integer memberId = member.getId();  
           // 获取套餐id  
           int setmealId = Integer.parseInt((String) map.get("setmealId"));  
           Order order = new Order(memberId, date, null, null, setmealId);  
           // 根据预约信息查询是否已经预约  
           List<Order> list = orderDao.findByCondition(order);  
           // 判断是否已经预约list不等于null，说明已经预约，不能重复预约  
           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);  
          }  
      }else{  
               // 如果不是会员：注册会员，向会员表中添加数据  
               member = new Member();  
               member.setName((String)map.get("name"));  
               member.setSex((String)map.get("sex"));  
               member.setPhoneNumber((String)map.get("telephone"));  
               member.setIdCard((String)map.get("idCard"));  
               member.setRegTime(new Date()); // 会员注册时间，当前时间  
               memberDao.add(member);  
          }  
        //（3）可以进行预约，更新预约设置表中预约人数的值，使其的值+1  
       //可以预约，设置预约人数加一  
       orderSetting.setReservations(orderSetting.getReservations() + 1);  
       orderSettingDao.editReservationsByOrderDate(orderSetting);  
​  
        
       //（4）可以进行预约，向预约表中添加1条数据  
       //保存预约信息到预约表  
       Order order = new Order();  
       order.setMemberId(member.getId()); //会员id  
       order.setOrderDate(date); // 预约时间  
       order.setOrderStatus(Order.ORDERSTATUS\_NO); // 预约状态（已出游/未出游）  
       order.setOrderType((String)map.get("orderType"));  
       order.setSetmealId(Integer.parseInt((String)map.get("setmealId")));  
       orderDao.add(order);  
​  
       return new Result(true, MessageConstant.ORDER\_SUCCESS, order);  
  }  
}

导入DateUtils放置到meinian\_common中

### 3.2.4. Dao接口

#### 3.2.4.1. OrderSettingDao.java

package com.atguigu.dao;  
​  
import com.atguigu.pojo.OrderSetting;  
import java.util.Date;  
import java.util.List;  
import java.util.Map;  
​  
public interface OrderSettingDao {  
   //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);  
}

#### 3.2.4.2. MemberDao.java

package com.atguigu.dao;  
​  
import com.atguigu.pojo.Member;  
​  
public interface MemberDao {  
   // 添加会员  
   public void add(Member member);  
   // 根据手机号查询会员信息（唯一）  
   public Member findByTelephone(String telephone);  
}

#### 3.2.4.3. OrderDao.java

package com.atguigu.dao;  
​  
import com.atguigu.pojo.Order;  
import java.util.List;  
​  
public interface OrderDao {  
   List<Order> findByCondition(Order order);  
   void add(Order order);  
}

### 3.2.5. Mapper映射文件

#### 3.2.5.1. OrderSettingDao.xml

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

#### 3.2.5.2. 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.atguigu.dao.MemberDao">  
​  
   <!--新增会员-->  
   <insert id="add" parameterType="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>  
​  
   <!--根据手机号查询会员-->  
   <select id="findByTelephone" parameterType="string" resultType="member">  
      select \* from t\_member where phoneNumber = #{phoneNumber}  
   </select>  
</mapper>

#### 3.2.5.3. 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.atguigu.dao.OrderDao">  
  
    <!--组织预约的查询条件，查询预约信息，  
      因为目前是三个预约条件查询，那么以后可能有4个，5个，甚至更多的条件查询，所以使用动态sql，全部查询  
   -->  
   <select id="findByCondition" parameterType="order" resultType="order">  
      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>  
​  
   <!--新增-->  
   <insert id="add" parameterType="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>  
​  
</mapper>

测试：

细节注意：

SQL语句 里面 为什么有and 不受影响

<select id="findByCondition" parameterType="order" resultType="order">  
       SELECT \* FROM t\_order  
      <where>  
          <if test="id != null">  
               and id = #{id}  
          </if>  
        </where>  
</select>  
  
有where标签，and会自动识别，where类似于where 1=1，这样后面就可以添加and了，SQL语句就变成如下：  
select \* from t\_order where 1=1 and member\_id =#{memberId}

### 【小结】

验证码

1.调用阿里服务, 发送成功后, 验证码存到Redis(存5分钟)

2.用户点击了预约, 需要把用户输入的验证码和redis里面存的验证码进行校验

预约业务

1.判断当前的日期是否可以预约

2.判断当前的日期预约是否已满

3.判断是否 是会员

* 如果是会员, 避免重复预约
* 不是会员, 自动注册成会员，t\_member 表插入一条记录

4.进行预约

* 向t\_order 表插入一条记录
* 向t\_ordersetting更新reservations+1

# 第四章. 预约成功页面展示

### 【目标】

前面已经完成了旅游预约，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（会员姓名、旅游套餐、旅游时间等）。

使用订单id，查询订单详细信息 在OrderDao.xml中会使用联合查询的sql，关联订单表，会员表，套餐表，以map数据结构返回。

orderSuccess.html

计算机生成了可选文字:
美年旅游
《旅游预约成功
预约信息
会员名：4444
旅游套者：音港九龙尖沙商双飞3一8日由行套
旅游日期：2020．02．04
预约型：有信预约
氵王意事1.页
1、每到一个地方，首先咨询游客服务中心相关事项，做到心中有数。
2、椠坐汽车公交车时，好安全带，氵意不要意更座位，以防急刹车或者突然
加速导致葬倒：不要将头手伸出窗外，上下车时要沌意前后来往车辆以兔发生危险。
3、按不同季节、地区、出游方式，杲好帚好个，人、防护用品，如纸门，衣物，折
伞，防晒箔等：昔用药物，如晕午药、风油佰，驱风油、感冒药等：以及镜、旅游
4、外出旅游时不要携帝大量现金和妥重物品：塗身份证，护照，信用卡，机等
麦重物品应妥善保管，芾在身上时不要脱离视线，或者放在酒店保险箱。
5、每次退秀前·检己所携带的行物品，特别意目己的证件以及一些蛊财
物，保证目己所有的东西都合完。
6、物和在迸行休闲娱乐氵肖或苔候车时要意巨身财物安全，保管好发票或凭
证，物时也勿兰众清数钞票，避免被小佝苷上。
7、在外旅游时要意身康，身旅游的本
8、在名胜古迹游玩时，要目觉守参观地点旅游规定。
9、在进行爬山或者幸等要爬高处的活讠了，既要防止不刂、心踟落壹伤，同时
也要预防脚被尖锐物扌L伤或被山区蛇虫咬历。

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);  
  }  
});

### 【路径】

前台代码编写

在/pages/orderSuccess.html

会员姓名：{{orderInfo.member}}

旅游套餐：{{orderInfo.setmeal}}

旅游日期：{{orderInfo.orderDate}}

预约类型：{{orderInfo.orderType}}

完成需求：  
1.页面输出订单相关人的信息  
2.使用订单id，查询订单详细信息。存放到orderInfo的模型中。

后台代码编写：

1.类OrderController.java

2.类OrderService.java

3.类OrderServiceImpl.java

4.类OrderDao.java

5.配置文件OrderDao.xml

使用订单id，查询订单详细信息  
在OrderDao.xml中会使用联合查询的sql，关联订单表，会员表，套餐表，以map数据结构返回。

### 【讲解】

## 4.1. 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">旅游预约成功</span>  
</div>  
<div class="notice-item">  
   <div class="item-title">旅游信息</div>  
   <div class="item-content">  
       <p>会员姓名：{{orderInfo.member}}</p>  
       <p>旅游套餐：{{orderInfo.setmeal}}</p>  
       <p>旅游日期：{{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>

## 4.2. 后台代码

### 4.2.1. Controller

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

package com.atguigu.controller;  
​  
import com.alibaba.dubbo.config.annotation.Reference;  
import com.atguigu.constant.MessageConstant;  
import com.atguigu.constant.RedisMessageConstant;  
import com.atguigu.entity.Result;  
import com.atguigu.pojo.Order;  
import com.atguigu.service.OrderService;  
import com.atguigu.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;  
​  
/\*\*  
\* OrderController  
\* @Description: 根据id查询预约信息，包括套餐信息和会员信息  
\*/  
@RestController  
@RequestMapping("/order")  
public class OrderController {  
​  
   @Autowired  
   private JedisPool jedisPool;  
​  
   @Reference  
   private OrderService orderService;  
​  
   @RequestMapping("/findById")  
   public Result findById(Integer id){  
       Map map =null;  
       try{  
           map = orderService.findById4Detail(id);  
           //查询预约信息成功  
           return new Result(true,MessageConstant.QUERY\_ORDER\_SUCCESS,map);  
      }catch (Exception e){  
           e.printStackTrace();  
           //查询预约信息失败  
           return new Result(false,MessageConstant.QUERY\_ORDER\_FAIL);  
      }  
  }

### 4.2.2. 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，包括人信息、套餐信息  
Map findById4Detail(Integer id);

### 4.2.3. 服务实现类

在 OrderServiceImpl 服务实现类中实现 findById4Detail 方法

package com.atguigu.service.impl;  
​  
import com.alibaba.dubbo.config.annotation.Service;  
import com.atguigu.constant.MessageConstant;  
import com.atguigu.dao.MemberDao;  
import com.atguigu.dao.OrderDao;  
import com.atguigu.dao.OrdersettingDao;  
import com.atguigu.entity.Result;  
import com.atguigu.pojo.Member;  
import com.atguigu.pojo.Order;  
import com.atguigu.pojo.OrderSetting;  
import com.atguigu.service.CheckItemService;  
import com.atguigu.service.OrderService;  
import com.atguigu.utils.DateUtils;  
import org.apache.poi.ss.usermodel.DateUtil;  
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;  
​  
  
@Service(interfaceClass = OrderService.class)  
@Transactional  
public class OrderServiceImpl implements OrderService {  
​  
   @Autowired  
   private OrdersettingDao ordersettingDao;  
​  
   @Autowired  
   private MemberDao memberDao;  
​  
   @Autowired  
   private OrderDao orderDao;  
​  
   //根据id查询预约信息，包括旅游人信息、套餐信息  
   public Map 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 map  
      }  
       return map;  
  }

### 4.2.4. Dao接口

在 OrderDao 接口中扩展 findById4Detail 方法

Map findById4Detail(Integer id);

### 4.2.5. 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，包括旅游人信息、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>会员姓名：{{orderInfo.member}}</p>  
<p>旅游套餐：{{orderInfo.setmeal}}</p>  
<p>旅游日期：{{orderInfo.orderDate}}</p>  
<p>预约类型：{{orderInfo.orderType}}</p>  
​

### 【小结】

使用订单id，查询订单详细信息 在OrderDao.xml中会使用联合查询的sql，关联订单表，会员表，套餐表，以map数据结构返回。