尚医通

版本：V1.0

# 上传医院接口

参考《尚医通API接口文档.docx》业务接口4.1上传医院

参考《医院接口模拟系统.docx》进行接口测试与数据上传

## 集成mongodb

### 添加依赖

service-hosp模块pom.xml添加依赖

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-data-mongodb</**artifactId**>  </**dependency**> |

### 添加配置

在application.properties文件添加配置

|  |
| --- |
| **spring.data.mongodb.uri**=**mongodb://192.168.44.165:27017/yygh\_hosp** |

说明：改为自己安装mongodb的ip地址

## 添加医院基础类

### 2.1 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.hosp.Hospital

### 2.2 添加Repository

|  |
| --- |
| @Repository  **public interface** HospitalRepository **extends** MongoRepository<Hospital,String> {    } |

### 2.3 添加service接口及实现类

1，添加com.atguigu.yygh.hosp.service.HospitalService接口

|  |
| --- |
| **public interface** HospitalService {    } |

2，添加com.atguigu.yygh.hosp.service.impl.HospitalServiceImpl接口实现

|  |
| --- |
| **package** com.atguigu.yygh.hosp.service.impl;  @Service  **public class** HospitalServiceImpl **implements** HospitalService {  @Autowired  **private** HospitalRepository **hospitalRepository**;  } |

### 2.4 添加controller

添加com.atguigu.yygh.hosp.api.ApiController

|  |
| --- |
| **package** com.atguigu.yygh.hosp.api;  @Api(tags = **"医院管理API接口"**)  @RestController  @RequestMapping(**"/api/hosp"**)  **public class** ApiController {  @Autowired  **private** HospitalService **hospitalService**;  } |

说明：平台对外开发的接口都写在该Controller类

## 上传医院

参考《尚医通API接口文档.doc》业务接口4.1上传医院

医院编号是平台分配的，全局唯一，上传医院接口可以多次调用，如果存在相同编号的为更新操作

### 3.1 接口数据分析

|  |
| --- |
| {  **"hoscode"**: **"1000\_0"**,  **"hosname"**: **"北京协和医院"**,  **"hostype"**: **"1"**,  **"provinceCode"**: **"110000"**,  **"cityCode"**: **"110100"**,  **"districtCode"**: **"110102"**,  **"address"**: **"大望路"**,  **"intro"**: **"北京协和医院是集医疗、教学、科研于一体的大型三级甲等综合医院，是国家卫生计生委...目标而继续努力。"**,  **"route"**: **"东院区乘车路线：106、...更多乘车路线详见须知。"**,  **"logoData"**: **"iVBORw0KGgoAAAA...NSUhEUg=="**,  **"bookingRule"**: {  **"cycle"**: **"1"**,  **"releaseTime"**: **"08:30"**,  **"stopTime"**: **"11:30"**,  **"quitDay"**: **"-1"**,  **"quitTime"**: **"15:30"**,  **"rule"**: [  **"西院区预约号取号地点：西院区门诊楼一层大厅挂号窗口取号"**,  **"东院区预约号取号地点：东院区老门诊楼一层大厅挂号窗口或新门诊楼各楼层挂号/收费窗口取号"**  ]  }  } |

说明：

1. 数据分为医院基本信息与预约规则信息
2. 医院logo转换为base64字符串
3. 预约规则信息属于医院基本信息的一个属性
4. 预约规则rule，以数组形式传递
5. 数据传递过来我们还要验证签名，只允许平台开通的医院可以上传数据，保证数据安全性

### 3.2 添加service接口

1、在HospitalService 类添加接口

|  |
| --- |
| */\*\**  *\* 上传医院信息*  *\** ***@param paramMap***  *\*/*  **void** save(Map<String, Object> paramMap); |

说明：参数使用Map，减少对象封装，有利于签名校验，后续会体验到

2、在HospitalServiceImpl类添加实现

|  |
| --- |
| @Override  **public void** save(Map<String, Object> paramMap) {  ***log***.info(JSONObject.*toJSONString*(paramMap));  Hospital hospital = JSONObject.*parseObject*(JSONObject.*toJSONString*(paramMap),Hospital.**class**);  *//判断是否存在*  Hospital targetHospital = **hospitalRepository**.getHospitalByHoscode(hospital.getHoscode());  **if**(**null** != targetHospital) {  hospital.setStatus(targetHospital.getStatus());  hospital.setCreateTime(targetHospital.getCreateTime());  hospital.setUpdateTime(**new** Date());  hospital.setIsDeleted(0);  **hospitalRepository**.save(hospital);  } **else** {  *//0：未上线 1：已上线*  hospital.setStatus(0);  hospital.setCreateTime(**new** Date());  hospital.setUpdateTime(**new** Date());  hospital.setIsDeleted(0);  **hospitalRepository**.save(hospital);  }  } |

说明：

Hospital hospital = JSONObject.parseObject(JSONObject.toJSONString(paramMap),Hospital.class);

Map转换为Hospital对象时，预约规则bookingRule为一个对象属性，rule为一个数组属性，因此在转换时我们要重新对应的set方法，不然转换不会成功

|  |
| --- |
| **public class** Hospital **extends** BaseMongoEntity {  **private static final long *serialVersionUID*** = 1L;  @ApiModelProperty(value = **"医院编号"**)  **private** String **hoscode**;  ...  *//预约规则*  @ApiModelProperty(value = **"预约规则"**)  **private** BookingRule **bookingRule**;  **public void** setBookingRule(String bookingRule) {  **this**.**bookingRule** = JSONObject.*parseObject*(bookingRule, BookingRule.**class**);  }  } |

|  |
| --- |
| **public class** BookingRule {  @ApiModelProperty(value = **"预约周期"**)  **private** Integer **cycle**;  ...  @ApiModelProperty(value = **"预约规则"**)  **private** List<String>**rule**;  */\*\**  *\**  *\** ***@param rule***  *\*/*  **public void** setRule(String rule) {  **if**(!StringUtils.*isEmpty*(rule)) {  **this**.**rule** = JSONArray.*parseArray*(rule, String.**class**);  }  }  } |

### 3.3 添加repository接口

在HospitalRepository类添加接口

|  |
| --- |
| Hospital getHospitalByHoscode(String hoscode); |

### 3.4 添加controller接口

在ApiController类添加接口

|  |
| --- |
| @ApiOperation(value = **"上传医院"**)  @PostMapping(**"saveHospital"**)  **public** Result saveHospital(HttpServletRequest request) {  Map<String, Object> paramMap = HttpRequestHelper.*switchMap*(request.getParameterMap());  **hospitalService**.save(paramMap);  **return** Result.*ok*();  } |

### 3.5 添加帮助类

在service-util模块添加HttpRequestHelper帮助类

|  |
| --- |
| **package** com.atguigu.yygh.common.helper;  @Slf4j  **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;  }  } |

### 3.6 使用Swagger2测试上传接口

略

## 参数签名

参考《尚医通API接口文档.doc》业务接口3.1传参说明

### 4.1 封装签名方法

在service-util模块HttpRequestHelper类添加方法

|  |
| --- |
| **public static void** main(String[] args) {  Map<String, Object> paramMap = **new** HashMap<>();  paramMap.put(**"d"**, **"4"**);  paramMap.put(**"b"**, **"2"**);  paramMap.put(**"c"**, **"3"**);  paramMap.put(**"a"**, **"1"**);  paramMap.put(**"timestamp"**, *getTimestamp*());  ***log***.info(*getSign*(paramMap, **"111111111"**));  }  */\*\**  *\* 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();  } |

### 4.2 上传医院添加签名校验

我们在医院设置的时候，为每个医院生成了医院编码与签名key，因此我在验证签名时要根据医院编码去动态获取签名key，然后再做签名校验

#### 4.2 .1 添加获取签名key接口

1，在HospitalSetService类添加接口

|  |
| --- |
| */\*\**  *\* 获取签名key*  *\** ***@param hoscode***  *\** ***@return***  *\*/*  String getSignKey(String hoscode); |

2，在HospitalSetServiceImpl类实现接口

|  |
| --- |
| @Override  **public** String getSignKey(String hoscode) {  HospitalSet hospitalSet = **this**.getByHoscode(hoscode);  **if**(**null** == hospitalSet) {  **throw new** YyghException(ResultCodeEnum.***HOSPITAL\_OPEN***);  }  **if**(hospitalSet.getStatus().intValue() == 0) {  **throw new** YyghException(ResultCodeEnum.***HOSPITAL\_LOCK***);  }  **return** hospitalSet.getSignKey();  }  */\*\**  *\* 根据hoscode获取医院设置*  *\** ***@param hoscode***  *\** ***@return***  *\*/*  **private** HospitalSet getByHoscode(String hoscode) {  **return hospitalSetMapper**.selectOne(**new** QueryWrapper<HospitalSet>().eq(**"hoscode"**, hoscode));  } |

#### 4.2.2 修改ApiController类上传医院接口

修改ApiController类上传医院接口

|  |
| --- |
| @ApiOperation(value = **"上传医院"**)  @PostMapping(**"saveHospital"**)  **public** Result saveHospital(HttpServletRequest request) {  Map<String, Object> paramMap = HttpRequestHelper.*switchMap*(request.getParameterMap());  *//必须参数校验 略*  String hoscode = (String)paramMap.get(**"hoscode"**);  **if**(StringUtils.*isEmpty*(hoscode)) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//签名校验*  **if**(!HttpRequestHelper.*isSignEquals*(paramMap, **hospitalSetService**.getSignKey(hoscode))) {  **throw new** YyghException(ResultCodeEnum.***SIGN\_ERROR***);  }  **hospitalService**.save(paramMap);  **return** Result.*ok*();  } |

## 图片base64编码

### 5.1 图片base64说明

图片的base64编码就是可以将一张图片数据编码成一串字符串，使用该字符串代替图像地址url

在前端页面中常见的base64图片的引入方式：

<img src="data:image/png;base64,iVBORw0..>

1. 优点
2. base64格式的图片是文本格式，占用内存小，转换后的大小比例大概为1/3，降低了资源服务器的消耗；

（2）网页中使用base64格式的图片时，不用再请求服务器调用图片资源，减少了服务器访问次数。

2. 缺点

（1）base64格式的文本内容较多，存储在数据库中增大了数据库服务器的压力；

（2）网页加载图片虽然不用访问服务器了，但因为base64格式的内容太多，所以加载网页的速度会降低，可能会影响用户的体验。

说明：医院logo图片小，因此上传医院logo是可以使用base64格式保存

### 5.2 图片base64工具类

在common-util模块添加工具类

添加com.atguigu.yygh.common.util.ImageBase64Util类

|  |
| --- |
| **package** com.atguigu.yygh.common.util;  **import** org.apache.commons.codec.binary.Base64;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.InputStream;  **public class** ImageBase64Util {  **public static void** main(String[] args) {  String imageFile= **"D:\\yygh\_work\\xh.png"**;*// 待处理的图片*  System.***out***.println(*getImageString*(imageFile));  }  **public static** String getImageString(String imageFile){  InputStream is = **null**;  **try** {  **byte**[] data = **null**;  is = **new** FileInputStream(**new** File(imageFile));  data = **new byte**[is.available()];  is.read(data);  **return new** String(Base64.*encodeBase64*(data));  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **if** (**null** != is) {  **try** {  is.close();  is = **null**;  } **catch** (Exception e) {  e.printStackTrace();  }  }  }  **return ""**;  }  } |

### 5.3 上传医院接口修正

图片转换为base64字符串时，该字符串中包含大量的加号“+”，服务器在解析数据时会把加号当成连接符，转换为空格，因此我们要做一下特殊处理

修改ApiController类上传接口

|  |
| --- |
| @ApiOperation(value = **"上传医院"**)  @PostMapping(**"saveHospital"**)  **public** Result saveHospital(HttpServletRequest request) {  Map<String, Object> paramMap = HttpRequestHelper.*switchMap*(request.getParameterMap());  *//必须参数校验 略*  String hoscode = (String)paramMap.get(**"hoscode"**);  **if**(StringUtils.*isEmpty*(hoscode)) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//传输过程中“+”转换为了“ ”，因此我们要转换回来*  String logoDataString = (String)paramMap.get(**"logoData"**);  **if**(!StringUtils.*isEmpty*(logoDataString)) {  String logoData = logoDataString.replaceAll(**""**, **"+"**);  paramMap.put(**"logoData"**, logoData);  }  *//签名校验*  **if**(!HttpRequestHelper.*isSignEquals*(paramMap, **hospitalSetService**.getSignKey(hoscode))) {  **throw new** YyghException(ResultCodeEnum.***SIGN\_ERROR***);  }  **hospitalService**.save(paramMap);  **return** Result.*ok*();  } |

## 6、集成测试

参考《医院接口模拟系统.docx》进行接口测试与数据上传，后续不做说明，需要测试时即可使用

# 查询医院接口

参考《尚医通API接口文档.docx》业务接口4.4查询医院

## 1、添加service接口

1，在HospitalService 类添加接口

|  |
| --- |
| */\*\**  *\* 查询医院*  *\** ***@param hoscode***  *\** ***@return***  *\*/*  Hospital getByHoscode(String hoscode); |

2，在HospitalServiceImpl类添加实现

|  |
| --- |
| @Override  **public** Hospital getByHoscode(String hoscode) {  **return hospitalRepository**.getHospitalByHoscode(hoscode);  } |

## 添加controller接口

在ApiController类添加接口

|  |
| --- |
| @ApiOperation(value = **"获取医院信息"**)  @PostMapping(**"hospital/show"**)  **public** Result hospital(HttpServletRequest request) {  Map<String, Object> paramMap = HttpRequestHelper.*switchMap*(request.getParameterMap());  *//必须参数校验 略*  String hoscode = (String)paramMap.get(**"hoscode"**);  **if**(StringUtils.*isEmpty*(hoscode)) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//签名校验*  **if**(!HttpRequestHelper.*isSignEquals*(paramMap, **hospitalSetService**.getSignKey(hoscode))) {  **throw new** YyghException(ResultCodeEnum.***SIGN\_ERROR***);  }  **return** Result.*ok*(**hospitalService**.getByHoscode((String)paramMap.get(**"hoscode"**)));  } |

# 上传科室接口

参考《尚医通API接口文档.docx》业务接口4.2上传科室

## 1、添加科室基础类

### 1.1 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.hosp.Department

### 1.2 添加repository

添加com.atguigu.yygh.hosp.repository.DepartmentRepository

|  |
| --- |
| **package** com.atguigu.yygh.hosp.repository;  @Repository  **public interface** DepartmentRepository **extends** MongoRepository<Department,String> {  } |

### 1.3 添加service接口及实现类

1，添加com.atguigu.yygh.hosp.service.DepartmentService接口

|  |
| --- |
| **package** com.atguigu.yygh.hosp.service;  **public interface** DepartmentService {  } |

2，添加com.atguigu.yygh.hosp.service.impl.DepartmentServiceImpl接口实现

|  |
| --- |
| **package** com.atguigu.yygh.hosp.service.impl;  @Service  @Slf4j  **public class** DepartmentServiceImpl **implements** DepartmentService {  @Autowired  **private** DepartmentRepository department**Repository**;  } |

## 2、上传科室

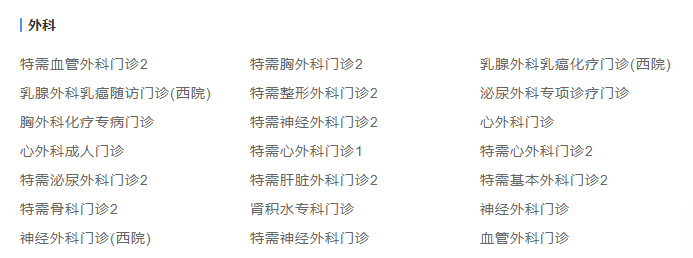
参考《尚医通API接口文档.doc》业务接口4.2上传科室

医院编号是平台分配的，全局唯一，科室编号为医院自己的编号，相对医院唯一，上传科室接口可以多次调用，如果医院编号与科室编号组合唯一为更新操作

### 2.1 接口数据分析

|  |
| --- |
| {  **"hoscode"**: **"1000\_0"**,  **"depcode"**: **"200050923"**,  **"depname"**: **"门诊部核酸检测门诊(东院)"**,  **"intro"**: **"门诊部核酸检测门诊(东院)"**,  **"bigcode"**: **"44f162029abb45f9ff0a5f743da0650d"**,  **"bigname"**: **"体检科"**  } |

说明：一个大科室下可以有多个小科室，如图：



### 2.2 添加service接口

1，在DepartmentService 类添加接口

|  |
| --- |
| */\*\**  *\* 上传科室信息*  *\** ***@param paramMap***  *\*/*  **void** save(Map<String, Object> paramMap); |

说明：参数使用Map，减少对象封装，有利于签名校验，后续会体验到

2，在DepartmentServiceImpl类添加实现

|  |
| --- |
| @Override  **public void** save(Map<String, Object> paramMap) {  Department department = JSONObject.*parseObject*(JSONObject.*toJSONString*(paramMap), Department.**class**);  Department targetDepartment = **departmentRepository**.getDepartmentByHoscodeAndDepcode(department.getHoscode(), department.getDepcode());  **if**(**null** != targetDepartment) {  *//copy不为null的值，该方法为自定义方法*  BeanUtils.*copyBean*(department, targetDepartment, Department.**class**);  **departmentRepository**.save(targetDepartment);  } **else** {  department.setCreateTime(**new** Date());  department.setUpdateTime(**new** Date());  department.setIsDeleted(0);  **departmentRepository**.save(department);  }  } |

### 2.3 添加repository接口

在DepartmentRepository添加方法

|  |
| --- |
| Department getDepartmentByHoscodeAndDepcode(String hoscode, String depcode); |

### 2.4 添加controller接口

在ApiController类添加接口

|  |
| --- |
| @Autowired  **private** DepartmentService **departmentService**; |
| @ApiOperation(value = **"上传科室"**)  @PostMapping(**"saveDepartment"**)  **public** Result saveDepartment(HttpServletRequest request) {  Map<String, Object> paramMap = HttpRequestHelper.*switchMap*(request.getParameterMap());  *//必须参数校验 略*  String hoscode = (String)paramMap.get(**"hoscode"**);  **if**(StringUtils.*isEmpty*(hoscode)) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//签名校验*  **if**(!HttpRequestHelper.*isSignEquals*(paramMap, **hospitalSetService**.getSignKey(hoscode))) {  **throw new** YyghException(ResultCodeEnum.***SIGN\_ERROR***);  }  **departmentService**.save(paramMap);  **return** Result.*ok*();  } |

# 查询科室接口

参考《尚医通API接口文档.docx》业务接口4.5查询医院

一个医院有多个科室，因此我们采取分页查询方式

## 1、添加service接口

1，在DepartmentService 类添加接口

|  |
| --- |
| */\*\**  *\* 分页查询*  *\** ***@param page*** *当前页码*  *\** ***@param limit*** *每页记录数*  *\** ***@param departmentQueryVo*** *查询条件*  *\** ***@return***  *\*/*  Page<Department> selectPage(Integer page, Integer limit, DepartmentQueryVo departmentQueryVo); |

2，在DepartmentServiceImpl类添加实现

|  |
| --- |
| @Override  **public** Page<Department> selectPage(Integer page, Integer limit, DepartmentQueryVo departmentQueryVo) {  Sort sort = Sort.*by*(Sort.Direction.***DESC***, **"createTime"**);  *//0为第一页*  Pageable pageable = PageRequest.*of*(page-1, limit, sort);  Department department = **new** Department();  BeanUtils.*copyProperties*(departmentQueryVo, department);  department.setIsDeleted(0);  *//创建匹配器，即如何使用查询条件*  ExampleMatcher matcher = ExampleMatcher.*matching*() *//构建对象*  .withStringMatcher(ExampleMatcher.StringMatcher.***CONTAINING***) *//改变默认字符串匹配方式：模糊查询*  .withIgnoreCase(**true**); *//改变默认大小写忽略方式：忽略大小写*  *//创建实例*  Example<Department> example = Example.*of*(department, matcher);  Page<Department> pages = **departmentRepository**.findAll(example, pageable);  **return** pages;  } |

## 2、添加controller接口

在ApiController类添加接口

|  |
| --- |
| @ApiOperation(value = **"获取分页列表"**)  @PostMapping(**"department/list"**)  **public** Result department(HttpServletRequest request) {  Map<String, Object> paramMap = HttpRequestHelper.*switchMap*(request.getParameterMap());  *//必须参数校验 略*  String hoscode = (String)paramMap.get(**"hoscode"**);  *//非必填*  String depcode = (String)paramMap.get(**"depcode"**);  **int** page = StringUtils.*isEmpty*(paramMap.get(**"page"**)) ? 1 : Integer.*parseInt*((String)paramMap.get(**"page"**));  **int** limit = StringUtils.*isEmpty*(paramMap.get(**"limit"**)) ? 10 : Integer.*parseInt*((String)paramMap.get(**"limit"**));  **if**(StringUtils.*isEmpty*(hoscode)) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//签名校验*  **if**(!HttpRequestHelper.*isSignEquals*(paramMap, **hospitalSetService**.getSignKey(hoscode))) {  **throw new** YyghException(ResultCodeEnum.***SIGN\_ERROR***);  }  DepartmentQueryVo departmentQueryVo = **new** DepartmentQueryVo();  departmentQueryVo.setHoscode(hoscode);  departmentQueryVo.setDepcode(depcode);  Page<Department> pageModel = **departmentService**.selectPage(page, limit, departmentQueryVo);  **return** Result.*ok*(pageModel);  } |

# 删除科室接口

参考《尚医通API接口文档.docx》业务接口4.7删除科室

根据医院编号与科室编号删除科室

## 1、添加service接口

1，在DepartmentService 类添加接口

|  |
| --- |
| */\*\**  *\* 删除科室*  *\** ***@param hoscode***  *\** ***@param depcode***  *\*/*  **void** remove(String hoscode, String depcode); |

2，在DepartmentServiceImpl类添加实现

|  |
| --- |
| @Override  **public void** remove(String hoscode, String depcode) {  Department department = **departmentRepository**.getDepartmentByHoscodeAndDepcode(hoscode, depcode);  **if**(**null** != department) {  *//departmentRepository.delete(department);*  **departmentRepository**.deleteById(department.getId());  }  } |

## 2、添加controller接口

在ApiController类添加接口

|  |
| --- |
| @ApiOperation(value = **"删除科室"**)  @PostMapping(**"department/remove"**)  **public** Result removeDepartment(HttpServletRequest request) {  Map<String, Object> paramMap = HttpRequestHelper.*switchMap*(request.getParameterMap());  *//必须参数校验 略*  String hoscode = (String)paramMap.get(**"hoscode"**);  *//必填*  String depcode = (String)paramMap.get(**"depcode"**);  **if**(StringUtils.*isEmpty*(hoscode)) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//签名校验*  **if**(!HttpRequestHelper.*isSignEquals*(paramMap, **hospitalSetService**.getSignKey(hoscode))) {  **throw new** YyghException(ResultCodeEnum.***SIGN\_ERROR***);  }  **departmentService**.remove(hoscode, depcode);  **return** Result.*ok*();  } |

# 上传排班接口

参考《尚医通API接口文档.docx》业务接口4.3上传排班

## 1、添加排班基础类

### 1.1 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.hosp.Schedule

### 1.2 添加repository

添加com.atguigu.yygh.hosp.repository.ScheduleRepository

|  |
| --- |
| **package** com.atguigu.yygh.hosp.repository;  @Repository  **public interface** ScheduleRepository **extends** MongoRepository<Schedule,String> {  } |

### 1.3 添加service接口及实现类

1，添加com.atguigu.yygh.hosp.service.ScheduleService接口

|  |
| --- |
| **package** com.atguigu.yygh.hosp.service;  **public interface** ScheduleService {  } |

2，添加com.atguigu.yygh.hosp.service.impl.ScheduleServiceImpl接口实现

|  |
| --- |
| **package** com.atguigu.yygh.hosp.service.impl;  @Service  @Slf4j  **public class** ScheduleServiceImpl **implements** ScheduleService {  @Autowired  **private** ScheduleRepository schedule**Repository**;  } |

## 2、上传排班

参考《尚医通API接口文档.doc》业务接口4.3上传排班

医院编号是平台分配的，全局唯一，排班编号为医院自己的编号，相对医院唯一，上传排班接口可以多次调用，如果医院编号与排班编号组合唯一为更新操作

### 2.1 接口数据分析

|  |
| --- |
| {  **"hoscode"**: **"1000\_0"**,  **"depcode"**: **"200040878"**,  **"title"**: **"医师"**,  **"docname"**: **""**,  **"skill"**: **"内分泌科常见病。"**,  **"workDate"**: **"2020-06-22"**,  **"workTime"**: 0,  **"reservedNumber"**: 33,  **"availableNumber"**: 22,  **"amount"**: **"100"**,  **"status"**: 1,  **"hosScheduleId"**: **"1"**  } |

### 2.2 添加service接口

1，在ScheduleService 类添加接口

|  |
| --- |
| */\*\**  *\* 上传排班信息*  *\** ***@param paramMap***  *\*/*  **void** save(Map<String, Object> paramMap); |

说明：参数使用Map，减少对象封装，有利于签名校验，后续会体验到

2，在ScheduleServiceImpl类添加实现

|  |
| --- |
| @Override  **public void** save(Map<String, Object> paramMap) {  Schedule schedule = JSONObject.*parseObject*(JSONObject.*toJSONString*(paramMap), Schedule.**class**);  Schedule targetSchedule = **scheduleRepository**.getScheduleByHoscodeAndHosScheduleId(schedule.getHoscode(), schedule.getHosScheduleId());  **if**(**null** != targetSchedule) {  *//值copy不为null的值，该方法为自定义方法*  BeanUtils.*copyBean*(schedule, targetSchedule, Schedule.**class**);  **scheduleRepository**.save(targetSchedule);  } **else** {  schedule.setCreateTime(**new** Date());  schedule.setUpdateTime(**new** Date());  schedule.setIsDeleted(0);  **scheduleRepository**.save(schedule);  }  } |

### 2.3 添加repository接口

在ScheduleRepository添加方法

|  |
| --- |
| Schedule getScheduleByHoscodeAndHosScheduleId(String hoscode, String hosScheduleId); |

### 2.4 添加controller接口

在ApiController类添加接口

|  |
| --- |
| @Autowired  **private** ScheduleService **scheduleService**; |
| @ApiOperation(value = **"上传排班"**)  @PostMapping(**"saveSchedule"**)  **public** Result saveSchedule(HttpServletRequest request) {  Map<String, Object> paramMap = HttpRequestHelper.*switchMap*(request.getParameterMap());  *//必须参数校验 略*  String hoscode = (String)paramMap.get(**"hoscode"**);  **if**(StringUtils.*isEmpty*(hoscode)) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//签名校验*  **if**(!HttpRequestHelper.*isSignEquals*(paramMap, **hospitalSetService**.getSignKey(hoscode))) {  **throw new** YyghException(ResultCodeEnum.***SIGN\_ERROR***);  }  **scheduleService**.save(paramMap);  **return** Result.*ok*();  } |

# 查询排班接口

参考《尚医通API接口文档.docx》业务接口4.6查询医院

一个科室有多个科室，因此我们采取分页查询方式

## 1、添加service接口

1，在ScheduleService 类添加接口

|  |
| --- |
| */\*\**  *\* 分页查询*  *\** ***@param page*** *当前页码*  *\** ***@param limit*** *每页记录数*  *\** ***@param scheduleQueryVo*** *查询条件*  *\** ***@return***  *\*/*  Page<Schedule> selectPage(Integer page, Integer limit, ScheduleQueryVo scheduleQueryVo); |

2，在ScheduleServiceImpl类添加实现

|  |
| --- |
| @Override  **public** Page<Schedule> selectPage(Integer page, Integer limit, ScheduleQueryVo scheduleQueryVo) {  Sort sort = Sort.*by*(Sort.Direction.***DESC***, **"createTime"**);  *//0为第一页*  Pageable pageable = PageRequest.*of*(page-1, limit, sort);  Schedule schedule = **new** Schedule();  BeanUtils.*copyProperties*(scheduleQueryVo, schedule);  schedule.setIsDeleted(0);  *//创建匹配器，即如何使用查询条件*  ExampleMatcher matcher = ExampleMatcher.*matching*() *//构建对象*  .withStringMatcher(ExampleMatcher.StringMatcher.***CONTAINING***) *//改变默认字符串匹配方式：模糊查询*  .withIgnoreCase(**true**); *//改变默认大小写忽略方式：忽略大小写*  *//创建实例*  Example<Schedule> example = Example.*of*(schedule, matcher);  Page<Schedule> pages = **scheduleRepository**.findAll(example, pageable);  **return** pages;  } |

## 2、添加controller接口

在ApiController类添加接口

|  |
| --- |
| @ApiOperation(value = **"获取排班分页列表"**)  @PostMapping(**"schedule/list"**)  **public** Result schedule(HttpServletRequest request) {  Map<String, Object> paramMap = HttpRequestHelper.*switchMap*(request.getParameterMap());  *//必须参数校验 略*  String hoscode = (String)paramMap.get(**"hoscode"**);  *//非必填*  String depcode = (String)paramMap.get(**"depcode"**);  **int** page = StringUtils.*isEmpty*(paramMap.get(**"page"**)) ? 1 : Integer.*parseInt*((String)paramMap.get(**"page"**));  **int** limit = StringUtils.*isEmpty*(paramMap.get(**"limit"**)) ? 10 : Integer.*parseInt*((String)paramMap.get(**"limit"**));  **if**(StringUtils.*isEmpty*(hoscode)) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//签名校验*  **if**(!HttpRequestHelper.*isSignEquals*(paramMap, **hospitalSetService**.getSignKey(hoscode))) {  **throw new** YyghException(ResultCodeEnum.***SIGN\_ERROR***);  }  ScheduleQueryVo scheduleQueryVo = **new** ScheduleQueryVo();  scheduleQueryVo.setHoscode(hoscode);  scheduleQueryVo.setDepcode(depcode);  Page<Schedule> pageModel = **scheduleService**.selectPage(page , limit, scheduleQueryVo);  **return** Result.*ok*(pageModel);  } |

# 删除排班接口

参考《尚医通API接口文档.docx》业务接口4.8删除科室

根据医院编号与排班编号删除科室

## 1、添加service接口

1，在ScheduleService 类添加接口

|  |
| --- |
| */\*\**  *\* 删除科室*  *\** ***@param hoscode***  *\** ***@param*** hosScheduleId  *\*/*  **void** remove(String hoscode, String hosScheduleId); |

2，在ScheduleServiceImpl类添加实现

|  |
| --- |
| @Override  **public void** remove(String hoscode, String hosScheduleId) {  Schedule schedule = **scheduleRepository**.getScheduleByHoscodeAndHosScheduleId(hoscode, hosScheduleId);  **if**(**null** != schedule) {  **scheduleRepository**.deleteById(schedule.getId());  }  } |

## 2、添加controller接口

在ApiController类添加接口

|  |
| --- |
| @ApiOperation(value = **"删除科室"**)  @PostMapping(**"schedule/remove"**)  **public** Result removeSchedule(HttpServletRequest request) {  Map<String, Object> paramMap = HttpRequestHelper.*switchMap*(request.getParameterMap());  *//必须参数校验 略*  String hoscode = (String)paramMap.get(**"hoscode"**);  *//必填*  String hosScheduleId = (String)paramMap.get(**"hosScheduleId"**);  **if**(StringUtils.*isEmpty*(hoscode)) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  *//签名校验*  **if**(!HttpRequestHelper.*isSignEquals*(paramMap, **hospitalSetService**.getSignKey(hoscode))) {  **throw new** YyghException(ResultCodeEnum.***SIGN\_ERROR***);  }  **scheduleService**.remove(hoscode, hosScheduleId);  **return** Result.*ok*();  } |