尚医通

版本：V1.0

# 一、预约下单

## 需求分析

### 订单表结构



### 下单分析

参考《尚医通API接口文档.docx》业务接口5.1预约下单

下单参数：就诊人id与排班id

1. 下单我们要获取就诊人信息
2. 获取排班下单信息与规则信息
3. 获取医院签名信息，然后通过接口去医院预约下单
4. 下单成功更新排班信息与发送短信

## 2、搭建service-order模块

### 2.1 搭建service-order模块

搭建过程参考service-user模块

### 2.2 修改配置

1、修改pom.xml，引入依赖

|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>com.atguigu</**groupId**>  <**artifactId**>service\_cmn\_client</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  </**dependency**>  </**dependencies**> |

2、添加配置文件application.properties

|  |
| --- |
| *# 服务端口*  **server.port**=**8206**  *# 服务名*  **spring.application.name**=**service-order**  *# 环境设置：dev、test、prod*  **spring.profiles.active**=**dev**  *# mysql数据库连接*  **spring.datasource.driver-class-name**=**com.mysql.jdbc.Driver**  **spring.datasource.url**=**jdbc:mysql://192.168.44.165:3306/yygh\_hosp?characterEncoding=utf-8&useSSL=false**  **spring.datasource.username**=**root**  **spring.datasource.password**=**root123**  *#返回json的全局时间格式*  **spring.jackson.date-format**=**yyyy-MM-dd HH:mm:ss**  **spring.jackson.time-zone**=**GMT+8**  **spring.data.mongodb.uri**=**mongodb://192.168.44.165:27017/yygh\_hosp**  *# nacos服务地址*  **spring.cloud.nacos.discovery.server-addr**=**127.0.0.1:8848**  *#rabbitmq地址*  **spring.rabbitmq.host**=**192.168.44.165**  **spring.rabbitmq.port**=**5672**  **spring.rabbitmq.username**=**guest**  **spring.rabbitmq.password**=**guest** |

### 2.3 启动类

|  |
| --- |
| @SpringBootApplication  @ComponentScan(basePackages = {**"com.atguigu"**})  @EnableDiscoveryClient  @EnableFeignClients(basePackages = {**"com.atguigu"**})  **public class** ServiceOrderApplication {  **public static void** main(String[] args) {  SpringApplication.*run*(ServiceOrderApplication.**class**, args);  }  } |

### 2.4配置网关

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

## 3、添加订单基础类

### 3.1 添加model

说明：由于实体对象没有逻辑，我们已经统一导入

com.atguigu.yygh.model.order.OrderInfo

### 3.2 添加Mapper

添加com.atguigu.yygh.order.mapper.OrderInfoMapper

|  |
| --- |
| **public interface** OrderInfoMapper **extends** BaseMapper<OrderInfo> {  } |

### 3.3 添加service接口及实现类

1、添加com.atguigu.yygh.order.service.OrderService接口

|  |
| --- |
| **public interface** OrderService **extends** IService<OrderInfo> {  *//保存订单*  Long saveOrder(String scheduleId, Long patientId);  } |

2、添加com.atguigu.yygh.order.service.impl.OrderServiceImpl接口实现

|  |
| --- |
| @Service  **public class** OrderServiceImpl **extends**  ServiceImpl<OrderMapper, OrderInfo> **implements** OrderService {  *//保存订单*  @Override  **public** Long saveOrder(String scheduleId, Long patientId) {  **return null**;  }  } |

### 3.4 添加controller

添加com.atguigu.yygh.order.api.OrderApiController类

|  |
| --- |
| @Api(tags = **"订单接口"**)  @RestController  @RequestMapping(**"/api/order/orderInfo"**)  **public class** OrderApiController {  @Autowired  **private** OrderService **orderService**;  @ApiOperation(value = **"创建订单"**)  @PostMapping(**"auth/submitOrder/{scheduleId}/{patientId}"**)  **public** Result submitOrder(  @ApiParam(name = **"scheduleId"**, value = **"排班id"**, required = **true**)  @PathVariable String scheduleId,  @ApiParam(name = **"patientId"**, value = **"就诊人id"**, required = **true**)  @PathVariable Long patientId) {  **return** Result.*ok*(**orderService**.saveOrder(scheduleId, patientId));  }  } |

## 封装Feign调用获取就诊人接口

### 4.1 获取就诊人信息api接口

操作模块：service-user

在PatientApiController类添加方法

|  |
| --- |
| @ApiOperation(value = **"获取就诊人"**)  @GetMapping(**"inner/get/{id}"**)  **public** Patient getPatientOrder(  @ApiParam(name = **"id"**, value = **"就诊人id"**, required = **true**)  @PathVariable(**"id"**) Long id) {  **return patientService**.getById(id);  } |

### 4.2 搭建service-user-client模块

#### 4.2.1 搭建service-user-client模块

搭建过程如service-cmn-client模块

#### 4.2.2添加Feign接口类

|  |
| --- |
| @FeignClient(value = **"service-user"**)  @Repository  **public interface** PatientFeignClient {  *//获取就诊人*  @GetMapping(**"/api/user/patient/inner/get/{id}"**)  Patient getPatient(@PathVariable(**"id"**) Long id);  } |

## 封装Feign调用获取排班下单信息接口

### 5.1 获取排班下单信息api接口

操作模块：service-hosp

#### 5.1.1 添加service接口与实现

1、在ScheduleService类添加接口

|  |
| --- |
| *//根据排班id获取预约下单数据*  ScheduleOrderVo getScheduleOrderVo(String scheduleId); |

2、在ScheduleServiceImpl类添加实现

|  |
| --- |
| *//根据排班id获取预约下单数据*  @Override  **public** ScheduleOrderVo getScheduleOrderVo(String scheduleId) {  ScheduleOrderVo scheduleOrderVo = **new** ScheduleOrderVo();  *//排班信息*  Schedule schedule = **baseMapper**.selectById(scheduleId);  **if**(**null** == schedule) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//获取预约规则信息*  Hospital hospital = **hospitalService**.getByHoscode(schedule.getHoscode());  **if**(**null** == hospital) {  **throw new** YyghException(ResultCodeEnum.***DATA\_ERROR***);  }  BookingRule bookingRule = hospital.getBookingRule();  **if**(**null** == bookingRule) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  scheduleOrderVo.setHoscode(schedule.getHoscode());  scheduleOrderVo.setHosname(**hospitalService**.getHospName(schedule.getHoscode()));  scheduleOrderVo.setDepcode(schedule.getDepcode());  scheduleOrderVo.setDepname(**departmentService**.getDepName(schedule.getHoscode(), schedule.getDepcode()));  scheduleOrderVo.setHosScheduleId(schedule.getHosScheduleId());  scheduleOrderVo.setAvailableNumber(schedule.getAvailableNumber());  scheduleOrderVo.setTitle(schedule.getTitle());  scheduleOrderVo.setReserveDate(schedule.getWorkDate());  scheduleOrderVo.setReserveTime(schedule.getWorkTime());  scheduleOrderVo.setAmount(schedule.getAmount());  *//退号截止天数（如：就诊前一天为-1，当天为0）*  **int** quitDay = bookingRule.getQuitDay();  DateTime quitTime = **this**.getDateTime(**new** DateTime(schedule.getWorkDate()).plusDays(quitDay).toDate(), bookingRule.getQuitTime());  scheduleOrderVo.setQuitTime(quitTime.toDate());  *//预约开始时间*  DateTime startTime = **this**.getDateTime(**new** Date(), bookingRule.getReleaseTime());  scheduleOrderVo.setStartTime(startTime.toDate());  *//预约截止时间*  DateTime endTime = **this**.getDateTime(**new** DateTime().plusDays(bookingRule.getCycle()).toDate(), bookingRule.getStopTime());  scheduleOrderVo.setEndTime(endTime.toDate());  *//当天停止挂号时间*  DateTime stopTime = **this**.getDateTime(**new** Date(), bookingRule.getStopTime());  scheduleOrderVo.setStartTime(startTime.toDate());  **return** scheduleOrderVo;  } |

#### 5.1.2 添加controller方法

在HospitalApiController类添加方法

|  |
| --- |
| @ApiOperation(value = **"根据排班id获取预约下单数据"**)  @GetMapping(**"inner/getScheduleOrderVo/{scheduleId}"**)  **public** ScheduleOrderVo getScheduleOrderVo(  @ApiParam(name = **"scheduleId"**, value = **"排班id"**, required = **true**)  @PathVariable(**"scheduleId"**) String scheduleId) {  **return scheduleService**.getScheduleOrderVo(scheduleId);  } |

### 5.2 获取下单引用签名信息接口

操作模块：service-hosp

#### 5.2.1 添加service接口与实现

1、在HospitalSetService类添加接口

|  |
| --- |
| *//获取医院签名信息*  SignInfoVo getSignInfoVo(String hoscode); |

2、在HospitalSetServiceImpl类添加实现

|  |
| --- |
| *//获取医院签名信息*  @Override  **public** SignInfoVo getSignInfoVo(String hoscode) {  QueryWrapper<HospitalSet> wrapper = **new** QueryWrapper<>();  wrapper.eq(**"hoscode"**,hoscode);  HospitalSet hospitalSet = **baseMapper**.selectOne(wrapper);  **if**(**null** == hospitalSet) {  **throw new** YyghException(ResultCodeEnum.***HOSPITAL\_OPEN***);  }  SignInfoVo signInfoVo = **new** SignInfoVo();  signInfoVo.setApiUrl(hospitalSet.getApiUrl());  signInfoVo.setSignKey(hospitalSet.getSignKey());  **return** signInfoVo;  } |

#### 5.2.2 添加controller方法

在HospitalApiController类添加方法

|  |
| --- |
| @ApiOperation(value = **"获取医院签名信息"**)  @GetMapping(**"inner/getSignInfoVo/{hoscode}"**)  **public** SignInfoVo getSignInfoVo(  @ApiParam(name = **"hoscode"**, value = **"医院code"**, required = **true**)  @PathVariable(**"hoscode"**) String hoscode) {  **return hospitalSetService**.getSignInfoVo(hoscode);  } |

### 5.3 搭建service-hosp-client模块

#### 5.3.1 搭建service-hosp-client模块

搭建过程如service-cmn-client模块

#### 5.3.2添加Feign接口类

|  |
| --- |
| @FeignClient(value = **"service-hosp"**)  @Repository  **public interface** HospitalFeignClient {  */\*\**  *\* 根据排班id获取预约下单数据*  *\*/*  @GetMapping(**"/api/hosp/hospital/inner/getScheduleOrderVo/{scheduleId}"**)  ScheduleOrderVo getScheduleOrderVo(@PathVariable(**"scheduleId"**) String scheduleId);  */\*\**  *\* 获取医院签名信息*  *\*/*  @GetMapping(**"/api/hosp/hospital/inner/getSignInfoVo/{hoscode}"**)  SignInfoVo getSignInfoVo(@PathVariable(**"hoscode"**) String hoscode);  } |

## 实现下单接口

操作模块：service-order

### 6.1 引入依赖

|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>com.atguigu</**groupId**>  <**artifactId**>service\_cmn\_client</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.atguigu</**groupId**>  <**artifactId**>service\_hosp\_client</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.atguigu</**groupId**>  <**artifactId**>service\_user\_client</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  </**dependency**>  </**dependencies**> |

### 6.2 封装下单工具类

封装HttpRequestHelper类，添加签名请求方法

|  |
| --- |
| **public class** HttpRequestHelper {  */\*\**  *\**  *\** ***@param paramMap***  *\** ***@return***  *\*/*  **public static** Map<String, Object> switchMap(Map<String, String[]> paramMap) {  Map<String, Object> resultMap = **new** HashMap<>();  **for** (Map.Entry<String, String[]> param : paramMap.entrySet()) {  resultMap.put(param.getKey(), param.getValue()[0]);  }  **return** resultMap;  }  */\*\**  *\* 请求数据获取签名*  *\** ***@param paramMap***  *\** ***@param signKey***  *\** ***@return***  *\*/*  **public static** String getSign(Map<String, Object> paramMap, String signKey) {  **if**(paramMap.containsKey(**"sign"**)) {  paramMap.remove(**"sign"**);  }  TreeMap<String, Object> sorted = **new** TreeMap<>(paramMap);  StringBuilder str = **new** StringBuilder();  **for** (Map.Entry<String, Object> param : sorted.entrySet()) {  str.append(param.getValue()).append(**"|"**);  }  str.append(signKey);  ***log***.info(**"加密前："** + str.toString());  String md5Str = MD5.*encrypt*(str.toString());  ***log***.info(**"加密后："** + md5Str);  **return** md5Str;  }  */\*\**  *\* 签名校验*  *\** ***@param paramMap***  *\** ***@param signKey***  *\** ***@return***  *\*/*  **public static boolean** isSignEquals(Map<String, Object> paramMap, String signKey) {  String sign = (String)paramMap.get(**"sign"**);  String md5Str = *getSign*(paramMap, signKey);  **if**(!sign.equals(md5Str)) {  **return false**;  }  **return true**;  }  */\*\**  *\* 获取时间戳*  *\** ***@return***  *\*/*  **public static long** getTimestamp() {  **return new** Date().getTime();  }  */\*\**  *\* 封装同步请求*  *\*/*  **public static** JSONObject sendRequest(Map<String, Object> paramMap, String url){  String result = **""**;  **try** {  *//封装post参数*  StringBuilder postdata = **new** StringBuilder();  **for** (Map.Entry<String, Object> param : paramMap.entrySet()) {  postdata.append(param.getKey()).append(**"="**)  .append(param.getValue()).append(**"&"**);  }  ***log***.info(String.*format*(**"--> 发送请求：post data %1s"**, postdata));  **byte**[] reqData = postdata.toString().getBytes(**"utf-8"**);  **byte**[] respdata = HttpUtil.*doPost*(url,reqData);  result = **new** String(respdata);  ***log***.info(String.*format*(**"--> 应答结果：result data %1s"**, result));  } **catch** (Exception ex) {  ex.printStackTrace();  }  **return** JSONObject.*parseObject*(result);  }  } |

### 6.3 实现下单接口

修改OrderServiceImpl类下单方法

|  |
| --- |
| @Service  **public class** OrderServiceImpl **extends**  ServiceImpl<OrderMapper, OrderInfo> **implements** OrderService {  @Autowired  **private** PatientFeignClient **patientFeignClient**;  @Autowired  **private** HospitalFeignClient **hospitalFeignClient**;  *//保存订单*  @Override  **public** Long saveOrder(String scheduleId, Long patientId) {  Patient patient = **patientFeignClient**.getPatient(patientId);  **if**(**null** == patient) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  ScheduleOrderVo scheduleOrderVo = **hospitalFeignClient**.getScheduleOrderVo(scheduleId);  **if**(**null** == scheduleOrderVo) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//当前时间不可以预约*  **if**(**new** DateTime(scheduleOrderVo.getStartTime()).isAfterNow()  || **new** DateTime(scheduleOrderVo.getEndTime()).isBeforeNow()) {  **throw new** YyghException(ResultCodeEnum.***TIME\_NO***);  }  SignInfoVo signInfoVo = **hospitalFeignClient**.getSignInfoVo(scheduleOrderVo.getHoscode());  **if**(**null** == scheduleOrderVo) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  **if**(scheduleOrderVo.getAvailableNumber() <= 0) {  **throw new** YyghException(ResultCodeEnum.***NUMBER\_NO***);  }  OrderInfo orderInfo = **new** OrderInfo();  BeanUtils.*copyProperties*(scheduleOrderVo, orderInfo);  String outTradeNo = System.*currentTimeMillis*() + **""**+ **new** Random().nextInt(100);  orderInfo.setOutTradeNo(outTradeNo);  orderInfo.setScheduleId(scheduleId);  orderInfo.setUserId(patient.getUserId());  orderInfo.setPatientId(patientId);  orderInfo.setPatientName(patient.getName());  orderInfo.setPatientPhone(patient.getPhone());  orderInfo.setOrderStatus(OrderStatusEnum.***UNPAID***.getStatus());  **this**.save(orderInfo);  Map<String, Object> paramMap = **new** HashMap<>();  paramMap.put(**"hoscode"**,orderInfo.getHoscode());  paramMap.put(**"depcode"**,orderInfo.getDepcode());  paramMap.put(**"hosScheduleId"**,orderInfo.getScheduleId());  paramMap.put(**"reserveDate"**,**new** DateTime(orderInfo.getReserveDate()).toString(**"yyyy-MM-dd"**));  paramMap.put(**"reserveTime"**, orderInfo.getReserveTime());  paramMap.put(**"amount"**,orderInfo.getAmount());  paramMap.put(**"name"**, patient.getName());  paramMap.put(**"certificatesType"**,patient.getCertificatesType());  paramMap.put(**"certificatesNo"**, patient.getCertificatesNo());  paramMap.put(**"sex"**,patient.getSex());  paramMap.put(**"birthdate"**, patient.getBirthdate());  paramMap.put(**"phone"**,patient.getPhone());  paramMap.put(**"isMarry"**, patient.getIsMarry());  paramMap.put(**"provinceCode"**,patient.getProvinceCode());  paramMap.put(**"cityCode"**, patient.getCityCode());  paramMap.put(**"districtCode"**,patient.getDistrictCode());  paramMap.put(**"address"**,patient.getAddress());  *//联系人*  paramMap.put(**"contactsName"**,patient.getContactsName());  paramMap.put(**"contactsCertificatesType"**, patient.getContactsCertificatesType());  paramMap.put(**"contactsCertificatesNo"**,patient.getContactsCertificatesNo());  paramMap.put(**"contactsPhone"**,patient.getContactsPhone());  paramMap.put(**"timestamp"**, HttpRequestHelper.*getTimestamp*());  String sign = HttpRequestHelper.*getSign*(paramMap, signInfoVo.getSignKey());  paramMap.put(**"sign"**, sign);  JSONObject result = HttpRequestHelper.*sendRequest*(paramMap, signInfoVo.getApiUrl()+**"/order/submitOrder"**);    **if**(result.getInteger(**"code"**) == 200) {  JSONObject jsonObject = result.getJSONObject(**"data"**);  *//预约记录唯一标识（医院预约记录主键）*  String hosRecordId = jsonObject.getString(**"hosRecordId"**);  *//预约序号*  Integer number = jsonObject.getInteger(**"number"**);;  *//取号时间*  String fetchTime = jsonObject.getString(**"fetchTime"**);;  *//取号地址*  String fetchAddress = jsonObject.getString(**"fetchAddress"**);;  *//更新订单*  orderInfo.setHosRecordId(hosRecordId);  orderInfo.setNumber(number);  orderInfo.setFetchTime(fetchTime);  orderInfo.setFetchAddress(fetchAddress);  **baseMapper**.updateById(orderInfo);  *//排班可预约数*  Integer reservedNumber = jsonObject.getInteger(**"reservedNumber"**);  *//排班剩余预约数*  Integer availableNumber = jsonObject.getInteger(**"availableNumber"**);  *//发送mq信息更新号源和短信通知*  } **else** {  **throw new** YyghException(result.getString(**"message"**), ResultCodeEnum.***FAIL***.getCode());  }  **return** orderInfo.getId();  }  } |

## 预约成功后处理逻辑

预约成功后我们要更新预约数和短信提醒预约成功，为了提高下单的并发性，这部分逻辑我们就交给mq为我们完成，预约成功发送消息即可

**（1）RabbitMQ简介**

以商品订单场景为例，

如果商品服务和订单服务是两个不同的微服务，在下单的过程中订单服务需要调用商品服务进行扣库存操作。按照传统的方式，下单过程要等到调用完毕之后才能返回下单成功，如果网络产生波动等原因使得商品服务扣库存延迟或者失败，会带来较差的用户体验，如果在高并发的场景下，这样的处理显然是不合适的，那怎么进行优化呢？这就需要消息队列登场了。

消息队列提供一个异步通信机制，消息的发送者不必一直等待到消息被成功处理才返回，而是立即返回。消息中间件负责处理网络通信，如果网络连接不可用，消息被暂存于队列当中，当网络畅通的时候在将消息转发给相应的应用程序或者服务，当然前提是这些服务订阅了该队列。如果在商品服务和订单服务之间使用消息中间件，既可以提高并发量，又降低服务之间的耦合度。

RabbitMQ就是这样一款消息队列。RabbitMQ是一个开源的消息代理的队列服务器，用来通过普通协议在完全不同的应用之间共享数据。

**（2）典型应用场景：**

**异步处理**。把消息放入消息中间件中，等到需要的时候再去处理。

**流量削峰**。例如秒杀活动，在短时间内访问量急剧增加，使用消息队列，当消息队列满了就拒绝响应，跳转到错误页面，这样就可以使得系统不会因为超负载而崩溃。

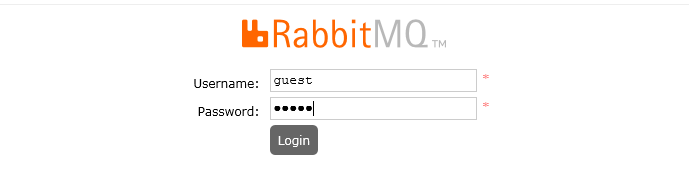
**日志处理**

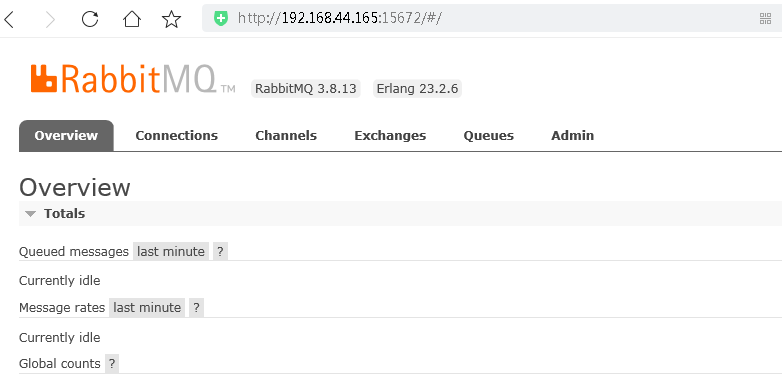
**应用解耦**

**（2）安装RabbitMQ**

|  |
| --- |
| docker pull rabbitmq:management  docker run -d -p 5672:5672 -p 15672:15672 --name rabbitmq rabbitmq:management |

**管理后台**：<http://IP:15672>





### 7.1 rabbit-util模块封装

由于后续可能多个模块都会使用mq，所以我们把它封装成一个模块，需要的地方直接引用即可

#### 7.1.1 搭建rabbit-util模块

在common模块下搭建，搭建过程如：common-util

#### 7.1.2 修改pom.xml

|  |
| --- |
| <**dependencies**>  *<!--rabbitmq消息队列-->*  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-actuator</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-bus-amqp</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>fastjson</**artifactId**>  </**dependency**>  </**dependencies**> |

#### 7.1.3 封装service方法

|  |
| --- |
| @Service  **public class** RabbitService {  @Autowired  **private** RabbitTemplate **rabbitTemplate**;  */\*\**  *\* 发送消息*  *\** ***@param exchange*** *交换机*  *\** ***@param routingKey*** *路由键*  *\** ***@param message*** *消息*  *\*/*  **public boolean** sendMessage(String exchange, String routingKey, Object message) {  **rabbitTemplate**.convertAndSend(exchange, routingKey, message);  **return true**;  }  } |

#### 7.1.4 配置mq消息转换器

|  |
| --- |
| @Configuration  **public class** MQConfig {  @Bean  **public** MessageConverter messageConverter(){  **return new** Jackson2JsonMessageConverter();  }  } |

说明：默认是字符串转换器

### 7.2 封装短信接口

操作模块：service-msm

#### 7.2.1 引入依赖

|  |
| --- |
| <**dependency**>  <**groupId**>com.atguigu</**groupId**>  <**artifactId**>rabbit\_util</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  </**dependency**> |

#### 7.2.2 添加配置

在resources/application.properties添加

|  |
| --- |
| *#rabbitmq地址*  **spring.rabbitmq.host**=**192.168.44.165**  **spring.rabbitmq.port**=**5672**  **spring.rabbitmq.username**=**guest**  **spring.rabbitmq.password**=**guest** |

#### 7.2.3 添加常量配置

在rabbit-util模块com.atguigu.yygh.common.constant.MqConst类添加

|  |
| --- |
| **public class** MqConst {  */\*\**  *\* 预约下单*  *\*/*  **public static final** String ***EXCHANGE\_DIRECT\_ORDER***  = **"exchange.direct.order"**;  **public static final** String ***ROUTING\_ORDER*** = **"order"**;  *//队列*  **public static final** String ***QUEUE\_ORDER*** = **"queue.order"**;  */\*\**  *\* 短信*  *\*/*  **public static final** String ***EXCHANGE\_DIRECT\_MSM*** = **"exchange.direct.msm"**;  **public static final** String ***ROUTING\_MSM\_ITEM*** = **"msm.item"**;  *//队列*  **public static final** String ***QUEUE\_MSM\_ITEM*** = **"queue.msm.item"**;  } |

#### 7.2.4 在model模块封装短信实体

|  |
| --- |
| @Data  @ApiModel(description = **"短信实体"**)  **public class** MsmVo {  @ApiModelProperty(value = **"phone"**)  **private** String **phone**;  @ApiModelProperty(value = **"短信模板code"**)  **private** String **templateCode**;  @ApiModelProperty(value = **"短信模板参数"**)  **private** Map<String,Object> **param**;  } |

说明：已统一引入

#### 7.2.5 封装service接口

1、在MsmService类添加接口

|  |
| --- |
| **boolean** send(MsmVo msmVo); |

2、在MsmServiceImpl类添加接口实现

|  |
| --- |
| @Override  **public boolean** send(MsmVo msmVo) {  **if**(!StringUtils.*isEmpty*(msmVo.getPhone())) {  String code = (String)msmVo.getParam().get(**"code"**);  **return this**.send(msmVo.getPhone(),code);  }  **return false**;  } |

#### 7.2.6 封装mq监听器

|  |
| --- |
| @Component  **public class** SmsReceiver {  @Autowired  **private** MsmService **msmService**;  @RabbitListener(bindings = @QueueBinding(  value = @Queue(value = MqConst.***QUEUE\_MSM\_ITEM***, durable = **"true"**),  exchange = @Exchange(value = MqConst.***EXCHANGE\_DIRECT\_MSM***),  key = {MqConst.***ROUTING\_MSM\_ITEM***}  ))  **public void** send(MsmVo msmVo, Message message, Channel channel) {  **msmService**.send(msmVo);  }  } |

### 7.3 封装更新排班数量

操作模块：service-hosp

#### 7.3.1 引入依赖

|  |
| --- |
| *<!--rabbitmq消息队列-->*  <**dependency**>  <**groupId**>com.atguigu.yygh</**groupId**>  <**artifactId**>rabbit-util</**artifactId**>  <**version**>1.0</**version**>  </**dependency**> |

#### 7.3.2 添加配置

在resources/application.properties添加

|  |
| --- |
| *#rabbitmq地址*  **spring.rabbitmq.host**=**192.168.44.165**  **spring.rabbitmq.port**=**5672**  **spring.rabbitmq.username**=**guest**  **spring.rabbitmq.password**=**guest** |

#### 7.3.3 添加常量配置

在rabbit-util模块com.atguigu.yygh.common.constant.MqConst类添加

|  |
| --- |
| */\*\**  *\* 预约下单*  *\*/*  **public static final** String ***EXCHANGE\_DIRECT\_ORDER*** = **"exchange.direct.order"**;  **public static final** String ***ROUTING\_ORDER*** = **"order"**;  *//队列*  **public static final** String ***QUEUE\_ORDER*** = **"queue.order"**; |

#### 7.3.4 在model模块封装更新排班实体

|  |
| --- |
| @Data  @ApiModel(description = **"OrderMqVo"**)  **public class** OrderMqVo {  @ApiModelProperty(value = **"可预约数"**)  **private** Integer **reservedNumber**;  @ApiModelProperty(value = **"剩余预约数"**)  **private** Integer **availableNumber**;  @ApiModelProperty(value = **"排班id"**)  **private** String **scheduleId**;  @ApiModelProperty(value = **"短信实体"**)  **private** MsmVo **msmVo**;  } |

说明：已统一引入，该对象放一个短信实体，预约下单成功后，我们发送一条消息，让mq来保证两个消息都发送成功

#### 7.3.5 封装service接口

1、在ScheduleService类添加接口

|  |
| --- |
| */\*\**  *\* 修改排班*  *\*/*  **void** update(Schedule schedule); |

2、在ScheduleServiceImpl类添加接口实现

|  |
| --- |
| @Override  **public void** update(Schedule schedule) {  schedule.setUpdateTime(**new** Date());  *//主键一致就是更新*  **scheduleRepository**.save(schedule);  } |

#### 7.3.6 封装mq监听器

|  |
| --- |
| @Component  **public class** HospitalReceiver {  @Autowired  **private** ScheduleService **scheduleService**;  @Autowired  **private** RabbitService **rabbitService**;  @RabbitListener(bindings = @QueueBinding(  value = @Queue(value = MqConst.***QUEUE\_ORDER***, durable = **"true"**),  exchange = @Exchange(value = MqConst.***EXCHANGE\_DIRECT\_ORDER***),  key = {MqConst.***ROUTING\_ORDER***}  ))  **public void** receiver(OrderMqVo orderMqVo, Message message, Channel channel) **throws** IOException {  *//下单成功更新预约数*  Schedule schedule = **scheduleService**.getScheduleId(orderMqVo.getScheduleId());  schedule.setReservedNumber(orderMqVo.getReservedNumber());  schedule.setAvailableNumber(orderMqVo.getAvailableNumber());  **scheduleService**.update(schedule);  *//发送短信*  MsmVo msmVo = orderMqVo.getMsmVo();  **if**(**null** != msmVo) {  **rabbitService**.sendMessage(MqConst.***EXCHANGE\_DIRECT\_MSM***, MqConst.***ROUTING\_MSM\_ITEM***, msmVo);  }  }  } |

### 7.4 调整下单接口

操作模块：service-order

#### 7.4.1 引入依赖

|  |
| --- |
| <**dependency**>  <**groupId**>com.atguigu</**groupId**>  <**artifactId**>rabbit\_util</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  </**dependency**> |

#### 7.4.2 添加配置

在resources/application.properties添加

|  |
| --- |
| *#rabbitmq地址*  **spring.rabbitmq.host**=**192.168.44.165**  **spring.rabbitmq.port**=**5672**  **spring.rabbitmq.username**=**guest**  **spring.rabbitmq.password**=**guest** |

#### 7.4.3 修改下单接口

修改OrderServiceImpl类下单方法

|  |
| --- |
| @Autowired  **private** RabbitService **rabbitService**; |
| @Transactional(rollbackFor = Exception.**class**)  @Override  **public** Long saveOrder(String scheduleId, Long patientId) {  .......  *//排班可预约数*  Integer reservedNumber = jsonObject.getInteger(**"reservedNumber"**);  *//排班剩余预约数*  Integer availableNumber = jsonObject.getInteger(**"availableNumber"**);  **//发送mq信息更新号源和短信通知**  **//发送mq信息更新号源**  **OrderMqVo orderMqVo = new OrderMqVo();**  **orderMqVo.setScheduleId(scheduleId);**  **orderMqVo.setReservedNumber(reservedNumber);**  **orderMqVo.setAvailableNumber(availableNumber);**  //短信提示  MsmVo msmVo = new MsmVo();  msmVo.setPhone(orderInfo.getPatientPhone());  msmVo.setTemplateCode("SMS\_194640721");  String reserveDate =  new DateTime(orderInfo.getReserveDate()).toString("yyyy-MM-dd")  + (orderInfo.getReserveTime()==0 ? "上午": "下午");  Map<String,Object> param = new HashMap<String,Object>(){{  put("title", orderInfo.getHosname()+"|"+orderInfo.getDepname()+"|"+orderInfo.getTitle());  put("amount", orderInfo.getAmount());  put("reserveDate", reserveDate);  put("name", orderInfo.getPatientName());  put("quitTime", new DateTime(orderInfo.getQuitTime()).toString("yyyy-MM-dd HH:mm"));  }};  msmVo.setParam(param);  orderMqVo.setMsmVo(msmVo);  rabbitService.sendMessage(MqConst.EXCHANGE\_DIRECT\_ORDER, MqConst.ROUTING\_ORDER, orderMqVo);  } **else** {  **throw new** YyghException(result.getString(**"message"**), ResultCodeEnum.***FAIL***.getCode());  }  **return** orderInfo.getId();  } |

# 订单管理

## 订单列表

### api接口

#### 1.1.1 添加service接口及实现类

1、在OrderInfoService类添加接口

|  |
| --- |
| */\*\**  *\* 分页列表*  *\*/*  IPage<OrderInfo> selectPage(Page<OrderInfo> pageParam, OrderQueryVo orderQueryVo); |

2、在OrderInfoServiceImpl类添加接口实现

|  |
| --- |
| *//订单列表（条件查询带分页）*  @Override  **public** IPage<OrderInfo> selectPage(Page<OrderInfo> pageParam, OrderQueryVo orderQueryVo) {  *//orderQueryVo获取条件值*  String name = orderQueryVo.getKeyword(); *//医院名称*  Long patientId = orderQueryVo.getPatientId(); *//就诊人名称*  String orderStatus = orderQueryVo.getOrderStatus(); *//订单状态*  String reserveDate = orderQueryVo.getReserveDate();*//安排时间*  String createTimeBegin = orderQueryVo.getCreateTimeBegin();  String createTimeEnd = orderQueryVo.getCreateTimeEnd();  *//对条件值进行非空判断*  QueryWrapper<OrderInfo> wrapper = **new** QueryWrapper<>();  **if**(!StringUtils.*isEmpty*(name)) {  wrapper.like(**"hosname"**,name);  }  **if**(!StringUtils.*isEmpty*(patientId)) {  wrapper.eq(**"patient\_id"**,patientId);  }  **if**(!StringUtils.*isEmpty*(orderStatus)) {  wrapper.eq(**"order\_status"**,orderStatus);  }  **if**(!StringUtils.*isEmpty*(reserveDate)) {  wrapper.ge(**"reserve\_date"**,reserveDate);  }  **if**(!StringUtils.*isEmpty*(createTimeBegin)) {  wrapper.ge(**"create\_time"**,createTimeBegin);  }  **if**(!StringUtils.*isEmpty*(createTimeEnd)) {  wrapper.le(**"create\_time"**,createTimeEnd);  }  *//调用mapper的方法*  IPage<OrderInfo> pages = **baseMapper**.selectPage(pageParam, wrapper);  *//编号变成对应值封装*  pages.getRecords().stream().forEach(item -> {  **this**.packOrderInfo(item);  });  **return** pages;  }  **private** OrderInfo packOrderInfo(OrderInfo orderInfo) {  orderInfo.getParam().put(**"orderStatusString"**, OrderStatusEnum.*getStatusNameByStatus*(orderInfo.getOrderStatus()));  **return** orderInfo;  } |

#### 1.1.2 添加controller

在OrderApiController类添加方法

|  |
| --- |
| *//订单列表（条件查询带分页）*  @GetMapping(**"auth/{page}/{limit}"**)  **public** Result list(@PathVariable Long page,  @PathVariable Long limit,  OrderQueryVo orderQueryVo, HttpServletRequest request) {  *//设置当前用户id*  orderQueryVo.setUserId(AuthContextHolder.*getUserId*(request));  Page<OrderInfo> pageParam = **new** Page<>(page,limit);  IPage<OrderInfo> pageModel =  **orderService**.selectPage(pageParam,orderQueryVo);  **return** Result.*ok*(pageModel);  }  @ApiOperation(value = **"获取订单状态"**)  @GetMapping(**"auth/getStatusList"**)  **public** Result getStatusList() {  **return** Result.*ok*(OrderStatusEnum.*getStatusList*());  } |

说明：订单状态我们是封装到枚举中的，页面搜索需要一个下拉列表展示，所以我们通过接口返回页面

### 前端

#### 1.2.1封装api请求

在/api/orderInfo.js添加方法

|  |
| --- |
| //订单列表  getPageList(page, limit, searchObj) {      return request({          url: `${api\_name}/auth/${page}/${limit}`,          method: `get`,          params: searchObj      })  },  //订单状态  getStatusList() {      return request({          url: `${api\_name}/auth/getStatusList`,          method: 'get'      })  }, |

#### 1.2.2 页面展示

创建/pages/order/index.vue组件

|  |
| --- |
| <template>    <!-- header -->    <div class="nav-container page-component">      <!--左侧导航 #start -->      <div class="nav left-nav">        <div class="nav-item ">          <span class="v-link clickable dark" onclick="javascript:window.location='/user'">实名认证 </span>        </div>        <div class="nav-item selected">          <span class="v-link selected dark" onclick="javascript:window.location='/order'"> 挂号订单 </span>        </div>        <div class="nav-item ">          <span class="v-link clickable dark" onclick="javascript:window.location='/patient'"> 就诊人管理 </span>        </div>        <div class="nav-item ">          <span class="v-link clickable dark"> 修改账号信息 </span>        </div>        <div class="nav-item ">          <span class="v-link clickable dark"> 意见反馈 </span>        </div>      </div>      <!-- 左侧导航 #end -->      <!-- 右侧内容 #start -->      <div class="page-container">        <div class="personal-order">          <div class="title"> 挂号订单</div>          <el-form :inline="true">            <el-form-item label="就诊人：">              <el-select v-model="searchObj.patientId" placeholder="请选择就诊人" class="v-select patient-select">                <el-option                  v-for="item in patientList"                  :key="item.id"                  :label="item.name + '【' + item.certificatesNo + '】'"                  :value="item.id">                </el-option>              </el-select>            </el-form-item>            <el-form-item label="订单状态：" style="margin-left: 80px">              <el-select v-model="searchObj.orderStatus" placeholder="全部" class="v-select patient-select" style="width: 200px;">                <el-option                  v-for="item in statusList"                  :key="item.status"                  :label="item.comment"                  :value="item.status">                </el-option>              </el-select>            </el-form-item>            <el-form-item>              <el-button type="text" class="search-button v-link highlight clickable selected" @click="fetchData()">                查询              </el-button>            </el-form-item>          </el-form>          <div class="table-wrapper table">            <el-table              :data="list"              stripe              style="width: 100%">              <el-table-column                label="就诊时间"                width="120">                <template slot-scope="scope">                  {{ scope.row.reserveDate }} {{ scope.row.reserveTime === 0 ? '上午' : '下午' }}                </template>              </el-table-column>              <el-table-column                prop="hosname"                label="医院"                width="100">              </el-table-column>              <el-table-column                prop="depname"                label="科室">              </el-table-column>              <el-table-column                prop="title"                label="医生">              </el-table-column>              <el-table-column                prop="amount"                label="医事服务费">              </el-table-column>              <el-table-column                prop="patientName"                label="就诊人">              </el-table-column>              <el-table-column                prop="param.orderStatusString"                label="订单状态">              </el-table-column>              <el-table-column label="操作">                <template slot-scope="scope">                  <el-button type="text" class="v-link highlight clickable selected" @click="show(scope.row.id)">详情</el-button>                </template>              </el-table-column>            </el-table>          </div>          <!-- 分页 -->          <el-pagination            class="pagination"            layout="prev, pager, next"            :current-page="page"            :total="total"            :page-size="limit"            @current-change="fetchData">          </el-pagination>        </div>      </div>      <!-- 右侧内容 #end -->    </div>    <!-- footer -->  </template>  <script>  import '~/assets/css/hospital\_personal.css'  import '~/assets/css/hospital.css'  import orderInfoApi from '@/api/order/orderInfo'  import patientApi from '@/api/user/patient'  export default {    data() {      return {        list: [], // banner列表        total: 0, // 数据库中的总记录数        page: 1, // 默认页码        limit: 10, // 每页记录数        searchObj: {}, // 查询表单对象        patientList: [],        statusList: []      }    },    created() {      this.orderId = this.$route.query.orderId      this.fetchData()      this.findPatientList()      this.getStatusList()    },    methods: {      fetchData(page = 1) {        this.page = page        orderInfoApi.getPageList(this.page, this.limit, this.searchObj).then(response => {          console.log(response.data);          this.list = response.data.records          this.total = response.data.total        })      },      findPatientList() {        patientApi.findList().then(response => {          this.patientList = response.data        })      },      getStatusList() {        orderInfoApi.getStatusList().then(response => {          this.statusList = response.data        })      },      changeSize(size) {        console.log(size)        this.limit = size        this.fetchData(1)      },      show(id) {        window.location.href = '/order/show?orderId=' + id      }    }  }  </script> |

## 订单详情

### 2.1 api接口

#### 2.1.1 添加service接口及实现类

1、在OrderInfoService类添加接口

|  |
| --- |
| */\*\**  *\* 获取订单详情*  *\*/*  OrderInfo getOrderInfo(Long id); |

2、在OrderInfoServiceImpl类添加接口实现

|  |
| --- |
| *//根据订单id查询订单详情*  @Override  **public** OrderInfo getOrder(String orderId) {  OrderInfo orderInfo = **baseMapper**.selectById(orderId);  **return this**.packOrderInfo(orderInfo);  } |

#### 2.1.2 添加controller

在OrderApiController类添加方法

|  |
| --- |
| *//根据订单id查询订单详情*  @GetMapping(**"auth/getOrders/{orderId}"**)  **public** Result getOrders(@PathVariable String orderId) {  OrderInfo orderInfo = **orderService**.getOrder(orderId);  **return** Result.*ok*(orderInfo);  } |

### 2.2前端

#### 2.2.1封装api请求

在/api/order/orderInfo.js添加方法

|  |
| --- |
| //订单详情      getOrders(orderId) {          return request({              url: `${api\_name}/auth/getOrders/${orderId}`,              method: `get`          })      }, |

#### 2.2.2 页面展示

创建/pages/order/show.vue组件

|  |
| --- |
| <template>    <!-- header -->    <div class="nav-container page-component">      <!--左侧导航 #start -->      <div class="nav left-nav">        <div class="nav-item ">          <span class="v-link clickable dark" onclick="javascript:window.location='/user'">实名认证 </span>        </div>        <div class="nav-item selected">          <span class="v-link selected dark" onclick="javascript:window.location='/order'"> 挂号订单 </span>        </div>        <div class="nav-item ">          <span class="v-link clickable dark" onclick="javascript:window.location='/patient'"> 就诊人管理 </span>        </div>        <div class="nav-item ">          <span class="v-link clickable dark"> 修改账号信息 </span>        </div>        <div class="nav-item ">          <span class="v-link clickable dark"> 意见反馈 </span>        </div>      </div>      <!-- 左侧导航 #end -->      <!-- 右侧内容 #start -->      <div class="page-container">        <div class="order-detail">          <div class="title"> 挂号详情</div>          <div class="status-bar">            <div class="left-wrapper">              <div class="status-wrapper BOOKING\_SUCCESS">                <span class="iconfont"></span> {{ orderInfo.param.orderStatusString }}              </div>            </div>            <div class="right-wrapper">              <img src="//img.114yygh.com/static/web/code\_order\_detail.png" class="code-img">              <div class="content-wrapper">                <div> 微信<span class="iconfont"></span>关注“北京114预约挂号”</div>                <div class="watch-wrapper"> 快速挂号，轻松就医</div>              </div>            </div>          </div>          <div class="info-wrapper">            <div class="title-wrapper">              <div class="block"></div>              <div>挂号信息</div>            </div>            <div class="info-form">              <el-form ref="form" :model="form">                <el-form-item label="就诊人信息：">                  <div class="content"><span>{{ orderInfo.patientName }}</span></div>                </el-form-item>                <el-form-item label="就诊日期：">                  <div class="content"><span>{{ orderInfo.reserveDate }} {{ orderInfo.reserveTime == 0 ? '上午' : '下午' }}</span></div>                </el-form-item>                <el-form-item label="就诊医院：">                  <div class="content"><span>{{ orderInfo.hosname }} </span></div>                </el-form-item>                <el-form-item label="就诊科室：">                  <div class="content"><span>{{ orderInfo.depname }} </span></div>                </el-form-item>                <el-form-item label="医生职称：">                  <div class="content"><span>{{ orderInfo.title }} </span></div>                </el-form-item>                <el-form-item label="医事服务费：">                  <div class="content">                    <div class="fee">{{ orderInfo.amount }}元                    </div>                  </div>                </el-form-item>                <el-form-item label="挂号单号：">                  <div class="content"><span>{{ orderInfo.outTradeNo }} </span></div>                </el-form-item>                <el-form-item label="挂号时间：">                  <div class="content"><span>{{ orderInfo.createTime }}</span></div>                </el-form-item>              </el-form>            </div>          </div>          <div class="rule-wrapper mt40">            <div class="rule-title"> 注意事项</div>            <div>1、请确认就诊人信息是否准确，若填写错误将无法取号就诊，损失由本人承担；<br>              <span style="color:red">2、【取号】就诊当天需在{{ orderInfo.fetchTime }}在医院取号，未取号视为爽约，该号不退不换；</span><br>              3、【退号】在{{ orderInfo.quitTime }}前可在线退号 ，逾期将不可办理退号退费；<br>              4、北京114预约挂号支持自费患者使用身份证预约，同时支持北京市医保患者使用北京社保卡在平台预约挂号。请于就诊当日，携带预约挂号所使用的有效身份证件到院取号；<br>              5、请注意北京市医保患者在住院期间不能使用社保卡在门诊取号。            </div>          </div>          <div class="bottom-wrapper mt60" v-if="orderInfo.orderStatus == 0 || orderInfo.orderStatus == 1">            <div class="button-wrapper">              <div class="v-button white" @click="cancelOrder()">取消预约</div>            </div>            <div class="button-wrapper ml20" v-if="orderInfo.orderStatus == 0">              <div class="v-button" @click="pay()">支付</div>            </div>          </div>        </div>      </div>      <!-- 右侧内容 #end -->      <!-- 微信支付弹出框 -->      <el-dialog :visible.sync="dialogPayVisible" style="text-align: left" :append-to-body="true" width="500px" @close="closeDialog">        <div class="container">          <div class="operate-view" style="height: 350px;">            <div class="wrapper wechat">              <div>                <img src="images/weixin.jpg" alt="">                  <div style="text-align: center;line-height: 25px;margin-bottom: 40px;">                  请使用微信扫一扫<br/>                  扫描二维码支付                </div>              </div>            </div>          </div>        </div>      </el-dialog>    </div>    <!-- footer -->  </template>  <script>  import '~/assets/css/hospital\_personal.css'  import '~/assets/css/hospital.css'  import orderInfoApi from '@/api/orderInfo'  export default {    data() {      return {        orderId: null,        orderInfo: {          param: {}        },        dialogPayVisible: false,        payObj: {},        timer: null  // 定时器名称      }    },    created() {      this.orderId = this.$route.query.orderId      this.init()    },    methods: {      init() {        orderInfoApi.getOrders(this.orderId).then(response => {          console.log(response.data);          this.orderInfo = response.data        })      }    }  }  </script>  <style>    .info-wrapper {      padding-left: 0;      padding-top: 0;    }    .content-wrapper {      color: #333;      font-size: 14px;      padding-bottom: 0;    }    .bottom-wrapper {      width: 100%;    }    .button-wrapper {      margin: 0;    }    .el-form-item {      margin-bottom: 5px;    }    .bottom-wrapper .button-wrapper {      margin-top: 0;    }  </style> |

# 平台订单管理

## 订单列表

### api接口

#### 添加controller方法

添加com.atguigu.yygh.order.controller.OrderController类

|  |
| --- |
| @Api(tags = **"订单接口"**)  @RestController  @RequestMapping(**"/admin/order/orderInfo"**)  **public class** OrderController {  @Autowired  **private** OrderService **orderService**;  @ApiOperation(value = **"获取分页列表"**)  @GetMapping(**"{page}/{limit}"**)  **public** Result index(  @ApiParam(name = **"page"**, value = **"当前页码"**, required = **true**)  @PathVariable Long page,  @ApiParam(name = **"limit"**, value = **"每页记录数"**, required = **true**)  @PathVariable Long limit,  @ApiParam(name = **"orderCountQueryVo"**, value = **"查询对象"**, required = **false**) OrderQueryVo orderQueryVo) {  Page<OrderInfo> pageParam = **new** Page<>(page, limit);  IPage<OrderInfo> pageModel = **orderService**.selectPage(pageParam, orderQueryVo);  **return** Result.*ok*(pageModel);  }  @ApiOperation(value = **"获取订单状态"**)  @GetMapping(**"getStatusList"**)  **public** Result getStatusList() {  **return** Result.*ok*(OrderStatusEnum.*getStatusList*());  }  } |

### 1.2前端

#### 1.2.1添加路由

在 src/router/index.js 文件添加路由

|  |
| --- |
| {    path: '/order',    component: Layout,    redirect: '/order/orderInfo/list',    name: 'BasesInfo',    meta: { title: '订单管理', icon: 'table' },    alwaysShow: true,    children: [        {          path: 'orderInfo/list',          name: '订单列表',          component: () =>import('@/views/order/orderInfo/list'),          meta: { title: '订单列表' }        }      ]  } |

#### 1.2.2封装api请求

添加/api/order/orderInfo.js文件

|  |
| --- |
| import request from '@/utils/request'  const api\_name = '/admin/order/orderInfo'  export default {    getPageList(page, limit, searchObj) {      return request({        url: `${api\_name}/${page}/${limit}`,        method: 'get',        params: searchObj      })    },    getStatusList() {      return request({        url: `${api\_name}/getStatusList`,        method: 'get'      })    }  } |

#### 1.2.3 添加组件

创建/views/order/orderInfo/list.vue组件

|  |
| --- |
| <template>    <div class="app-container">      <el-form :inline="true" class="demo-form-inline">        <el-form-item>          <el-input v-model="searchObj.hosname" placeholder="医院名称"/>        </el-form-item>        <el-form-item>          <el-input v-model="searchObj.outTradeNo" placeholder="订单号"/>        </el-form-item>        <el-form-item>          <el-input v-model="searchObj.patientName" placeholder="就诊人名称"/>        </el-form-item>        <el-form-item>          <el-date-picker            v-model="searchObj.createTimeBegin"            type="date"            placeholder="选择开始日期"            value-format="yyyy-MM-dd" />        </el-form-item>        <el-form-item>          <el-date-picker            v-model="searchObj.createTimeEnd"            type="date"            placeholder="选择截止日期"            value-format="yyyy-MM-dd" />        </el-form-item>        <el-form-item>          <el-date-picker            v-model="searchObj.reserveDate"            type="date"            placeholder="就诊日期"            value-format="yyyy-MM-dd" />        </el-form-item>        <el-form-item>          <el-select v-model="searchObj.orderStatus" placeholder="订单状态" class="v-select patient-select">            <el-option              v-for="item in statusList"              :key="item.status"              :label="item.comment"              :value="item.status">            </el-option>          </el-select>        </el-form-item>        <el-button type="primary" icon="el-icon-search" @click="fetchData()">查询</el-button>        <el-button type="default" @click="resetData()">清空</el-button>      </el-form>      <!-- 列表 -->      <el-table        v-loading="listLoading"        :data="list"        border        fit        highlight-current-row>        <el-table-column          label="序号"          width="60"          align="center">          <template slot-scope="scope">            {{ (page - 1) \* limit + scope.$index + 1 }}          </template>        </el-table-column>        <el-table-column prop="outTradeNo" label="订单交易号" width="160"/>        <el-table-column prop="hosname" label="医院名称" width="160"/>        <el-table-column prop="depname" label="科室名称" width="160"/>        <el-table-column prop="title" label="医生职称" />        <el-table-column label="安排时间" width="130">          <template slot-scope="scope">            {{ scope.row.reserveDate }} {{ scope.row.reserveTime === 0 ? '上午' : '下午' }}          </template>        </el-table-column>        <el-table-column prop="patientName" label="就诊人" />        <el-table-column prop="number" label="预约号序" width="80"/>        <el-table-column prop="amount" label="服务费" width="70"/>        <el-table-column prop="param.orderStatusString" label="订单状态" />        <el-table-column prop="createTime" label="创建时间" width="156"/>        <el-table-column label="操作" width="100" align="center">          <template slot-scope="scope">            <router-link :to="'/order/orderInfo/show/'+scope.row.id">              <el-button type="primary" size="mini" icon="el-icon-edit">查看</el-button>            </router-link>          </template>        </el-table-column>      </el-table>      <!-- 分页组件 -->      <el-pagination        :current-page="page"        :total="total"        :page-size="limit"        :page-sizes="[5, 10, 20, 30, 40, 50, 100]"        style="padding: 30px 0; text-align: center;"        layout="sizes, prev, pager, next, jumper, ->, total, slot"        @current-change="fetchData"        @size-change="changeSize"      />    </div>  </template>  <script>  import orderInfoApi from '@/api/order/orderInfo'  export default {    data() {      return {        listLoading: true, // 数据是否正在加载        list: null, // banner列表        total: 0, // 数据库中的总记录数        page: 1, // 默认页码        limit: 10, // 每页记录数        searchObj: {} // 查询表单对象      }    },    // 生命周期函数：内存准备完毕，页面尚未渲染    created() {      this.fetchData()      this.getStatusList()    },    methods: {      // 当页码发生改变的时候      changeSize(size) {        console.log(size)        this.limit = size        this.fetchData(1)      },      // 加载banner列表数据      fetchData(page = 1) {        console.log('翻页。。。' + page)        // 异步获取远程数据（ajax）        this.page = page        orderInfoApi.getPageList(this.page, this.limit, this.searchObj).then(          response => {            debugger            this.list = response.data.records            this.total = response.data.total            // 数据加载并绑定成功            this.listLoading = false          }        )      },      getStatusList() {        orderInfoApi.getStatusList().then(response => {          this.statusList = response.data        })      },      // 重置查询表单      resetData() {        console.log('重置查询表单')        this.searchObj = {}        this.fetchData()      }    }  }  </script> |

## 订单详情

### 2.1 api接口

#### 2.1.1 添加service接口及实现类

1、在OrderInfoService类添加接口

|  |
| --- |
| */\*\**  *\* 订单详情*  *\** ***@param orderId***  *\** ***@return***  *\*/*  Map<String,Object> show(Long orderId); |

2、在OrderInfoServiceImpl类添加接口实现

|  |
| --- |
| @Override  **public** Map<String, Object> show(Long orderId) {  Map<String, Object> map = **new** HashMap<>();  OrderInfo orderInfo = **this**.packOrderInfo(**this**.getById(orderId));  map.put(**"orderInfo"**, orderInfo);  Patient patient  = **patientFeignClient**.getPatient(orderInfo.getPatientId());  map.put(**"patient"**, patient);  **return** map;  } |

#### 2.1.2 添加controller

在OrderController类添加方法

|  |
| --- |
| @ApiOperation(value = **"获取订单"**)  @GetMapping(**"show/{id}"**)  **public** Result get(  @ApiParam(name = **"orderId"**, value = **"订单id"**, required = **true**)  @PathVariable Long id) {  **return** Result.*ok*(**orderService**.show(id));  } |

### 2.2前端

#### 2.2.1添加路由

在 src/router/index.js 文件添加路由

|  |
| --- |
| {    path: '/order',    component: Layout,    redirect: '/order/orderInfo/list',    name: 'BasesInfo',    meta: { title: '订单管理', icon: 'table' },    alwaysShow: true,    children: [        {          path: 'orderInfo/list',          name: '订单列表',          component: () =>import('@/views/order/orderInfo/list'),          meta: { title: '订单列表' }        },        {          path: 'orderInfo/show/:id',          name: '查看',                component: () =>import('@/views/order/orderInfo/show'),          meta: { title: '查看', noCache: true },          hidden: true        }      ]  } |

#### 2.2.2封装api请求

在/api/order/orderInfo.js文件添加方法

|  |
| --- |
| getById(id) {      return request({        url: `${api\_name}/show/${id}`,        method: 'get'      })    } |

#### 2.2.3 修改列表组件

修改/views/order/orderInfo/list.vue组件

|  |
| --- |
| <**el-table-column label="操作"width="100"align="center"**>  <**template slot-scope="scope"**>  <**router-link :to="**'/order/orderInfo/show/'+scope.row.**id"**>  <**el-button type="primary" size="mini" icon="el-icon-edit"**>查看</**el-button**>  </**router-link**>  </**template**>  </**el-table-column**> |

#### 2.2.4 添加组件

创建/views/order/orderInfo/show.vue组件

|  |
| --- |
| <template>    <div class="app-container">      <h4>订单信息</h4>      <table class="table table-striped table-condenseda table-bordered" width="100%">        <tbody>        <tr>          <th width="15%">订单交易号</th>          <td width="35%"><b style="font-size: 14px">{{ orderInfo.outTradeNo }}</b> </td>          <th width="15%">医院名称</th>          <td width="35%">{{ orderInfo.hosname }}</td>        </tr>        <tr>          <th>科室名称</th>          <td>{{ orderInfo.depname }}</td>          <th>医生职称</th>          <td>{{ orderInfo.title }}</td>        </tr>        <tr>          <th>安排日期</th>          <td>{{ orderInfo.reserveDate }} {{ orderInfo.reserveTime === 0 ? '上午' : '下午' }}</td>          <th>预约号序</th>          <td>{{ orderInfo.number }}</td>        </tr>        <tr>          <th>医事服务费</th>          <td>{{ orderInfo.amount }}</td>          <th>建议取号时间</th>          <td>{{ orderInfo.fetchTime }}</td>        </tr>        <tr>          <th>取号地点</th>          <td>{{ orderInfo.fetchAddress }}</td>          <th>退号时间</th>          <td>{{ orderInfo.quitTime }}</td>        </tr>        <tr>          <th>订单状态</th>          <td >{{ orderInfo.param.orderStatusString }}</td>          <th>预约时间</th>          <td>{{ orderInfo.createTime }}</td>        </tr>        </tbody>      </table>      <h4>就诊人信息</h4>      <table class="table table-striped table-condenseda table-bordered" width="100%">        <tbody>        <tr>          <th width="15%">姓名</th>          <td width="35%">{{ patient.name }}天</td>          <th width="15%">证件类型</th>          <td width="35%">{{ patient.param.certificatesTypeString }}</td>        </tr>        <tr>          <th>证件编号</th>          <td>{{ patient.certificatesNo }}</td>          <th>性别</th>          <td>{{ patient.sex === 1 ? '男' : '女' }}</td>        </tr>        <tr>          <th>出生年月</th>          <td>{{ patient.birthdate }}</td>          <th>手机</th>          <td>{{ patient.phone }}</td>        </tr>        <tr>          <th>是否结婚</th>          <td>{{ patient.isMarry === 1 ? '是' : '否' }}</td>          <th>地址</th>          <td>{{ patient.param.fullAddress }}</td>        </tr>        <tr>          <th>联系人姓名</th>          <td>{{ patient.contactsName }}</td>          <th>联系人证件类型</th>          <td>{{ patient.param.contactsCertificatesTypeString }}</td>        </tr>        <tr>          <th>联系人证件号</th>          <td>{{ orderInfo.contactsCertificatesNo }}</td>          <th>联系人手机</th>          <td>{{ orderInfo.contactsPhone }}</td>        </tr>        <br>        <el-row>          <el-button @click="back">返回</el-button>        </el-row>        </tbody>      </table>    </div>  </template>  <script>  // 引入组件  import orderInfoApi from '@/api/order/orderInfo'  export default {    data() {      return {        orderInfo: null,        patient: null      }    },    // 生命周期方法（在路由切换，组件不变的情况下不会被调用）    created() {      console.log('form created ......')      this.init()    },    methods: {      // 表单初始化      init() {        const id = this.$route.params.id        this.fetchDataById(id)      },      // 根据id查询记录      fetchDataById(id) {        orderInfoApi.getById(id).then(response => {          this.orderInfo = response.data.orderInfo          this.patient = response.data.patient        })      },      back() {        this.$router.push({ path: '/order/orderInfo/list' })      }    }  }  </script> |