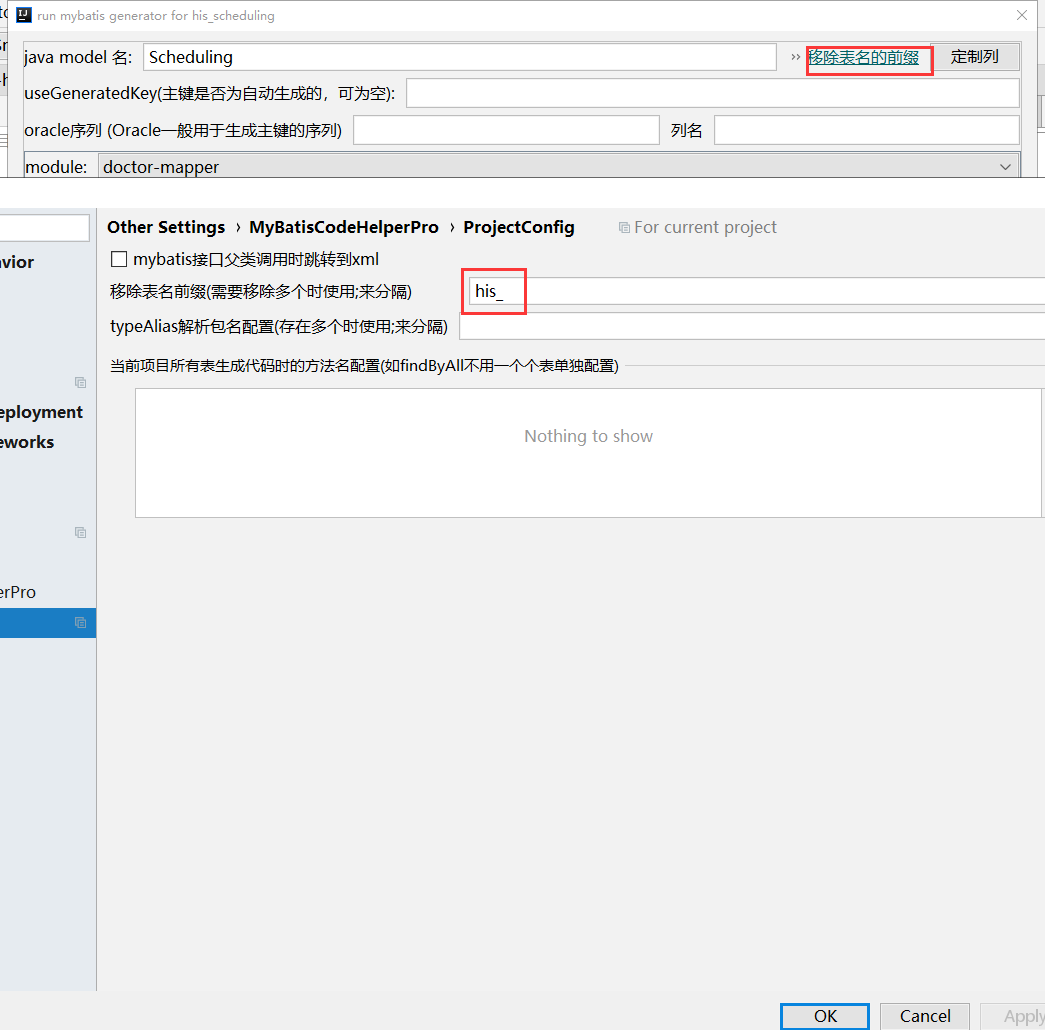
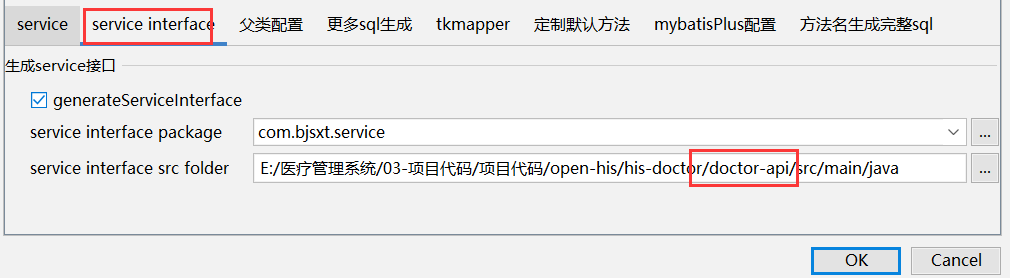
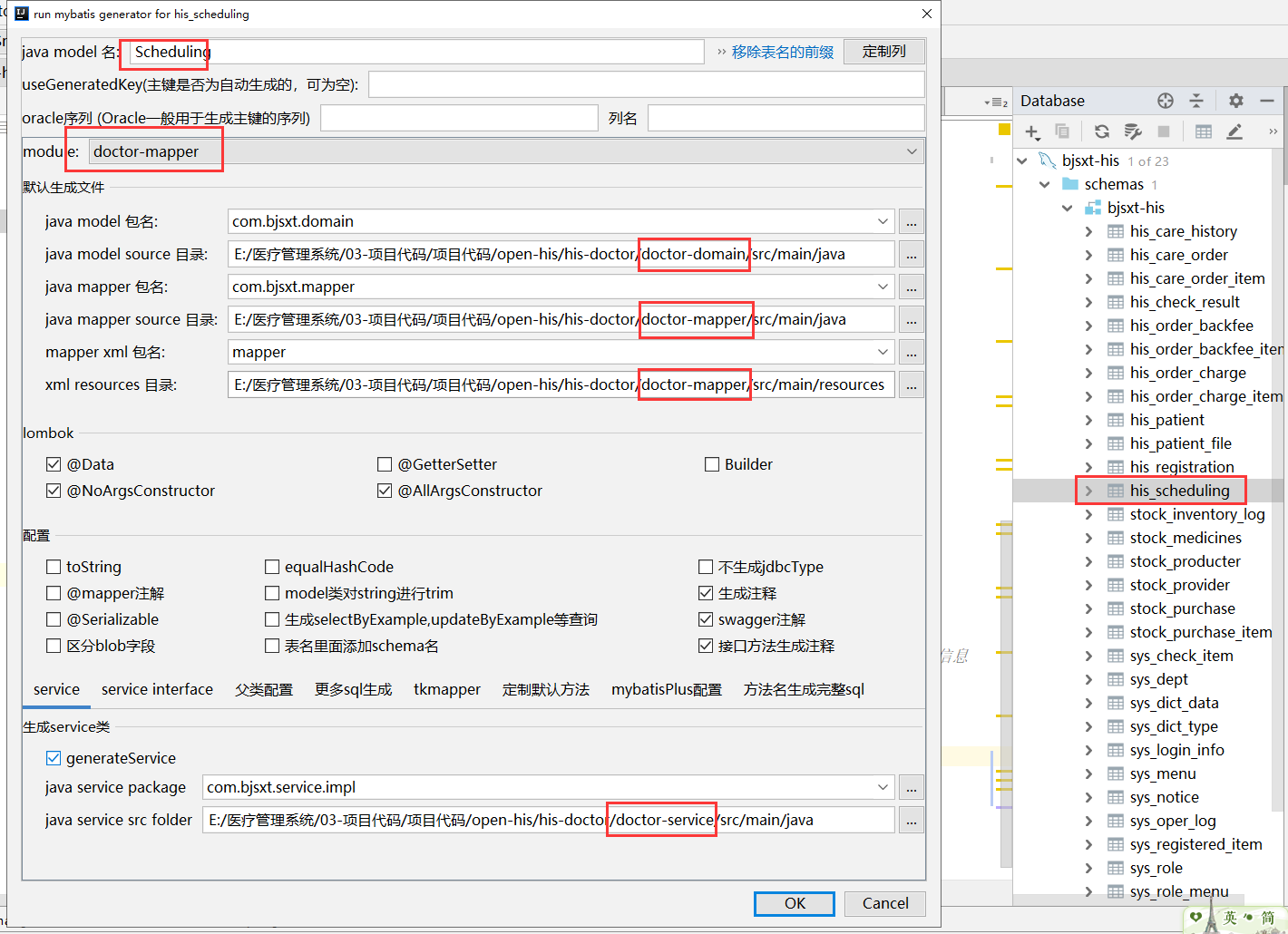
**Open-his看病就诊系统后端开发笔记**

# 医生排班及我的排班

## 生成排班表相关代码



### Scheduling【doctor-domain项目】

不动

### SchedulingMapper【doctor-mapper项目】

不动

### SchedulingMapper.xml【doctor-mapper项目】

不动

## 修改pom.xml解决system-web无法使用DTO的问题【system-web项目】

|  |
| --- |
| *<!--依赖doctor-api-->* <**dependency**>  <**groupId**>com.bjsxt</**groupId**>  <**artifactId**>doctor-api</**artifactId**>  <**version**>1.0</**version**> </**dependency**> |

## 创建SchedulingDto【doctor-domain项目】

|  |
| --- |
| **package** com.bjsxt.dto;  **import** com.fasterxml.jackson.annotation.JsonIgnore; **import** io.swagger.annotations.ApiModel; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.NoArgsConstructor;  **import** java.io.Serializable; **import** java.util.Collection; **import** java.util.Map;  */\*\*  \** ***@author:*** *尚学堂 雷哥  \* 封装数据使用  \*\*/* @ApiModel(value=**"com-bjsxt-dto-SchedulingDto"**) @Data @AllArgsConstructor @NoArgsConstructor **public class** SchedulingDto **implements** Serializable {  **private** Long **userId**;  **private** Long **deptId**;  **private** String **subsectionType**; *//上午 下午 晚上  //星期的值班值* **private** Collection<String> **schedulingType**;  *//星期值班的记录 key 是日期* @JsonIgnore  **private** Map<String,String> **record** ;    **public** SchedulingDto(Long userId, Long deptId, String subsectionType, Map<String,String> map) {  **this**.**userId** = userId;  **this**.**subsectionType** = subsectionType;  **this**.**record** = map;  **this**.**deptId**=deptId;  } } |

## 创建SchedulingFormDto【doctor-domain项目】

|  |
| --- |
| **package** com.bjsxt.dto;  **import** com.bjsxt.domain.SimpleUser; **import** io.swagger.annotations.ApiModel; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.NoArgsConstructor;  **import** java.io.Serializable; **import** java.util.Collection; **import** java.util.List;  */\*\*  \** ***@author:*** *尚学堂 雷哥  \* 表单数据提交对象  \*\*/* @ApiModel(value=**"com-bjsxt-dto-SchedulingFormDto"**) @Data @AllArgsConstructor @NoArgsConstructor **public class** SchedulingFormDto **implements** Serializable {   **private** SimpleUser **simpleUser**;   **private** String **beginDate**;   **private** List<SchedulingData> **data**;   @Data  **public static class** SchedulingData **implements** Serializable{  **private** Long **userId**;  **private** Long **deptId**;  **private** String **subsectionType**; *//上午 下午 晚上  //星期的值班值* **private** Collection<String> **schedulingType**;  } } |

## 创建SchedulingQueryDto【doctor-domain项目】

|  |
| --- |
| **package** com.bjsxt.dto;  **import** io.swagger.annotations.ApiModel; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.NoArgsConstructor;  **import** java.io.Serializable;  */\*\* \** ***@Author:*** *尚学堂 雷哥  \*排班信息表查询对象 \*/* @ApiModel(value=**"com-bjsxt-dto-SchedulingQueryDto"**) @Data @AllArgsConstructor @NoArgsConstructor **public class** SchedulingQueryDto **implements** Serializable {  *//用户ID* **private** Long **userId**;  *//部门ID* **private** Long **deptId**;  *//页面传过来的的上一周下一周* **private** String **queryDate**;   *//开始时间* **private** String **beginDate**;  *//结束时间* **private** String **endDate**; } |

## 编写SchedulingController【system-web项目】

|  |
| --- |
| 、**package** com.bjsxt.controller.doctor;  **import** cn.hutool.core.date.DateTime; **import** cn.hutool.core.date.DateUtil; **import** com.bjsxt.controller.BaseController; **import** com.bjsxt.domain.Scheduling; **import** com.bjsxt.domain.User; **import** com.bjsxt.dto.SchedulingDto; **import** com.bjsxt.dto.SchedulingFormDto; **import** com.bjsxt.dto.SchedulingQueryDto; **import** com.bjsxt.service.SchedulingService; **import** com.bjsxt.service.UserService; **import** com.bjsxt.utils.ShiroSecurityUtils; **import** com.bjsxt.vo.AjaxResult; **import** com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand; **import** org.apache.commons.lang3.StringUtils; **import** org.apache.dubbo.config.annotation.Reference; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.web.bind.annotation.\*;  **import** java.util.\*;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \*/* @RestController @RequestMapping(**"doctor/scheduling"**) **public class** SchedulingController **extends** BaseController {    @Reference  **private** SchedulingService **schedulingService**;   @Autowired  **private** UserService **userService**;   */\*\*  \* 查询要排班的医生信息  \* 如果部门ID为空，那么查询所有要排班的用户信息  \*/* @GetMapping(**"queryUsersNeedScheduling"**)  **public** AjaxResult queryUsersNeedScheduling(Long deptId) {  List<User> users = **this**.**userService**.queryUsersNeedScheduling(**null**, deptId);  **return** AjaxResult.*success*(users);  }   */\*\*  \* 查询要排班的医生的排班信息  \*/* @GetMapping(**"queryScheduling"**)  @HystrixCommand  **public** AjaxResult queryScheduling(SchedulingQueryDto schedulingQueryDto) {  *//根据部门ID用户ID查询用户信息 如果用户ID和部门ID都为空，那么就查询所有要排班的用户信息* List<User> users = **userService**.queryUsersNeedScheduling(schedulingQueryDto.getUserId(), schedulingQueryDto.getDeptId());  **return** getSchedulingAjaxResult(schedulingQueryDto, users);  }   */\*\*  \* 查询当前登陆用户的的排班信息  \*/* @GetMapping(**"queryMyScheduling"**)  @HystrixCommand  **public** AjaxResult queryMyScheduling(SchedulingQueryDto schedulingQueryDto) {  *//根据部门ID用户ID查询用户信息 如果用户ID和部门ID都为空，那么就查询所有要排班的用户信息* List<User> users = **userService**.queryUsersNeedScheduling(Long.*valueOf*(ShiroSecurityUtils.*getCurrentSimpleUser*().getUserId().toString()), schedulingQueryDto.getDeptId());  **return** getSchedulingAjaxResult(schedulingQueryDto, users);  }   */\*\*  \* 核心的构造页面数据的方法  \*  \** ***@param schedulingQueryDto*** *\** ***@param users*** *\** ***@return*** *\*/* **private** AjaxResult getSchedulingAjaxResult(SchedulingQueryDto schedulingQueryDto, List<User> users) {  *//得到当前时间* Date date = DateUtil.*date*();  **if** (StringUtils.*isNotBlank*(schedulingQueryDto.getQueryDate())) {  date = DateUtil.*parse*(schedulingQueryDto.getQueryDate(), **"yyyy-MM-dd"**);  *//根据传过来的日期判断是周几* **int** i = DateUtil.*dayOfWeek*(date);*//1表示周日，2表示周一* **if** (i == 1) {  date = DateUtil.*offsetDay*(date, 1);*//下一周的周一的日期* } **else** {  date = DateUtil.*offsetDay*(date, -1);*//上一周的周天的日期* }  }  *//计算一周的开始日期和结束日期* DateTime beginTime = DateUtil.*beginOfWeek*(date);  DateTime endTime = DateUtil.*endOfWeek*(date);  *//设置开始日期和结束日期到schedulingQueryDto* schedulingQueryDto.setBeginDate(DateUtil.*format*(beginTime, **"yyyy-MM-dd"**));  schedulingQueryDto.setEndDate(DateUtil.*format*(endTime, **"yyyy-MM-dd"**));  *//根据开始时间和结束时间去查询数据库里面存在的数据* List<Scheduling> list = **this**.**schedulingService**.queryScheduling(schedulingQueryDto);  *//存放页面需要的数据的对象* List<SchedulingDto> schedulingDtos = **new** ArrayList<>();  *//根据用户来循环* **for** (User user : users) {  SchedulingDto schedulingDto1 = **new** SchedulingDto(user.getUserId(), user.getDeptId(), **"1"**, initMap(beginTime));  SchedulingDto schedulingDto2 = **new** SchedulingDto(user.getUserId(), user.getDeptId(), **"2"**, initMap(beginTime));  SchedulingDto schedulingDto3 = **new** SchedulingDto(user.getUserId(), user.getDeptId(), **"3"**, initMap(beginTime));  *//一个用户的循环放三条数据* schedulingDtos.add(schedulingDto1);  schedulingDtos.add(schedulingDto2);  schedulingDtos.add(schedulingDto3);  **for** (Scheduling scheduling : list) {  Long userId = scheduling.getUserId();*//获取数据库的用户ID* String subsectionType = scheduling.getSubsectionType();*//得到早中晚的类型* String schedulingDay = scheduling.getSchedulingDay();*//值班日期* **if** (user.getUserId().equals(userId)) {  **switch** (subsectionType) {  **case "1"**:  Map<String, String> record1 = schedulingDto1.getRecord();  record1.put(schedulingDay, scheduling.getSchedulingType());  **break**;  **case "2"**:  Map<String, String> record2 = schedulingDto2.getRecord();  record2.put(schedulingDay, scheduling.getSchedulingType());  **break**;  **case "3"**:  Map<String, String> record3 = schedulingDto3.getRecord();  record3.put(schedulingDay, scheduling.getSchedulingType());  **break**;  }  }   }  *//把map转成数组数据放到schedulingType* schedulingDto1.setSchedulingType(schedulingDto1.getRecord().values());  schedulingDto2.setSchedulingType(schedulingDto2.getRecord().values());  schedulingDto3.setSchedulingType(schedulingDto3.getRecord().values());  }  *//组装返回的对象* Map<String, Object> resMap = **new** HashMap<>();  resMap.put(**"tableData"**, schedulingDtos);   Map<String, Object> schedulingData = **new** HashMap<>();  schedulingData.put(**"startTimeThisWeek"**, schedulingQueryDto.getBeginDate());  schedulingData.put(**"endTimeThisWeek"**, schedulingQueryDto.getEndDate());  resMap.put(**"schedulingData"**, schedulingData);  resMap.put(**"labelNames"**, initLabel(beginTime));  **return** AjaxResult.*success*(resMap);  }   */\*\*  \* 初始化labelNames  \*  \** ***@param beginTime*** *\** ***@return*** *\*/* **private** String[] initLabel(DateTime beginTime) {  String[] labelNames = **new** String[7];  **for** (**int** i = 0; i < 7; i++) {  DateTime d = DateUtil.*offsetDay*(beginTime, i);  labelNames[i] = DateUtil.*format*(d, **"yyyy-MM-dd"**) + formatterWeek(i);  }  **return** labelNames;  }   */\*\*  \* 翻译周  \*  \** ***@param i*** *\** ***@return*** *\*/* **private** String formatterWeek(**int** i) {  **switch** (i) {  **case** 0:  **return "(周一)"**;  **case** 1:  **return "(周二)"**;  **case** 2:  **return "(周三)"**;  **case** 3:  **return "(周四)"**;  **case** 4:  **return "(周五)"**;  **case** 5:  **return "(周六)"**;  **default**:  **return "(周日)"**;  }  }   */\*\*  \* 初始化值班记录  \*  \** ***@param beginTime*** *当周的第一天  \** ***@return*** *\*/* **private** Map<String, String> initMap(DateTime beginTime) {  Map<String, String> map = **new** TreeMap<>();*//按顺序放 2020-08-03 - 2020-08-09* **for** (**int** i = 0; i < 7; i++) {  DateTime d = DateUtil.*offsetDay*(beginTime, i);  String key = DateUtil.*format*(d, **"yyyy-MM-dd"**);  map.put(key, **""**);  }  **return** map;  }    */\*\*  \* 保存排班数据  \*/* @PostMapping(**"saveScheduling"**) *// @HystrixCommand* **public** AjaxResult saveScheduling(@RequestBody SchedulingFormDto schedulingFormDto){  schedulingFormDto.setSimpleUser(ShiroSecurityUtils.*getCurrentSimpleUser*());  **return** AjaxResult.*toAjax*(**this**.**schedulingService**.saveScheduling(schedulingFormDto));  }    } |

## 修改UserService【system-api项目】

|  |
| --- |
| */\*\*  \* 根据部门ID和用户ID查询用户信息，如果用户ID和部门ID为空，那么就查询所有需要班的医生信息  \** ***@param userId*** *\** ***@param deptId*** *\** ***@return*** *\*/* List<User> querySchedulingUsers(Long userId, Long deptId); |

## 修改UserServiceImpl【system-api项目】

|  |
| --- |
| @Override **public** List<User> querySchedulingUsers(Long userId, Long deptId) {  QueryWrapper<User> qw=**new** QueryWrapper<>();  qw.eq(**null**!=deptId, User.***COL\_DEPT\_ID***,deptId);  qw.eq(**null**!=userId, User.***COL\_USER\_ID***,userId);  qw.eq(User.***COL\_SCHEDULING\_FLAG***, Constants.***SCHEDULING\_FLAG\_TRUE***);  **return this**.**userMapper**.selectList(qw); } |

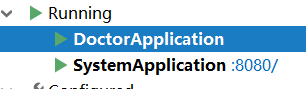
## 创建SchedulingService【doctor-api项目】

|  |
| --- |
| **package** com.bjsxt.service;  **import** com.bjsxt.domain.Scheduling; **import** com.bjsxt.dto.SchedulingFormDto; **import** com.bjsxt.dto.SchedulingQueryDto;  **import** java.util.List;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \*/* **public interface** SchedulingService {    */\*\*  \* 查询排班的数据  \** ***@param schedulingDto*** *\** ***@return*** *\*/* List<Scheduling> queryScheduling(SchedulingQueryDto schedulingDto);   */\*\*  \* 保存排班的数据  \** ***@param schedulingFormDto*** *\** ***@return*** *\*/* **int** saveScheduling(SchedulingFormDto schedulingFormDto); } |

## 创建SchedulingServiceImpl【doctor-service项目】

|  |
| --- |
| **package** com.bjsxt.service.impl;  **import** cn.hutool.core.date.DateTime; **import** cn.hutool.core.date.DateUtil; **import** com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; **import** com.bjsxt.domain.Scheduling; **import** com.bjsxt.dto.SchedulingFormDto; **import** com.bjsxt.dto.SchedulingQueryDto; **import** com.bjsxt.mapper.SchedulingMapper; **import** com.bjsxt.service.SchedulingService; **import** org.apache.commons.lang3.StringUtils; **import** org.apache.dubbo.config.annotation.Service; **import** org.springframework.beans.factory.annotation.Autowired;  **import** java.util.ArrayList; **import** java.util.Date; **import** java.util.List;  */\*\* \** ***@Author:*** *尚学堂 雷哥 \*/* @Service*//注意是dubbo的service注解* **public class** SchedulingServiceImpl **implements** SchedulingService{   @Autowired  **private** SchedulingMapper **schedulingMapper**;   @Override  **public** List<Scheduling> queryScheduling(SchedulingQueryDto schedulingDto) {  QueryWrapper<Scheduling> qw=**new** QueryWrapper<>();  qw.eq(**null**!=schedulingDto.getUserId(),Scheduling.***COL\_USER\_ID***,schedulingDto.getUserId());  qw.eq(**null**!=schedulingDto.getDeptId(),Scheduling.***COL\_DEPT\_ID***,schedulingDto.getDeptId());  qw.ge(StringUtils.*isNotBlank*(schedulingDto.getBeginDate()),Scheduling.***COL\_SCHEDULING\_DAY***,schedulingDto.getBeginDate());  qw.le(StringUtils.*isNotBlank*(schedulingDto.getEndDate()),Scheduling.***COL\_SCHEDULING\_DAY***,schedulingDto.getEndDate());  **return this**.**schedulingMapper**.selectList(qw);  }   @Override  **public int** saveScheduling(SchedulingFormDto schedulingFormDto) {  **if**(**null**!=schedulingFormDto.getData()&&schedulingFormDto.getData().size()>0){  DateTime dateTime = DateUtil.*parse*(schedulingFormDto.getBeginDate(), **"yyyy-MM-dd"**);  DateTime date = DateUtil.*beginOfWeek*(dateTime);  String beginDate = DateUtil.*format*(date,**"yyyy-MM-dd"**);  String endDate = DateUtil.*format*(DateUtil.*endOfWeek*(dateTime),**"yyyy-MM-dd"**);  SchedulingFormDto.SchedulingData schedulingData = schedulingFormDto.getData().get(0);  Long userId=schedulingData.getUserId();  Long deptId=schedulingData.getDeptId();  **if**(**null**!=userId){  *//删除原来的排班* QueryWrapper<Scheduling> qw=**new** QueryWrapper<>();  qw.eq(Scheduling.***COL\_USER\_ID***,userId);  qw.eq(Scheduling.***COL\_DEPT\_ID***,deptId);  qw.between(Scheduling.***COL\_SCHEDULING\_DAY***,beginDate,endDate);  **this**.**schedulingMapper**.delete(qw);  *//添加新的排班* List<String> schedulingDays = *initSchedulingDay*(date);  **for** (SchedulingFormDto.SchedulingData d : schedulingFormDto.getData()) {  Scheduling scheduling;  **int** i=0;  **for** (String s : d.getSchedulingType()) {  **if**(StringUtils.*isNotBlank*(s)){  scheduling=**new** Scheduling(userId,deptId,schedulingDays.get(i),d.getSubsectionType(),s,**new** Date(),schedulingFormDto.getSimpleUser().getUserName());  **this**.**schedulingMapper**.insert(scheduling);  }  i++;  }  }  **return** 1;*//受影响的行数* }**else**{  **return** 0;  }  }**else**{  **return** 0;  }  }  */\*\*  \* 初始化值班记录  \** ***@param startDate*** *当周第一天  \** ***@return*** *\*/* **private static** List<String> initSchedulingDay(Date startDate){  List<String> list = **new** ArrayList<>();  **for** (**int** i=0;i<7;i++){  DateTime d = DateUtil.*offsetDay*(startDate,i);  String key = DateUtil.*format*(d, **"yyyy-MM-dd"**);  list.add(key);  }  **return** list;  } } |

## 启动DoctorApplication和SystemApplication

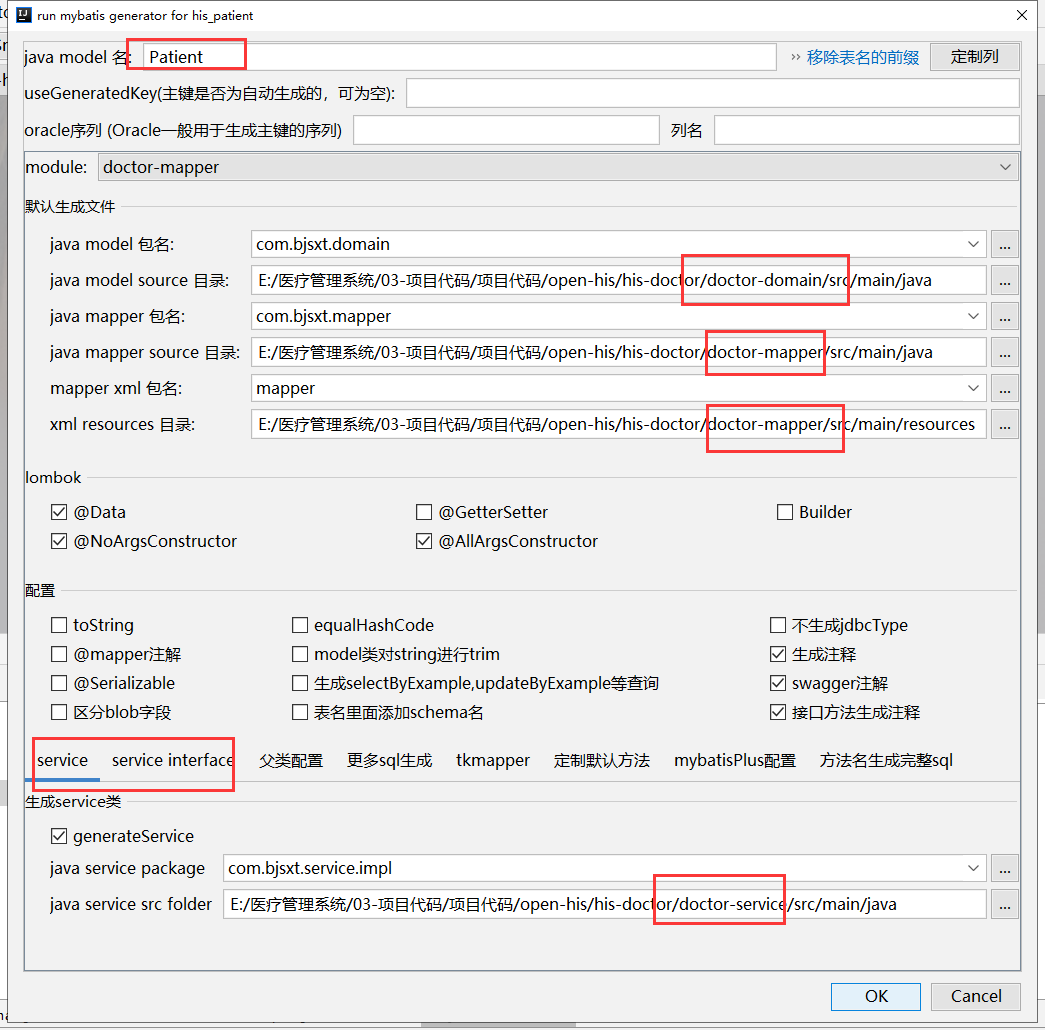


## yapi测试



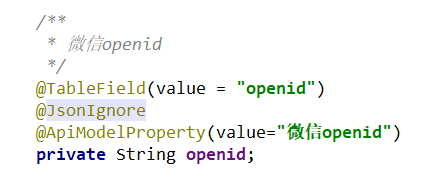
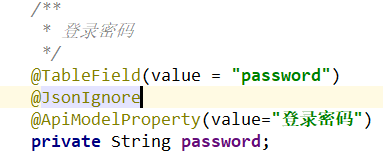
# 患者库管理

## 生成患者表相关代码



### 生成Patient【doctor-domain项目】

不动 使用JsonIgnore 忽略到相关信息



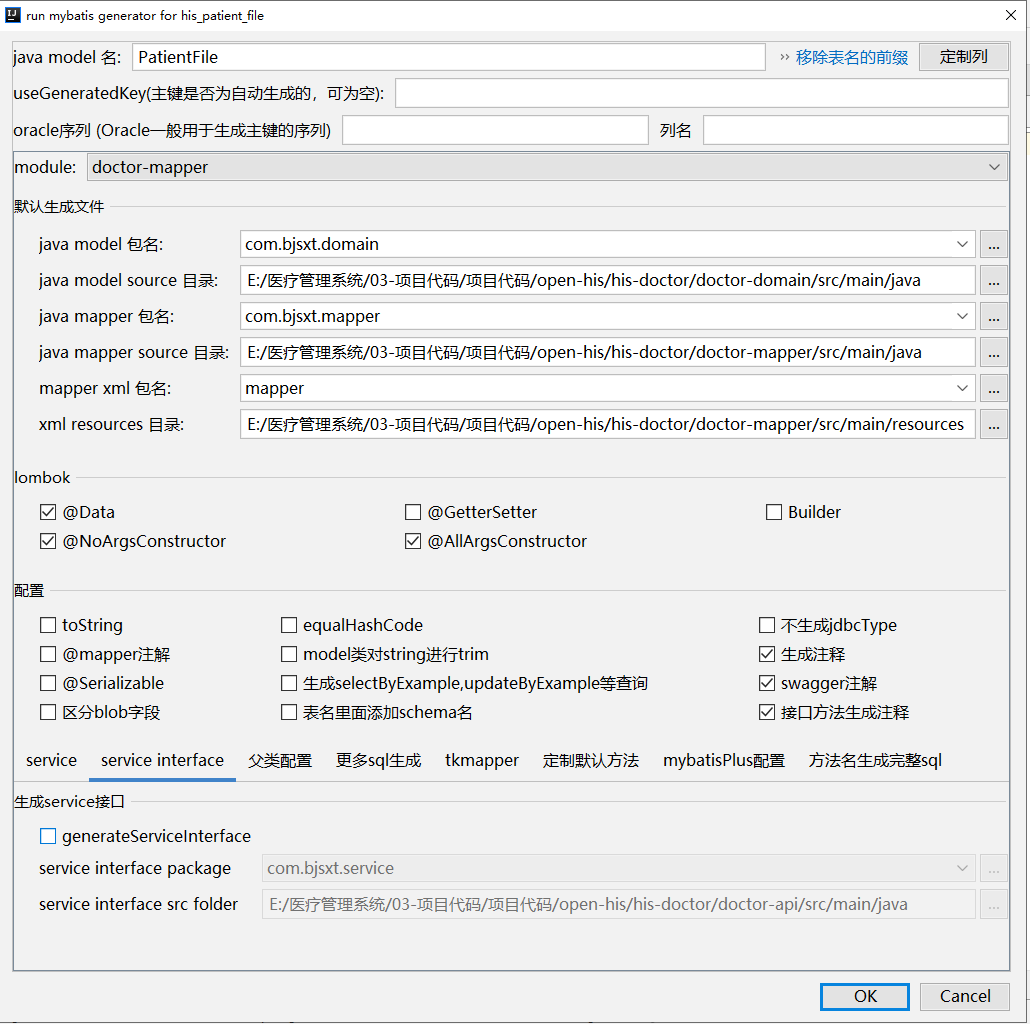
### 生成PatientMapper【doctor-mapper项目】

不动

### 生成PatientMapper.xml【doctor-mapper项目】

不动

## 生成患者档案表相关代码



### 生成PatientFile【doctor-domain项目】

不动

### 生成PatientFileMapper【doctor-mapper项目】

不动

### 生成PatientFileMapper.xml【doctor-mapper项目】

不动

## 创建PatientDto【doctor-domain项目】

|  |
| --- |
| **package** com.bjsxt.dto;  **import** io.swagger.annotations.ApiModel; **import** io.swagger.annotations.ApiModelProperty; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.EqualsAndHashCode; **import** lombok.NoArgsConstructor;  **import** javax.validation.constraints.NotBlank;  */\*\* \** ***@Author:*** *尚学堂 雷哥  \* 患者信息表 \*/* @ApiModel(value=**"com-bjsxt-dto-PatientDto"**) @Data @EqualsAndHashCode(callSuper=**true**) @AllArgsConstructor @NoArgsConstructor **public class** PatientDto **extends** BaseDto {  */\*\*  \* 主键  \*/* @ApiModelProperty(value=**"主键"**)  **private** String **patientId**;   */\*\*  \* 患者姓名  \*/* @NotBlank(message = **"患者姓名不能为空"**)  @ApiModelProperty(value=**"患者姓名"**)  **private** String **name**;   */\*\*  \* 患者电话  \*/* @NotBlank(message = **"患者电话不能为空"**)  @ApiModelProperty(value=**"患者电话"**)  **private** String **phone**;   */\*\*  \* 患者性别0男1女 2未知字典表 sys\_user\_sex  \*/* @NotBlank(message = **"患者性别不能为空"**)  @ApiModelProperty(value=**"患者性别0男1女 2未知字典表 sys\_user\_sex"**)  **private** String **sex**;   */\*\*  \* 出生年月  \*/* @NotBlank(message = **"患者出生年月不能为空"**)  @ApiModelProperty(value=**"出生年月"**)  **private** String **birthday**;   */\*\*  \* 身份证号[认证ID]  \*/* @NotBlank(message = **"身份证号不能为空"**)  @ApiModelProperty(value=**"身份证号[认证ID]"**)  **private** String **idCard**;   */\*\*  \* 地址信息  \*/* @ApiModelProperty(value=**"地址信息"**)  **private** String **address**;   */\*\*  \* 过敏信息  \*/* @ApiModelProperty(value=**"过敏信息"**)  **private** String **allergyInfo**;  } |

## 创建PatientService【doctor-api项目】

|  |
| --- |
| **package** com.bjsxt.service;  **import** com.bjsxt.domain.Patient; **import** com.bjsxt.domain.PatientFile; **import** com.bjsxt.dto.PatientDto; **import** com.bjsxt.vo.DataGridView;  */\*\* \** ***@Author:*** *尚学堂 雷哥 \*/* **public interface** PatientService {   */\*\*  \* 分页查询  \** ***@param patientDto*** *\** ***@return*** *\*/* DataGridView listPatientForPage(PatientDto patientDto);   */\*\*  \* 根据患者ID查询患者信息  \** ***@param patientId*** *\** ***@return*** *\*/* Patient getPatientById(String patientId);   */\*\*  \* 根据患者ID查询患者档案  \** ***@param patientId*** *\** ***@return*** *\*/* PatientFile getPatientFileById(String patientId); } |

## 创建PatientServiceImpl【doctor-service项目】

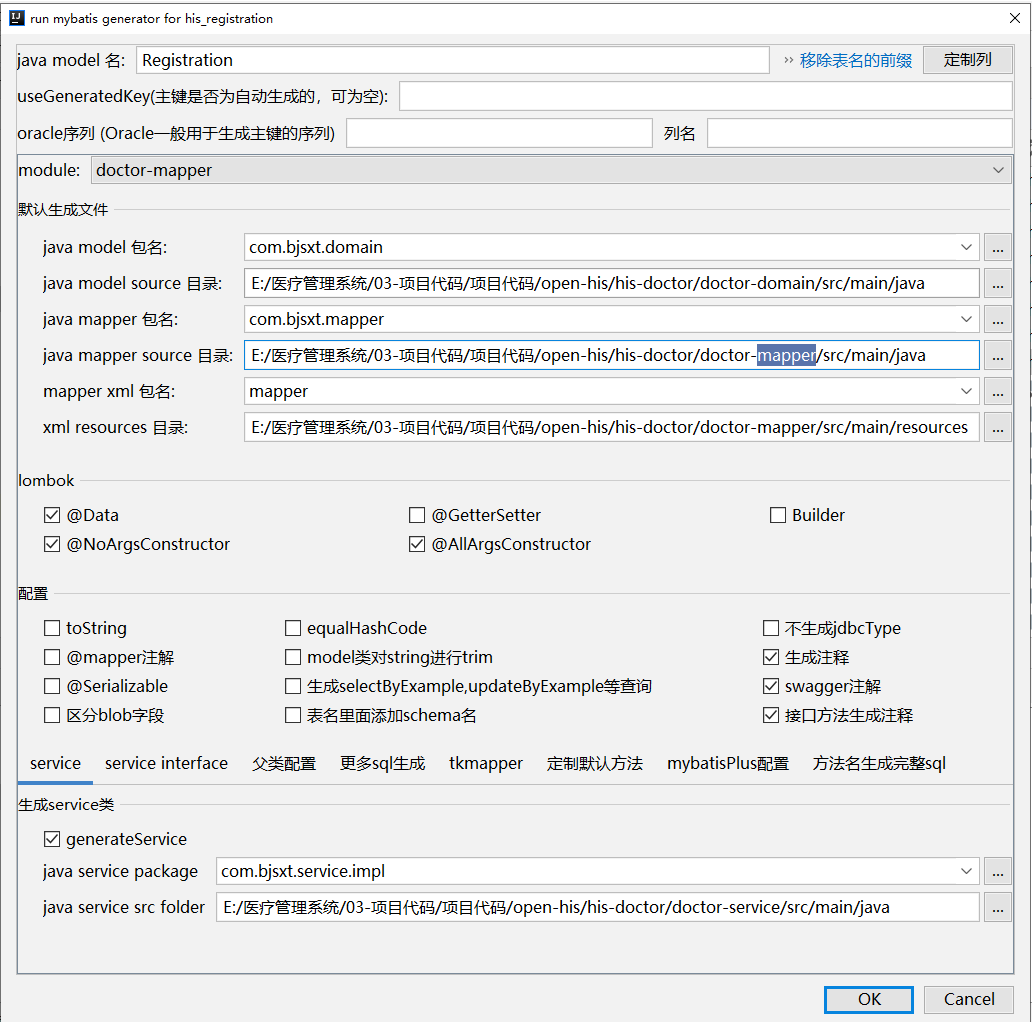
|  |
| --- |
| **package** com.bjsxt.service.impl;  **import** com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; **import** com.baomidou.mybatisplus.extension.plugins.pagination.Page; **import** com.bjsxt.domain.Patient; **import** com.bjsxt.domain.PatientFile; **import** com.bjsxt.dto.PatientDto; **import** com.bjsxt.mapper.PatientFileMapper; **import** com.bjsxt.mapper.PatientMapper; **import** com.bjsxt.service.PatientService; **import** com.bjsxt.vo.DataGridView; **import** org.apache.commons.lang3.StringUtils; **import** org.apache.dubbo.config.annotation.Service; **import** org.springframework.beans.factory.annotation.Autowired;  */\*\* \** ***@Author:*** *尚学堂 雷哥 \*/* @Service **public class** PatientServiceImpl **implements** PatientService{   @Autowired  **private** PatientMapper **patientMapper**;   @Autowired  **private** PatientFileMapper **patientFileMapper**;   @Override  **public** DataGridView listPatientForPage(PatientDto patientDto) {  Page<Patient> page=**new** Page<>(patientDto.getPageNum(),patientDto.getPageSize());  QueryWrapper<Patient> qw=**new** QueryWrapper<>();  qw.like(StringUtils.*isNotBlank*(patientDto.getName()),Patient.***COL\_NAME***,patientDto.getName());  qw.like(StringUtils.*isNotBlank*(patientDto.getIdCard()),Patient.***COL\_ID\_CARD***,patientDto.getIdCard());  qw.like(StringUtils.*isNotBlank*(patientDto.getPhone()),Patient.***COL\_PHONE***,patientDto.getPhone());  **this**.**patientMapper**.selectPage(page,qw);  **return new** DataGridView(page.getTotal(),page.getRecords());  }   @Override  **public** Patient getPatientById(String patientId) {  **return this**.**patientMapper**.selectById(patientId);  }   @Override  **public** PatientFile getPatientFileById(String patientId) {  **if** (**null**==patientId){  **return null**;  }  QueryWrapper<PatientFile> qw=**new** QueryWrapper<>();  qw.eq(PatientFile.***COL\_PATIENT\_ID***,patientId);  **return this**.**patientFileMapper**.selectOne(qw);  } } |

## 创建PatientController【system-web】

|  |
| --- |
| **package** com.bjsxt.controller.doctor;  **import** com.bjsxt.controller.BaseController; **import** com.bjsxt.domain.Patient; **import** com.bjsxt.domain.PatientFile; **import** com.bjsxt.dto.PatientDto; **import** com.bjsxt.service.PatientService; **import** com.bjsxt.vo.AjaxResult; **import** com.bjsxt.vo.DataGridView; **import** com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand; **import** org.apache.dubbo.config.annotation.Reference; **import** org.springframework.web.bind.annotation.GetMapping; **import** org.springframework.web.bind.annotation.PathVariable; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController;  */\*\*  \*  \** ***@Auther:*** *尚学堂 雷哥  \** ***@Description:*** *患者库控制器  \*/* @RestController @RequestMapping(**"doctor/patient"**) **public class** PatientController **extends** BaseController {   @Reference  **private** PatientService **patientService**;   */\*\*  \* 分页查询  \*/* @GetMapping(**"listPatientForPage"**)  @HystrixCommand  **public** AjaxResult listPatientForPage(PatientDto patientDto){  DataGridView gridView = **this**.**patientService**.listPatientForPage(patientDto);  **return** AjaxResult.*success*(**"查询成功"**,gridView.getData(),gridView.getTotal());  }   */\*\*  \* 查询一个  \*/* @GetMapping(**"getPatientById/{patientId}"**)  @HystrixCommand  **public** AjaxResult getPatientById(@PathVariable String patientId){  Patient patient=**this**.**patientService**.getPatientById(patientId);  **return** AjaxResult.*success*(patient);  }   */\*\*  \* 根据ID查询患者档案  \*/* @GetMapping(**"getPatientFileById/{patientId}"**)  @HystrixCommand  **public** AjaxResult getPatientFileById(@PathVariable String patientId){  PatientFile patientFile=**this**.**patientService**.getPatientFileById(patientId);  **return** AjaxResult.*success*(patientFile);  }  */\*\*  \* 根据患者ID查询患者信息 患者档案信息 历史病例  \*/* @GetMapping(**"getPatientAllMessageByPatientId/{patientId}"**)  **public** AjaxResult getPatientAllMessageByPatientId(@PathVariable String patientId){  **return** AjaxResult.*success*();  } } |

# 门诊挂号

## 生成挂号表相关代码



### 生成Registration【doctor-domain项目】

不动

### 生成RegistrationMapper【doctor-mapper项目】

不动

### 生成RegistrationMapper.xml【doctor-mapper项目】

不动

## 挂号相关修改

### 创建RegistrationDto

|  |
| --- |
| **package** com.bjsxt.dto;  **import** io.swagger.annotations.ApiModel; **import** io.swagger.annotations.ApiModelProperty; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.EqualsAndHashCode; **import** lombok.NoArgsConstructor;  **import** javax.validation.constraints.NotBlank; **import** javax.validation.constraints.NotNull; **import** java.math.BigDecimal;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \*/* @ApiModel(value = **"com-bjsxt-dto-RegistrationDto"**) @Data @EqualsAndHashCode(callSuper = **true**) @AllArgsConstructor @NoArgsConstructor **public class** RegistrationDto **extends** BaseDto {  */\*\*  \* 挂号流水  \*/* @ApiModelProperty(value = **"挂号流水"**)  **private** String **regId**;   */\*\*  \* 患者ID  \*/* @ApiModelProperty(value = **"患者ID"**)  **private** String **patientId**;   */\*\*  \* 患者姓名  \*/* @ApiModelProperty(value = **"患者姓名"**)  **private** String **patientName**;   */\*\*  \* 接诊医生ID  \*/* @ApiModelProperty(value = **"接诊医生ID"**)  **private** Long **userId**;   */\*\*  \* 接诊医生姓名  \*/* @ApiModelProperty(value = **"接诊医生姓名"**)  **private** String **doctorName**;   */\*\*  \* 科室ID  \*/* @ApiModelProperty(value = **"科室ID"**)  **private** Long **deptId**;   */\*\*  \* 挂号费用ID  \*/* @NotBlank(message = **"挂号费用ID不能为空"**)  @ApiModelProperty(value = **"挂号费用ID"**)  **private** Long **regItemId**;   */\*\*  \* 挂号总金额  \*/* @NotNull(message = **"挂号总金额不能为空"**)  @ApiModelProperty(value = **"挂号总金额"**)  **private** BigDecimal **regItemAmount**;   */\*\*  \* 挂号编号  \*/* @NotNull(message = **"挂号编号不能为空"**)  @ApiModelProperty(value = **"挂号编号"**)  **private** Integer **regNumber**;   */\*\*  \* 就诊日期  \*/* @NotBlank(message = **"就诊日期不能为空"**)  @ApiModelProperty(value = **"就诊日期"**)  **private** String **visitDate**;   */\*\*  \* 排班类型1 门诊 2 急诊 字典表数据翻译  \*/* @NotBlank(message = **"排班类型不能为空"**)  @ApiModelProperty(value = **"排班类型1 门诊 2 急诊 字典表数据翻译"**)  **private** String **schedulingType**;   */\*\*  \* 排班时段1上午 2下午 3晚上 字典表数据翻译  \*/* @NotBlank(message = **"排班时段不能为空"**)  @ApiModelProperty(value = **"排班时段1上午 2下午 3晚上 字典表数据翻译"**)  **private** String **subsectionType**;   */\*\*  \* 挂号状态0未收费,1待就诊，2,就诊中，3，就诊完成，4，已退号，5 作废  \*/* @ApiModelProperty(value = **"挂号状态0未收费,1待就诊，2,就诊中，3，就诊完成，4，已退号，5 作废"**)  **private** String **regStatus**; } |

### 创建RegistrationFormDto

|  |
| --- |
| **package** com.bjsxt.dto;  **import** io.swagger.annotations.ApiModel; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.EqualsAndHashCode; **import** lombok.NoArgsConstructor;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \*/* @ApiModel(value = **"com-bjsxt-dto-RegistrationFormDto"**) @Data @EqualsAndHashCode(callSuper = **true**) @AllArgsConstructor @NoArgsConstructor **public class** RegistrationFormDto **extends** BaseDto {  **private** PatientDto **patientDto**;  **private** RegistrationDto **registrationDto**; } |

### 创建RegistrationQueryDto

|  |
| --- |
| **package** com.bjsxt.dto;  **import** io.swagger.annotations.ApiModel; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.EqualsAndHashCode; **import** lombok.NoArgsConstructor;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \*/* @ApiModel(value = **"com-bjsxt-dto-RegistrationQueryDto"**) @Data @EqualsAndHashCode(callSuper = **true**) @AllArgsConstructor @NoArgsConstructor **public class** RegistrationQueryDto **extends** BaseDto {  **private** Long **deptId**;  **private** String **schedulingType**;*//挂号类型* **private** String **subsectionType**;*//挂号时段* **private** String **schedulingDay**;*//查询时间* } |

### 创建RegistrationController

|  |
| --- |
| **package** com.bjsxt.controller.doctor;  **import** cn.hutool.core.date.DateUtil; **import** com.bjsxt.constants.Constants; **import** com.bjsxt.domain.Dept; **import** com.bjsxt.domain.Patient; **import** com.bjsxt.domain.Registration; **import** com.bjsxt.dto.PatientDto; **import** com.bjsxt.dto.RegistrationDto; **import** com.bjsxt.dto.RegistrationFormDto; **import** com.bjsxt.dto.RegistrationQueryDto; **import** com.bjsxt.service.DeptService; **import** com.bjsxt.service.PatientService; **import** com.bjsxt.service.RegistrationService; **import** com.bjsxt.service.SchedulingService; **import** com.bjsxt.utils.IdGeneratorSnowflake; **import** com.bjsxt.utils.ShiroSecurityUtils; **import** com.bjsxt.vo.AjaxResult; **import** com.bjsxt.vo.DataGridView; **import** lombok.extern.log4j.Log4j2; **import** org.apache.commons.lang3.StringUtils; **import** org.apache.dubbo.config.annotation.Reference; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.validation.annotation.Validated; **import** org.springframework.web.bind.annotation.\*;  **import** java.util.Collections; **import** java.util.List;  */\*\*  \*  \** ***@Auther:*** *尚学堂 雷哥  \** ***@Description:*** *挂号相关的控制器  \*/* @Log4j2 @RestController @RequestMapping(**"doctor/registration"**) **public class** RegistrationController {   @Reference  **private** RegistrationService **registrationService**;   @Reference  **private** SchedulingService **schedulingService**;   @Reference  **private** PatientService **patientService**;   @Autowired  **private** DeptService **deptService**;   */\*\*  \* 分页查询出有排班的部门列表  \*/* @GetMapping(**"listDeptForScheduling"**)  **public** AjaxResult listDeptForScheduling(RegistrationQueryDto registrationQueryDto){  Long deptId = registrationQueryDto.getDeptId();  String schedulingDay = registrationQueryDto.getSchedulingDay();  String schedulingType = registrationQueryDto.getSchedulingType();  String subsectionType = registrationQueryDto.getSubsectionType();  List<Long> deptIds=**this**.**schedulingService**.queryHasSchedulingDeptIds(deptId,schedulingDay,schedulingType,subsectionType);  **if**(deptIds==**null**||deptIds.size()==0){  **return** AjaxResult.*success*(Collections.***EMPTY\_LIST***);  }**else** {  List<Dept> list = **this**.**deptService**.listDeptByDeptIds(deptIds);  **return** AjaxResult.*success*(list);  }  }   */\*\*  \*根据身份证号查询患者信息  \*/* @GetMapping(**"getPatientByIdCard/{idCard}"**)  **public** AjaxResult getPatientByIdCard(@PathVariable String idCard){  Patient patient=**this**.**patientService**.getPatientByIdCard(idCard);  **if**(**null**==patient){  **return** AjaxResult.*fail*(idCard+**"对应的患者不存在，请在下面新建患者信息"**);  }**else**{  **return** AjaxResult.*success*(patient);  }  }   */\*\*  \* 添加挂号信息  \* 如果患者ID为空，则先添加患者信息  \*/* @PostMapping(**"addRegistration"**)  **public** AjaxResult addRegistration(@RequestBody @Validated RegistrationFormDto registrationFormDto){  PatientDto patientDto = registrationFormDto.getPatientDto();  RegistrationDto registrationDto = registrationFormDto.getRegistrationDto();  Patient patient=**null**;  *//保存患者信息* **if**(StringUtils.*isBlank*(patientDto.getPatientId())){  patientDto.setPatientId(IdGeneratorSnowflake.*generatorIdWithProfix*(**"HZ"**));  patientDto.setSimpleUser(ShiroSecurityUtils.*getCurrentSimpleUser*());  patient=**patientService**.addPatient(patientDto);  }**else**{  patient=**this**.**patientService**.getPatientById(patientDto.getPatientId());  }  **if**(**null**==patient){  **return** AjaxResult.*fail*(**"当前患者ID不存在，请确认后再提交"**);  }  *//修改部门挂号* Dept dept = **deptService**.getOne(registrationDto.getDeptId());  *//保存挂号信息* registrationDto.setRegId(IdGeneratorSnowflake.*generatorIdWithProfix*(**"GH"**));  registrationDto.setPatientId(patient.getPatientId());  registrationDto.setPatientName(patient.getName());  registrationDto.setRegNumber(dept.getRegNumber()+1);  registrationDto.setVisitDate(registrationDto.getVisitDate().substring(0,10));  registrationDto.setSimpleUser(ShiroSecurityUtils.*getCurrentSimpleUser*());  **this**.**registrationService**.addRegistration(registrationDto);  *//修改部门挂号信息编号* dept.setRegNumber(dept.getRegNumber()+1);  **this**.**deptService**.updateDeptRegNumber(dept.getDeptId(),dept.getRegNumber());  *//返回挂号编号* **return** AjaxResult.*success*(**""**,registrationDto.getRegId());  }    *//分页加载挂号列表【默认当天的】* @GetMapping(**"queryRegistrationForPage"**)  **public** AjaxResult queryRegistrationForPage(RegistrationDto registrationDto){  **if**(registrationDto.getBeginTime()==**null**||registrationDto.getEndTime()==**null**){  registrationDto.setVisitDate(DateUtil.*format*(DateUtil.*date*(),**"yyyy-MM-dd"**));  }  DataGridView gridView=**this**.**registrationService**.queryRegistrationForPage(registrationDto);  **return** AjaxResult.*success*(**"查询成功"**,gridView.getData(),gridView.getTotal());  }   */\*\*  \* 收费  \*/* @PostMapping(**"collectFee/{registrationId}"**)  **public** AjaxResult collectFee(@PathVariable String registrationId){  Registration registration=**this**.**registrationService**.queryRegistrationByRegId(registrationId);  **if**(**null**==registration){  **return** AjaxResult.*fail*(**"当前挂号单ID对应的挂号单不存在，请核对后再查询"**);  }  **if**(!registration.getRegStatus().equals(Constants.***REG\_STATUS\_0***)){  **return** AjaxResult.*fail*(**"当前挂号单状态不是未收费状态，不能收费"**);  }  registration.setRegStatus(Constants.***REG\_STATUS\_1***);  **return** AjaxResult.*toAjax*(**this**.**registrationService**.updateRegistrationById(registration));   }} |

### 创建RegistrationService

|  |
| --- |
| **package** com.bjsxt.service;  **import** com.bjsxt.domain.Registration; **import** com.bjsxt.dto.RegistrationDto; **import** com.bjsxt.vo.DataGridView;  */\*\* \** ***@Author:*** *尚学堂 雷哥 \*/* **public interface** RegistrationService{   */\*\*  \* 保存挂号单信息  \** ***@param registrationDto*** *\*/* **void** addRegistration(RegistrationDto registrationDto);   */\*\*  \* 分页加载挂号列表【默认当天的】  \** ***@param registrationDto*** *\** ***@return*** *\*/* DataGridView queryRegistrationForPage(RegistrationDto registrationDto);   */\*\*  \* 更新挂号单信息  \** ***@param registration*** *\** ***@return*** *\*/* **int** updateRegistrationById(Registration registration);   */\*\*  \* 根据挂号ID查询挂号信息  \** ***@param registrationId*** *\** ***@return*** *\*/* Registration queryRegistrationByRegId(String registrationId); } |

### 创建RegistrationServiceImpl

|  |
| --- |
| **package** com.bjsxt.service.impl;  **import** cn.hutool.core.bean.BeanUtil; **import** cn.hutool.core.date.DateUtil; **import** com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; **import** com.baomidou.mybatisplus.extension.plugins.pagination.Page; **import** com.bjsxt.constants.Constants; **import** com.bjsxt.domain.Registration; **import** com.bjsxt.dto.RegistrationDto; **import** com.bjsxt.mapper.RegistrationMapper; **import** com.bjsxt.service.RegistrationService; **import** com.bjsxt.vo.DataGridView; **import** org.apache.commons.lang3.StringUtils; **import** org.apache.dubbo.config.annotation.Service; **import** org.springframework.beans.factory.annotation.Autowired;  */\*\* \** ***@Author:*** *尚学堂 雷哥 \*/* @Service **public class** RegistrationServiceImpl **implements** RegistrationService{   @Autowired  **private** RegistrationMapper **registrationMapper**;   @Override  **public void** addRegistration(RegistrationDto registrationDto) {  Registration registration=**new** Registration();  BeanUtil.*copyProperties*(registrationDto,registration);  registration.setRegStatus(Constants.***REG\_STATUS\_0***);  registration.setCreateBy(registrationDto.getSimpleUser().getUserName());  registration.setCreateTime(DateUtil.*date*());  **this**.**registrationMapper**.insert(registration);  }   @Override  **public** DataGridView queryRegistrationForPage(RegistrationDto registrationDto) {  Page<Registration> page=**new** Page<>(registrationDto.getPageNum(),registrationDto.getPageSize());  QueryWrapper<Registration> qw=**new** QueryWrapper<>();  qw.eq(registrationDto.getDeptId()!=**null**,Registration.***COL\_DEPT\_ID***,registrationDto.getDeptId());  qw.eq(StringUtils.*isNotBlank*(registrationDto.getSchedulingType()),Registration.***COL\_SCHEDULING\_TYPE***,registrationDto.getSchedulingType());  qw.eq(StringUtils.*isNotBlank*(registrationDto.getSubsectionType()),Registration.***COL\_SUBSECTION\_TYPE***,registrationDto.getSubsectionType());  qw.eq(StringUtils.*isNotBlank*(registrationDto.getRegStatus()), Registration.***COL\_REG\_STATUS***,registrationDto.getRegStatus());  qw.eq(StringUtils.*isNotBlank*(registrationDto.getVisitDate()),Registration.***COL\_VISIT\_DATE***,registrationDto.getVisitDate());  qw.ge(registrationDto.getBeginTime()!=**null**,Registration.***COL\_VISIT\_DATE***,DateUtil.*format*(registrationDto.getBeginTime(),**"yyyy-MM-dd"**));  qw.le(registrationDto.getEndTime()!=**null**,Registration.***COL\_VISIT\_DATE***,DateUtil.*format*(registrationDto.getEndTime(),**"yyyy-MM-dd"**));  qw.orderByDesc(Registration.***COL\_CREATE\_TIME***);  **this**.**registrationMapper**.selectPage(page,qw);  **return new** DataGridView(page.getTotal(),page.getRecords());  }   @Override  **public int** updateRegistrationById(Registration registration) {  **return this**.**registrationMapper**.updateById(registration);  }   @Override  **public** Registration queryRegistrationByRegId(String registrationId) {  **return this**.**registrationMapper**.selectById(registrationId);  } } |

## 排班相关修改

### 修改SchedulingService

|  |
| --- |
| */\*\*  \* 查询有排班的部门信息  \** ***@param deptId*** *\** ***@param schedulingDay*** *\** ***@param schedulingType*** *\** ***@param subsectionType*** *\** ***@return*** *\*/* List<Long> queryHasSchedulingDeptIds(Long deptId, String schedulingDay, String schedulingType, String subsectionType); |

### 修改SchedulingServiceImpl

|  |
| --- |
| @Override **public** List<Long> queryHasSchedulingDeptIds(Long deptId, String schedulingDay, String schedulingType, String subsectionType) {  **return this**.**schedulingMapper**.queryHasSchedulingDeptIds(deptId,schedulingDay,schedulingType,subsectionType); } |

### 修改SchedulingMapper

|  |
| --- |
| */\*\*  \** ***@param deptId*** *\** ***@param schedulingDay*** *\** ***@param schedulingType*** *\** ***@param subsectionType*** *\** ***@return*** *\*/* List<Long> queryHasSchedulingDeptIds(@Param(**"deptId"**) Long deptId, @Param(**"schedulingDay"**) String schedulingDay, @Param(**"schedulingType"**) String schedulingType, @Param(**"subsectionType"**) String subsectionType); |

### 修改SchedulingMapper.xml

|  |
| --- |
| <**select id="queryHasSchedulingDeptIds" resultType="java.lang.Long"**>  select distinct **dept\_id** from his\_scheduling t <**where**>  <**if test="deptId != null"**>  t.**dept\_id** =#{deptId}  </**if**>  <**if test="schedulingType!= null"**>  and t.**scheduling\_type**=#{schedulingType}  </**if**>  <**if test="schedulingDay!=null"**>  and t.**scheduling\_day**=#{schedulingDay}  </**if**>  <**if test="subsectionType!=null"**>  and t.**subsection\_type**=#{subsectionType}  </**if**> </**where**> </**select**> |

## 部门查询相关修改

### 修改DeptService

|  |
| --- |
| */\*\*  \* 根据部门ID查询部门信息  \** ***@param deptIds*** *\** ***@return*** *\*/* List<Dept> listDeptByDeptIds(List<Long> deptIds);  */\*\*  \* 根据部门ID更新部门挂号编号  \** ***@param deptId*** *\** ***@param regNumber*** *\*/* **void** updateDeptRegNumber(Long deptId, Integer regNumber); |

### 修改DeptServiceImpl

|  |
| --- |
| @Override **public** List<Dept> listDeptByDeptIds(List<Long> deptIds) {  QueryWrapper<Dept> qw=**new** QueryWrapper<>();  qw.in(Dept.***COL\_DEPT\_ID***,deptIds);  **return this**.**deptMapper**.selectList(qw); }  @Override **public void** updateDeptRegNumber(Long deptId, Integer regNumber) {  Dept dept=**new** Dept();  dept.setDeptId(deptId);  dept.setRegNumber(regNumber);  **this**.**deptMapper**.updateById(dept); } |

## 患者服务相关修改

### 修改PatientService

|  |
| --- |
| */\*\*  \* 根据身份证号查询患者信息  \** ***@param idCard*** *\** ***@return*** *\*/* Patient getPatientByIdCard(String idCard);  */\*\*  \* 保存患者信息  \** ***@param patientDto*** *\*/* Patient addPatient(PatientDto patientDto); |

### 修改PatientServiceImpl

|  |
| --- |
| @Override **public** Patient getPatientByIdCard(String idCard) {  QueryWrapper<Patient> qw=**new** QueryWrapper<>();  qw.eq(Patient.***COL\_ID\_CARD***,idCard);  **return this**.**patientMapper**.selectOne(qw); }  @Override **public** Patient addPatient(PatientDto patientDto) {  Patient patient=**new** Patient();  BeanUtil.*copyProperties*(patientDto,patient);  patient.setCreateTime(DateUtil.*date*());  patient.setIsFinal(Constants.***IS\_FINAL\_FALSE***);  String defaultPwd=patient.getPhone().substring(5);  patient.setPassword(AppMd5Utils.*md5*(defaultPwd,patient.getPhone(),2));  **this**.**patientMapper**.insert(patient);  **return** patient; } |

## 创建RegistrationController

|  |
| --- |
| **package** com.bjsxt.controller.doctor;  **import** com.bjsxt.constants.Constants; **import** com.bjsxt.controller.BaseController; **import** com.bjsxt.domain.Dept; **import** com.bjsxt.domain.Patient; **import** com.bjsxt.domain.Registration; **import** com.bjsxt.dto.PatientDto; **import** com.bjsxt.dto.RegistrationDto; **import** com.bjsxt.dto.RegistrationFormDto; **import** com.bjsxt.dto.RegistrationQueryDto; **import** com.bjsxt.service.DeptService; **import** com.bjsxt.service.PatientService; **import** com.bjsxt.service.RegistrationService; **import** com.bjsxt.service.SchedulingService; **import** com.bjsxt.utils.IdGeneratorSnowflake; **import** com.bjsxt.utils.ShiroSecurityUtils; **import** com.bjsxt.vo.AjaxResult; **import** com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand; **import** org.apache.commons.lang3.StringUtils; **import** org.apache.dubbo.config.annotation.Reference; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.validation.annotation.Validated; **import** org.springframework.web.bind.annotation.\*;  **import** java.util.Collections; **import** java.util.List;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \*/* @RestController @RequestMapping(**"doctor/registration"**) **public class** RegistrationController **extends** BaseController {    @Reference  **private** SchedulingService **schedulingService**;   @Autowired  **private** DeptService **deptService**;   @Reference  **private** PatientService **patientService**;   @Reference  **private** RegistrationService **registrationService**;  */\*\*  \* 查询出所有的排班的部门列表  \* 1.从排班表里面查询有排班的部门编号集合  \* 2.根据查询出来的部门编号集合再去查询部门  \*/* @GetMapping(**"listDeptForScheduling"**)  @HystrixCommand  **public** AjaxResult listDeptForScheduling(@Validated RegistrationQueryDto registrationQueryDto){  Long deptId = registrationQueryDto.getDeptId();  String subsectionType = registrationQueryDto.getSubsectionType();  String schedulingType = registrationQueryDto.getSchedulingType();*//挂号类型* String schedulingDay = registrationQueryDto.getSchedulingDay().substring(0,10);*//时间* List<Long> deptIds=**schedulingService**.queryHasSchedulingDeptIds(deptId,schedulingDay,schedulingType,subsectionType);   **if**(**null**==deptIds||deptIds.size()==0){  **return** AjaxResult.*success*(Collections.***EMPTY\_LIST***);  }**else**{  List<Dept> list=**this**.**deptService**.listDeptByDeptIds(deptIds);  **return** AjaxResult.*success*(list);  }  }    */\*\*  \* 根据身份证号查询患者信息  \*/* @GetMapping(**"getPatientByIdCard/{idCard}"**)  **public** AjaxResult getPatientByIdCard(@PathVariable String idCard){   Patient patient=**this**.**patientService**.getPatientByIdCard(idCard);  **if**(**null**==patient){  **return** AjaxResult.*fail*(**"【"**+idCard+**"】对应的患者不存在，请在下面新建患者信息"**);  }**else**{  **return** AjaxResult.*success*(patient);  }   }   */\*\*  \* 挂号  \*/* @PostMapping(**"addRegistration"**) *// @HystrixCommand* **public** AjaxResult addRegistration(@RequestBody @Validated RegistrationFormDto registrationFormDto){  PatientDto patientDto = registrationFormDto.getPatientDto();  RegistrationDto registrationDto = registrationFormDto.getRegistrationDto();  Patient patient=**null**;  **if**(StringUtils.*isBlank*(patientDto.getPatientId())){*//说明要新添加患者* patientDto.setPatientId(IdGeneratorSnowflake.*generatorIdWithProfix*(Constants.***ID\_PROFIX\_HZ***));  patientDto.setSimpleUser(ShiroSecurityUtils.*getCurrentSimpleUser*());  patient=**patientService**.addPatient(patientDto);  }**else**{  *//如果有患者编号，就把患者信息查询出来* patient=**this**.**patientService**.getPatientById(patientDto.getPatientId());  }  **if**(**null**==patient){  **return** AjaxResult.*fail*(**"当前患者ID不存在，请确认后再提交"**);  }  *//查询部门信息* Dept dept=**this**.**deptService**.getOne(registrationDto.getDeptId());  *//保存患者信息* registrationDto.setSimpleUser(ShiroSecurityUtils.*getCurrentSimpleUser*());  registrationDto.setRegId(IdGeneratorSnowflake.*generatorIdWithProfix*(Constants.***ID\_PROFIX\_GH***));  registrationDto.setPatientId(patient.getPatientId());  registrationDto.setPatientName(patient.getName());  registrationDto.setRegNumber(dept.getRegNumber()+1);*//编号+1* **this**.**registrationService**.addRegistration(registrationDto);  *//更新当天的科室号段* **this**.**deptService**.updateDeptRegNumber(dept.getDeptId(),dept.getRegNumber()+1);  *//返回挂号编号给前端* **return** AjaxResult.*success*(**""**,registrationDto.getRegId());  }   */\*\*  \* 收费  \*/* @PostMapping(**"collectFee/{regId}"**) *// @HystrixCommand* **public** AjaxResult collectFee(@PathVariable String regId){  *//查询挂号单* Registration registration=**this**.**registrationService**.queryRegistrationByRegId(regId);  **if**(**null**==registration){  **return** AjaxResult.*fail*(**"当前挂号单【"**+regId+**"】对应的挂号单不存在，请核对后再查询"**);  }  *//如果挂号单的状态不是未收费* **if**(!registration.getRegStatus().equals(Constants.***REG\_STATUS\_0***)){  **return** AjaxResult.*fail*(**"当前挂号单【"**+regId+**"】的状态不是未收费状态，不能进行收费"**);  }  *//收费，更新挂号单的状态* registration.setRegStatus(Constants.***REG\_STATUS\_1***);  **return** AjaxResult.*toAjax*(**this**.**registrationService**.updateRegistrationByRegId(registration));  }  } |

# 挂号列表

## 修改RegistrationController【system-web】

|  |
| --- |
| */\*\*  \* 分页查询挂号信息  \*/* @GetMapping(**"queryRegistrationForPage"**) @HystrixCommand **public** AjaxResult queryRegistrationForPage(RegistrationDto registrationDto){  DataGridView gridView=**this**.**registrationService**.queryRegistrationForPage(registrationDto);  **return** AjaxResult.*success*(**"查询成功"**,gridView.getData(),gridView.getTotal()); }    */\*\*  \* 作废  \*/* @PostMapping(**"doInvalid/{regId}"**) @HystrixCommand **public** AjaxResult doInvalid(@PathVariable String regId){  Registration registration=**this**.**registrationService**.queryRegistrationByRegId(regId);  **if**(**null**==registration){  **return** AjaxResult.*fail*(**"当前挂号单【"**+regId+**"】对应的挂号单不存在，请核对后再查询"**);  }  *//如果挂号单的状态不是未收费* **if**(!registration.getRegStatus().equals(Constants.***REG\_STATUS\_0***)){  **return** AjaxResult.*fail*(**"当前挂号单【"**+regId+**"】的状态不是未收费状态，不能作废"**);  }  *//收费，更新挂号单的状态* registration.setRegStatus(Constants.***REG\_STATUS\_5***);  **return** AjaxResult.*toAjax*(**this**.**registrationService**.updateRegistrationByRegId(registration)); }  */\*\*  \* 退号  \*/* @PostMapping(**"doReturn/{regId}"**) @HystrixCommand **public** AjaxResult doReturn(@PathVariable String regId){  Registration registration=**this**.**registrationService**.queryRegistrationByRegId(regId);  **if**(**null**==registration){  **return** AjaxResult.*fail*(**"当前挂号单【"**+regId+**"】对应的挂号单不存在，请核对后再查询"**);  }  *//如果挂号单的状态不是未收费* **if**(!registration.getRegStatus().equals(Constants.***REG\_STATUS\_1***)){  **return** AjaxResult.*fail*(**"当前挂号单【"**+regId+**"】的状态不是待就诊状态，不能退号"**);  }  *//收费，更新挂号单的状态* registration.setRegStatus(Constants.***REG\_STATUS\_4***);  **return** AjaxResult.*toAjax*(**this**.**registrationService**.updateRegistrationByRegId(registration)); } |

## 修改RegistrationService【doctor-api】

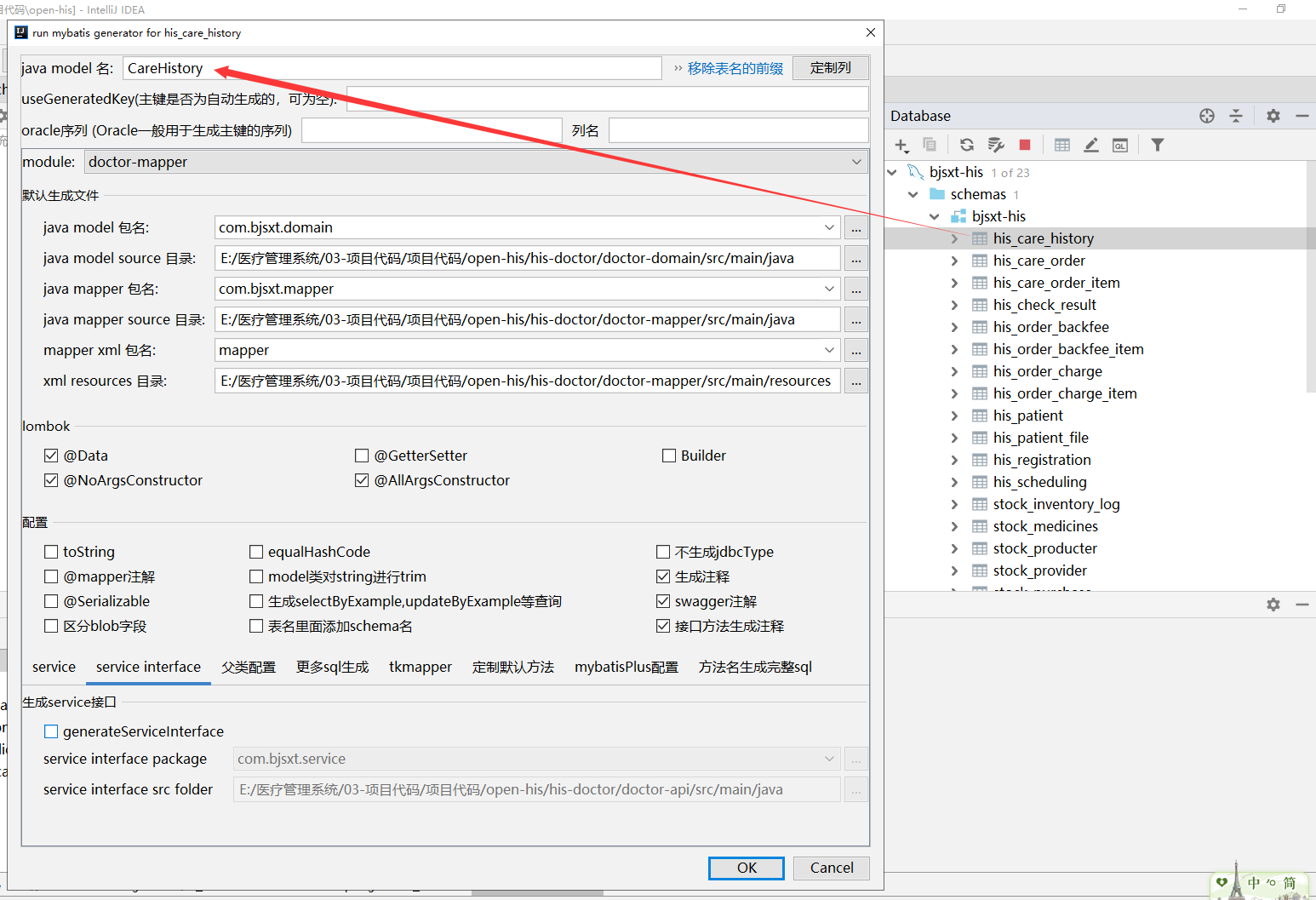
|  |
| --- |
| */\*\*  \* 分页查询挂号单  \** ***@param registrationDto*** *\** ***@return*** *\*/* DataGridView queryRegistrationForPage(RegistrationDto registrationDto); |

## 修改RegistrationServiceImpl【doctor-service】

|  |
| --- |
| @Override **public** DataGridView queryRegistrationForPage(RegistrationDto registrationDto) {  Page<Registration> page=**new** Page<>(registrationDto.getPageNum(),registrationDto.getPageSize());  QueryWrapper<Registration> qw=**new** QueryWrapper<>();  qw.eq(registrationDto.getDeptId()!=**null**,Registration.***COL\_DEPT\_ID***,registrationDto.getDeptId());  qw.like(StringUtils.*isNotBlank*(registrationDto.getPatientName()),Registration.***COL\_PATIENT\_NAME***,registrationDto.getPatientName());  qw.eq(StringUtils.*isNotBlank*(registrationDto.getSchedulingType()),Registration.***COL\_SCHEDULING\_TYPE***,registrationDto.getSchedulingType());  qw.eq(StringUtils.*isNotBlank*(registrationDto.getSubsectionType()),Registration.***COL\_SUBSECTION\_TYPE***,registrationDto.getSubsectionType());  qw.eq(StringUtils.*isNotBlank*(registrationDto.getRegStatus()),Registration.***COL\_REG\_STATUS***,registrationDto.getRegStatus());  qw.eq(StringUtils.*isNotBlank*(registrationDto.getVisitDate()),Registration.***COL\_VISIT\_DATE***,registrationDto.getVisitDate());  qw.orderByDesc(Registration.***COL\_CREATE\_TIME***);  **this**.**registrationMapper**.selectPage(page,qw);  **return new** DataGridView(page.getTotal(),page.getRecords()); } |

# 新开就诊第一部分

## 生成病例表相关代码



### 生成CareHistory【doctor-domain项目】

不动

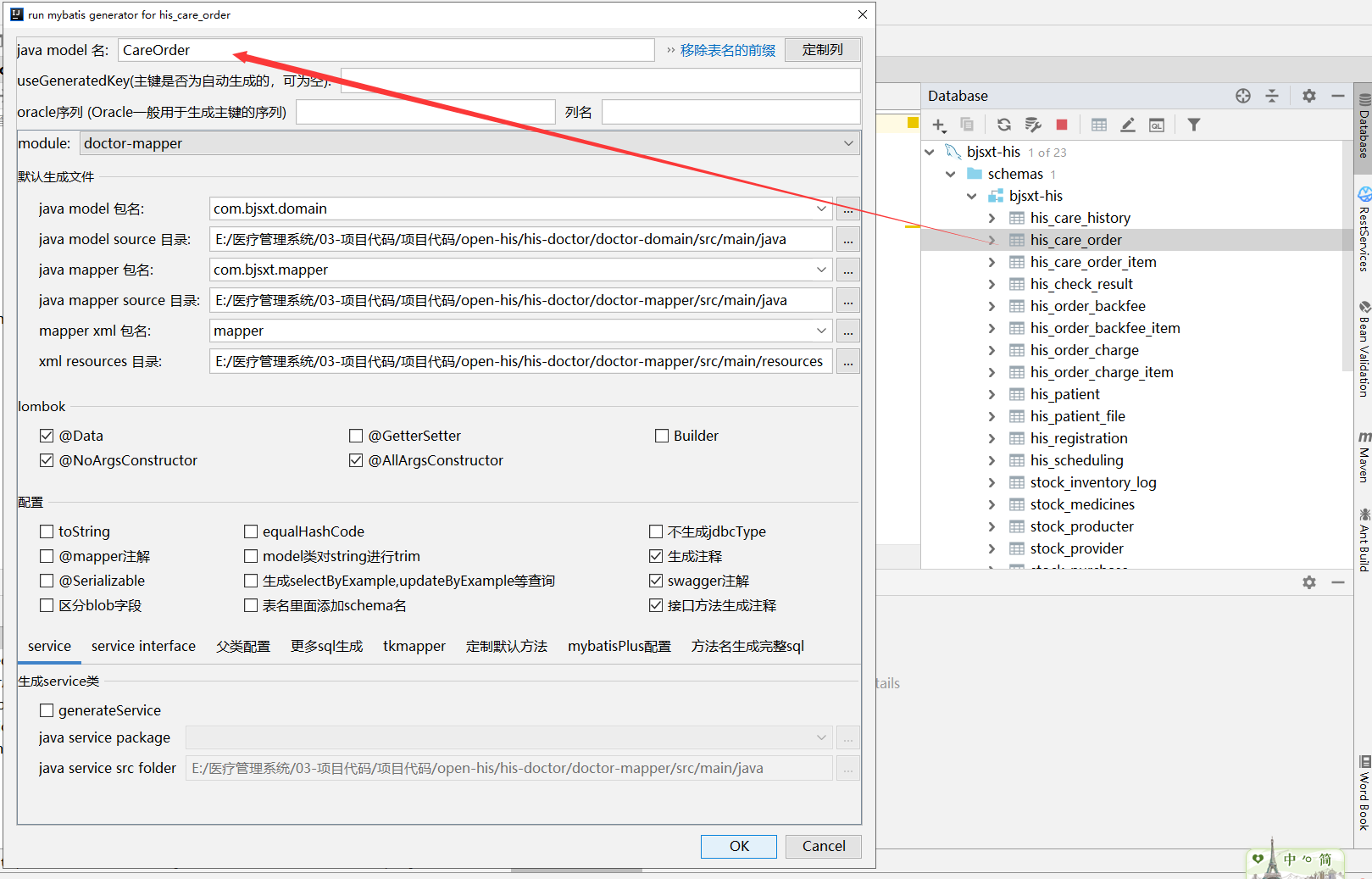
### 生成CareHistoryMapper【doctor-mapper项目】

不动

### 生成CareHistoryMapper.xml【doctor-mapper项目】

不动

## 生成处方表相关代码



### 生成CareOrder【doctor-domain项目】

不动

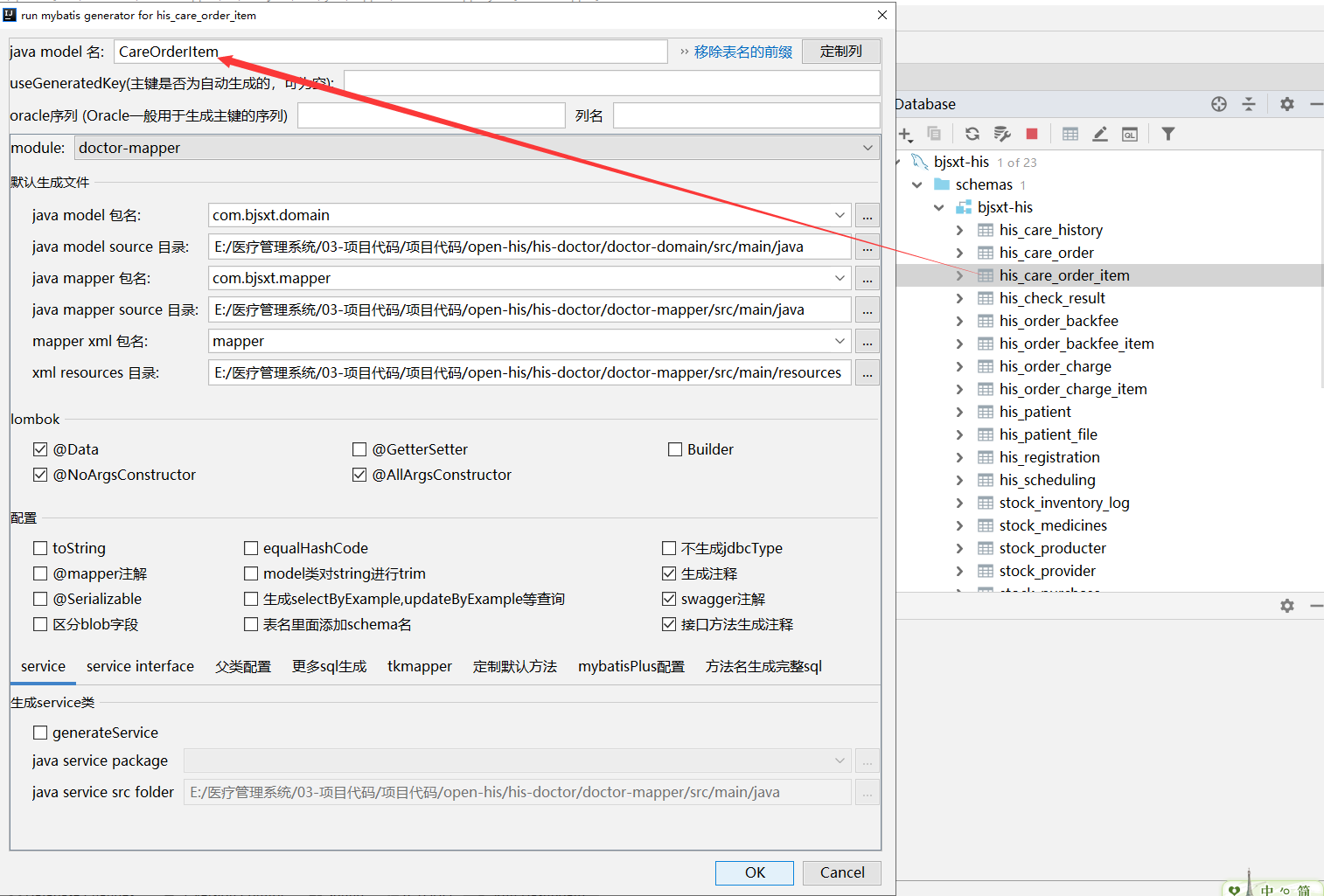
### 生成CareOrderMapper【doctor-mapper项目】

不动

### 生成CareOrderMapper.xml【doctor-mapper项目】

不动

## 生成处方详情方表相关代码



### 生成CareOrderItem【doctor-domain项目】

不动

### 生成CareOrderItemMapper【doctor-mapper项目】

不动

### 生成CareOrderItemMapper.xml【doctor-mapper项目】

不动

## 创建HisDateUtils【his-commons】

|  |
| --- |
| **package** com.bjsxt.utils;  **import** cn.hutool.core.date.DateUtil;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \*/* **public class** HisDateUtils {  */\*\*  \* 得到时段  \* 1上午  \* 2下午  \* 3晚上  \*/* **public static** String getCurrentTimeType(){  **int** hour = DateUtil.*hour*(DateUtil.*date*(), **true**);  **if**(hour>=8&&hour<12){  **return "1"**;  }**else if**(hour>=12&& hour<18){  **return "2"**;  }**else** {  **return "3"**;  }  } } |

## 创建CareController【system-web】

|  |
| --- |
| **package** com.bjsxt.controller.doctor;  **import** com.bjsxt.constants.Constants; **import** com.bjsxt.controller.BaseController; **import** com.bjsxt.domain.CareHistory; **import** com.bjsxt.domain.Patient; **import** com.bjsxt.domain.PatientFile; **import** com.bjsxt.domain.Registration; **import** com.bjsxt.service.CareService; **import** com.bjsxt.service.PatientService; **import** com.bjsxt.service.RegistrationService; **import** com.bjsxt.utils.HisDateUtils; **import** com.bjsxt.utils.ShiroSecurityUtils; **import** com.bjsxt.vo.AjaxResult; **import** org.apache.dubbo.config.annotation.Reference; **import** org.springframework.web.bind.annotation.\*;  **import** java.util.HashMap; **import** java.util.List; **import** java.util.Map;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \** ***@Description:*** *新开就诊控制器  \*/* @RestController @RequestMapping(**"doctor/care"**) **public class** CareController **extends** BaseController {   @Reference  **private** RegistrationService **registrationService**;   @Reference  **private** PatientService **patientService**;   @Reference  **private** CareService **careService**;    */\*\*  \* 查询待就诊的挂号信息  \* GET/doctor/care/queryToBeSeenRegistration/{scheudlingType}  \*/* @GetMapping(**"queryToBeSeenRegistration/{scheudlingType}"**)  **public** AjaxResult queryToBeSeenRegistration(@PathVariable String scheudlingType){  *//得到当前用户的部门ID* Long deptId= ShiroSecurityUtils.*getCurrentUser*().getDeptId();  *//设置要查询的状态 只能是挂号单的待就诊的挂号信息* String regStatus= Constants.***REG\_STATUS\_1***;  *//计算时段* String subsectionType= HisDateUtils.*getCurrentTimeType*();  *//查询* Long userId=**null**;  List<Registration> list=**this**.**registrationService**.queryRegistration(deptId,subsectionType,scheudlingType,regStatus,userId);  **return** AjaxResult.*success*(list);  }    */\*\*  \* 查询就诊中的挂号信息  \* GET/doctor/care/queryVisitingRegistration/{scheudlingType}  \*/* @GetMapping(**"queryVisitingRegistration/{scheudlingType}"**)  **public** AjaxResult queryVisitingRegistration(@PathVariable String scheudlingType){  *//得到当前用户的部门ID* Long deptId= ShiroSecurityUtils.*getCurrentUser*().getDeptId();  *//设置要查询的状态 只能是挂号单的待就诊的挂号信息* String regStatus= Constants.***REG\_STATUS\_2***;  *//计算时段 // String subsectionType= HisDateUtils.getCurrentTimeType();  //查询* Long userId=ShiroSecurityUtils.*getCurrentUser*().getUserId();  List<Registration> list=**this**.**registrationService**.queryRegistration(deptId,**null**,scheudlingType,regStatus,userId);  **return** AjaxResult.*success*(list);  }   */\*\*  \* 查询就诊完成的挂号信息  \* GET/doctor/care/queryVisitCompletedRegistration/{scheudlingType}  \*/* @GetMapping(**"queryVisitCompletedRegistration/{scheudlingType}"**)  **public** AjaxResult queryVisitCompletedRegistration(@PathVariable String scheudlingType){  *//得到当前用户的部门ID* Long deptId= ShiroSecurityUtils.*getCurrentUser*().getDeptId();  *//设置要查询的状态 只能是挂号单的待就诊的挂号信息* String regStatus= Constants.***REG\_STATUS\_3***;  *//计算时段 // String subsectionType= HisDateUtils.getCurrentTimeType();  //查询* Long userId=ShiroSecurityUtils.*getCurrentUser*().getUserId();  List<Registration> list=**this**.**registrationService**.queryRegistration(deptId,**null**,scheudlingType,regStatus,userId);  **return** AjaxResult.*success*(list);  }   */\*\*  \* 接诊  \* POST/doctor/care/receivePatient/{regId}  \*/* @PostMapping(**"receivePatient/{regId}"**)  **public** AjaxResult receivePatient(@PathVariable String regId){  **synchronized** (**this**){ *//防止并发接诊的问题  //根据挂号单ID查询挂号信息* Registration registration=**this**.**registrationService**.queryRegistrationByRegId(regId);  **if**(**null**==registration){  **return** AjaxResult.*fail*(**"【"**+regId+**"】挂号单的不存在，不能接诊"**);  }  *//只有当挂号单的状态 regStatus为 待就诊时可以接诊* **if**(registration.getRegStatus().equals(Constants.***REG\_STATUS\_1***)){  registration.setRegStatus(Constants.***REG\_STATUS\_2***);*//就诊中* registration.setUserId(ShiroSecurityUtils.*getCurrentUser*().getUserId());  registration.setDoctorName(ShiroSecurityUtils.*getCurrentUser*().getUserName());  **return** AjaxResult.*toAjax*(**this**.**registrationService**.updateRegistrationByRegId(registration));  }**else**{  **return** AjaxResult.*fail*(**"【"**+regId+**"】挂号单的状态不是待就诊状态，不能接诊"**);  }  }  }    */\*\*  \* 根据患者ID获取患者信息、档案信息、病历信息  \* GET/doctor/care/getPatientAllMessageByPatientId/{patientId}  \*/* @GetMapping(**"getPatientAllMessageByPatientId/{patientId}"**)  **public** AjaxResult getPatientAllMessageByPatientId(@PathVariable String patientId){  *//查询患者信息* Patient patient=**this**.**patientService**.getPatientById(patientId);  *//查询档案* PatientFile patientFile=**this**.**patientService**.getPatientFileById(patientId);  *//查询病历表* List<CareHistory> careHistories=**this**.**careService**.queryCareHistoryByPatientId(patientId);  Map<String,Object> res=**new** HashMap<>();  res.put(**"patient"**,patient);  res.put(**"patientFile"**,patientFile);  res.put(**"careHistoryList"**,careHistories);  **return** AjaxResult.*success*(res);  }  } |

## 修改RegistrationService【doctor-api】

|  |
| --- |
| */\*\*  \* 根据条件查询挂号信息  \** ***@param deptId*** *部门  \** ***@param subsectionType*** *时段  \** ***@param scheudlingType*** *类型 门诊 急诊  \** ***@param regStatus*** *挂号单状态  \** ***@param userId*** *医生ID  \** ***@return*** *\*/* List<Registration> queryRegistration(Long deptId, String subsectionType, String scheudlingType, String regStatus, Long userId); |

## 创建RegistrationServiceImpl【doctor-service】

|  |
| --- |
| */\*\*  \* 根据条件查询挂号信息  \** ***@param deptId*** *部门  \** ***@param subsectionType*** *时段  \** ***@param scheudlingType*** *类型 门诊 急诊  \** ***@param regStatus*** *挂号单状态  \** ***@param userId*** *医生ID  \** ***@return*** *\*/* @Override **public** List<Registration> queryRegistration(Long deptId, String subsectionType, String scheudlingType, String regStatus, Long userId) {  QueryWrapper<Registration> qw=**new** QueryWrapper<>();  qw.eq(Registration.***COL\_DEPT\_ID***,deptId);  qw.eq(StringUtils.*isNotBlank*(subsectionType),Registration.***COL\_SUBSECTION\_TYPE***,subsectionType);  qw.eq(Registration.***COL\_SCHEDULING\_TYPE***,scheudlingType);  qw.eq(Registration.***COL\_REG\_STATUS***,regStatus);  qw.eq(**null**!=userId,Registration.***COL\_USER\_ID***,userId);  qw.orderByAsc(Registration.***COL\_REG\_NUMBER***);  **return this**.**registrationMapper**.selectList(qw); } |

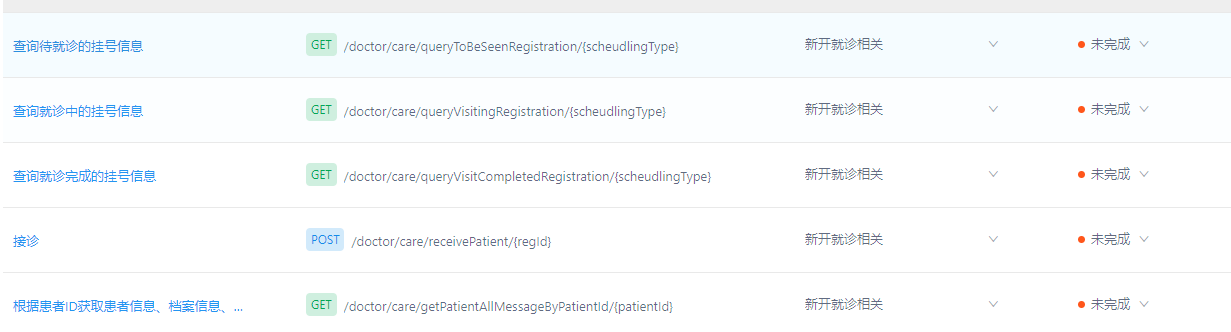
## 创建CareService【doctor-api】

|  |
| --- |
| **package** com.bjsxt.service;  **import** com.bjsxt.domain.CareHistory;  **import** java.util.List;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \* 就诊相关的接口  \*/* **public interface** CareService {  */\*\*  \* 根据患者ID查询历史病历  \** ***@param patientId*** *\** ***@return*** *\*/* List<CareHistory> queryCareHistoryByPatientId(String patientId); } |

## 创建CareServiceImpl【doctor-service】

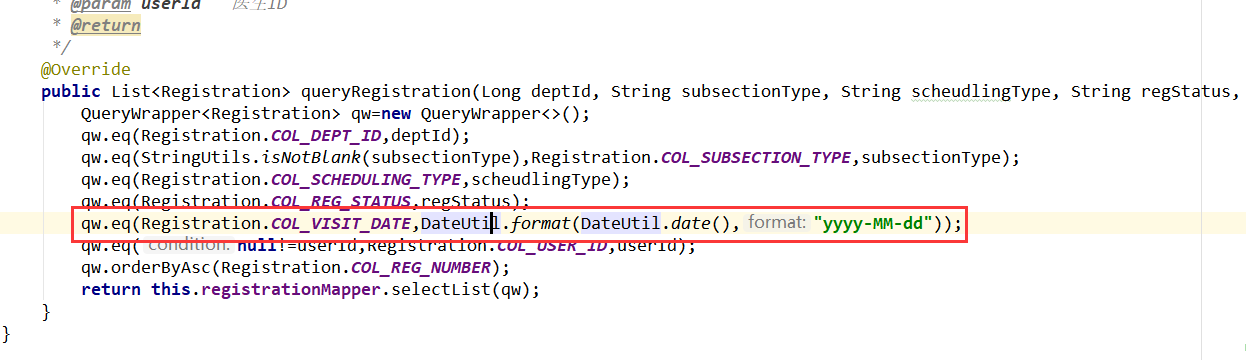
|  |
| --- |
| **package** com.bjsxt.service.impl;   **import** com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; **import** com.bjsxt.domain.CareHistory; **import** com.bjsxt.mapper.CareHistoryMapper; **import** com.bjsxt.mapper.CareOrderItemMapper; **import** com.bjsxt.mapper.CareOrderMapper; **import** com.bjsxt.service.CareService; **import** org.apache.dubbo.config.annotation.Service; **import** org.springframework.beans.factory.annotation.Autowired;  **import** java.util.List;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \*/* @Service **public class** CareServiceImpl **implements** CareService {  @Autowired  **private** CareHistoryMapper **careHistoryMapper**;    @Override  **public** List<CareHistory> queryCareHistoryByPatientId(String patientId) {  QueryWrapper<CareHistory> qw = **new** QueryWrapper<>();  qw.eq(CareHistory.***COL\_PATIENT\_ID***, patientId);  **return this**.**careHistoryMapper**.selectList(qw);  } } |

## yapi测试



## 解决挂号的日期BUG

### 修改RegistrationServiceImpl【doctor-service】



# 新开就诊第二部分

## 保存病例

### 创建CareHistoryDto【doctor-domain项目】

|  |
| --- |
| **package** com.bjsxt.dto;  **import** com.baomidou.mybatisplus.annotation.TableField; **import** io.swagger.annotations.ApiModel; **import** io.swagger.annotations.ApiModelProperty; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.EqualsAndHashCode; **import** lombok.NoArgsConstructor;  **import** javax.validation.constraints.NotBlank; **import** java.util.Date;  */\*\* \** ***@Author:*** *尚学堂 雷哥  \* 病例表 \*/* @ApiModel(value=**"com-bjsxt-dto-CareHistoryDto"**) @Data @EqualsAndHashCode(callSuper=**true**) @AllArgsConstructor @NoArgsConstructor **public class** CareHistoryDto **extends** BaseDto {  */\*\*  \* 病历ID  \*/* @ApiModelProperty(value=**"病历ID"**)  **private** String **chId**;   */\*\*  \* 医生id  \*/* @ApiModelProperty(value=**"医生id"**)  **private** Long **userId**;   */\*\*  \* 医生姓名  \*/* @ApiModelProperty(value=**"医生姓名"**)  **private** String **userName**;   */\*\*  \* 患者id  \*/* @NotBlank(message = **"患者编号不能为空"**)  @ApiModelProperty(value=**"患者id"**)  **private** String **patientId**;   */\*\*  \* 患者姓名  \*/* @NotBlank(message = **"患者姓名不能为空"**)  @ApiModelProperty(value=**"患者姓名"**)  **private** String **patientName**;   */\*\*  \* 科室id  \*/* @ApiModelProperty(value=**"科室id"**)  **private** Long **deptId**;   */\*\*  \* 科室名称  \*/* @ApiModelProperty(value=**"科室名称"**)  **private** String **deptName**;   */\*\*  \* 接诊类型：0初诊，1复诊 字典表属性his\_receive\_type  \*/* @NotBlank(message = **"接诊类型不能为空"**)  @ApiModelProperty(value=**"接诊类型：0初诊，1复诊 字典表属性his\_receive\_type"**)  **private** String **receiveType**;   */\*\*  \* 是否传染，0否，1是 字典表属性his\_contagious\_status  \*/* @NotBlank(message = **"是否传染不能为空"**)  @ApiModelProperty(value=**"是否传染，0否，1是 字典表属性his\_contagious\_status"**)  **private** String **isContagious**;   */\*\*  \* 就诊时间  \*/* @ApiModelProperty(value=**"就诊时间"**)  **private** Date **careDate**;   */\*\*  \* 发病日期  \*/* @TableField(value = **"case\_date"**)  @NotBlank(message = **"发病日期不能为空"**)  @ApiModelProperty(value=**"发病日期"**)  **private** String **caseDate**;   */\*\*  \* 挂号单号  \*/* @NotBlank(message = **"挂号单号不能为空"**)  @ApiModelProperty(value=**"挂号单号"**)  **private** String **regId**;   */\*\*  \* 主诉  \*/* @ApiModelProperty(value=**"主诉"**)  **private** String **caseTitle**;   */\*\*  \* 诊断信息  \*/* @ApiModelProperty(value=**"诊断信息"**)  **private** String **caseResult**;   */\*\*  \* 医生建议  \*/* @ApiModelProperty(value=**"医生建议"**)  **private** String **doctorTips**;   */\*\*  \* 备注  \*/* @ApiModelProperty(value=**"备注"**)  **private** String **remark**; } |

### 修改CareController【system-web项目】

|  |
| --- |
| */\*\*  \* 保存病例信息  \*/* @PostMapping(**"saveCareHistory"**) **public** AjaxResult saveCareHistory(@RequestBody CareHistoryDto careHistoryDto){  careHistoryDto.setUserId(ShiroSecurityUtils.*getCurrentUser*().getUserId());  careHistoryDto.setUserName(ShiroSecurityUtils.*getCurrentUserName*());  careHistoryDto.setDeptId(ShiroSecurityUtils.*getCurrentUser*().getDeptId());  Dept dept=**this**.**deptService**.getOne(ShiroSecurityUtils.*getCurrentUser*().getDeptId());  careHistoryDto.setDeptName(dept.getDeptName());  careHistoryDto.setCareDate(DateUtil.*date*());  CareHistory ch = **this**.**careService**.saveOrUpdateCareHistory(careHistoryDto);  **return** AjaxResult.*success*(ch); } |

### 修改CareService【doctor-api项目】

|  |
| --- |
| */\*\*  \* 保存或更新病例信息  \** ***@param careHistoryDto*** *\** ***@return*** *\*/* CareHistory saveOrUpdateCareHistory(CareHistoryDto careHistoryDto); |

### 修改Constants【his-commons项目】

|  |
| --- |
| String ***ID\_PROFIX\_CH*** =**"CH"**;*//病历* |

### 修改CareServiceImpl【doctor-service项目】

|  |
| --- |
| @Override **public** CareHistory saveOrUpdateCareHistory(CareHistoryDto careHistoryDto) {  CareHistory careHistory=**new** CareHistory();  BeanUtil.*copyProperties*(careHistoryDto,careHistory);  **if** (StringUtils.*isNotBlank*(careHistory.getChId())) {  **this**.**careHistoryMapper**.updateById(careHistory);  } **else** {  careHistory.setChId(IdGeneratorSnowflake.*generatorIdWithProfix*(Constants.***ID\_PROFIX\_CH***));  **this**.**careHistoryMapper**.insert(careHistory);  }  **return** careHistory; } |

## 查询病历和处方信息

### 修改CareController【system-web项目】

|  |
| --- |
| */\*\*  \* 根据挂号单位ID查询对应的病历信息  \*/* @GetMapping(**"getCareHistoryByRegId/{regId}"**) @HystrixCommand **public** AjaxResult getCareHistoryByRegId(@PathVariable String regId){  CareHistory careHistory=**this**.**careService**.queryCareHistoryByRegId(regId);  **return** AjaxResult.*success*(careHistory); }   */\*\*  \* 根据病历ID查询处方信息及详情信息  \*/* @GetMapping(**"queryCareOrdersByChId/{chId}"**) @HystrixCommand **public** AjaxResult queryCareOrdersByChId(@PathVariable String chId){  List<CareOrder> careOrders=**this**.**careService**.queryCareOrdersByChId(chId);  List<Map<String,Object>> res=**new** ArrayList<>();  **for** (CareOrder careOrder : careOrders) {  Map<String,Object> map=**new** HashMap<>();  map.put(**"careOrder"**,careOrder);  List<CareOrderItem> careOrderItems=**this**.**careService**.queryCareOrderItemsByCoId(careOrder.getCoId());  map.put(**"careOrderItems"**,careOrderItems);  res.add(map);  }  **return** AjaxResult.*success*(res); } |

### 修改CareService【doctor-api项目】

|  |
| --- |
| */\*\*  \* 根据挂号单位ID查询对应的病历信息  \** ***@param regId*** *\** ***@return*** *\*/* CareHistory queryCareHistoryByRegId(String regId);  */\*\*  \* 根据病历信息查询处方信息  \** ***@param chId*** *\** ***@return*** *\*/* List<CareOrder> queryCareOrdersByChId(String chId);  */\*\*  \* 根据处方ID查询处方详情信息  \** ***@param coId*** *\** ***@return*** *\*/* List<CareOrderItem> queryCareOrderItemsByCoId(String coId); |

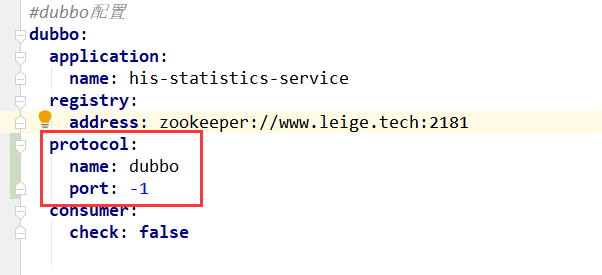
### 修改CareServiceImpl【doctor-service项目】

|  |
| --- |
| @Override **public** CareHistory queryCareHistoryByRegId(String regId) {  QueryWrapper<CareHistory> qw=**new** QueryWrapper<>();  qw.eq(CareHistory.***COL\_REG\_ID***,regId);  **return this**.**careHistoryMapper**.selectOne(qw); }  @Override **public** List<CareOrder> queryCareOrdersByChId(String chId) {  QueryWrapper<CareOrder> qw=**new** QueryWrapper<>();  qw.eq(CareOrder.***COL\_CH\_ID***,chId);  **return this**.**careOrderMapper**.selectList(qw); }  @Override **public** List<CareOrderItem> queryCareOrderItemsByCoId(String coId) {  QueryWrapper<CareOrderItem> qw=**new** QueryWrapper<>();  qw.eq(CareOrderItem.***COL\_CO\_ID***,coId);  **return this**.**careOrderItemMapper**.selectList(qw); } |

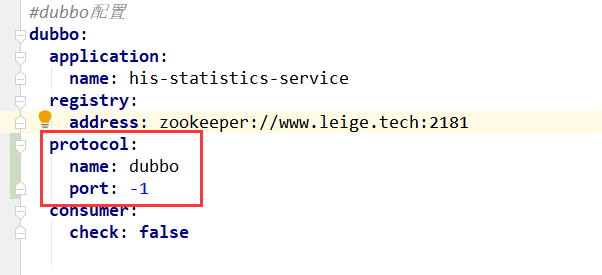
# 新开就诊第三部分

## 解决服务提供者端口占用的问题

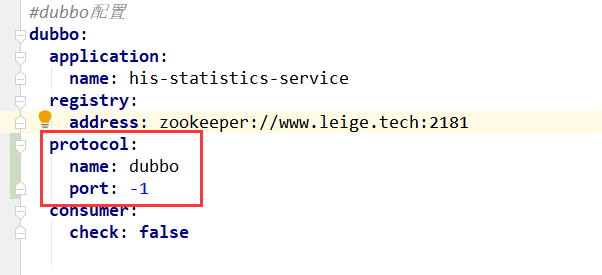
### 创建application-dev.yaml【doctor-service项目】



### 创建application-dev.yaml【erp-service项目】



### 创建application-dev.yaml【statistics-service项目】



## 保存处方及详情，删除详情，完成就诊接口编写

### 创建CareOrderDto【doctor-domain项目】

|  |
| --- |
| **package** com.bjsxt.dto;  **import** io.swagger.annotations.ApiModel; **import** io.swagger.annotations.ApiModelProperty; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.EqualsAndHashCode; **import** lombok.NoArgsConstructor;  **import** javax.validation.constraints.NotBlank; **import** javax.validation.constraints.NotNull; **import** java.math.BigDecimal;  */\*\* \** ***@Author:*** *尚学堂 雷哥 \*/* @ApiModel(value=**"com-bjsxt-dto-CareOrderDto"**) @Data @EqualsAndHashCode(callSuper=**true**) @AllArgsConstructor @NoArgsConstructor **public class** CareOrderDto **extends** BaseDto {  */\*\*  \* 处方ID  \*/* @ApiModelProperty(value=**"处方ID"**)  **private** String **coId**;   */\*\*  \* 处方类型0药用处方1检查处方  \*/* @NotBlank(message = **"处方类型不能为空"**)  @ApiModelProperty(value=**"处方类型0药用处方1检查处方"**)  **private** String **coType**;   */\*\*  \* 医生id  \*/* @ApiModelProperty(value=**"医生id"**)  **private** Long **userId**;   */\*\*  \* 患者id  \*/* @ApiModelProperty(value=**"患者id"**)  **private** String **patientId**;   */\*\*  \* 患者姓名  \*/* @ApiModelProperty(value=**"患者姓名"**)  **private** String **patientName**;   */\*\*  \* 关联病历id  \*/* @NotBlank(message = **"病历ID不能为空"**)  @ApiModelProperty(value=**"关联病历id"**)  **private** String **chId**;   */\*\*  \* 处方全额  \*/* @NotNull(message = **"处方全额不能为空"**)  @ApiModelProperty(value=**"处方全额"**)  **private** BigDecimal **allAmount**; } |

### 创建CareOrderItemDto【doctor-domain项目】

|  |
| --- |
| **package** com.bjsxt.dto;  **import** io.swagger.annotations.ApiModel; **import** io.swagger.annotations.ApiModelProperty; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.NoArgsConstructor;  **import** javax.validation.constraints.NotBlank; **import** javax.validation.constraints.NotNull; **import** java.io.Serializable; **import** java.math.BigDecimal;  */\*\* \** ***@Author:*** *尚学堂 雷哥 \*/* @ApiModel(value=**"com-bjsxt-dto-CareOrderItemDto"**) @Data @AllArgsConstructor @NoArgsConstructor **public class** CareOrderItemDto **implements** Serializable {  */\*\*  \* 开诊明细ID  \*/* @ApiModelProperty(value=**"开诊明细ID"**)  **private** String **itemId**;   */\*\*  \* 所属处方id  \*/* @ApiModelProperty(value=**"所属处方id"**)  **private** String **coId**;   */\*\*  \* 药品或者检查项目id  \*/* @NotBlank(message = **"药品或检查项目ID不能为空"**)  @ApiModelProperty(value=**"药品或者检查项目id"**)  **private** String **itemRefId**;   */\*\*  \* 药品或检查项目名称  \*/* @NotBlank(message = **"药品或检查项目名称不能为空"**)  @ApiModelProperty(value=**"药品或检查项目名称"**)  **private** String **itemName**;   */\*\*  \* 项目类型0药品 还是1检查项  \*/* @NotBlank(message = **"项目类型不能为空"**)  @ApiModelProperty(value=**"项目类型0药品 还是1检查项"**)  **private** String **itemType**;   */\*\*  \* 数量  \*/* @NotNull(message = **"数量不能为空"**)  @ApiModelProperty(value=**"数量"**)  **private** BigDecimal **num**;   */\*\*  \* 单价  \*/* @NotNull(message = **"单价不能为空"**)  @ApiModelProperty(value=**"单价"**)  **private** BigDecimal **price**;   */\*\*  \* 金额  \*/* @NotNull(message = **"金额不能为空"**)  @ApiModelProperty(value=**"金额"**)  **private** BigDecimal **amount**;   */\*\*  \* 备注  \*/* @ApiModelProperty(value=**"备注"**)  **private** String **remark**;  } |

### 创建CareOrderFormDto【doctor-domain项目】

|  |
| --- |
| **package** com.bjsxt.dto;  **import** io.swagger.annotations.ApiModel; **import** lombok.AllArgsConstructor; **import** lombok.Data; **import** lombok.NoArgsConstructor;  **import** javax.validation.constraints.NotEmpty; **import** java.io.Serializable; **import** java.util.List;  */\*\*  \** ***@Author:*** *尚学堂 雷哥  \*/* @ApiModel(value=**"com-bjsxt-dto-CareOrderFormDto"**) @Data @AllArgsConstructor @NoArgsConstructor **public class** CareOrderFormDto **implements** Serializable {  *//处方* **private** CareOrderDto **careOrder**;  *//处方详情* @NotEmpty(message = **"处方详情不能为空"**)  **private** List<CareOrderItemDto> **careOrderItems**; } |

### 修改Constants【his-commons项目】

|  |
| --- |
| String ***ID\_PROFIX\_CO*** = **"CO"**; *//病历* String ***ID\_PROFIX\_ITEM*** = **"ITEM"**; *//病历详情* |

### 修改CareOrderController【system-web项目】

|  |
| --- |
| */\*\*  \* 保存处方及详情信息  \*/* @PostMapping(**"saveCareOrderItem"**) @HystrixCommand **public** AjaxResult saveCareOrderItem(@RequestBody @Validated CareOrderFormDto careOrderFormDto){  *//根据病例ID查询病历信息* CareHistory careHistory=**this**.**careService**.queryCareHistoryByChId(careOrderFormDto.getCareOrder().getChId());  **if**(**null**==careHistory){  **return** AjaxResult.*fail*(**"病历ID不存在，请核对后再提交"**);  }  careOrderFormDto.getCareOrder().setCoId(IdGeneratorSnowflake.*generatorIdWithProfix*(Constants.***ID\_PROFIX\_CO***));  careOrderFormDto.getCareOrder().setPatientId(careHistory.getPatientId());  careOrderFormDto.getCareOrder().setPatientName(careHistory.getPatientName());  careOrderFormDto.getCareOrder().setUserId(ShiroSecurityUtils.*getCurrentUser*().getUserId());  careOrderFormDto.getCareOrder().setSimpleUser(ShiroSecurityUtils.*getCurrentSimpleUser*());  **return** AjaxResult.*toAjax*(**this**.**careService**.saveCareOrderItem(careOrderFormDto)); }  */\*\*  \* 根据处方详情ID删除处方详情【只能删除未支付的】  \*/* @DeleteMapping(**"deleteCareOrderItemById/{itemId}"**) @HystrixCommand **public** AjaxResult deleteCareOrderItemById(@PathVariable String itemId){  CareOrderItem careOrderItem=**this**.**careService**.queryCareOrderItemByItemId(itemId);  **if**(**null**==careOrderItem){  **return** AjaxResult.*fail*(**"处方详情ID不存在"**);  }  **if**(!careOrderItem.getStatus().equals(Constants.***ORDER\_DETAILS\_STATUS\_0***)){  **return** AjaxResult.*fail*(**"【"**+itemId+**"】不是未支付状态，不能删除"**);  }  **return** AjaxResult.*toAjax*(**this**.**careService**.deleteCareOrderItemByItemId(itemId)); }  */\*\*  \*完成就诊  \*/* @PostMapping(**"visitComplete/{regId}"**) @HystrixCommand **public** AjaxResult visitComplete(@PathVariable String regId){  Registration registration=**this**.**registrationService**.queryRegistrationByRegId(regId);  **if**(**null**==registration){  **return** AjaxResult.*fail*(**"【"**+regId+**"】挂号单号不存在，请核对后再提交"**);  }  **if**(!registration.getRegStatus().equals(Constants.***REG\_STATUS\_2***)){  **return** AjaxResult.*fail*(**"【"**+regId+**"】状态不是就诊中状态，不能完成就诊"**);  }  *//更改挂号单的状态* **return** AjaxResult.*toAjax*(**this**.**careService**.visitComplete(regId)); } |

### 修改CareService【doctor-api项目】

|  |
| --- |
| */\*\*  \* 根据病例ID查询病历信息  \** ***@param chId*** *\** ***@return*** *\*/* CareHistory queryCareHistoryByChId(String chId);  */\*\*  \* 保存处方及详情信息  \** ***@param careOrderFormDto*** *\** ***@return*** *\*/* **int** saveCareOrderItem(CareOrderFormDto careOrderFormDto);  */\*\*  \* 根据处方详情ID查询处方详情数据  \** ***@param itemId*** *\** ***@return*** *\*/* CareOrderItem queryCareOrderItemByItemId(String itemId); */\*\*  \* 根据详情ID删除详情信息  \** ***@param itemId*** *\** ***@return*** *\*/* **int** deleteCareOrderItemByItemId(String itemId);  */\*\*  \* 完成就诊  \** ***@param regId*** *\** ***@return*** *\*/* **int** visitComplete(String regId); |

### 修改CareServiceImpl【doctor-service项目】

|  |
| --- |
| @Override **public** CareHistory queryCareHistoryByChId(String chId) {  **return this**.**careHistoryMapper**.selectById(chId); }  @Override **public int** saveCareOrderItem(CareOrderFormDto careOrderFormDto) {  CareOrderDto careOrderDto = careOrderFormDto.getCareOrder();  CareOrder careOrder=**new** CareOrder();  BeanUtil.*copyProperties*(careOrderDto,careOrder);  careOrder.setCreateBy(careOrderDto.getSimpleUser().getUserName());  careOrder.setCreateTime(DateUtil.*date*());  **int** i=**this**.**careOrderMapper**.insert(careOrder);*//保存处方主表* List<CareOrderItemDto> careOrderItems = careOrderFormDto.getCareOrderItems();  *//保存详情数据* **for** (CareOrderItemDto careOrderItemDto : careOrderItems) {  CareOrderItem careOrderItem=**new** CareOrderItem();  BeanUtil.*copyProperties*(careOrderItemDto,careOrderItem);  careOrderItem.setCoId(careOrder.getCoId());  careOrderItem.setCreateTime(DateUtil.*date*());  careOrderItem.setStatus(Constants.***ORDER\_DETAILS\_STATUS\_0***);*//未支付* careOrderItem.setItemId(IdGeneratorSnowflake.*generatorIdWithProfix*(Constants.***ID\_PROFIX\_ITEM***));  **this**.**careOrderItemMapper**.insert(careOrderItem);  }  **return** i; }  @Override **public** CareOrderItem queryCareOrderItemByItemId(String itemId) {  **return this**.**careOrderItemMapper**.selectById(itemId); }   @Override **public int** deleteCareOrderItemByItemId(String itemId) {  *//注意点，如果删除了，要更新careOrder主表的all\_amount* CareOrderItem careOrderItem=**this**.**careOrderItemMapper**.selectById(itemId);  String coId=careOrderItem.getCoId();*//取出主表ID  //删除* **int** i=**this**.**careOrderItemMapper**.deleteById(itemId);   *//再根据coID查询还存在的详情数据* QueryWrapper<CareOrderItem> qw=**new** QueryWrapper<>();  qw.eq(CareOrderItem.***COL\_CO\_ID***,coId);  List<CareOrderItem> careOrderItems=**this**.**careOrderItemMapper**.selectList(qw);  **if**(careOrderItems!=**null**&&careOrderItems.size()>0){  *//重新计算总价格* BigDecimal allAmount=**new** BigDecimal(**"0"**);  **for** (CareOrderItem orderItem : careOrderItems) {  allAmount=allAmount.add(orderItem.getAmount());  }  *//再根据coId查询主表的数据* CareOrder careOrder=**this**.**careOrderMapper**.selectById(coId);  *//更新主表的数据* careOrder.setAllAmount(allAmount);  **this**.**careOrderMapper**.updateById(careOrder);  }**else**{  *//说明没有详情了，直接干掉主表里面的数据* **this**.**careOrderMapper**.deleteById(coId);  }  **return** i; }  @Override **public int** visitComplete(String regId) {  Registration registration=**new** Registration();  registration.setRegId(regId);  registration.setRegStatus(Constants.***REG\_STATUS\_3***);  **return this**.**registrationMapper**.updateById(registration); } |

# 患者管理-病例查询

## 根据患者ID查询所有病历的接口

### 修改PatientController

|  |
| --- |
| @Reference **private** CareService **careService**;  */\*\*  \* 根据患者ID查询患者信息 患者档案信息 历史病例   \*/* @GetMapping(**"getPatientAllMessageByPatientId/{patientId}"**) **public** AjaxResult getPatientAllMessageByPatientId(@PathVariable String patientId){  *//查询病历* List<CareHistory> careHistories=**this**.**careService**.queryCareHistoryByPatientId(patientId);  List<Map<String, Object>> res = **new** ArrayList<>();  **for** (CareHistory careHistory : careHistories) {  Map<String, Object> careHistoryMap = BeanUtil.*beanToMap*(careHistory);  careHistoryMap.put(**"careOrders"**, Collections.***EMPTY\_LIST***);  List<Map<String, Object>> reCareOrders = **new** ArrayList<>();  *//根据病历ID查询处方列表* List<CareOrder> careOrders = **this**.**careService**.queryCareOrdersByChId(careHistory.getChId());  **for** (CareOrder order : careOrders) {  Map<String, Object> careOrder = BeanUtil.*beanToMap*(order);  List<CareOrderItem> careOrderItems = **this**.**careService**.queryCareOrderItemsByCoId(order.getCoId());  careOrder.put(**"careOrderItems"**, careOrderItems);  reCareOrders.add(careOrder);  }  careHistoryMap.put(**"careOrders"**, reCareOrders);  res.add(careHistoryMap);  }  **return** AjaxResult.*success*(res); } |