Spring Boot: Yeb(backend)

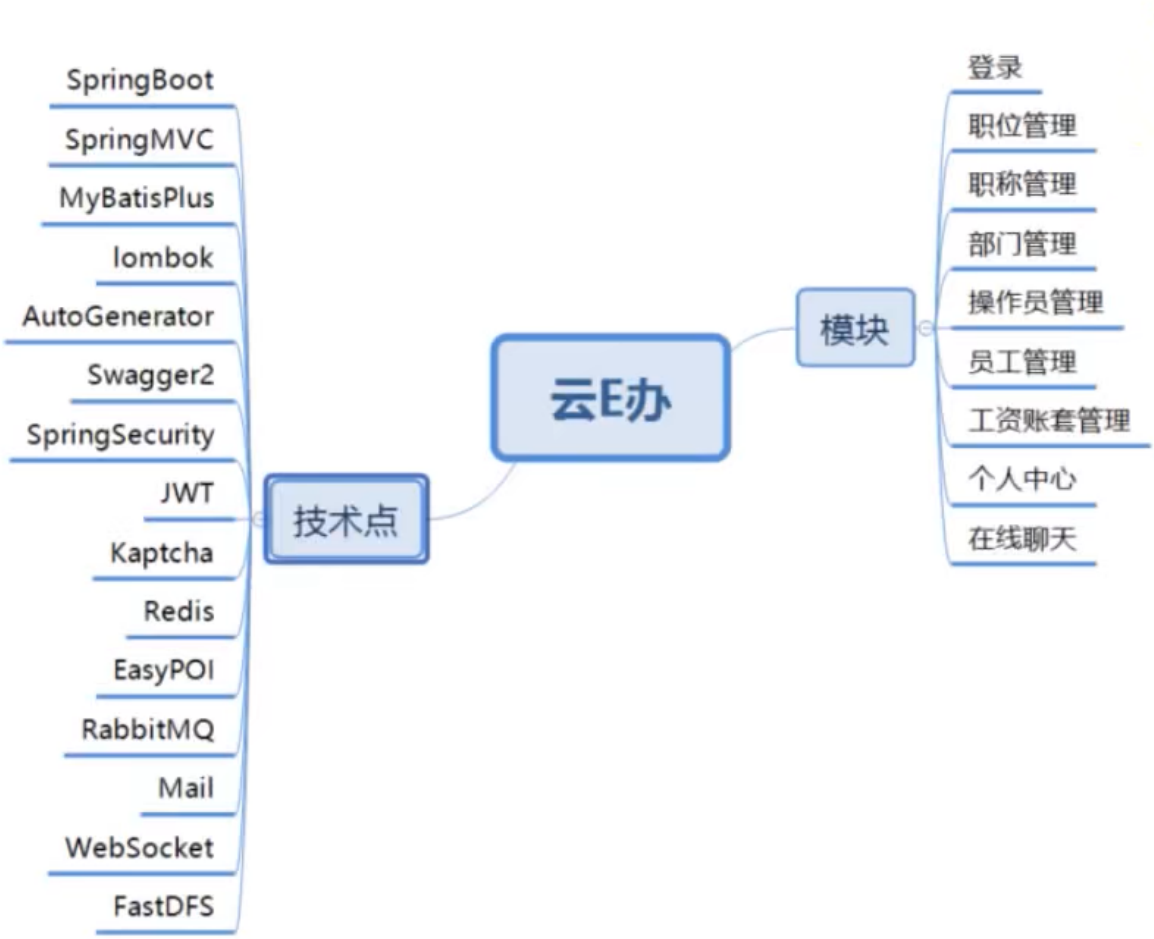
https://blog.csdn.net/m0_73813319/article/details/127265246

> ## SpringBoot2.6.x集成swagger: Failed to start bean ‘documentationPluginsBootstrapper问题解决
>
> https://cloud.tencent.com/developer/article/1975750

1. 项目介绍

本项目目的是实现中小型企业的在线办公系统，云E办在线办公系统是一个用来管理日常的办公事务的一个系统，他能够管的内容有：日常的各种流程审批，新闻，通知，公告，文件信息，财务，人事，费用，资产，行政，项目，移动办公等等。它的作用就是通过软件的方式，方便管理，更加简单，更加扁平。更加高效，更加规范，能够提高整体的管理运营水平。

本项目在技术方面采用最主流的前后端分离开发模式，使用业界最流行、社区非常活跃的开源框架Spring Boot来构建后端，旨在实现云E办在线办公系统。包括职位管理、职称管理、部门管理、员工管理、工资管理、在线聊天等模块。项目中还会使用业界主流的第三方组件扩展大家的知识面和技能池。



- Swagger2：接口文档
- SpringSecurity：安全框架
- JWT：令牌
- Kaptcha：图形验证码
- Redis：缓存器
- EasyPOI：文档导入导出
- RabbitMQ：消息队列。异步处理
- Mail：发送邮件
- WebSocket：在线聊天功能
- FastDFS：文件服务器，静态资源相应的文件。

2. 数据库

2.1 创建数据库

```
# 使用容器创建mysql数据库
$ docker run -d -p 3306:3306 --name mysql -v /mysqldata/mysql/log:/var/log/mysql  -v
/mysqldata/mysql/data:/var/lib/mysql  -v /mysqldata/mysql/conf:/etc/mysql -e
MYSQL_ROOT_PASSWORD=password   mysql:5.7

$ docker exec -it mysql-master /bin/bash
# mysql -u root -p
# create databases yeb;
# show databases;
# use yeb
```

2.2 初始化数据库

```
/*
 Navicat Premium Data Transfer

 Source Server         : localhost
 Source Server Type    : MySQL
 Source Server Version : 80018
 Source Host           : localhost:3306
 Source Schema         : voa2

 Target Server Type    : MySQL
 Target Server Version : 80018
 File Encoding         : 65001

 Date: 09/05/2020 09:52:59

 all table :
t_admin,t_admin_role,t_appraise,t_department,t_employee,t_employee_ec,t_employee_remov
e,t_employee_train,t_joblevel,t_mail_log,t_menu,t_menu_role,t_nation,t_oplog,t_politic
s_status,t_position,t_role,t_salary,t_salary_adjust,t_sys_msg,t_sys_msg_content
*/

SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;

-- ----------------------------
-- Table structure for t_admin
-- ----------------------------
DROP TABLE IF EXISTS `t_admin`;
CREATE TABLE `t_admin`  (
                            `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                            `name` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '姓名',
                            `phone` char(11) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '手机号码',
                            `telephone` varchar(16) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '住宅电话',
                            `address` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '联系地址',
                            `enabled` tinyint(1) NULL DEFAULT 1 COMMENT '是否启用',
                            `username` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '用户名',
                            `password` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '密码',
                            `userFace` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '用户头像',
                            `remark` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '备注',
                           PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 15 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_bin
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_admin
-- ----------------------------
```

```
INSERT INTO `t_admin` VALUES (1, '系统管理员', '13812361398', '71937538', '香港特别行政区强县长寿柳州路p座123', 1, 'admin',
'$2a$10$ogvUqZZAxrBwrmVI/e7.SuFYyx8my8d.9zJ6bs9lPKWvbD9eefyCe',
'http://192.168.10.100:8888/group1/M00/00/00/wKgKZF6oHzuAXnw9AABaLsrkrQQ148.jpg',
NULL);
INSERT INTO `t_admin` VALUES (2, '何淑华', '18875971675', '41413109', '河北省秀荣市萧山长沙街p座 737268', 0, 'taomeng',
'$2a$10$oE39aG10kB/rFu2vQeCJTu/V/v4n6DRR0f8WyXRiAYvBpmadoOBE.',
'https://timgsa.baidu.com/timg?
image&quality=80&size=b9999_10000&sec=1585830947922&di=60b35821fb9112d0aad6915efe982c8d&imgtype=0&src=http%3A%2F%2Fb-
ssl.duitang.com%2Fuploads%2Fitem%2F201703%2F26%2F20170326161532_aGteC.jpeg', NULL);
INSERT INTO `t_admin` VALUES (3, '安淑华123', '14588110811', '50603155', '山东省凤英县长寿银川街l座', 1, 'naqiao', '$2a$10$oE39aG10kB/rFu2vQeCJTu/V/v4n6DRR0f8WyXRiAYvBpmadoOBE.',
'https://timgsa.baidu.com/timg?
image&quality=80&size=b9999_10000&sec=1517070040185&di=be0375e0c3db6c311b837b28c208f318&imgtype=0&src=http%3A%2F%2Fimg2.soyoung.com%2Fpost%2F20150213%2F6%2F20150213141918532.jpg', NULL);
INSERT INTO `t_admin` VALUES (4, '林宇', '15761248727', '25546253', '宁夏回族自治区帆市翔安昆明路b座 672985', 0, 'leisu',
'$2a$10$oE39aG10kB/rFu2vQeCJTu/V/v4n6DRR0f8WyXRiAYvBpmadoOBE.',
'https://timgsa.baidu.com/timg?
image&quality=80&size=b9999_10000&sec=1585830829995&di=0fc5f8313a734b401d20a57bc9bdd421&imgtype=0&src=http%3A%2F%2Fpic4.zhimg.com%2F50%2Fv2-
7fece9a613445edb78271216c8c20c6d_hd.jpg', NULL);
INSERT INTO `t_admin` VALUES (5, '武军', '18030710396', '27523842', '宁夏回族自治区秀兰县涪城邯郸路t座 618651', 0, 'hanli',
'$2a$10$oE39aG10kB/rFu2vQeCJTu/V/v4n6DRR0f8WyXRiAYvBpmadoOBE.',
'https://timgsa.baidu.com/timg?
image&quality=80&size=b9999_10000&sec=1585830877372&di=9ae7236e73ff24c756ac30722b6e84b1&imgtype=0&src=http%3A%2F%2Fb-
ssl.duitang.com%2Fuploads%2Fitem%2F201704%2F10%2F20170410095843_SEvMy.thumb.700_0.jpeg
', NULL);

-- ----------------------------
-- Table structure for t_admin_role
-- ----------------------------
DROP TABLE IF EXISTS `t_admin_role`;
CREATE TABLE `t_admin_role`  (
                                 `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                                 `adminId` int(11) NULL DEFAULT NULL COMMENT '用户id',
                                 `rid` int(11) NULL DEFAULT NULL COMMENT '权限id',
                                 PRIMARY KEY (`id`) USING BTREE,
                                 INDEX `rid`(`rid`) USING BTREE,
                                 INDEX `adminId`(`adminId`) USING BTREE,
                                 CONSTRAINT `t_admin_role_ibfk_1` FOREIGN KEY
(`adminId`) REFERENCES `t_admin` (`id`) ON DELETE CASCADE ON UPDATE RESTRICT,
                                 CONSTRAINT `t_admin_role_ibfk_2` FOREIGN KEY (`rid`)
REFERENCES `t_role` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB AUTO_INCREMENT = 68 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_bin
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_admin_role
-- ----------------------------
INSERT INTO `t_admin_role` VALUES (2, 5, 4);
INSERT INTO `t_admin_role` VALUES (3, 5, 3);
INSERT INTO `t_admin_role` VALUES (4, 5, 2);
```

```
INSERT INTO `t_admin_role` VALUES (5, 4, 3);
INSERT INTO `t_admin_role` VALUES (6, 4, 2);
INSERT INTO `t_admin_role` VALUES (7, 4, 4);
INSERT INTO `t_admin_role` VALUES (8, 4, 5);
INSERT INTO `t_admin_role` VALUES (16, 1, 6);
INSERT INTO `t_admin_role` VALUES (55, 3, 3);
INSERT INTO `t_admin_role` VALUES (56, 3, 4);
INSERT INTO `t_admin_role` VALUES (67, 2, 3);
INSERT INTO `t_admin_role` VALUES (68, 2, 4);

-- ----------------------------
-- Table structure for t_appraise
-- ----------------------------
DROP TABLE IF EXISTS `t_appraise`;
CREATE TABLE `t_appraise`  (
                               `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                               `eid` int(11) NULL DEFAULT NULL COMMENT '员工id',
                               `appDate` date NULL DEFAULT NULL COMMENT '考评日期',
                               `appResult` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '考评结果',
                               `appContent` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '考评内容',
                               `remark` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '备注',
                              PRIMARY KEY (`id`) USING BTREE,
                              INDEX `eid`(`eid`) USING BTREE,
                              CONSTRAINT `t_appraise_ibfk_1` FOREIGN KEY (`eid`)
REFERENCES `t_employee` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Dynamic;

-- ----------------------------
-- Table structure for t_department
-- ----------------------------
DROP TABLE IF EXISTS `t_department`;
CREATE TABLE `t_department`  (
                                 `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                                 `name` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '部门名称',
                                 `parentId` int(11) NULL DEFAULT NULL COMMENT '父id',
                                 `depPath` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '路径',
                                 `enabled` tinyint(1) NULL DEFAULT 1 COMMENT '是否启用',
                                 `isParent` tinyint(1) NULL DEFAULT 0 COMMENT '是否上
级',
                                PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 149 CHARACTER SET = utf8 COLLATE = utf8_general_ci
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_department
-- ----------------------------
INSERT INTO `t_department` VALUES (1, '股东会', -1, '.1', 1, 1);
INSERT INTO `t_department` VALUES (2, '董事会', 1, '.1.2', 1, 1);
INSERT INTO `t_department` VALUES (3, '总办', 2, '.1.2.3', 1, 1);
INSERT INTO `t_department` VALUES (4, '财务部', 3, '.1.2.3.4', 1, 0);
INSERT INTO `t_department` VALUES (5, '市场部', 3, '.1.2.3.5', 1, 1);
INSERT INTO `t_department` VALUES (6, '华东市场部', 5, '1.2.3.5.6', 1, 1);
INSERT INTO `t_department` VALUES (7, '华南市场部', 5, '1.2.3.5.7', 1, 0);
```

```
INSERT INTO `t_department` VALUES (8, '上海市场部', 6, '1.2.3.5.6.8', 1, 0);
INSERT INTO `t_department` VALUES (9, '西北市场部', 5, '.1.2.3.5.9', 1, 1);
INSERT INTO `t_department` VALUES (10, '贵阳市场', 9, '.1.2.3.5.9.10', 1, 1);
INSERT INTO `t_department` VALUES (11, '乌当区市场', 10, '.1.2.3.5.9.10.11', 1, 0);
INSERT INTO `t_department` VALUES (12, '技术部', 3, '.1.2.3.12', 1, 0);
INSERT INTO `t_department` VALUES (13, '运维部', 3, '.1.2.3.13', 1, 0);

-- ----------------------------
-- Table structure for t_employee
-- ----------------------------
DROP TABLE IF EXISTS `t_employee`;
CREATE TABLE `t_employee`  (
                               `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '员工编号',
                               `name` varchar(10) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '员工姓名',
                               `gender` char(4) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '性别',
                               `birthday` date NULL DEFAULT NULL COMMENT '出生日期',
                               `idCard` char(18) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '身份证号',
                               `wedlock` enum('已婚','未婚','离异') CHARACTER SET utf8
COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '婚姻状况',
                               `nationId` int(8) NULL DEFAULT NULL COMMENT '民族',
                               `nativePlace` varchar(20) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '籍贯',
                               `politicId` int(8) NULL DEFAULT NULL COMMENT '政治面貌',
                               `email` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '邮箱',
                               `phone` varchar(11) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '电话号码',
                               `address` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '联系地址',
                               `departmentId` int(11) NULL DEFAULT NULL COMMENT '所属部
门',
                               `jobLevelId` int(11) NULL DEFAULT NULL COMMENT '职称ID',
                               `posId` int(11) NULL DEFAULT NULL COMMENT '职位ID',
                               `engageForm` varchar(8) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '聘用形式',
                               `tiptopDegree` enum('博士','硕士','本科','大专','高中','初
中','小学','其他') CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT
'最高学历',
                               `specialty` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '所属专业',
                               `school` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '毕业院校',
                               `beginDate` date NULL DEFAULT NULL COMMENT '入职日期',
                               `workState` enum('在职','离职') CHARACTER SET utf8
COLLATE utf8_general_ci NULL DEFAULT '在职' COMMENT '在职状态',
                               `workID` char(8) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '工号',
                               `contractTerm` double NULL DEFAULT NULL COMMENT '合同期
限',
                               `conversionTime` date NULL DEFAULT NULL COMMENT '转正日
期',
                               `notWorkDate` date NULL DEFAULT NULL COMMENT '离职日期',
                               `beginContract` date NULL DEFAULT NULL COMMENT '合同起始
日期',
```

```
                                `endContract` date NULL DEFAULT NULL COMMENT '合同终止日
期',
                                `workAge` int(11) NULL DEFAULT NULL COMMENT '工龄',
                                `salaryId` int(11) NULL DEFAULT NULL COMMENT '工资账套
ID',
                                PRIMARY KEY (`id`) USING BTREE,
                                INDEX `departmentId`(`departmentId`) USING BTREE,
                                INDEX `jobId`(`jobLevelId`) USING BTREE,
                                INDEX `posId`(`posId`) USING BTREE,
                                INDEX `nationId`(`nationId`) USING BTREE,
                                INDEX `politicId`(`politicId`) USING BTREE,
                                INDEX `workId`(`workID`) USING BTREE,
                                INDEX `salaryId`(`salaryId`) USING BTREE,
                                CONSTRAINT `t_employee_ibfk_1` FOREIGN KEY
(`departmentId`) REFERENCES `t_department` (`id`) ON DELETE RESTRICT ON UPDATE
RESTRICT,
                                CONSTRAINT `t_employee_ibfk_2` FOREIGN KEY
(`jobLevelId`) REFERENCES `t_joblevel` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,
                                CONSTRAINT `t_employee_ibfk_3` FOREIGN KEY (`posId`)
REFERENCES `t_position` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,
                                CONSTRAINT `t_employee_ibfk_4` FOREIGN KEY (`nationId`)
REFERENCES `t_nation` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,
                                CONSTRAINT `t_employee_ibfk_5` FOREIGN KEY
(`politicId`) REFERENCES `t_politics_status` (`id`) ON DELETE RESTRICT ON UPDATE
RESTRICT,
                                CONSTRAINT `t_employee_ibfk_6` FOREIGN KEY (`salaryId`)
REFERENCES `t_salary` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB AUTO_INCREMENT = 418 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_bin
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_employee
-- ----------------------------
INSERT INTO `t_employee` VALUES (1, '韦梅', '女', '1999-11-20', '341502198810196427',
'未婚', 1, '英市', 11, 'xia53@gangjing.cn', '15567487644', '贵州省洁市清城汕尾街d座
502246', 3, 5, 5, '劳动合同', '博士', '电子工程', '中国科学技术大学', '2018-10-16', '在职',
'00000001', 9.31, '2018-08-29', NULL, '2017-09-03', '2019-08-26', NULL, 4);
INSERT INTO `t_employee` VALUES (2, '王丹', '女', '1992-03-25', '350481197304037905',
'未婚', 1, '关岭市', 2, 'jieqian@yahoo.com', '18762780051', '山西省合肥县西峰香港街C座
302114', 4, 8, 2, '劳动合同', '博士', '无', '北京大学', '2018-06-25', '在职', '00000002',
9.5, '2017-08-05', NULL, '2017-05-31', '2020-01-06', NULL, 1);
INSERT INTO `t_employee` VALUES (3, '刘俊', '男', '1996-07-18', '130224197009132687',
'未婚', 1, '萍县', 13, 'qiangfang@yahoo.com', '18663579680', '西藏自治区秀荣市海陵张家港街m座
576579', 1, 6, 1, '劳务合同', '博士', '护理学', '南京大学', '2017-08-15', '在职',
'00000003', 2.53, '2016-05-25', NULL, '2015-12-21', '2019-04-21', NULL, 1);
INSERT INTO `t_employee` VALUES (4, '刘玉珍', '女', '1993-07-10', '512000196701014288',
'未婚', 1, '雷县', 5, 'fangyang@pinggang.cn', '13972309788', '西藏自治区张家港市双滦凌街V座
614280', 4, 8, 1, '劳动合同', '博士', '市场营销', '上海交通大学', '2019-02-21', '在职',
'00000004', 5.68, '2018-06-26', NULL, '2019-02-27', '2019-11-06', NULL, 4);
INSERT INTO `t_employee` VALUES (5, '孟秀兰', '女', '2000-01-08', '130601195204145457',
'未婚', 1, '荆门县', 10, 'xliu@yahoo.com', '13319975239', '天津市辽阳市金平傅街j座 850761',
8, 4, 3, '劳动合同', '博士', '电子工程', '浙江大学', '2016-12-13', '在职', '00000005', 9.33,
'2018-03-09', NULL, '2016-11-28', '2019-06-25', NULL, 3);
INSERT INTO `t_employee` VALUES (6, '袁秀英', '男', '1990-05-07', '110107198510132428',
'未婚', 1, '合肥县', 4, 'nawu@hotmail.com', '15984781796', '重庆市兰英市高明沈阳街d座
430329', 9, 7, 2, '劳务合同', '博士', '无', '中国人民大学', '2015-10-15', '在职',
'00000006', 9.2, '2016-08-30', NULL, '2019-03-21', '2019-11-30', NULL, 2);
```

```
INSERT INTO `t_employee` VALUES (7, '沈璐', '男', '1992-05-07', '520400196705056989',
'未婚', 1, '合山县', 8, 'gaotao@hotmail.com', '14715158775', '浙江省坤县西峰余街C座
221697', 5, 6, 5, '劳务合同', '博士', '室内装修设计', '中国科学院大学', '2016-07-30', '在职',
'00000007', 7.96, '2018-12-22', NULL, '2016-09-28', '2020-02-05', NULL, 3);
INSERT INTO `t_employee` VALUES (8, '丁艳', '男', '2001-01-29', '542100199401152966',
'未婚', 1, '玲市', 10, 'yong36@gmail.com', '18878249984', '江西省南昌县双滦王路E座 490503',
8, 6, 3, '劳动合同', '博士', '信息管理与信息系统', '中国科学技术大学', '2016-05-21', '在职',
'00000008', 0.25, '2017-05-06', NULL, '2015-07-28', '2019-06-05', NULL, 2);
INSERT INTO `t_employee` VALUES (9, '张瑞', '男', '1994-01-11', '330782198309077942',
'未婚', 1, '银川市', 10, 'ping21@zengyi.net', '15510694655', '江西省太原市翔安刘街J座
815655', 12, 4, 1, '劳务合同', '博士', '信息管理与信息系统', '上海交通大学', '2018-01-15', '在
职', '00000009', 4.54, '2016-04-30', NULL, '2017-02-03', '2019-08-04', NULL, 3);
INSERT INTO `t_employee` VALUES (10, '黄柳', '男', '1999-10-03', '451422194208186 91X',
'未婚', 1, '北镇市', 3, 'juanliu@jieming.cn', '14523008198', '江苏省宁县南溪王路x座
227133', 1, 1, 2, '劳务合同', '博士', '市场营销', '南京大学', '2017-06-17', '在职',
'00000010', 1.46, '2016-11-03', NULL, '2016-06-07', '2019-08-24', NULL, 4);
INSERT INTO `t_employee` VALUES (11, '王慧', '女', '1994-07-09', '441223196008184592',
'未婚', 1, '贵阳县', 8, 'xiahu@dengpan.net', '18288493616', '青海省长沙县山亭廖街v座
439792', 5, 1, 1, '劳务合同', '博士', '电子工程', '浙江大学', '2015-10-22', '在职',
'00000011', 9.54, '2016-07-28', NULL, '2018-03-02', '2019-09-11', NULL, 4);
INSERT INTO `t_employee` VALUES (12, '田龙', '男', '1991-07-21', '621026195512050072',
'未婚', 1, '敏县', 11, 'epan@hotmail.com', '13130911244', '安徽省帆市东城唐路d座 819867',
2, 7, 4, '劳务合同', '博士', '电子工程', '中国科学技术大学', '2016-06-13', '在职',
'00000012', 4.95, '2019-02-15', NULL, '2018-07-28', '2019-08-27', NULL, 2);
INSERT INTO `t_employee` VALUES (13, '徐桂香', '男', '1997-03-06', '220101195011153893',
'未婚', 1, '桂芝市', 8, 'mingxiong@guiyingna.cn', '15239681245', '广东省巢湖县和平王路V座
418151', 4, 5, 3, '劳务合同', '博士', '信息管理与信息系统', '中国人民大学', '2018-03-01', '在
职', '00000013', 8.58, '2018-04-13', NULL, '2018-01-09', '2020-03-29', NULL, 2);
INSERT INTO `t_employee` VALUES (14, '韩桂花', '女', '1993-03-13', '451200193609248217',
'已婚', 1, '丽丽市', 13, 'vzhao@la.cn', '13091676162', '澳门特别行政区东莞市高港关岭街Z座
113957', 9, 3, 1, '劳务合同', '博士', '护理学', '清华大学', '2015-08-09', '在职',
'00000014', 5.9, '2016-08-08', NULL, '2017-09-04', '2019-06-23', NULL, 4);
INSERT INTO `t_employee` VALUES (15, '方勇', '女', '1993-03-31', '500226195607290 08X',
'已婚', 1, '银川县', 12, 'juan44@hotmail.com', '13599861266', '吉林省芳县南湖兰路y座
907387', 10, 3, 5, '劳务合同', '硕士', '室内装修设计', '北京大学', '2016-01-17', '在职',
'00000015', 7.9, '2018-11-06', NULL, '2017-09-22', '2019-12-16', NULL, 2);
INSERT INTO `t_employee` VALUES (16, '徐桂芝', '女', '1999-12-07', '350627193810292241',
'已婚', 1, '玉梅县', 2, 'min11@hotmail.com', '18060545344', '海南省桂荣县锡山拉萨路m座
889598', 3, 8, 4, '劳动合同', '硕士', '市场营销', '南京大学', '2015-10-11', '在职',
'00000016', 8.5, '2017-10-08', NULL, '2016-01-25', '2019-09-07', NULL, 3);
INSERT INTO `t_employee` VALUES (17, '郭玉英', '女', '1991-05-13', '532626196803147428',
'已婚', 1, '荆门市', 5, 'taozhang@hotmail.com', '13498131990', '贵州省玉英市长寿席路j座
607847', 4, 6, 4, '劳动合同', '硕士', '护理学', '清华大学', '2017-09-23', '在职',
'00000017', 6.9, '2019-03-31', NULL, '2017-10-23', '2019-07-17', NULL, 2);
INSERT INTO `t_employee` VALUES (18, '张波', '男', '1995-07-20', '341523193305302515',
'已婚', 1, '淑英县', 4, 'chao28@yahoo.com', '13198353039', '甘肃省勇县金平合山街V座 343550',
6, 1, 3, '劳务合同', '硕士', '信息管理与信息系统', '中国科学院大学', '2015-05-24', '在职',
'00000018', 6.51, '2018-04-10', NULL, '2016-07-14', '2019-07-21', NULL, 4);
INSERT INTO `t_employee` VALUES (19, '陈桂英', '女', '1998-07-24', '320300198302021032',
'已婚', 1, '西宁市', 12, 'leixiuying@lijun.cn', '15196790642', '江西省梧州县西峰林街g座
890108', 12, 2, 1, '劳务合同', '硕士', '室内装修设计', '中国人民大学', '2019-02-19', '在职',
'00000019', 9.4, '2018-01-31', NULL, '2016-04-20', '2019-12-24', NULL, 2);
INSERT INTO `t_employee` VALUES (20, '郭慧', '男', '1997-12-27', '370784196907163913',
'已婚', 1, '冬梅市', 5, 'jingyi@lilong.cn', '18748925191', '浙江省淮安市西峰周路Q座 231298',
1, 2, 3, '劳务合同', '硕士', '室内装修设计', '复旦大学', '2018-05-20', '在职', '00000020',
1.23, '2018-02-28', NULL, '2016-02-03', '2019-05-13', NULL, 2);
```

```
INSERT INTO `t_employee` VALUES (21, '王兰英', '女', '2001-01-14', '130628194602015 40X',
'已婚', 1, '北镇县', 2, 'nren@kc.cn', '13697605585', '河北省荆门县东丽徐路w座 733493', 9, 8,
3, '劳动合同', '硕士', '市场营销', '国防科技大学', '2017-11-22', '在职', '00000021', 7.2,
'2017-09-14', NULL, '2018-05-04', '2019-10-11', NULL, 1);
INSERT INTO `t_employee` VALUES (22, '张丽丽', '女', '1993-01-11', '440114197501194 69X',
'已婚', 1, '桂花市', 13, 'vcao@hotmail.com', '13499132244', '江苏省颖县黄浦吴路f座 348086',
1, 6, 5, '劳动合同', '硕士', '信息管理与信息系统', '中国科学技术大学', '2018-07-27', '在职',
'00000022', 1.57, '2016-08-17', NULL, '2019-03-29', '2019-12-01', NULL, 2);
INSERT INTO `t_employee` VALUES (23, '陈红', '女', '1991-05-07', '120101194008275 509',
'已婚', 1, '永安市', 4, 'fgu@hotmail.com', '14504492015', '宁夏回族自治区帆县山亭黄路Q座
528477', 9, 2, 5, '劳动合同', '硕士', '市场营销', '南京大学', '2018-06-24', '在职',
'00000023', 2.32, '2016-11-21', NULL, '2018-09-15', '2020-02-11', NULL, 4);
INSERT INTO `t_employee` VALUES (24, '范凤英', '女', '1994-11-12', '510122194703163 917',
'已婚', 1, '秀兰县', 6, 'yong19@34.cn', '13973512992', '海南省梧州市滨城李路f座 504377', 8,
6, 4, '劳动合同', '硕士', '电子工程', '北京大学', '2017-07-23', '在职', '00000024', 3.52,
'2017-07-30', NULL, '2016-02-23', '2019-12-07', NULL, 2);
INSERT INTO `t_employee` VALUES (25, '张兵', '男', '1990-09-06', '420701196603064 012',
'已婚', 1, '宜都市', 3, 'edeng@rd.cn', '15904360492', '西藏自治区桂珍市友好昆明街Y座 634021',
9, 6, 3, '劳务合同', '硕士', '市场营销', '中国人民大学', '2017-03-20', '在职', '00000025',
0.5, '2016-11-07', NULL, '2017-06-02', '2020-01-24', NULL, 2);
INSERT INTO `t_employee` VALUES (26, '黄宁', '男', '1995-06-12', '510303198712060 557',
'已婚', 1, '玉英县', 11, 'xiuyingpan@gmail.com', '13377122856', '广东省莉市朝阳台北街x座
190715', 6, 4, 4, '劳动合同', '硕士', '护理学', '中国科学院大学', '2017-05-09', '在职',
'00000026', 7.72, '2017-01-24', NULL, '2018-06-07', '2019-10-31', NULL, 4);
INSERT INTO `t_employee` VALUES (27, '黄荣', '女', '1997-08-04', '140603198703253 16X',
'已婚', 1, '大冶市', 1, 'ihao@yahoo.com', '14528832529', '江西省六安县和平永安街r座 137243',
9, 4, 1, '劳务合同', '硕士', '无', '浙江大学', '2018-09-29', '在职', '00000027', 6.16,
'2017-11-25', NULL, '2017-12-14', '2019-12-29', NULL, 2);
INSERT INTO `t_employee` VALUES (28, '周雷', '男', '2001-12-03', '411201197212305 874',
'已婚', 1, '六安市', 7, 'ping89@wg.cn', '14550266014', '浙江省丹县黄浦北镇路G座 186275', 6,
4, 3, '劳动合同', '硕士', '电子工程', '中国科学院大学', '2017-10-04', '在职', '00000028',
8.14, '2018-04-18', NULL, '2018-03-07', '2020-03-20', NULL, 2);
INSERT INTO `t_employee` VALUES (29, '周静', '女', '2001-03-14', '640105199111105 559',
'已婚', 1, '雪梅县', 6, 'xiajing@yahoo.com', '18912358599', '山西省瑜县怀柔北京路p座
691913', 6, 6, 4, '劳务合同', '硕士', '护理学', '北京大学', '2018-02-28', '在职',
'00000029', 1.4, '2018-10-20', NULL, '2017-05-04', '2020-01-05', NULL, 4);
INSERT INTO `t_employee` VALUES (30, '师婷婷', '男', '1998-09-14', '500241199402197 190',
'已婚', 1, '沈阳县', 2, 'pcui@hotmail.com', '14572285539', '福建省海口县江北陈路H座 259607',
5, 4, 4, '劳动合同', '本科', '信息管理与信息系统', '南京大学', '2017-02-22', '在职',
'00000030', 8.7, '2018-11-17', NULL, '2018-07-07', '2019-05-26', NULL, 1);
INSERT INTO `t_employee` VALUES (31, '邹平', '男', '1995-07-18', '469023197609079 864',
'已婚', 1, '沈阳县', 7, 'qiang09@hotmail.com', '15711044116', '甘肃省丽丽县永川石家庄街w座
788572', 13, 2, 2, '劳务合同', '本科', '市场营销', '浙江大学', '2019-02-27', '在职',
'00000031', 0.22, '2019-01-15', NULL, '2017-04-12', '2019-05-02', NULL, 3);
INSERT INTO `t_employee` VALUES (32, '蒋桂兰', '男', '2002-01-03', '451000198103141 393',
'已婚', 1, '通辽县', 2, 'xuli@yahoo.com', '14747861689', '台湾省斌县大兴潮州街Q座 620273',
5, 6, 1, '劳务合同', '本科', '电子工程', '浙江大学', '2019-01-30', '在职', '00000032', 4.11,
'2018-09-26', NULL, '2015-05-16', '2019-05-13', NULL, 3);
INSERT INTO `t_employee` VALUES (33, '何燕', '女', '1995-10-03', '211081194006172 759',
'已婚', 1, '伟市', 9, 'pqin@yahoo.com', '14510585247', '宁夏回族自治区健市和平蔡街X座
666462', 9, 6, 1, '劳动合同', '本科', '护理学', '复旦大学', '2018-05-08', '在职',
'00000033', 9.4, '2017-03-17', NULL, '2017-07-12', '2019-10-22', NULL, 3);
INSERT INTO `t_employee` VALUES (34, '马淑珍', '女', '2001-07-07', '130604198910150 088',
'已婚', 1, '成县', 7, 'leizou@yahoo.com', '13164981755', '湖北省太原县魏都安街c座 530374',
9, 8, 3, '劳动合同', '本科', '市场营销', '中国科学院大学', '2019-03-12', '在职', '00000034',
2.67, '2018-10-18', NULL, '2018-12-22', '2020-04-04', NULL, 4);
```

```
INSERT INTO `t_employee` VALUES (35, '杨秀芳', '女', '1994-04-13', '130581196312068381',
'已婚', 1, '合肥县', 11, 'pqian@jiena.cn', '13873572680', '重庆市璐县大东孙街P座 155477',
1, 4, 3, '劳动合同', '本科', '信息管理与信息系统', '中国科学院大学', '2017-07-21', '在职',
'00000035', 4.4, '2017-08-12', NULL, '2016-01-07', '2019-07-22', NULL, 2);
INSERT INTO `t_employee` VALUES (36, '邱洁', '女', '2000-09-29', '130101194910161225',
'已婚', 1, '婷婷县', 8, 'czhou@yongluo.cn', '13841978850', '安徽省磊县南溪刘街U座 968626',
7, 5, 3, '劳务合同', '本科', '市场营销', '复旦大学', '2016-06-01', '在职', '00000036', 1.63,
'2018-04-25', NULL, '2015-05-18', '2019-12-12', NULL, 4);
INSERT INTO `t_employee` VALUES (37, '王桂花', '女', '1992-05-08', '140723194212128260',
'已婚', 1, '呼和浩特市', 8, 'ping62@yahoo.com', '18029671014', '内蒙古自治区英县魏都台北街h座
836168', 8, 6, 1, '劳动合同', '本科', '中国语言文学', '浙江大学', '2015-09-26', '在职',
'00000037', 9.93, '2017-10-24', NULL, '2017-06-27', '2019-07-13', NULL, 2);
INSERT INTO `t_employee` VALUES (38, '刘畅', '女', '1995-12-27', '511000193311215215',
'已婚', 1, '兴安盟县', 1, 'naxiong@yahoo.com', '13377530628', '河北省呼和浩特县大东王街A座
556099', 12, 1, 4, '劳动合同', '本科', '信息管理与信息系统', '中国科学技术大学', '2017-12-11',
'在职', '00000038', 6.5, '2018-10-09', NULL, '2018-05-08', '2019-04-20', NULL, 4);
INSERT INTO `t_employee` VALUES (39, '蒋欣', '男', '1994-04-02', '450225195711013399',
'已婚', 1, '南宁市', 7, 'chenli@75.cn', '13808748993', '福建省杭州市南长天津街m座 127451',
4, 3, 5, '劳务合同', '本科', '无', '北京大学', '2016-04-15', '在职', '00000039', 7.55,
'2016-10-15', NULL, '2018-10-04', '2020-03-28', NULL, 3);
INSERT INTO `t_employee` VALUES (40, '王超', '男', '1991-03-18', '652901197112045159',
'已婚', 1, '重庆市', 5, 'mengyong@xiongyu.cn', '18159666889', '广东省秀华县浔阳高街Q座
723731', 7, 5, 3, '劳务合同', '本科', '无', '中国人民大学', '2016-10-09', '在职',
'00000040', 9.67, '2017-05-01', NULL, '2015-06-11', '2019-04-19', NULL, 1);
INSERT INTO `t_employee` VALUES (41, '龚东', '女', '1993-04-15', '140107193503215615',
'已婚', 1, '成都市', 5, 'guiying04@leilai.cn', '13611631710', '台湾省东莞县龙潭潜江路d座
593600', 11, 4, 1, '劳动合同', '本科', '室内装修设计', '南京大学', '2016-05-09', '在职',
'00000041', 8.68, '2016-05-05', NULL, '2017-08-22', '2020-01-21', NULL, 2);
INSERT INTO `t_employee` VALUES (42, '侯欢', '女', '1990-08-09', '341824193311150186',
'已婚', 1, '龙县', 8, 'yangsu@fangpeng.net', '13821459885', '河南省哈尔滨县东丽呼和浩特路C座
743024', 7, 8, 1, '劳动合同', '本科', '中国语言文学', '复旦大学', '2015-11-18', '在职',
'00000042', 2.62, '2018-11-17', NULL, '2017-04-15', '2020-01-26', NULL, 4);
INSERT INTO `t_employee` VALUES (43, '赵玉华', '女', '1998-01-09', '341525197108234630',
'未婚', 1, '博县', 2, 'xiuying33@dt.cn', '18682469543', '香港特别行政区晨县上街刘街e座
463619', 3, 1, 1, '劳动合同', '本科', '护理学', '南京大学', '2016-02-11', '在职',
'00000043', 2.66, '2016-07-21', NULL, '2016-03-11', '2019-07-04', NULL, 4);
INSERT INTO `t_employee` VALUES (44, '刘杨', '男', '1993-05-06', '610404193406165281',
'未婚', 1, '玉兰县', 11, 'fangren@xg.cn', '15750031625', '台湾省玉兰市门头沟海门街p座
470818', 5, 2, 2, '劳动合同', '本科', '无', '上海交通大学', '2015-10-21', '在职',
'00000044', 1.63, '2016-07-16', NULL, '2018-07-14', '2020-04-08', NULL, 2);
INSERT INTO `t_employee` VALUES (45, '谢秀兰', '女', '1993-10-29', '450203200203284294',
'未婚', 1, '六安县', 6, 'yangmo@59.cn', '15970698995', '澳门特别行政区秀芳市安次张路1座
926668', 11, 8, 2, '劳动合同', '本科', '室内装修设计', '国防科技大学', '2019-02-09', '在职',
'00000045', 7.66, '2016-11-15', NULL, '2016-04-25', '2019-12-25', NULL, 4);
INSERT INTO `t_employee` VALUES (46, '徐秀云', '男', '1997-07-22', '520621193005292541',
'未婚', 1, '六盘水市', 8, 'zhengyan@yahoo.com', '15017588555', '山西省太原县滨城潜江路a座
670289', 10, 2, 5, '劳务合同', '大专', '护理学', '国防科技大学', '2015-10-26', '在职',
'00000046', 2.41, '2017-01-17', NULL, '2016-02-23', '2019-10-04', NULL, 3);
INSERT INTO `t_employee` VALUES (47, '秦艳', '男', '1993-11-26', '420581194004273164',
'未婚', 1, '兵县', 3, 'guiying60@shikang.cn', '15048749906', '北京市建军县沈北新齐齐哈尔路a座
711455', 5, 5, 5, '劳动合同', '大专', '电子工程', '复旦大学', '2018-10-20', '在职',
'00000047', 9.55, '2018-01-13', NULL, '2019-01-11', '2020-02-18', NULL, 2);
INSERT INTO `t_employee` VALUES (48, '李瑞', '男', '1990-06-27', '231085198709305690',
'未婚', 1, '建军县', 2, 'mindong@yahoo.com', '13974711811', '贵州省马鞍山市龙潭朱街L座
375799', 6, 8, 4, '劳务合同', '大专', '信息管理与信息系统', '国防科技大学', '2017-08-11', '在
职', '00000048', 7.61, '2016-09-08', NULL, '2018-06-16', '2019-08-18', NULL, 2);
```

```
INSERT INTO `t_employee` VALUES (49, '周倩', '男', '2001-01-10', '140926195508093145',
'未婚', 1, '巢湖县', 10, 'qianjuan@yan.cn', '18792735131', '西藏自治区重庆市江北石家庄路G座
574433', 12, 1, 5, '劳动合同', '大专', '信息管理与信息系统', '复旦大学', '2018-11-09', '在
职', '00000049', 7.79, '2018-10-07', NULL, '2018-06-25', '2019-08-10', NULL, 2);
INSERT INTO `t_employee` VALUES (50, '宋阳', '女', '1993-05-17', '510921199204091277',
'未婚', 1, '昆明市', 3, 'mwen@minyu.cn', '18826244663', '湖南省丽丽县清城李路P座 172795',
10, 8, 2, '劳动合同', '大专', '无', '国防科技大学', '2015-08-20', '在职', '00000050', 6.69,
'2019-02-23', NULL, '2017-08-18', '2019-11-12', NULL, 2);
INSERT INTO `t_employee` VALUES (51, '毛玉', '男', '1995-05-24', '230901193512038662',
'未婚', 1, '哈尔滨县', 10, 'tshao@yaoliu.cn', '13921802030', '广东省晶县房山郑州路t座
474776', 2, 1, 2, '劳务合同', '大专', '护理学', '北京大学', '2016-11-16', '在职',
'00000051', 9.3, '2019-01-05', NULL, '2017-03-16', '2020-04-05', NULL, 4);
INSERT INTO `t_employee` VALUES (52, '陈红梅', '男', '1993-01-15', '36078120010301542X',
'未婚', 1, '明市', 9, 'ptian@guiying.cn', '15660538227', '澳门特别行政区淑珍县华龙田路s座
526080', 3, 2, 2, '劳务合同', '大专', '市场营销', '南京大学', '2016-03-14', '在职',
'00000052', 7.99, '2018-11-16', NULL, '2016-08-13', '2020-04-15', NULL, 3);
INSERT INTO `t_employee` VALUES (53, '萧春梅', '女', '1994-07-14', '130225193503290451',
'未婚', 1, '华县', 7, 'guiyingzeng@gmail.com', '15280462787', '河北省昆明市长寿顾街o座
728886', 3, 3, 3, '劳务合同', '大专', '中国语言文学', '国防科技大学', '2017-07-25', '在职',
'00000053', 8.1, '2017-04-08', NULL, '2017-06-16', '2020-04-05', NULL, 2);
INSERT INTO `t_employee` VALUES (54, '蔡琳', '女', '2001-04-07', '230801197307256970',
'未婚', 1, '雪县', 6, 'yongkong@gmail.com', '13658227544', '澳门特别行政区小红市西夏刘路v座
917143', 4, 4, 1, '劳务合同', '大专', '中国语言文学', '复旦大学', '2016-02-17', '在职',
'00000054', 1.32, '2018-09-27', NULL, '2017-07-31', '2019-11-19', NULL, 1);
INSERT INTO `t_employee` VALUES (55, '谢杰', '男', '1990-07-04', '532524199902246363',
'未婚', 1, '晨市', 12, 'junjin@yahoo.com', '13637320900', '内蒙古自治区辽阳市淄川刘路w座
282976', 13, 4, 2, '劳务合同', '大专', '市场营销', '中国人民大学', '2016-03-23', '在职',
'00000055', 4.3, '2016-07-16', NULL, '2017-03-09', '2019-11-05', NULL, 3);
INSERT INTO `t_employee` VALUES (56, '都利', '男', '2001-12-03', '410601196110227963',
'未婚', 1, '东莞市', 3, 'wzeng@yahoo.com', '13388584730', '湖北省西宁县沙湾沈阳街Y座
767974', 6, 4, 5, '劳务合同', '大专', '信息管理与信息系统', '北京大学', '2016-07-26', '在职',
'00000056', 2.91, '2019-02-11', NULL, '2016-11-19', '2019-10-02', NULL, 1);
INSERT INTO `t_employee` VALUES (57, '刘颖', '男', '1992-01-01', '130403198604090550',
'未婚', 1, '六盘水县', 11, 'zhaofang@84.cn', '15110335073', '澳门特别行政区嘉禾市白云周路J座
881702', 4, 3, 1, '劳务合同', '大专', '中国语言文学', '中国科学技术大学', '2016-10-01', '在
职', '00000057', 3.3, '2016-12-05', NULL, '2018-03-10', '2020-01-17', NULL, 4);
INSERT INTO `t_employee` VALUES (58, '刁桂芳', '女', '2001-09-13', '150302197202283875',
'未婚', 1, '六安县', 5, 'rdong@hotmail.com', '15357876400', '新疆维吾尔自治区佳县南溪宜都街R
座 771673', 12, 8, 3, '劳务合同', '大专', '市场营销', '清华大学', '2017-10-29', '在职',
'00000058', 4.85, '2018-08-12', NULL, '2015-05-03', '2019-06-22', NULL, 3);
INSERT INTO `t_employee` VALUES (59, '许明', '女', '1996-02-01', '540121196305265449',
'未婚', 1, '海口市', 2, 'dlu@yahoo.com', '15131877702', '贵州省香港市黄浦佛山街P座 941339',
3, 4, 2, '劳动合同', '大专', '无', '南京大学', '2016-04-15', '在职', '00000059', 4.3,
'2019-02-08', NULL, '2017-04-02', '2019-10-09', NULL, 4);
INSERT INTO `t_employee` VALUES (60, '张瑞', '男', '1997-11-07', '220104197208285134',
'未婚', 1, '龙市', 5, 'yong72@36.net', '13034769245', '山西省桂荣县海陵洪街u座 729157', 8,
3, 1, '劳务合同', '大专', '室内装修设计', '浙江大学', '2015-05-30', '在职', '00000060',
5.37, '2018-03-25', NULL, '2015-12-25', '2020-04-07', NULL, 4);
INSERT INTO `t_employee` VALUES (61, '周荣', '女', '1999-08-11', '450304196910099779',
'未婚', 1, '天津县', 12, 'chao45@30.cn', '15278236594', '宁夏回族自治区娟县合川辛集街1座
407633', 7, 1, 2, '劳务合同', '大专', '无', '国防科技大学', '2015-05-17', '在职',
'00000061', 9.96, '2018-06-02', NULL, '2015-09-18', '2020-03-28', NULL, 2);
INSERT INTO `t_employee` VALUES (62, '徐飞', '女', '1993-03-27', '350600195104028506',
'未婚', 1, '芳市', 5, 'chaoyuan@yahoo.com', '18975736884', '香港特别行政区合肥县翔安陈街s座
253286', 12, 3, 1, '劳动合同', '大专', '市场营销', '复旦大学', '2017-12-14', '在职',
'00000062', 9.6, '2017-11-26', NULL, '2019-04-09', '2019-09-04', NULL, 3);
```

```
INSERT INTO `t_employee` VALUES (63, '金健', '男', '1991-04-26', '610125197105094719',
'未婚', 1, '呼和浩特市', 9, 'juan18@hotmail.com', '15328888611', '浙江省颖县涪城福州街Z座
270167', 9, 8, 4, '劳务合同', '大专', '信息管理与信息系统', '南京大学', '2017-09-02', '在职',
'00000063', 8.84, '2018-01-08', NULL, '2018-05-09', '2019-12-15', NULL, 3);
INSERT INTO `t_employee` VALUES (64, '刘红梅', '女', '1991-09-18', '653001195201108337',
'未婚', 1, '柳州县', 2, 'pwei@yuan.cn', '13716052290', '贵州省济南县安次林路r座 659793', 11,
6, 2, '劳务合同', '大专', '电子工程', '中国人民大学', '2018-11-09', '在职', '00000064',
7.27, '2018-12-14', NULL, '2018-09-05', '2019-08-06', NULL, 4);
INSERT INTO `t_employee` VALUES (65, '彭静', '女', '1994-03-14', '420105196211114116',
'未婚', 1, '秀华县', 11, 'jinggu@yongzeng.net', '13342125691', '湖南省燕县友好阜新路Z座
892528', 1, 5, 5, '劳动合同', '大专', '无', '中国人民大学', '2017-12-17', '在职',
'00000065', 7.48, '2018-09-08', NULL, '2017-09-01', '2019-10-10', NULL, 3);
INSERT INTO `t_employee` VALUES (66, '朱燕', '男', '1998-12-25', '140724199512227964',
'未婚', 1, '兰州县', 12, 'nguo@gmail.com', '14747652210', '贵州省呼和浩特市六枝特宗街k座
609064', 6, 2, 4, '劳动合同', '大专', '无', '中国人民大学', '2017-07-28', '在职',
'00000066', 1.52, '2017-10-23', NULL, '2016-05-31', '2020-01-22', NULL, 1);
INSERT INTO `t_employee` VALUES (67, '成健', '女', '1996-03-30', '230108200104147249',
'未婚', 1, '武汉市', 7, 'nxiao@uu.cn', '13209643778', '香港特别行政区邯郸市浔阳呼和浩特路U座
929024', 7, 4, 2, '劳务合同', '大专', '中国语言文学', '中国人民大学', '2017-05-25', '在职',
'00000067', 2.1, '2017-01-27', NULL, '2018-01-30', '2019-07-21', NULL, 1);
INSERT INTO `t_employee` VALUES (68, '高成', '男', '1994-04-09', '410000198807233933',
'离异', 1, '辽阳市', 10, 'baijun@gmail.com', '15061826099', '广西壮族自治区桂芳市沈河济南路Y
座 175803', 13, 6, 1, '劳务合同', '大专', '信息管理与信息系统', '浙江大学', '2015-10-07', '在
职', '00000068', 1.24, '2017-01-12', NULL, '2016-10-15', '2020-03-09', NULL, 4);
INSERT INTO `t_employee` VALUES (69, '杜海燕', '男', '1992-02-29', '421127195707202229',
'离异', 1, '台北市', 4, 'chaoyi@kd.cn', '13961620974', '青海省洁县沙湾罗街V座 370522', 8,
6, 4, '劳动合同', '高中', '无', '浙江大学', '2017-03-15', '在职', '00000069', 4.13, '2017-
09-20', NULL, '2019-02-20', '2019-11-29', NULL, 4);
INSERT INTO `t_employee` VALUES (70, '徐芳', '男', '1998-09-25', '621223197609010852',
'离异', 1, '娜县', 11, 'dzou@yinsun.org', '13372235190', '云南省秀梅市东丽淮安街j座 774716',
3, 3, 1, '劳动合同', '高中', '室内装修设计', '中国人民大学', '2018-08-18', '在职',
'00000070', 8.79, '2017-06-28', NULL, '2015-09-02', '2019-12-27', NULL, 2);
INSERT INTO `t_employee` VALUES (71, '徐宇', '女', '1996-05-19', '62102519520401847X',
'离异', 1, '红梅市', 7, 'na38@hotmail.com', '14754588063', '重庆市磊县花溪南京街Z座 323661',
11, 4, 2, '劳务合同', '高中', '市场营销', '中国科学技术大学', '2015-08-23', '在职',
'00000071', 7.83, '2019-03-29', NULL, '2016-02-15', '2020-04-13', NULL, 4);
INSERT INTO `t_employee` VALUES (72, '孟璐', '男', '1991-01-26', '220105195307205862',
'离异', 1, '婷婷县', 11, 'yangtang@gmail.com', '13267664247', '四川省贵阳县闵行北京路y座
418617', 5, 8, 4, '劳动合同', '高中', '信息管理与信息系统', '复旦大学', '2017-11-01', '在职',
'00000072', 5.18, '2018-02-26', NULL, '2019-01-22', '2019-10-25', NULL, 1);
INSERT INTO `t_employee` VALUES (73, '王鹏', '女', '2000-06-06', '330281198803115744',
'离异', 1, '桂英县', 6, 'gang62@pan.cn', '18660120065', '浙江省武汉市怀柔王路v座 294443', 6,
2, 5, '劳动合同', '高中', '电子工程', '复旦大学', '2018-05-13', '在职', '00000073', 6.85,
'2017-02-05', NULL, '2018-04-10', '2019-04-22', NULL, 1);
INSERT INTO `t_employee` VALUES (74, '刘金凤', '男', '2001-02-16', '330203194112083504',
'离异', 1, '涛市', 2, 'bliang@gmail.com', '15159155928', '山东省峰市大兴潘路x座 917961', 3,
4, 2, '劳务合同', '高中', '市场营销', '浙江大学', '2015-11-05', '在职', '00000074', 1.78,
'2018-12-17', NULL, '2016-09-20', '2020-03-23', NULL, 2);
INSERT INTO `t_employee` VALUES (75, '苏琴', '男', '1994-12-21', '222404198310016558',
'离异', 1, '桂花市', 4, 'xiangguiying@gmail.com', '18692275625', '四川省璐县永川邢街O座
973198', 5, 1, 4, '劳动合同', '高中', '护理学', '北京大学', '2015-06-30', '在职',
'00000075', 5.3, '2018-05-05', NULL, '2017-04-08', '2020-03-29', NULL, 1);
INSERT INTO `t_employee` VALUES (76, '毛帆', '男', '1998-09-07', '450903196006129649',
'离异', 1, '佛山市', 13, 'juan48@xia.cn', '18652337157', '浙江省凤英市高明宜都路l座 164063',
10, 3, 4, '劳务合同', '高中', '无', '中国科学技术大学', '2016-04-15', '在职', '00000076',
4.11, '2016-08-27', NULL, '2018-05-15', '2019-10-01', NULL, 4);
```

```
INSERT INTO `t_employee` VALUES (77, '张秀云', '男', '1998-10-26', '431123197807174937',
'离异', 1, '重庆市', 5, 'vsu@haoliao.cn', '18029370756', '陕西省太原市朝阳王路X座 220974',
11, 6, 5, '劳务合同', '高中', '无', '浙江大学', '2017-03-10', '在职', '00000077', 5.17,
'2018-01-08', NULL, '2017-08-27', '2020-03-31', NULL, 4);
INSERT INTO `t_employee` VALUES (78, '高想', '女', '2000-11-30', '371723198203262056',
'离异', 1, '哈尔滨市', 7, 'vxiang@hotmail.com', '15044991465', '西藏自治区辉市秀英张街T座
564316', 8, 2, 5, '劳务合同', '高中', '电子工程', '国防科技大学', '2016-06-01', '在职',
'00000078', 1.13, '2017-06-06', NULL, '2017-11-07', '2019-09-13', NULL, 4);
INSERT INTO `t_employee` VALUES (79, '朱宁', '女', '1992-05-05', '211322196507123213',
'离异', 1, '呼和浩特县', 4, 'lei13@tao.cn', '15222017286', '四川省广州县南湖邹路U座 729670',
12, 1, 2, '劳务合同', '高中', '护理学', '北京大学', '2017-07-01', '在职', '00000079', 2.55,
'2018-12-28', NULL, '2015-09-12', '2019-04-29', NULL, 4);
INSERT INTO `t_employee` VALUES (80, '陈楠', '男', '1992-04-28', '230129196805237522',
'离异', 1, '桂兰市', 13, 'gangkang@hotmail.com', '13406718891', '台湾省林市清浦周路k座
115493', 7, 2, 2, '劳动合同', '高中', '护理学', '中国科学院大学', '2015-08-15', '在职',
'00000080', 8.88, '2016-11-22', NULL, '2018-09-28', '2019-12-28', NULL, 2);
INSERT INTO `t_employee` VALUES (81, '方桂珍', '女', '1998-07-07', '230108194311177987',
'离异', 1, '帅市', 6, 'nli@hotmail.com', '15127153492', '青海省春梅县壁山马鞍山路J座
120691', 7, 7, 1, '劳务合同', '高中', '市场营销', '国防科技大学', '2017-10-22', '在职',
'00000081', 5.63, '2017-12-27', NULL, '2017-11-07', '2019-11-17', NULL, 4);
INSERT INTO `t_employee` VALUES (82, '李佳', '女', '2001-06-21', '440606196708251869',
'离异', 1, '萍市', 1, 'donglei@yahoo.com', '18989396343', '山东省娟县海陵孙街D座 339304',
3, 7, 3, '劳动合同', '高中', '中国语言文学', '中国科学院大学', '2016-10-30', '在职',
'00000082', 0.89, '2017-05-20', NULL, '2018-09-24', '2019-09-02', NULL, 2);
INSERT INTO `t_employee` VALUES (83, '朱丽华', '女', '1994-06-20', '230705193206128541',
'离异', 1, '博县', 6, 'yinxiulan@fangsong.cn', '15767432260', '台湾省香港市东城杭州街L座
231062', 6, 6, 5, '劳动合同', '高中', '室内装修设计', '南京大学', '2018-05-07', '在职',
'00000083', 7.35, '2018-06-24', NULL, '2017-05-05', '2019-08-31', NULL, 2);
INSERT INTO `t_employee` VALUES (84, '丛婷婷', '男', '1990-07-15', '341323195608243637',
'离异', 1, '六安市', 6, 'leisong@gmail.com', '15058065854', '黑龙江省昆明市崇文南昌街k座
929460', 3, 4, 4, '劳务合同', '高中', '中国语言文学', '国防科技大学', '2018-09-17', '在职',
'00000084', 5.4, '2017-05-20', NULL, '2017-07-27', '2019-05-07', NULL, 1);
INSERT INTO `t_employee` VALUES (85, '阙亮', '男', '1990-05-12', '542221195510068340',
'离异', 1, '汕尾县', 2, 'minfang@gmail.com', '15728517177', '香港特别行政区永安县西峰王路n座
719026', 2, 1, 5, '劳务合同', '高中', '信息管理与信息系统', '南京大学', '2017-01-10', '在职',
'00000085', 6.33, '2017-01-11', NULL, '2016-05-05', '2019-08-23', NULL, 1);
INSERT INTO `t_employee` VALUES (86, '范红霞', '男', '1999-02-14', '440704198110164308',
'离异', 1, '辽阳县', 3, 'iyuan@tao.cn', '18943020154', '贵州省秀兰市徐汇东莞街Z座 140503',
3, 1, 5, '劳务合同', '高中', '电子工程', '上海交通大学', '2018-05-03', '在职', '00000086',
5.91, '2018-08-23', NULL, '2017-12-18', '2019-04-29', NULL, 2);
INSERT INTO `t_employee` VALUES (87, '曹娜', '男', '1993-08-04', '431321195103314793',
'离异', 1, '秀荣县', 9, 'lei99@xiulanjiang.cn', '14535758888', '广东省娟县崇文彭街p座
303594', 6, 2, 5, '劳动合同', '高中', '护理学', '浙江大学', '2018-08-02', '在职',
'00000087', 6.79, '2017-08-31', NULL, '2018-04-24', '2020-03-25', NULL, 1);
INSERT INTO `t_employee` VALUES (88, '秦龙', '男', '1991-08-23', '420601199107076880',
'离异', 1, '春梅县', 6, 'dingyong@hotmail.com', '13035350393', '北京市拉萨县西夏哈尔滨路l座
940104', 3, 5, 4, '劳动合同', '高中', '信息管理与信息系统', '南京大学', '2015-07-04', '在职',
'00000088', 7.36, '2018-01-09', NULL, '2017-08-24', '2019-09-09', NULL, 1);
INSERT INTO `t_employee` VALUES (89, '刘欢', '男', '2000-04-14', '360124197901038461',
'离异', 1, '呼和浩特县', 4, 'leiluo@hotmail.com', '18650425363', '台湾省秀珍县门头沟张路k座
785416', 2, 3, 4, '劳务合同', '高中', '信息管理与信息系统', '中国科学技术大学', '2018-08-28',
'在职', '00000089', 6.76, '2017-01-25', NULL, '2019-03-23', '2020-02-05', NULL, 2);
INSERT INTO `t_employee` VALUES (90, '邵刚', '女', '1992-05-31', '450401194505080019',
'离异', 1, '丽市', 7, 'junxiong@hotmail.com', '15138260125', '青海省建军市海陵戴街x座
562142', 4, 5, 1, '劳动合同', '高中', '市场营销', '中国人民大学', '2018-01-17', '在职',
'00000090', 3.92, '2019-03-17', NULL, '2019-04-05', '2019-11-20', NULL, 4);
```

```
INSERT INTO `t_employee` VALUES (91, '石坤', '男', '1991-12-06', '211301200012288558',
'离异', 1, '超县', 9, 'jun14@yahoo.com', '13380161242', '江西省昆明市丰都苏路b座 863383',
8, 5, 1, '劳动合同', '高中', '市场营销', '上海交通大学', '2018-05-31', '在职', '00000091',
6.93, '2017-09-24', NULL, '2015-06-13', '2020-04-14', NULL, 4);
INSERT INTO `t_employee` VALUES (92, '余建华', '女', '1992-07-17', '520302193402030347',
'离异', 1, '阳市', 3, 'mingding@28.cn', '13435060183', '重庆市金凤市江北杭州路V座 700174',
6, 3, 3, '劳务合同', '高中', '市场营销', '清华大学', '2018-11-27', '在职', '00000092', 0.29,
'2018-06-10', NULL, '2018-08-04', '2019-08-10', NULL, 2);
INSERT INTO `t_employee` VALUES (93, '宋伟', '女', '1999-11-09', '330300198210308601',
'离异', 1, '北京市', 11, 'juanliang@gmail.com', '18862667374', '吉林省亮市秀英拉萨路w座
463882', 5, 1, 2, '劳务合同', '高中', '市场营销', '复旦大学', '2018-09-23', '在职',
'00000093', 2.57, '2018-01-01', NULL, '2018-02-01', '2020-03-14', NULL, 1);
INSERT INTO `t_employee` VALUES (94, '韦成', '男', '1991-02-13', '130421198801238342',
'离异', 1, '雷县', 10, 'zhaoxia@kt.cn', '14575813594', '青海省宁德县普陀淮安路W座 410428',
7, 6, 3, '劳务合同', '高中', '电子工程', '清华大学', '2018-07-14', '在职', '00000094', 2.27,
'2017-03-04', NULL, '2016-01-26', '2019-12-06', NULL, 2);
INSERT INTO `t_employee` VALUES (95, '陈鹏', '男', '2000-07-19', '620721197805215584',
'离异', 1, '兴安盟市', 9, 'juan93@04.cn', '13709426633', '云南省桂英县沙湾黄路m座 314538',
5, 3, 4, '劳动合同', '高中', '信息管理与信息系统', '复旦大学', '2015-09-29', '在职',
'00000095', 2.22, '2018-10-14', NULL, '2018-01-12', '2019-12-01', NULL, 3);
INSERT INTO `t_employee` VALUES (96, '叶楠', '女', '1990-12-03', '532931197605230863',
'离异', 1, '太原县', 9, 'ming60@yahoo.com', '14770621573', '陕西省辉市朝阳佛山街r座 140974',
5, 5, 5, '劳务合同', '高中', '信息管理与信息系统', '南京大学', '2018-01-10', '在职',
'00000096', 9.78, '2016-09-29', NULL, '2016-11-21', '2019-12-25', NULL, 1);
INSERT INTO `t_employee` VALUES (97, '原红霞', '男', '1997-04-07', '341701193905207661',
'已婚', 1, '刚县', 5, 'yan03@ye.cn', '13243818433', '辽宁省桂荣市沙湾陈路E座 758823', 1, 7,
5, '劳务合同', '高中', '无', '浙江大学', '2016-11-18', '在职', '00000097', 2.3, '2017-06-
03', NULL, '2016-09-26', '2020-01-16', NULL, 3);
INSERT INTO `t_employee` VALUES (98, '王玉英', '女', '1998-03-12', '370282193508208316',
'已婚', 1, '英县', 7, 'liaochao@gmail.com', '14798666739', '湖北省贵阳县花溪汕尾街e座
501635', 13, 5, 3, '劳动合同', '高中', '中国语言文学', '中国科学技术大学', '2017-12-19', '在
职', '00000098', 2.23, '2017-11-02', NULL, '2018-05-08', '2019-10-21', NULL, 3);
INSERT INTO `t_employee` VALUES (99, '张红梅', '女', '1991-07-18', '350923199911240828',
'已婚', 1, '燕市', 2, 'yili@hotmail.com', '13506023869', '浙江省兰州县大兴侯路x座 732564',
2, 8, 4, '劳动合同', '高中', '电子工程', '上海交通大学', '2017-11-27', '在职', '00000099',
2.4, '2018-03-13', NULL, '2019-01-30', '2019-07-04', NULL, 1);
INSERT INTO `t_employee` VALUES (100, '李强', '男', '1994-02-17', '330127200203300889',
'已婚', 1, '淑兰县', 4, 'fang76@ds.cn', '15703517874', '重庆市凤兰市山亭马路l座 594007', 5,
2, 1, '劳务合同', '高中', '无', '上海交通大学', '2015-12-14', '在职', '00000100', 5.96,
'2017-09-06', NULL, '2017-06-16', '2019-06-08', NULL, 3);

-- ----------------------------
-- Table structure for t_employee_ec
-- ----------------------------
DROP TABLE IF EXISTS `t_employee_ec`;
CREATE TABLE `t_employee_ec`  (
                                  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                                  `eid` int(11) NULL DEFAULT NULL COMMENT '员工编号',
                                  `ecDate` date NULL DEFAULT NULL COMMENT '奖罚日期',
                                  `ecReason` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '奖罚原因',
                                  `ecPoint` int(11) NULL DEFAULT NULL COMMENT '奖罚分',
                                  `ecType` int(11) NULL DEFAULT NULL COMMENT '奖罚类别,
0:奖, 1:罚',
                                  `remark` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '备注',
                                  PRIMARY KEY (`id`) USING BTREE,
                                  INDEX `eid`(`eid`) USING BTREE,
```

```
                                  CONSTRAINT `t_employee_ec_ibfk_1` FOREIGN KEY
(`eid`) REFERENCES `t_employee` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Dynamic;

-- ----------------------------
-- Table structure for t_employee_remove
-- ----------------------------
DROP TABLE IF EXISTS `t_employee_remove`;
CREATE TABLE `t_employee_remove`  (
                                      `id` int(11) NOT NULL AUTO_INCREMENT COMMENT
'id',
                                      `eid` int(11) NULL DEFAULT NULL COMMENT '员工id',
                                      `afterDepId` int(11) NULL DEFAULT NULL COMMENT
'调动后部门',
                                      `afterJobId` int(11) NULL DEFAULT NULL COMMENT
'调动后职位',
                                      `removeDate` date NULL DEFAULT NULL COMMENT '调动
日期',
                                      `reason` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '调动原因',
                                      `remark` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '备注',
                                      PRIMARY KEY (`id`) USING BTREE,
                                      INDEX `eid`(`eid`) USING BTREE,
                                      CONSTRAINT `t_employee_remove_ibfk_1` FOREIGN
KEY (`eid`) REFERENCES `t_employee` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Dynamic;

-- ----------------------------
-- Table structure for t_employee_train
-- ----------------------------
DROP TABLE IF EXISTS `t_employee_train`;
CREATE TABLE `t_employee_train`  (
                                      `id` int(11) NOT NULL AUTO_INCREMENT COMMENT
'id',
                                      `eid` int(11) NULL DEFAULT NULL COMMENT '员工编号',
                                      `trainDate` date NULL DEFAULT NULL COMMENT '培训日
期',
                                      `trainContent` varchar(255) CHARACTER SET utf8
COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '培训内容',
                                      `remark` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '备注',
                                     PRIMARY KEY (`id`) USING BTREE,
                                     INDEX `eid`(`eid`) USING BTREE,
                                     CONSTRAINT `t_employee_train_ibfk_1` FOREIGN KEY
(`eid`) REFERENCES `t_employee` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Dynamic;

-- ----------------------------
-- Table structure for t_joblevel
-- ----------------------------
DROP TABLE IF EXISTS `t_joblevel`;
CREATE TABLE `t_joblevel`  (
                                `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                                `name` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '职称名称',
                                `titleLevel` enum('正高级','副高级','中级','初级','员级')
CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '职称等级',
```

```
                              `createDate` timestamp(0) NULL DEFAULT
CURRENT_TIMESTAMP(0) COMMENT '创建时间',
                              `enabled` tinyint(1) NULL DEFAULT 1 COMMENT '是否启用',
                              PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 33 CHARACTER SET = utf8 COLLATE = utf8_general_ci
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_joblevel
-- ----------------------------
INSERT INTO `t_joblevel` VALUES (1, '教授', '正高级', '2020-03-31 16:20:34', 1);
INSERT INTO `t_joblevel` VALUES (2, '副教授', '副高级', '2020-03-31 16:20:34', 1);
INSERT INTO `t_joblevel` VALUES (3, '助教', '初级', '2020-03-31 16:20:34', 1);
INSERT INTO `t_joblevel` VALUES (4, '讲师', '中级', '2020-03-31 16:20:34', 0);
INSERT INTO `t_joblevel` VALUES (5, '初级工程师', '初级', '2020-03-31 16:20:34', 1);
INSERT INTO `t_joblevel` VALUES (6, '中级工程师', '中级', '2020-03-31 16:20:34', 1);
INSERT INTO `t_joblevel` VALUES (7, '高级工程师', '副高级', '2020-03-31 16:20:34', 1);
INSERT INTO `t_joblevel` VALUES (8, '骨灰级工程师', '正高级', '2020-03-31 16:20:34', 1);

-- ----------------------------
-- Table structure for t_mail_log
-- ----------------------------
DROP TABLE IF EXISTS `t_mail_log`;
CREATE TABLE `t_mail_log`  (
                              `msgId` varchar(64) CHARACTER SET utf8mb4 COLLATE
utf8mb4_bin NOT NULL COMMENT '消息id',
                              `eid` int(11) NULL DEFAULT NULL COMMENT '接收员工id',
                              `status` int(1) NULL DEFAULT NULL COMMENT '状态（0:消息投
递中 1:投递成功 2:投递失败）',
                              `routeKey` varchar(20) CHARACTER SET utf8mb4 COLLATE
utf8mb4_bin NULL DEFAULT NULL COMMENT '路由键',
                              `exchange` varchar(20) CHARACTER SET utf8mb4 COLLATE
utf8mb4_bin NULL DEFAULT NULL COMMENT '交换机',
                              `count` int(1) NULL DEFAULT NULL COMMENT '重试次数',
                              `tryTime` datetime(0) NULL DEFAULT NULL COMMENT '重试时
间',
                              `createTime` datetime(0) NULL DEFAULT NULL COMMENT '创建
时间',
                              `updateTime` datetime(0) NULL DEFAULT NULL COMMENT '更新
时间',
                              UNIQUE INDEX `msgId`(`msgId`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_bin ROW_FORMAT = Dynamic;

-- ----------------------------
-- Table structure for t_menu
-- ----------------------------
DROP TABLE IF EXISTS `t_menu`;
CREATE TABLE `t_menu`  (
                           `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                           `url` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT 'url',
                           `path` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT 'path',
                           `component` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '组件',
                           `name` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '菜单名',
```

```
                          `iconCls` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '图标',
                          `keepAlive` tinyint(1) NULL DEFAULT NULL COMMENT '是否保持激
活',
                          `requireAuth` tinyint(1) NULL DEFAULT NULL COMMENT '是否要求
权限',
                          `parentId` int(11) NULL DEFAULT NULL COMMENT '父id',
                          `enabled` tinyint(1) NULL DEFAULT 1 COMMENT '是否启用',
                          PRIMARY KEY (`id`) USING BTREE,
                          INDEX `parentId`(`parentId`) USING BTREE,
                          CONSTRAINT `menu_ibfk_1` FOREIGN KEY (`parentId`)
REFERENCES `t_menu` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB AUTO_INCREMENT = 28 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_bin
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_menu
-- ----------------------------
INSERT INTO `t_menu` VALUES (1, '/', NULL, NULL, '所有', NULL, NULL, NULL, NULL, 1);
INSERT INTO `t_menu` VALUES (2, '/', '/home', 'Home', '员工资料', 'fa fa-user-circle-o',
NULL, 1, 1, 1);
INSERT INTO `t_menu` VALUES (3, '/', '/home', 'Home', '人事管理', 'fa fa-address-card-
o', NULL, 1, 1, 1);
INSERT INTO `t_menu` VALUES (4, '/', '/home', 'Home', '薪资管理', 'fa fa-money', NULL,
1, 1, 1);
INSERT INTO `t_menu` VALUES (5, '/', '/home', 'Home', '统计管理', 'fa fa-bar-chart',
NULL, 1, 1, 1);
INSERT INTO `t_menu` VALUES (6, '/', '/home', 'Home', '系统管理', 'fa fa-windows', NULL,
1, 1, 1);
INSERT INTO `t_menu` VALUES (7, '/employee/basic/**', '/emp/basic', 'EmpBasic', '基本资
料', NULL, NULL, 1, 2, 1);
INSERT INTO `t_menu` VALUES (8, '/employee/advanced/**', '/emp/adv', 'EmpAdv', '高级资
料', NULL, NULL, 1, 2, 1);
INSERT INTO `t_menu` VALUES (9, '/personnel/emp/**', '/per/emp', 'PerEmp', '员工资料',
NULL, NULL, 1, 3, 1);
INSERT INTO `t_menu` VALUES (10, '/personnel/ec/**', '/per/ec', 'PerEc', '员工奖惩',
NULL, NULL, 1, 3, 1);
INSERT INTO `t_menu` VALUES (11, '/personnel/train/**', '/per/train', 'PerTrain', '员工
培训', NULL, NULL, 1, 3, 1);
INSERT INTO `t_menu` VALUES (12, '/personnel/salary/**', '/per/salary', 'PerSalary',
'员工调薪', NULL, NULL, 1, 3, 1);
INSERT INTO `t_menu` VALUES (13, '/personnel/remove/**', '/per/mv', 'PerMv', '员工调动',
NULL, NULL, 1, 3, 1);
INSERT INTO `t_menu` VALUES (14, '/salary/sob/**', '/sal/sob', 'SalSob', '工资账套管理',
NULL, NULL, 1, 4, 1);
INSERT INTO `t_menu` VALUES (15, '/salary/sobcfg/**', '/sal/sobcfg', 'SalSobCfg', '员工
账套设置', NULL, NULL, 1, 4, 1);
INSERT INTO `t_menu` VALUES (16, '/salary/table/**', '/sal/table', 'SalTable', '工资表管
理', NULL, NULL, 1, 4, 1);
INSERT INTO `t_menu` VALUES (17, '/salary/month/**', '/sal/month', 'SalMonth', '月末处
理', NULL, NULL, 1, 4, 1);
INSERT INTO `t_menu` VALUES (18, '/salary/search/**', '/sal/search', 'SalSearch', '工资
表查询', NULL, NULL, 1, 4, 1);
INSERT INTO `t_menu` VALUES (19, '/statistics/all/**', '/sta/all', 'StaAll', '综合信息统
计', NULL, NULL, 1, 5, 1);
INSERT INTO `t_menu` VALUES (20, '/statistics/score/**', '/sta/score', 'StaScore', '员
工积分统计', NULL, NULL, 1, 5, 1);
```

```
INSERT INTO `t_menu` VALUES (21, '/statistics/personnel/**', '/sta/pers', 'StaPers',
'人事信息统计', NULL, NULL, 1, 5, 1);
INSERT INTO `t_menu` VALUES (22, '/statistics/recored/**', '/sta/record', 'StaRecord',
'人事记录统计', NULL, NULL, 1, 5, 1);
INSERT INTO `t_menu` VALUES (23, '/system/basic/**', '/sys/basic', 'SysBasic', '基础信息
设置', NULL, NULL, 1, 6, 1);
INSERT INTO `t_menu` VALUES (24, '/system/cfg/**', '/sys/cfg', 'SysCfg', '系统管理',
NULL, NULL, 1, 6, 1);
INSERT INTO `t_menu` VALUES (25, '/system/log/**', '/sys/log', 'SysLog', '操作日志管理',
NULL, NULL, 1, 6, 1);
INSERT INTO `t_menu` VALUES (26, '/system/admin/**', '/sys/admin', 'SysAdmin', '操作员管
理', NULL, NULL, 1, 6, 1);
INSERT INTO `t_menu` VALUES (27, '/system/data/**', '/sys/data', 'SysData', '备份恢复数
据库', NULL, NULL, 1, 6, 1);
INSERT INTO `t_menu` VALUES (28, '/system/init/**', '/sys/init', 'SysInit', '初始化数据
库', NULL, NULL, 1, 6, 1);

-- ----------------------------
-- Table structure for t_menu_role
-- ----------------------------
DROP TABLE IF EXISTS `t_menu_role`;
CREATE TABLE `t_menu_role`  (
                                `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                                `mid` int(11) NULL DEFAULT NULL COMMENT '菜单id',
                                `rid` int(11) NULL DEFAULT NULL COMMENT '权限id',
                                PRIMARY KEY (`id`) USING BTREE,
                                INDEX `mid`(`mid`) USING BTREE,
                                INDEX `rid`(`rid`) USING BTREE,
                                CONSTRAINT `t_menu_role_ibfk_1` FOREIGN KEY (`mid`)
REFERENCES `t_menu` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,
                                CONSTRAINT `t_menu_role_ibfk_2` FOREIGN KEY (`rid`)
REFERENCES `t_role` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB AUTO_INCREMENT = 747 CHARACTER SET = utf8mb4 COLLATE =
utf8mb4_general_ci ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_menu_role
-- ----------------------------
INSERT INTO `t_menu_role` VALUES (1, 7, 3);
INSERT INTO `t_menu_role` VALUES (2, 7, 6);
INSERT INTO `t_menu_role` VALUES (3, 9, 6);
INSERT INTO `t_menu_role` VALUES (4, 10, 6);
INSERT INTO `t_menu_role` VALUES (5, 11, 6);
INSERT INTO `t_menu_role` VALUES (6, 12, 6);
INSERT INTO `t_menu_role` VALUES (7, 13, 6);
INSERT INTO `t_menu_role` VALUES (8, 14, 6);
INSERT INTO `t_menu_role` VALUES (9, 15, 6);
INSERT INTO `t_menu_role` VALUES (10, 16, 6);
INSERT INTO `t_menu_role` VALUES (11, 17, 6);
INSERT INTO `t_menu_role` VALUES (12, 18, 6);
INSERT INTO `t_menu_role` VALUES (13, 19, 6);
INSERT INTO `t_menu_role` VALUES (14, 20, 6);
INSERT INTO `t_menu_role` VALUES (15, 21, 6);
INSERT INTO `t_menu_role` VALUES (16, 22, 6);
INSERT INTO `t_menu_role` VALUES (17, 23, 6);
INSERT INTO `t_menu_role` VALUES (18, 25, 6);
INSERT INTO `t_menu_role` VALUES (19, 26, 6);
INSERT INTO `t_menu_role` VALUES (20, 27, 6);
```

```
INSERT INTO `t_menu_role` VALUES (21, 28, 6);
INSERT INTO `t_menu_role` VALUES (22, 24, 6);
INSERT INTO `t_menu_role` VALUES (26, 7, 2);
INSERT INTO `t_menu_role` VALUES (27, 8, 2);
INSERT INTO `t_menu_role` VALUES (28, 9, 2);
INSERT INTO `t_menu_role` VALUES (29, 10, 2);
INSERT INTO `t_menu_role` VALUES (30, 12, 2);
INSERT INTO `t_menu_role` VALUES (31, 13, 2);
INSERT INTO `t_menu_role` VALUES (32, 7, 1);
INSERT INTO `t_menu_role` VALUES (33, 8, 1);
INSERT INTO `t_menu_role` VALUES (34, 9, 1);
INSERT INTO `t_menu_role` VALUES (35, 10, 1);
INSERT INTO `t_menu_role` VALUES (36, 11, 1);
INSERT INTO `t_menu_role` VALUES (37, 12, 1);
INSERT INTO `t_menu_role` VALUES (38, 13, 1);
INSERT INTO `t_menu_role` VALUES (39, 14, 1);
INSERT INTO `t_menu_role` VALUES (40, 15, 1);
INSERT INTO `t_menu_role` VALUES (41, 16, 1);
INSERT INTO `t_menu_role` VALUES (42, 17, 1);
INSERT INTO `t_menu_role` VALUES (43, 18, 1);
INSERT INTO `t_menu_role` VALUES (44, 19, 1);
INSERT INTO `t_menu_role` VALUES (45, 20, 1);
INSERT INTO `t_menu_role` VALUES (46, 21, 1);
INSERT INTO `t_menu_role` VALUES (47, 22, 1);
INSERT INTO `t_menu_role` VALUES (48, 23, 1);
INSERT INTO `t_menu_role` VALUES (49, 24, 1);
INSERT INTO `t_menu_role` VALUES (50, 25, 1);
INSERT INTO `t_menu_role` VALUES (51, 26, 1);
INSERT INTO `t_menu_role` VALUES (52, 27, 1);
INSERT INTO `t_menu_role` VALUES (53, 28, 1);
INSERT INTO `t_menu_role` VALUES (346, 11, 4);
INSERT INTO `t_menu_role` VALUES (347, 8, 4);
INSERT INTO `t_menu_role` VALUES (348, 7, 4);

-- ----------------------------
-- Table structure for t_nation
-- ----------------------------
DROP TABLE IF EXISTS `t_nation`;
CREATE TABLE `t_nation`  (
                             `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                             `name` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '民族',
                             PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 56 CHARACTER SET = utf8 COLLATE = utf8_general_ci
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_nation
-- ----------------------------
INSERT INTO `t_nation` VALUES (1, '汉族');
INSERT INTO `t_nation` VALUES (2, '蒙古族');
INSERT INTO `t_nation` VALUES (3, '回族');
INSERT INTO `t_nation` VALUES (4, '藏族');
INSERT INTO `t_nation` VALUES (5, '维吾尔族');
INSERT INTO `t_nation` VALUES (6, '苗族');
INSERT INTO `t_nation` VALUES (7, '彝族');
INSERT INTO `t_nation` VALUES (8, '壮族');
INSERT INTO `t_nation` VALUES (9, '布依族');
```

```
INSERT INTO `t_nation` VALUES (10, '朝鲜族');
INSERT INTO `t_nation` VALUES (11, '满族');
INSERT INTO `t_nation` VALUES (12, '侗族');
INSERT INTO `t_nation` VALUES (13, '瑶族');
INSERT INTO `t_nation` VALUES (14, '白族');
INSERT INTO `t_nation` VALUES (15, '土家族');
INSERT INTO `t_nation` VALUES (16, '哈尼族');
INSERT INTO `t_nation` VALUES (17, '哈萨克族');
INSERT INTO `t_nation` VALUES (18, '傣族');
INSERT INTO `t_nation` VALUES (19, '黎族');
INSERT INTO `t_nation` VALUES (20, '傈僳族');
INSERT INTO `t_nation` VALUES (21, '佤族');
INSERT INTO `t_nation` VALUES (22, '畲族');
INSERT INTO `t_nation` VALUES (23, '高山族');
INSERT INTO `t_nation` VALUES (24, '拉祜族');
INSERT INTO `t_nation` VALUES (25, '水族');
INSERT INTO `t_nation` VALUES (26, '东乡族');
INSERT INTO `t_nation` VALUES (27, '纳西族');
INSERT INTO `t_nation` VALUES (28, '景颇族');
INSERT INTO `t_nation` VALUES (29, '柯尔克孜族');
INSERT INTO `t_nation` VALUES (30, '土族');
INSERT INTO `t_nation` VALUES (31, '达斡尔族');
INSERT INTO `t_nation` VALUES (32, '仫佬族');
INSERT INTO `t_nation` VALUES (33, '羌族');
INSERT INTO `t_nation` VALUES (34, '布朗族');
INSERT INTO `t_nation` VALUES (35, '撒拉族');
INSERT INTO `t_nation` VALUES (36, '毛难族');
INSERT INTO `t_nation` VALUES (37, '仡佬族');
INSERT INTO `t_nation` VALUES (38, '锡伯族');
INSERT INTO `t_nation` VALUES (39, '阿昌族');
INSERT INTO `t_nation` VALUES (40, '普米族');
INSERT INTO `t_nation` VALUES (41, '塔吉克族');
INSERT INTO `t_nation` VALUES (42, '怒族');
INSERT INTO `t_nation` VALUES (43, '乌孜别克族');
INSERT INTO `t_nation` VALUES (44, '俄罗斯族');
INSERT INTO `t_nation` VALUES (45, '鄂温克族');
INSERT INTO `t_nation` VALUES (46, '崩龙族');
INSERT INTO `t_nation` VALUES (47, '保安族');
INSERT INTO `t_nation` VALUES (48, '裕固族');
INSERT INTO `t_nation` VALUES (49, '京族');
INSERT INTO `t_nation` VALUES (50, '塔塔尔族');
INSERT INTO `t_nation` VALUES (51, '独龙族');
INSERT INTO `t_nation` VALUES (52, '鄂伦春族');
INSERT INTO `t_nation` VALUES (53, '赫哲族');
INSERT INTO `t_nation` VALUES (54, '门巴族');
INSERT INTO `t_nation` VALUES (55, '珞巴族');
INSERT INTO `t_nation` VALUES (56, '基诺族');

-- ----------------------------
-- Table structure for t_oplog
-- ----------------------------
DROP TABLE IF EXISTS `t_oplog`;
CREATE TABLE `t_oplog`  (
                            `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                            `addDate` date NULL DEFAULT NULL COMMENT '添加日期',
                            `operate` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '操作内容',
                            `adminid` int(11) NULL DEFAULT NULL COMMENT '操作员ID',
```

```
                            PRIMARY KEY (`id`) USING BTREE,
                            INDEX `adminid`(`adminid`) USING BTREE,
                            CONSTRAINT `t_oplog_ibfk_1` FOREIGN KEY (`adminid`)
REFERENCES `t_admin` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Dynamic;

-- ----------------------------
-- Table structure for t_politics_status
-- ----------------------------
DROP TABLE IF EXISTS `t_politics_status`;
CREATE TABLE `t_politics_status`  (
                                      `id` int(11) NOT NULL AUTO_INCREMENT COMMENT
'id',
                                      `name` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '政治面貌',
                                      PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 13 CHARACTER SET = utf8 COLLATE = utf8_general_ci
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_politics_status
-- ----------------------------
INSERT INTO `t_politics_status` VALUES (1, '中共党员');
INSERT INTO `t_politics_status` VALUES (2, '中共预备党员');
INSERT INTO `t_politics_status` VALUES (3, '共青团员');
INSERT INTO `t_politics_status` VALUES (4, '民革党员');
INSERT INTO `t_politics_status` VALUES (5, '民盟盟员');
INSERT INTO `t_politics_status` VALUES (6, '民建会员');
INSERT INTO `t_politics_status` VALUES (7, '民进会员');
INSERT INTO `t_politics_status` VALUES (8, '农工党党员');
INSERT INTO `t_politics_status` VALUES (9, '致公党党员');
INSERT INTO `t_politics_status` VALUES (10, '九三学社社员');
INSERT INTO `t_politics_status` VALUES (11, '台盟盟员');
INSERT INTO `t_politics_status` VALUES (12, '无党派民主人士');
INSERT INTO `t_politics_status` VALUES (13, '普通公民');

-- ----------------------------
-- Table structure for t_position
-- ----------------------------
DROP TABLE IF EXISTS `t_position`;
CREATE TABLE `t_position`  (
                               `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                               `name` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '职位',
                               `createDate` timestamp(0) NULL DEFAULT
CURRENT_TIMESTAMP(0) COMMENT '创建时间',
                               `enabled` tinyint(1) NULL DEFAULT 1 COMMENT '是否启用',
                               PRIMARY KEY (`id`) USING BTREE,
                               UNIQUE INDEX `name`(`name`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 14 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_bin
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_position
-- ----------------------------
INSERT INTO `t_position` VALUES (1, '技术总监', '2020-03-31 16:20:34', 1);
INSERT INTO `t_position` VALUES (2, '运营总监', '2020-03-31 16:20:34', 1);
INSERT INTO `t_position` VALUES (3, '市场总监', '2020-03-31 16:20:34', 1);
```

```
INSERT INTO `t_position` VALUES (4, '研发工程师', '2020-03-31 16:20:34', 1);
INSERT INTO `t_position` VALUES (5, '运维工程师', '2020-03-31 16:20:34', 1);

-- ----------------------------
-- Table structure for t_role
-- ----------------------------
DROP TABLE IF EXISTS `t_role`;
CREATE TABLE `t_role`  (
                           `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                           `name` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '名称',
                           `nameZh` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '角色名称',
                           PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 9 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_bin
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_role
-- ----------------------------
INSERT INTO `t_role` VALUES (1, 'ROLE_manager', '部门经理');
INSERT INTO `t_role` VALUES (2, 'ROLE_personnel', '人事专员');
INSERT INTO `t_role` VALUES (3, 'ROLE_recruiter', '招聘主管');
INSERT INTO `t_role` VALUES (4, 'ROLE_train', '培训主管');
INSERT INTO `t_role` VALUES (5, 'ROLE_performance', '薪酬绩效主管');
INSERT INTO `t_role` VALUES (6, 'ROLE_admin', '系统管理员');
INSERT INTO `t_role` VALUES (8, 'ROLE_test', '测试角色');

-- ----------------------------
-- Table structure for t_salary
-- ----------------------------
DROP TABLE IF EXISTS `t_salary`;
CREATE TABLE `t_salary`  (
                             `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                             `basicSalary` int(11) NULL DEFAULT NULL COMMENT '基本工资',
                             `bonus` int(11) NULL DEFAULT NULL COMMENT '奖金',
                             `lunchSalary` int(11) NULL DEFAULT NULL COMMENT '午餐补助',
                             `trafficSalary` int(11) NULL DEFAULT NULL COMMENT '交通补
助',
                             `allSalary` int(11) NULL DEFAULT NULL COMMENT '应发工资',
                             `pensionBase` int(11) NULL DEFAULT NULL COMMENT '养老金基
数',
                             `pensionPer` float NULL DEFAULT NULL COMMENT '养老金比率',
                             `createDate` timestamp(0) NULL DEFAULT NULL COMMENT '启用
时间',
                             `medicalBase` int(11) NULL DEFAULT NULL COMMENT '医疗基数',
                             `medicalPer` float NULL DEFAULT NULL COMMENT '医疗保险比
率',
                             `accumulationFundBase` int(11) NULL DEFAULT NULL COMMENT
'公积金基数',
                             `accumulationFundPer` float NULL DEFAULT NULL COMMENT '公
积金比率',
                             `name` varchar(32) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '名称',
                             PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 23 CHARACTER SET = utf8 COLLATE = utf8_general_ci
ROW_FORMAT = Dynamic;
```

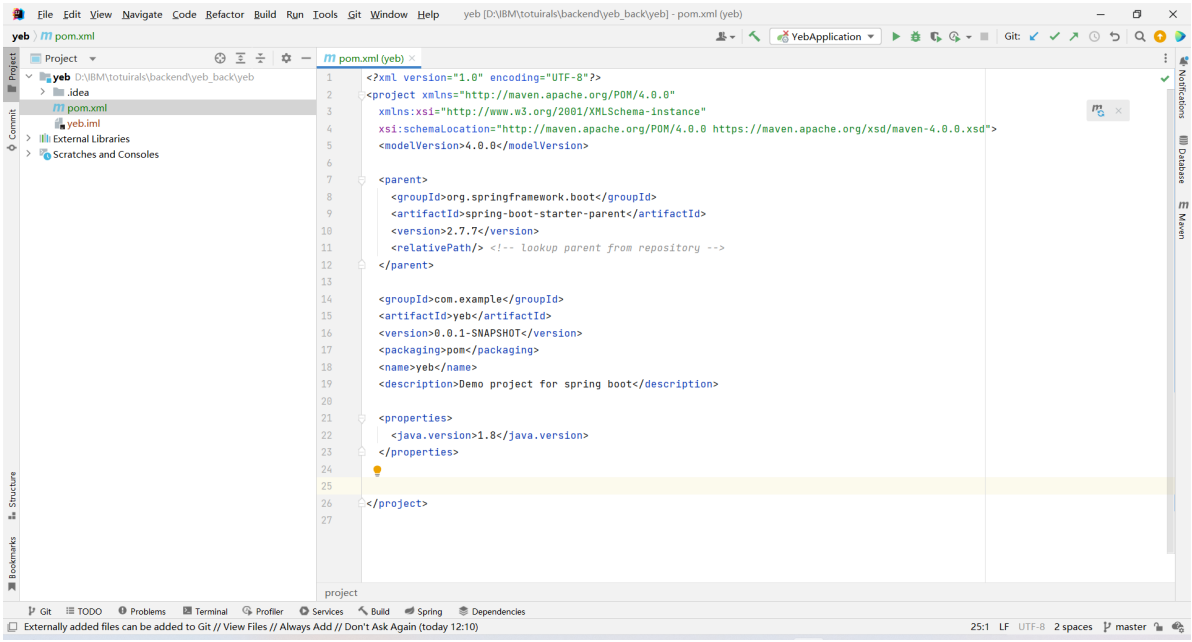
```
-- ----------------------------
-- Records of t_salary
-- ----------------------------
INSERT INTO `t_salary` VALUES (1, 8000, 500, 800, 400, NULL, 1000, 0.06, '2018-01-24
00:00:00', 1000, 0.06, 1000, 0.06, '市场部工资账套');
INSERT INTO `t_salary` VALUES (2, 3000, 500, 500, 500, NULL, 1800, 0.06, '2018-01-01
00:00:00', 2200, 0.06, 3200, 0.06, '人事部工资账套');
INSERT INTO `t_salary` VALUES (3, 9000, 500, 1000, 1000, NULL, 3000, 0.06, '2018-01-25
00:00:00', 3000, 0.06, 3000, 0.06, '运维部工资账套');
INSERT INTO `t_salary` VALUES (4, 5000, 500, 500, 500, NULL, 500, 0.06, '2020-04-10
14:15:45', 500, 0.06, 500, 0.06, '财务部工资账套');

-- ----------------------------
-- Table structure for t_salary_adjust
-- ----------------------------
DROP TABLE IF EXISTS `t_salary_adjust`;
CREATE TABLE `t_salary_adjust`  (
                                    `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                                    `eid` int(11) NULL DEFAULT NULL COMMENT '员工ID',
                                    `asDate` date NULL DEFAULT NULL COMMENT '调薪日期',
                                    `beforeSalary` int(11) NULL DEFAULT NULL COMMENT
'调前薪资',
                                    `afterSalary` int(11) NULL DEFAULT NULL COMMENT '调
后薪资',
                                    `reason` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '调薪原因',
                                    `remark` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '备注',
                                    PRIMARY KEY (`id`) USING BTREE,
                                    INDEX `eid`(`eid`) USING BTREE,
                                    CONSTRAINT `t_salary_adjust_ibfk_1` FOREIGN KEY
(`eid`) REFERENCES `t_employee` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Dynamic;

-- ----------------------------
-- Table structure for t_sys_msg
-- ----------------------------
DROP TABLE IF EXISTS `t_sys_msg`;
CREATE TABLE `t_sys_msg`  (
                              `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
                              `mid` int(11) NULL DEFAULT NULL COMMENT '消息id',
                              `type` int(11) NULL DEFAULT 0 COMMENT '0表示群发消息',
                              `adminid` int(11) NULL DEFAULT NULL COMMENT '这条消息是给谁
的',
                              `state` int(11) NULL DEFAULT 0 COMMENT '0 未读 1 已读',
                              PRIMARY KEY (`id`) USING BTREE,
                              INDEX `adminid`(`adminid`) USING BTREE,
                              INDEX `mid`(`mid`) USING BTREE,
                              CONSTRAINT `t_sys_msg_ibfk_1` FOREIGN KEY (`mid`)
REFERENCES `t_sys_msg_content` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,
                              CONSTRAINT `t_sys_msg_ibfk_2` FOREIGN KEY (`adminid`)
REFERENCES `t_admin` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_bin ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_sys_msg
-- ----------------------------
INSERT INTO `t_sys_msg` VALUES (1, 1, 0, 1, 1);
```

```
INSERT INTO `t_sys_msg` VALUES (2, 1, 0, 2, 1);
INSERT INTO `t_sys_msg` VALUES (3, 1, 0, 3, 1);
INSERT INTO `t_sys_msg` VALUES (4, 1, 0, 4, 0);
INSERT INTO `t_sys_msg` VALUES (5, 1, 0, 5, 0);
INSERT INTO `t_sys_msg` VALUES (6, 2, 0, 1, 1);
INSERT INTO `t_sys_msg` VALUES (7, 2, 0, 2, 1);
INSERT INTO `t_sys_msg` VALUES (8, 2, 0, 3, 1);
INSERT INTO `t_sys_msg` VALUES (9, 2, 0, 4, 0);
INSERT INTO `t_sys_msg` VALUES (10, 2, 0, 5, 0);
INSERT INTO `t_sys_msg` VALUES (11, 3, 0, 1, 1);
INSERT INTO `t_sys_msg` VALUES (12, 3, 0, 2, 1);
INSERT INTO `t_sys_msg` VALUES (13, 3, 0, 3, 1);
INSERT INTO `t_sys_msg` VALUES (14, 3, 0, 4, 0);
INSERT INTO `t_sys_msg` VALUES (15, 3, 0, 5, 0);

-- ----------------------------
-- Table structure for t_sys_msg_content
-- ----------------------------
DROP TABLE IF EXISTS `t_sys_msg_content`;
CREATE TABLE `t_sys_msg_content`  (
                                      `id` int(11) NOT NULL AUTO_INCREMENT COMMENT
'id',
                                      `title` varchar(64) CHARACTER SET utf8 COLLATE
utf8_general_ci NULL DEFAULT NULL COMMENT '标题',
                                      `message` varchar(255) CHARACTER SET utf8
COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '内容',
                                      `createDate` timestamp(0) NOT NULL DEFAULT
CURRENT_TIMESTAMP(0) COMMENT '创建时间',
                                      PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 18 CHARACTER SET = utf8 COLLATE = utf8_general_ci
ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of t_sys_msg_content
-- ----------------------------
INSERT INTO `t_sys_msg_content` VALUES (1, '通知标题1', '通知内容1', '2020-03-31
16:20:34');
INSERT INTO `t_sys_msg_content` VALUES (2, '通知标题2', '通知内容2', '2020-03-31
16:20:34');
INSERT INTO `t_sys_msg_content` VALUES (3, '通知标题3', '通知内容3', '2020-03-31
16:20:34');

-- ----------------------------
-- Procedure structure for addDep
-- ----------------------------
DROP PROCEDURE IF EXISTS `addDep`;
delimiter ;;
CREATE PROCEDURE `addDep`(in depName varchar(32),in parentId int,in enabled
boolean,out result int,out result2 int)
begin
  declare did int;
  declare pDepPath varchar(64);
insert into t_department set name=depName,parentId=parentId,enabled=enabled;
select row_count() into result;
select last_insert_id() into did;
set result2=did;
select depPath into pDepPath from t_department where id=parentId;
update t_department set depPath=concat(pDepPath,'.',did) where id=did;
```

```
update t_department set isParent=true where id=parentId;
end
;;
delimiter ;

-- ----------------------------
-- Procedure structure for deleteDep
-- ----------------------------
DROP PROCEDURE IF EXISTS `deleteDep`;
delimiter ;;
CREATE PROCEDURE `deleteDep`(in did int,out result int)
begin
  declare ecount int;
  declare pid int;
  declare pcount int;
  declare a int;
select count(*) into a from t_department where id=did and isParent=false;
if a=0 then set result=-2;
else
select count(*) into ecount from t_employee where departmentId=did;
if ecount>0 then set result=-1;
else
select parentId into pid from t_department where id=did;
delete from t_department where id=did and isParent=false;
select row_count() into result;
select count(*) into pcount from t_department where parentId=pid;
if pcount=0 then update t_department set isParent=false where id=pid;
end if;
end if;
end if;
end
;;
delimiter ;

SET FOREIGN_KEY_CHECKS = 1;
```

3. 创建yeb项目

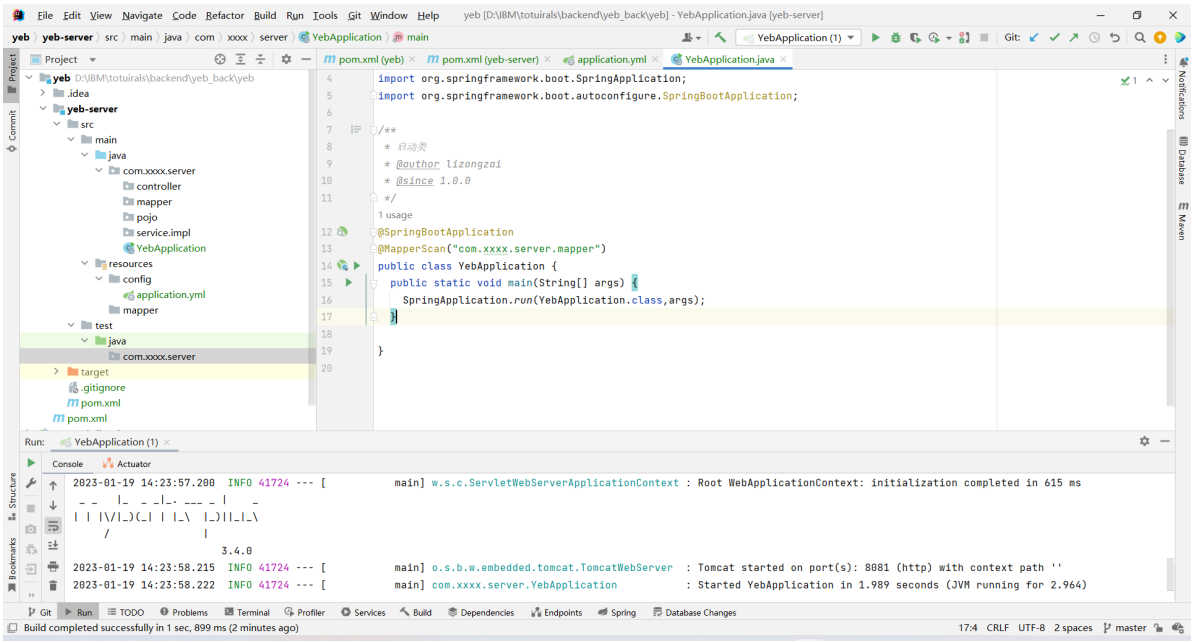
3.1 父工程



父工程：引入pom依赖包

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <modules>
    <module>yeb-server</module>
    <module>yeb-generator</module>
  </modules>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.7</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <groupId>com.xxxx</groupId>
  <artifactId>yeb</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>yeb</name>
  <description>demo project for spring boot</description>

  <properties>
    <java.version>1.8</java.version>
  </properties>
</project>
```

3.2 子工程



`application.yml`

```yaml
server:
  port: 8081

spring:
  main:
    allow-circular-references: true
  mvc:
    pathmatch:
      matching-strategy: ANT_PATH_MATCHER
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: "jdbc:mysql://localhost:3306/yeb_back?useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai"
    username: root
    password: password
    redis:
      timeout: 10000ms
      host: localhost
      port: 6379
      database: 0 # 选择哪个库，默认0库
      password: root
      lettuce:
        pool:
          max-active: 1024 # 最大连接数，默认 8
          max-wait: 10000ms # 最大连接阻塞等待时间，单位毫秒，默认 -1
          max-idle: 200 # 最大空闲连接，默认 8
          min-idle: 5
    # rabbitmq配置
    rabbitmq:
      # 用户名
      username: guest
      # 密码
      password: guest
      # 服务器地址
```

```yaml
      host: localhost
      # 端口
      port: 5672
      # 消息确认回调
      publisher-confirm-type: correlated
      # 消息失败回调
      publisher-returns: true

    hikari:
      # 连接池名
      pool-name: DateHikariCP
      # 最小空闲连接数
      minimum-idle: 5
      # 空闲连接存活最大时间，默认值为6000000
      idle-timeout: 180000
      # 最大连接数,默认10
      maximum-pool-size: 10
      # 从连接池返回的连接的自动提交
      auto-commit: true
      # 连接最大存活时间，0表示永久存活，默认1800000(30分钟)
      max-lifetime: 1800000
      # 连接超时时间, 默认30000(30秒)
      connection-timeout: 30000
      # 测试连接是否可用的查询语句
      connection-test-query: SELECT 1

# Mybatis-plus配置
mybatis-plus:
  # 配置Mapper映射文件
  mapper-locations: classpath*:/mapper/*Mapper.xml
  #配置mybatis数据返回类型别名
  type-aliases-package: com.ibm.server.pojo
  configuration:
    # 自动驼峰命名
    map-underscore-to-camel-case: false

# Mybatis SQL打印(方法接口所在的包，不是Mapper.xml所在的包)
logging:
  level:
    com.xxxx.server.mapper: debug

jwt:
  # Jwt存储的请求头
  tokenHeader: Authorization
  # Jwt加密秘钥
  secret: yeb-secret
  # Jwt 的超期限时间（60*60）*24
  expiration: 604800
  # Jwt负载中拿到开头
  tokenHead: Bearer
```

创建启动类

```java
package com.xxxx.server;

import org.mybatis.spring.annotation.MapperScan;
```

```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 * 启动类
 * @author lizongzai
 * @since 1.0.0
 */
@SpringBootApplication
@MapperScan("com.xxxx.server.mapper")
public class YebApplication {
  public static void main(String[] args) {
    SpringApplication.run(YebApplication.class,args);
  }

}
```

4. 代码生成器

4.1 官方地址

https://baomidou.com/pages/d357af/

4.2 引入maven依赖包

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <artifactId>yeb</artifactId>
    <groupId>com.xxxx</groupId>
    <version>0.0.1-SNAPSHOT</version>
  </parent>

  <groupId>com.xxxx</groupId>
  <artifactId>yeb-generator</artifactId>
```

```
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>yeb-generator</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>


  <dependencies>
    <!--Web 依赖-->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <!--mybatis-plus 依赖-->
    <dependency>
      <groupId>com.baomidou</groupId>
      <artifactId>mybatis-plus-boot-starter</artifactId>
      <version>3.4.0</version>
    </dependency>
    <!--mybatis-plus 代码生成器依赖-->
    <dependency>
      <groupId>com.baomidou</groupId>
      <artifactId>mybatis-plus-generator</artifactId>
      <version>3.4.1</version>
    </dependency>
    <!--freemarker 依赖-->
    <dependency>
      <groupId>org.freemarker</groupId>
      <artifactId>freemarker</artifactId>
      <version>2.3.28</version>
    </dependency>
    <!--mysql 依赖-->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
    </dependency>
  </dependencies>
</project>
```

4.3 生成代码

```
package com.xxxx.generator;

import com.baomidou.mybatisplus.core.exceptions.MybatisPlusException;
import com.baomidou.mybatisplus.core.toolkit.StringPool;
import com.baomidou.mybatisplus.core.toolkit.StringUtils;
import com.baomidou.mybatisplus.generator.AutoGenerator;
import com.baomidou.mybatisplus.generator.InjectionConfig;
import com.baomidou.mybatisplus.generator.config.*;
```

```
import com.baomidou.mybatisplus.generator.config.po.TableInfo;
import com.baomidou.mybatisplus.generator.config.rules.NamingStrategy;
import com.baomidou.mybatisplus.generator.engine.FreemarkerTemplateEngine;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class CodeGenerator {

    /**
     * <p>
     * 读取控制台内容
     * </p>
     */
    public static String scanner(String tip) {
        Scanner scanner = new Scanner(System.in);
        StringBuilder help = new StringBuilder();
        help.append("请输入" + tip + "：");
        System.out.println(help.toString());
        if (scanner.hasNext()) {
            String ipt = scanner.next();
            if (StringUtils.isNotBlank(ipt)) {
                return ipt;
            }
        }
        throw new MybatisPlusException("请输入正确的" + tip + "！");
    }

    public static void main(String[] args) {
        // 代码生成器
        AutoGenerator mpg = new AutoGenerator();

        // 全局配置
        GlobalConfig gc = new GlobalConfig();
        String projectPath = System.getProperty("user.dir");
        gc.setOutputDir(projectPath + "/yeb-generator/src/main/java");
        //作者
        gc.setAuthor("lizongzai");
        //打开输出目录
        gc.setOpen(false);
        //xml开启BaseResultMap
        gc.setBaseResultMap(true);
        //xml 开启BaseColumnList
        gc.setBaseColumnList(true);
        //实体属性 Swagger2 注解
        gc.setSwagger2(true);
        mpg.setGlobalConfig(gc);

        // 数据源配置
        DataSourceConfig dsc = new DataSourceConfig();
        dsc.setUrl("jdbc:mysql://localhost:3306/yeb_back?
useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai");
        // dsc.setSchemaName("public");
        dsc.setDriverName("com.mysql.cj.jdbc.Driver");
        dsc.setUsername("root");
        dsc.setPassword("password");
        mpg.setDataSource(dsc);
```

```
        // 包配置
        PackageConfig pc = new PackageConfig();
        //pc.setModuleName(scanner("模块名"));
        pc.setParent("com.xxxx.server")
                .setEntity("pojo")
                .setMapper("mapper")
                .setService("service")
                .setServiceImpl("service.impl")
                .setController("controller");
        mpg.setPackageInfo(pc);

        // 自定义配置
        InjectionConfig cfg = new InjectionConfig() {
            @Override
            public void initMap() {
                // to do nothing
            }
        };

        // 如果模板引擎是 freemarker
        String templatePath = "/templates/mapper.xml.ftl";
        // 如果模板引擎是 velocity
        // String templatePath = "/templates/mapper.xml.vm";

        // 自定义输出配置
        List<FileOutConfig> focList = new ArrayList<>();
        // 自定义配置会被优先输出
        focList.add(new FileOutConfig(templatePath) {
            @Override
            public String outputFile(TableInfo tableInfo) {
                // 自定义输出文件名 ， 如果你 Entity 设置了前后缀、此处注意 xml 的名称会跟着发生
变化！！
                return projectPath + "/yeb-generator/src/main/resources/mapper/" +
pc.getModuleName()
                        + "/" + tableInfo.getEntityName() + "Mapper" +
StringPool.DOT_XML;
            }
        });
        /*
        cfg.setFileCreate(new IFileCreate() {
            @Override
            public boolean isCreate(ConfigBuilder configBuilder, FileType fileType,
String filePath) {
                // 判断自定义文件夹是否需要创建
                checkDir("调用默认方法创建的目录，自定义目录用");
                if (fileType == FileType.MAPPER) {
                    // 已经生成 mapper 文件判断存在，不想重新生成返回 false
                    return !new File(filePath).exists();
                }
                // 允许生成模板文件
                return true;
            }
        });
        */
        cfg.setFileOutConfigList(focList);
        mpg.setCfg(cfg);

        // 配置模板
```

```
        TemplateConfig templateConfig = new TemplateConfig();

        // 配置自定义输出模板
        //指定自定义模板路径，注意不要带上.ftl/.vm, 会根据使用的模板引擎自动识别
        // templateConfig.setEntity("templates/entity2.java");
        // templateConfig.setService();
        // templateConfig.setController();

        templateConfig.setXml(null);
        mpg.setTemplate(templateConfig);

        // 策略配置
        StrategyConfig strategy = new StrategyConfig();
        //数据库表映射到实体的命名策略
        strategy.setNaming(NamingStrategy.underline_to_camel);
        //数据库表字段映射到实体的命名策略
        strategy.setColumnNaming(NamingStrategy.no_change);
        //strategy.setSuperEntityClass("你自己的父类实体,没有就不用设置!");
        //lombok模型
        strategy.setEntityLombokModel(true);
        //生成RestController
        strategy.setRestControllerStyle(true);
        // 公共父类
        //strategy.setSuperControllerClass("你自己的父类控制器,没有就不用设置!");
        // 写于父类中的公共字段
        //strategy.setSuperEntityColumns("id");
        strategy.setInclude(scanner("表名，多个英文逗号分割").split(","));
        strategy.setControllerMappingHyphenStyle(true);
        //表前缀
        strategy.setTablePrefix("t_");
        mpg.setStrategy(strategy);
        mpg.setTemplateEngine(new FreemarkerTemplateEngine());
        mpg.execute();
    }
}
```

4.4 拷贝代码到项目

忽略

4.5 引入swagger2依赖包

```xml
<!--swagger2 依赖-->
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.7.0</version>
</dependency>
<!--swagger 第三方ui依赖-->
<dependency>
  <groupId>com.github.xiaoymin</groupId>
  <artifactId>swagger-bootstrap-ui</artifactId>
  <version>1.9.6</version>
</dependency>
```

5. 登录功能

5.1 引入spring security依赖包

```xml
<!--security 依赖-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<!--JWT Token 依赖-->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
```

5.2 引入JWT配置

```yaml
jwt:
  # Jwt存储的请求头
  tokenHeader: Authorization
  # Jwt加密秘钥
  secret: yeb-secret
  # Jwt 的超期限时间（60*60）*24
  expiration: 604800
  # Jwt负载中拿到开头
  tokenHead: Bearer
```

5.3 引入JWT工具类(★★★)

```java
package com.xxxx.server.config.jwt;


import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import java.util.Date;
import java.util.HashMap;
```

```java
import java.util.Map;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;

@Component
public class JwtTokenUtil {

  private static final String CLAIM_KEY_USERNAME = "sub";
  private static final String CLAIM_KEY_CREATED = "created";
  @Value("${jwt.secret}")
  private String secret;
  @Value("${jwt.expiration}")
  private long expiration;

  /**
   * @return java.lang.String
   * @Author lizongzai
   * @Description //TODO 根据用户信息生成token
   * @Date 2023/1/4 21:13
   * @Param [userDetails]
   **/
  public String generateToken(UserDetails userDetails) {
    Map<String, Object> claims = new HashMap<>(16);
    claims.put(CLAIM_KEY_USERNAME, userDetails.getUsername());
    claims.put(CLAIM_KEY_CREATED, new Date());
    return generateToken(claims);
  }

  /**
   * @return java.lang.String
   * @Author lizongzai
   * @Description //TODO  根据负载生成JWT Token
   @Date 2023/1/4 21:13
    * @Param [claims]
   **/
  private String generateToken(Map<String, Object> claims) {
    return Jwts.builder()
        .setClaims(claims)
        .setExpiration(generateExpirationDate())
        .signWith(SignatureAlgorithm.HS512, secret)
        .compact();
  }

  /**
   * @return java.util.Date
   * @Author lizongzai
   * @Description //TODO 生成token过期时间
   * @Date 2023/1/4 21:13
   * @Param []
   **/
  private Date generateExpirationDate() {
    return new Date(System.currentTimeMillis() + expiration * 1000);
  }

  /**
   * @return java.lang.String
   * @Author lizongzai
```

```
     * @Description //TODO 从token中获取登录用户名
     * @Date 2023/1/4 21:13
     * @Param [token]
     **/
    public String getUserNameFromToken(String token) {
      String username = null;
      try {
        Claims claims = getClaimsFromToken(token);
        username = claims.getSubject();
      } catch (Exception e) {
        username = null;
      }
      return username;
    }

    /**
     * @return io.jsonwebtoken.Claims
     * @Author lizongzai
     * @Description //TODO 从token中获取JWT中的负载
     * @Date 2023/1/4 21:13
     * @Param [token]
     **/
    private Claims getClaimsFromToken(String token) {
      Claims claims = null;
      try {
        claims = Jwts.parser()
            .setSigningKey(secret)
            .parseClaimsJws(token)
            .getBody();
      } catch (Exception e) {
        e.printStackTrace();
      }
      return claims;
    }

    /**
     * @return java.lang.boolean
     * @Author lizongzai
     * @Description //TODO 验证 token 是否有效
     * @Date 2023/1/4 21:13
     * @Param [userDetails, token]
     **/
    public boolean validateToken(String token, UserDetails userDetails) {
      String userName = getUserNameFromToken(token);
      return userName.equals(userDetails.getUsername()) && !isTokenExpired(token);
    }

    /**
     * @return java.lang.boolean
     * @Author lizongzai
     * @Description //TODO 判断token是否失效
     * @Date 2023/1/4 21:13
     * @Param [token]
     **/
    private boolean isTokenExpired(String token) {
      Date expiredDate = getExpiredDateFromToken(token);
      return expiredDate.before(new Date());
    }
```

```
    /**
     * @return java.util.Date
     * @Author lizongzai
     * @Description //TODO 从token中获取过期时间
     * @Date 2023/1/4 21:13
     * @Param [token]
     **/
    private Date getExpiredDateFromToken(String token) {
        Claims claims = getClaimsFromToken(token);
        return claims.getExpiration();
    }

    /**
     * @return java.lang.boolean
     * @Author lizongzai
     * @Description //TODO 判断token是否可以被刷新
     * @Date 2023/1/4 21:13
     * @Param [token]
     **/
    public boolean canRefresh(String token) {
        return !isTokenExpired(token);
    }

    /**
     * @return java.lang.String
     * @Author lizongzai
     * @Description //TODO 刷新token
     * @Date 2023/1/4 21:13
     * @Param [token]
     **/
    public String refreshToken(String token) {
        Claims claims = getClaimsFromToken(token);
        claims.put(CLAIM_KEY_CREATED, new Date());
        return generateToken(claims);
    }

}
```

6. 公共返回对象

6.1 RespBean

```
package com.ibm.server.pojo;

import com.baomidou.mybatisplus.extension.api.R;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * 公共返回对象
 *
 * @author lizongzai
```

```
 * @since 1.0.0
 */

@Data
@NoArgsConstructor
@AllArgsConstructor
@ApiModel(value = "RespBean对象", description = "")
public class RespBean {

  private long code;
  private String message;
  private Object obj;

  /**
   * 成功返回结果
   *
   * @param message
   * @param obj
   * @return
   */
  public static RespBean success(String message, Object obj) {
    return new RespBean(200, message, obj);
  }

  /**
   * 失败返回结果
   *
   * @param message
   * @return
   */
  public static RespBean error(String message) {
    return new RespBean(500, message, null);
  }

  /**
   * 失败返回结果
   * @param message
   * @param obj
   * @return
   */
  public static RespBean error(String message, Object obj) {
    return new RespBean(500, message, obj);
  }
}
```

7. 登录之后返回Token

7.1 Admin

- [ ] UserDetails类

```
public class Admin implements Serializable, UserDetails {

}
```

- [ ] 实现UserDetails类的方法

```
@Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return null;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return enabled;
    }
```

```
package com.ibm.server.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import java.util.Collection;
import lombok.Data;
import lombok.EqualsAndHashCode;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
```

```
 * @since 2023-01-01
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_admin")
@ApiModel(value="Admin对象", description="")
public class Admin implements Serializable, UserDetails {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "姓名")
    private String name;

    @ApiModelProperty(value = "手机号码")
    private String phone;

    @ApiModelProperty(value = "住宅电话")
    private String telephone;

    @ApiModelProperty(value = "联系地址")
    private String address;

    @ApiModelProperty(value = "是否启用")
    private Boolean enabled;

    @ApiModelProperty(value = "用户名")
    private String username;

    @ApiModelProperty(value = "密码")
    private String password;

    @ApiModelProperty(value = "用户头像")
    private String userFace;

    @ApiModelProperty(value = "备注")
    private String remark;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return null;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
```

```
        return true;
    }

    @Override
    public boolean isEnabled() {
        return enabled;
    }
}
```

8. 用户登录实体类

8.1 自定义AdminLoginParam实体

```
package com.xxxx.server.pojo;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.experimental.Accessors;

/**
 * 用户登录实体类
 */

@Data
@EqualsAndHashCode(callSuper = false)
@Accessors(chain = true)
@ApiModel(value = "AdminLogin对象", description = "")
public class AdminLoginParam {

  @ApiModelProperty(value = "用户名", required = true)
  private String username;
  @ApiModelProperty(value = "密码", required = true)
  private String password;
}
```

9. 登录用户信息

9.1 LoginController

```
package com.xxxx.server.controller;

import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.AdminLoginParam;
import com.xxxx.server.pojo.RespBean;
import com.xxxx.server.service.IAdminService;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import java.security.Principal;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * 登录功能
 */

@RestController
@Api(tags = "LoginController")
public class LoginController {

  @Autowired
  private IAdminService adminService;

  @ApiOperation(value = "用户登录之后返回token")
  @PostMapping("/login")
  public RespBean login(AdminLoginParam adminLoginParam, HttpServletRequest request) {
    return adminService.login(adminLoginParam.getUsername(),
adminLoginParam.getPassword(),
        request);
  }

  @ApiOperation(value = "获取当前登录用户的信息")
  @GetMapping("/admin/info")
  public Admin getAdminInfo(Principal principal) {
    //判断principal对象是否为空
    if (principal == null) {
      return null;
    }
    //获取用户名
    String username = principal.getName();
    Admin admin = adminService.getAdminByUsername(username);
    admin.setPassword(null);
    return admin;

  }

  @ApiOperation(value = "退出登录")
  @PostMapping("/logout")
  public RespBean logout() {
    return RespBean.success("注销成功");
  }

}
```

9.2 IAdminService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.RespBean;
import javax.servlet.http.HttpServletRequest;

/**
```

```
 * <p>
 * 服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-19
 */
public interface IAdminService extends IService<Admin> {

  /**
   * 用户登录之后返回token
   *
   * @param username
   * @param password
   * @param request
   * @return
   */
  RespBean login(String username, String password, HttpServletRequest request);

  /**
   * 获取当前登录用户的信息
   *
   * @param username
   * @return
   */
  Admin getAdminByUsername(String username);
}
```

9.3 AdminServiceImpl

```
package com.ibm.server.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.extension.conditions.query.QueryChainWrapper;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import com.ibm.server.config.security.JwtTokenUtil;
import com.ibm.server.mapper.AdminMapper;
import com.ibm.server.pojo.Admin;
import com.ibm.server.pojo.RespBean;
import com.ibm.server.service.IAdminService;
import java.util.HashMap;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

/**
```

```
 * <p>
 *  服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-01
 */
@Service
public class AdminServiceImpl extends ServiceImpl<AdminMapper, Admin> implements
IAdminService {

  @Autowired
  private AdminMapper adminMapper;

  @Autowired
  private UserDetailsService userDetailsService;

  @Autowired
  private PasswordEncoder passwordEncoder;

  @Autowired
  private JwtTokenUtil jwtTokenUtil;

  @Value("${jwt.tokenHead}")
  private String tokenHead;

  /**
   * 登录之后返回token
   * @param username
   * @param password
   * @param httpServletRequest
   * @return
   */
  @Override
  public RespBean login(String username, String password, HttpServletRequest
httpServletRequest) {

    UserDetails userDetails = userDetailsService.loadUserByUsername(username);
    //判断用户和密码是否有效
    if (userDetails == null ||
!passwordEncoder.matches(password,userDetails.getPassword())) {
      return RespBean.error("用户名或密码不正确");
    }

    //判断账号是否被禁用
    if (!userDetails.isEnabled()) {
      return RespBean.error("账号被禁用, 请联系管理员");
    }

    //更新Security登录用户对象(用户信息，用户密码，凭证)
    UsernamePasswordAuthenticationToken authenticationToken = new
UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
    //将token放进全局里面，并且更新权限
    SecurityContextHolder.getContext().setAuthentication(authenticationToken);

    //获取token信息
    String token = jwtTokenUtil.generateToken(userDetails);
    Map<String,String> tokenMap = new HashMap<>();
```

```
        tokenMap.put("token",token);
        tokenMap.put("tokenHead",tokenHead);

        return RespBean.success("登录成功", tokenMap);
    }

    /**
     * 根据用户获取用户名
     * @param username
     * @return
     */
    @Override
    public Admin getAdminByUserName(String username) {
        return adminMapper.selectOne(new QueryWrapper<Admin>().eq("username",
username).eq("enabled",true));
    }
}
```

10. 配置Security登录授权过滤器

10.1 SecurityConfig(★★★★★)

```
package com.xxxx.server.config.security;

import com.xxxx.server.pojo.Admin;
import com.xxxx.server.service.IAdminService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationM
anagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurer
Adapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private IAdminService adminService;
    @Autowired
    private RestAuthorizationEntryPoint restAuthorizationEntryPoint;
    @Autowired
    private RestfulAccessDeniedHandler restfulAccessDeniedHandler;

    @Override
```

```
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService()).passwordEncoder(passwordEncoder());
  }

  @Override
  protected void configure(HttpSecurity http) throws Exception {

    //使用JWT,不需要csrf
    http.csrf().disable()

        //基于token,不需要session

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()

        .authorizeRequests().antMatchers("/login", "/logout").permitAll()

        //除了上面，其它请求都需要认证
        .anyRequest().authenticated().and()

        //禁止缓存
        .headers().cacheControl();

    //添加jwt授权拦截器
    http.addFilterBefore(jwtAuthenticationTokenFilter(),
        UsernamePasswordAuthenticationFilter.class);

    //添加自定义未授权和未登录返回结果
    http.exceptionHandling().accessDeniedHandler(restfulAccessDeniedHandler)
        .authenticationEntryPoint(restAuthorizationEntryPoint);

  }

  @Override
  @Bean
  public UserDetailsService userDetailsService() {
    return username -> {
      Admin admin = adminService.getAdminByUsername(username);
      if (admin != null) {
        return admin;
      }
      return null;
    };
  }

  @Bean
  public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
  }

  public JwtAuthenticationTokenFilter jwtAuthenticationTokenFilter() {
    return new JwtAuthenticationTokenFilter();
  }
}
```

10.2 JwtAuthenticationTokenFilter(★★★)

```
package com.xxxx.server.config.security;


import java.io.IOException;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.web.filter.OncePerRequestFilter;

/**
 * JWT登录授权拦截器
 */
public class JwtAuthenticationTokenFilter extends OncePerRequestFilter {

  @Value("${jwt.tokenHeader}")
  private String tokenHeader;
  @Value("${jwt.tokenHead}")
  private String tokenHead;
  @Autowired
  private JwtTokenUtil jwtTokenUtil;
  @Autowired
  private UserDetailsService userDetailsService;

  @Override
  protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
response,
      FilterChain filterChain) throws ServletException, IOException {

    //通过http request获取请求头
    String authHeader = request.getHeader(tokenHeader);
    //若authHeader不为空,并且authHeader以tokenHead开发
    if (authHeader != null && authHeader.startsWith(tokenHead)) {
      //获取authToken
      String authToken = authHeader.substring(tokenHead.length());
      //通过authToken获取用户名username
      String username = jwtTokenUtil.getUsernameFromToken(authToken);
      //token存在用户,但是未登录
      if (username != null && SecurityContextHolder.getContext().getAuthentication()
== null) {
        //登录
        UserDetails userDetails = userDetailsService.loadUserByUsername(username);
        if (jwtTokenUtil.validateToken(authToken, userDetails)) {
          //更新security用户对象
          UsernamePasswordAuthenticationToken authenticationToken = new
UsernamePasswordAuthenticationToken(
              userDetails, null, userDetails.getAuthorities());
```

```
            //重新设置请求内容
            authenticationToken.setDetails(
                new WebAuthenticationDetailsSource().buildDetails(request));
            //设置全局上下文
            SecurityContextHolder.getContext().setAuthentication(authenticationToken);
          }
        }
      }
      //放行所有
      filterChain.doFilter(request,response);
    }
}
```

10.3 RestAuthorizationEntryPoint(★★★)

```
package com.xxxx.server.config.security;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.xxxx.server.pojo.RespBean;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

/**
 * @Description //TODO 当未登录或token失效访问接口时,自定义返回的结果
 * @Author lizongzai
 * @Since 1.0.0
 */
@Component
public class RestAuthorizationEntryPoint implements AuthenticationEntryPoint {

    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response,
        AuthenticationException authException) throws IOException, ServletException {
      response.setCharacterEncoding("UTF-8");
      response.setContentType("application/json");
      PrintWriter out = response.getWriter();
      RespBean bean = RespBean.error("尚未登录,请登录!");
      bean.setCode(401);
      out.write(new ObjectMapper().writeValueAsString(bean));
      out.flush();
      out.close();
    }
}
```

10.4 RestfulAccessDeniedHandler(★★★)

```
package com.xxxx.server.config.security;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.xxxx.server.pojo.RespBean;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.security.access.AccessDeniedException;
import org.springframework.security.web.access.AccessDeniedHandler;
import org.springframework.stereotype.Component;

/**
 * @Description //TODO 当访问接口无权限时，自定义返回结果
 * @Author lizongzai
 * @Since 1.0.0
 */
@Component
public class RestfulAccessDeniedHandler implements AccessDeniedHandler {

  @Override
  public void handle(HttpServletRequest request, HttpServletResponse response,
      AccessDeniedException accessDeniedException) throws IOException,
ServletException {
    response.setCharacterEncoding("UTF-8");
    response.setContentType("application/json");
    PrintWriter out = response.getWriter();
    RespBean bean = RespBean.error("权限不足,请联系管理员!");
    bean.setCode(403);
    out.write(new ObjectMapper().writeValueAsString(bean));
    out.flush();
    out.close();
  }
}
```

11. Swagger2配置

11.1 引入swagger依赖包

```
    <!--swagger2 依赖-->
    <dependency>
      <groupId>io.springfox</groupId>
      <artifactId>springfox-swagger2</artifactId>
      <version>2.7.0</version>
    </dependency>

    <!--swagger 第三方ui依赖-->
    <dependency>
      <groupId>com.github.xiaoymin</groupId>
      <artifactId>swagger-bootstrap-ui</artifactId>
      <version>1.9.6</version>
    </dependency>
```

11.2 SwaggerConfig配置(★)

```
package com.xxxx.server.config.swagger;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

  @Bean
  public Docket createRestApi() {
    return new Docket(DocumentationType.SWAGGER_2)
        .apiInfo(apiInfo())
        .select()
        .apis(RequestHandlerSelectors.basePackage("com.xxxx.server.controller"))
        .paths(PathSelectors.any())
        .build();
  }

  private ApiInfo apiInfo() {
    return new ApiInfoBuilder()
        .title("Restful API's Document")
        .description("Restful API's Document")
        .contact(new Contact("lizongzai", "http://localhost:8081/doc.html",
"lizongzai@gmail.com"))
        .version("1.0")
        .build();
  }

}
```

11.3 Swagger添加Auhorize(★★★)

```
package com.xxxx.server.config.swagger;

import java.util.ArrayList;
import java.util.List;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.ApiKey;
import springfox.documentation.service.AuthorizationScope;
import springfox.documentation.service.Contact;
import springfox.documentation.service.SecurityReference;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spi.service.contexts.SecurityContext;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class Swagger2Config {

  @Bean
  public Docket createRestApi() {
    return new Docket(DocumentationType.SWAGGER_2)
        .apiInfo(apiInfo())
        .select()
        .apis(RequestHandlerSelectors.basePackage("com.xxxx.server.controller"))
        .paths(PathSelectors.any())
        .build()
        .securityContexts(securityContexts())
        .securitySchemes(securitySchemes());
  }


  private ApiInfo apiInfo() {
    return new ApiInfoBuilder()
        .title("Restful API's Document")
        .description("Restful API's Document")
        .contact(new Contact("lizongzai", "http://localhost:8086/doc.html",
"lizongzai@gmail.com"))
        .build();
  }

  private List<SecurityContext> securityContexts() {
    List<SecurityContext> result = new ArrayList<>();
    result.add(getContextByPath("/hello/.*"));
    return result;
  }

  private SecurityContext getContextByPath(String pathRegex) {
    return SecurityContext.builder().securityReferences(defaultAuth())
        .forPaths(PathSelectors.regex(pathRegex)).build();
  }
```

```
  private List<SecurityReference> defaultAuth() {
    List<SecurityReference> result = new ArrayList<>();
    //授权范围
    AuthorizationScope authorizationScope = new
AuthorizationScope("global","accessEverything");
    AuthorizationScope[] authorizationScopes = new AuthorizationScope[1];
    authorizationScopes[0] =authorizationScope;
    result.add(new SecurityReference("Authorization",authorizationScopes));
    return result;
  }

  private List<ApiKey> securitySchemes() {
    //设置请求头信息
    List<ApiKey> result = new ArrayList<>();
    ApiKey apiKey = new ApiKey("Authorization", "Authorization", "Header");
    result.add(apiKey);
    return result;
  }
}
```

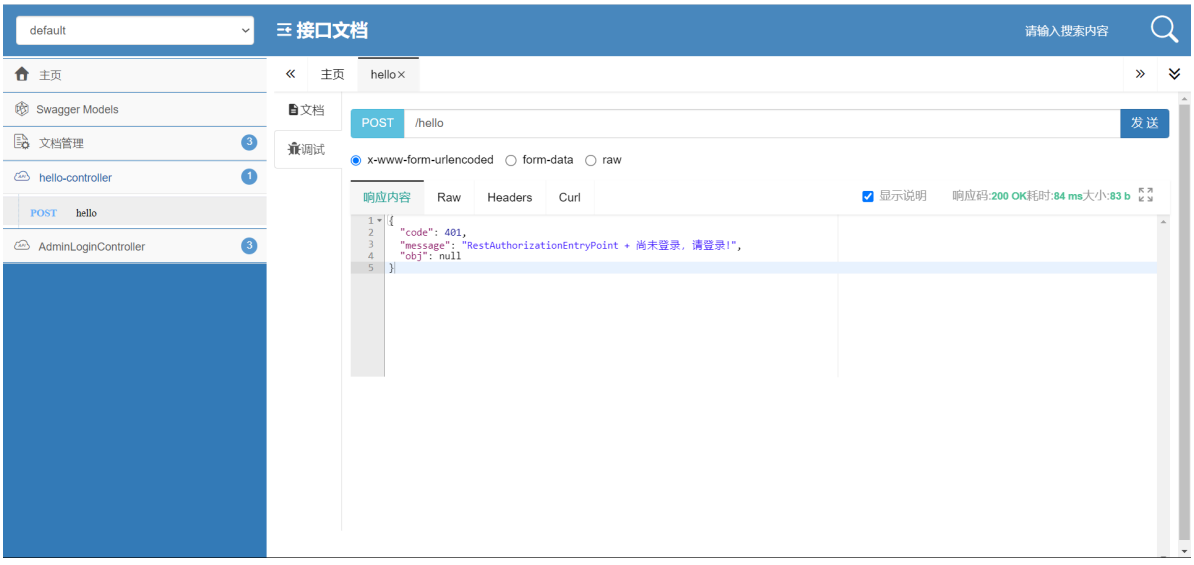
11.4 SecurityConfig 放行信息

```
@Override
public void configure(WebSecurity web) throws Exception {
 web.ignoring().antMatchers(
      "/login",
      "/logout",
      "/ws/**",
      "/css/**",
      "/js/**",
      "/index.html",
      "/img/**",
      "/fonts/**",
      "favicon.ico",
      "/doc.html",
      "/webjars/**",
      "/swagger-resources/**",
      "/v2/api-docs/**",
      "/captcha",
      "/ws/**"
 );
}
```

```
package com.xxxx.server.config.security;

import com.xxxx.server.pojo.Admin;
import com.xxxx.server.service.IAdminService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationM
anagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurer
Adapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {

  @Autowired
  private IAdminService adminService;
  @Autowired
  private RestAuthorizationEntryPoint restAuthorizationEntryPoint;
  @Autowired
  private RestfulAccessDeniedHandler restfulAccessDeniedHandler;

  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService()).passwordEncoder(passwordEncoder());
  }

  @Override
  public void configure(WebSecurity web) throws Exception {
    web.ignoring().antMatchers(
        "/login",
        "/logout",
        "/ws/**",
        "/css/**",
        "/js/**",
        "/index.html",
        "/img/**",
        "/fonts/**",
        "favicon.ico",
        "/doc.html",
        "/webjars/**",
        "/swagger-resources/**",
        "/v2/api-docs/**",
        "/captcha",
        "/ws/**"
    );
  }

  @Override
  protected void configure(HttpSecurity http) throws Exception {

    //使用JWT,不需要csrf
    http.csrf().disable()
        //基于token,不需要session
```

```
.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
        .authorizeRequests()
        //.antMatchers("/login", "/logout").permitAll()
        //除了上面，其它请求都需要认证
        .anyRequest().authenticated().and()
        //禁止缓存
        .headers().cacheControl();

    //添加jwt授权拦截器
    http.addFilterBefore(jwtAuthenticationTokenFilter(),
        UsernamePasswordAuthenticationFilter.class);

    //添加自定义未授权和未登录返回结果
    http.exceptionHandling().accessDeniedHandler(restfulAccessDeniedHandler)
        .authenticationEntryPoint(restAuthorizationEntryPoint);

  }

  @Override
  @Bean
  public UserDetailsService userDetailsService() {
    return username -> {
      Admin admin = adminService.getAdminByUsername(username);
      if (admin != null) {
        return admin;
      }
      return null;
    };
  }

  @Bean
  public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
  }

  public JwtAuthenticationTokenFilter jwtAuthenticationTokenFilter() {
    return new JwtAuthenticationTokenFilter();
  }
}
```



- [x] JAXB API是java EE 的API，因此在java SE 9.0 中不再包含这个 Jar 包。 java 9 中引入了模块的概念，默认情况下，Java SE中将不再包含java EE 的Jar包 而在 java 6/7/8 时关于这个API
  都是捆绑在一起的
- [x] [dispatcherServlet] in context with path [] threw exception [Handler dispatch failed; nested exception is java.lang.NoClassDefFoundError: javax/xml/bind/DatatypeConverter] with root cause

https://www.twle.cn/t/19216

```
<dependency>
 <groupId>javax.xml.bind</groupId>
 <artifactId>jaxb-api</artifactId>
 <version>2.3.1</version>
</dependency>
```

https://blog.csdn.net/zhangruibo_code/article/details/119675229

11.5 心跳测试

```
package com.xxxx.server.controller;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@Api(tags = "HelloController")
public class HelloController {

  @ApiOperation(value = "测试")
  @GetMapping("/hello")
  public String hello() {
    return "hello world";
  }
}
```

12. 生成验证码

12.1 添加依赖

```
<!-- google kaptcha依赖 -->
<dependency>
   <groupId>com.github.axet</groupId>
   <artifactId>kaptcha</artifactId>
   <version>0.0.9</version>
</dependency>
```

12.2 验证码配置类

```
package com.xxxx.server.config.captcha;

import com.google.code.kaptcha.impl.DefaultKaptcha;

import com.google.code.kaptcha.util.Config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;


import java.util.Properties;

/**
 * 验证码配置
 *
 * @author zhanglishen
 * @since 1.0.0
 */

@Configuration
public class CaptchaConfig {

    @Bean
    public DefaultKaptcha getDefaultKaptcha() {
        //验证码生成器
        DefaultKaptcha defaultKaptcha = new DefaultKaptcha();
        //配置
        Properties properties = new Properties();
        //是否有边框
        properties.setProperty("kaptcha.border", "yes");
        //设置边框颜色
        properties.setProperty("kaptcha.border.color", "105,179,90");
        //边框粗细度，默认为1
        // properties.setProperty("kaptcha.border.thickness","1");
        //验证码
        properties.setProperty("kaptcha.session.key", "code");
        //验证码文本字符颜色 默认为黑色
        properties.setProperty("kaptcha.textproducer.font.color", "blue");
        //设置字体样式
        properties.setProperty("kaptcha.textproducer.font.names", "宋体,楷体,微软雅 黑");
        //字体大小，默认40
        properties.setProperty("kaptcha.textproducer.font.size", "30");
        //验证码文本字符内容范围 默认为abced2345678gfynmnpwx
        // properties.setProperty("kaptcha.textproducer.char.string", "");
        //字符长度，默认为5
        properties.setProperty("kaptcha.textproducer.char.length", "4");
        //字符间距 默认为2
        properties.setProperty("kaptcha.textproducer.char.space", "4");
        //验证码图片宽度 默认为200
        properties.setProperty("kaptcha.image.width", "100");
        //验证码图片高度 默认为40
        properties.setProperty("kaptcha.image.height", "40");
        Config config = new Config(properties);
        defaultKaptcha.setConfig(config);
        return defaultKaptcha;
```

```
    }
}
```

12.3 提供验证码生成接口

> ☐ 需要将验证码输出到jpeg图片,否则会出现乱码
>
> ```
> produces = "image/jpeg"
> ```

```
package com.xxxx.server.controller;

import com.google.code.kaptcha.impl.DefaultKaptcha;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import java.awt.image.BufferedImage;
import java.io.IOException;
import javax.imageio.ImageIO;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * @Description //TODO 验证码
 * @Author lizongzai
 * @Since 1.0.0
 */
@RestController
@Api(tags = "CaptchaController")
public class CaptchaController {

  @Autowired
  private DefaultKaptcha defaultKaptcha;

  @ApiOperation(value = "验证码")
  @GetMapping(value = "/captcha", produces = "image/jpeg")
  public void captcha(HttpServletRequest request, HttpServletResponse response) {

    // 定义response输出类型为image/jpeg类型
    response.setDateHeader("Expires", 0);
    // Set standard HTTP/1.1 no-cache headers.
    response.setHeader("Cache-Control", "no-store, no-cache, must-revalidate");
    // Set IE extended HTTP/1.1 no-cache headers (use addHeader).
    response.addHeader("Cache-Control", "post-check=0, pre-check=0");
    // Set standard HTTP/1.0 no-cache header.
    response.setHeader("Pragma", "no-cache");
    // return a jpeg
    response.setContentType("image/jpeg");

    //获取验证码文本内容
    String text = defaultKaptcha.createText();
```

```
    System.out.println("验证码内容: " + text);
    //将文本验证码内放入session
    request.getSession().setAttribute("captcha",text);
    //根据文本验证码生成图形验证码
    BufferedImage image = defaultKaptcha.createImage(text);

    ServletOutputStream outputStream = null;
    try {
      outputStream = response.getOutputStream();
      ImageIO.write(image,"jpg",outputStream);
      outputStream.flush();
    } catch (IOException e) {
      e.printStackTrace();
    }finally {
      if (outputStream != null){
        try {
          outputStream.close();
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
    }

  }

}
```

12.4 放行验证码接口

`SecurityConfig`

```
@Override
  public void configure(WebSecurity web) throws Exception {
    web.ignoring().antMatchers(
        "/login",
        "/logout",
        "/ws/**",
        "/css/**",
        "/js/**",
        "/index.html",
        "/img/**",
        "/fonts/**",
        "favicon.ico",
        "/doc.html",
        "/webjars/**",
        "/swagger-resources/**",
        "/v2/api-docs/**",
        "/captcha"
    );
  }
```

13. 校验验证码

13.1 登录参数对象添加验证码属性

13.1.1 AdminLoginParam

```
@Data
@EqualsAndHashCode(callSuper = false)
@Accessors(chain = true)
@ApiModel(value = "AdminLogin对象", description = "")
public class AdminLoginParam {
    @ApiModelProperty(value = "用户名", required = true)
    private String username;
    @ApiModelProperty(value = "密码", required = true)
    private String password;
    @ApiModelProperty(value = "验证码", required = true)
    private String code;
}
```

13.2 登录方法添加验证码判断

13.2.1 AdminLoginController

```
@ApiOperation(value = "登录之后返回token")
  @PostMapping("/login")
  public RespBean login(@RequestBody AdminLoginParam adminLoginParam,
HttpServletRequest request) {
    return adminService.login(adminLoginParam.getUsername(),
adminLoginParam.getPassword(),
        adminLoginParam.getCode(), request);
  }
```

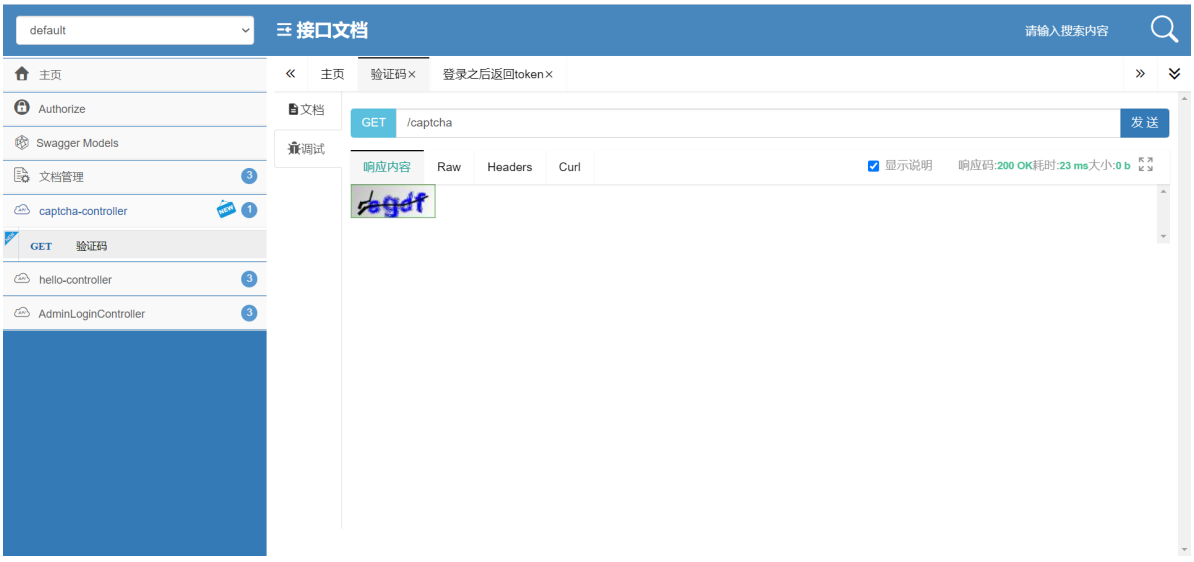
13.2.2 IAdminService

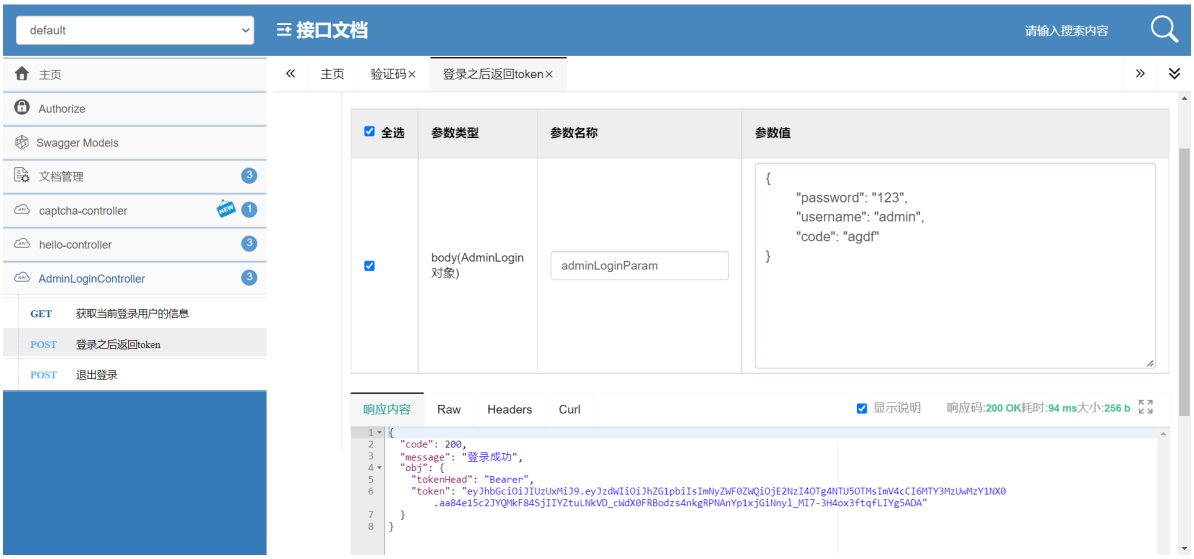
```
/**
   * 登录之后返回token
   *
   * @param username
   * @param password
   * @param code
   * @param request
   * @return
   */
  RespBean login(String username, String password, String code, HttpServletRequest
request);
```

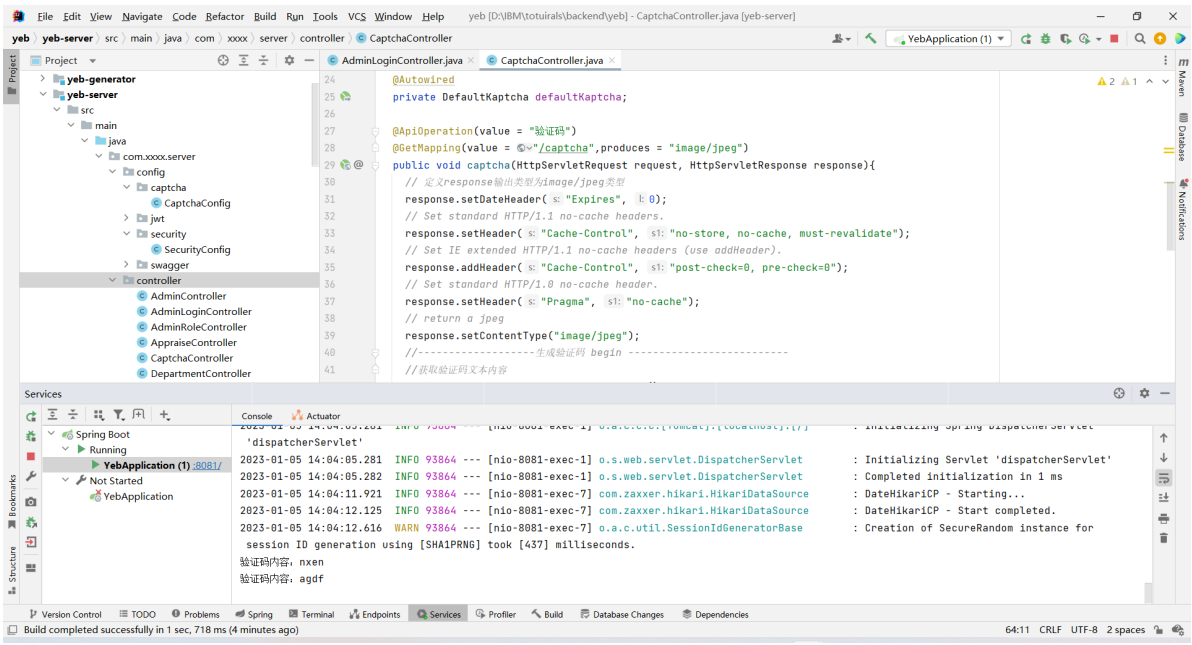
13.2.3 AdminServiceImpl

```
/**
   * 登录之后返回token
   *
   * @param username
```

```
     * @param password
     * @param code
     * @param request
     * @return
     */
    @Override
    public RespBean login(String username, String password, String code,
HttpServletRequest request) {
        //验证码
        String captcha = (String) request.getSession().getAttribute("captcha");
        if (StringUtils.isEmpty(code)||!captcha.equalsIgnoreCase(code)){
            return RespBean.error("验证码输入错误，请重新输入验证!");
        }

        //登录
        UserDetails userDetails = userDetailsService.loadUserByUsername(username);
        if (username == null || !passwordEncoder.matches(password,
userDetails.getPassword())) {
            return RespBean.error("用户名和密码不正确!");
        }
        if (!userDetails.isEnabled()) {
            return RespBean.error("账号被禁用, 请联系管理员");
        }

        //更新security登录用户对象
        UsernamePasswordAuthenticationToken authenticationToken = new
UsernamePasswordAuthenticationToken(
                userDetails, null, userDetails.getAuthorities());
        SecurityContextHolder.getContext().setAuthentication(authenticationToken);

        String token = jwtTokenUtil.generateToken(userDetails);
        Map<String, String> tokenMap = new HashMap<>();
        tokenMap.put("token", token);
        tokenMap.put("tokenHead", tokenHead);
        return RespBean.success("登录成功", tokenMap);
    }
```

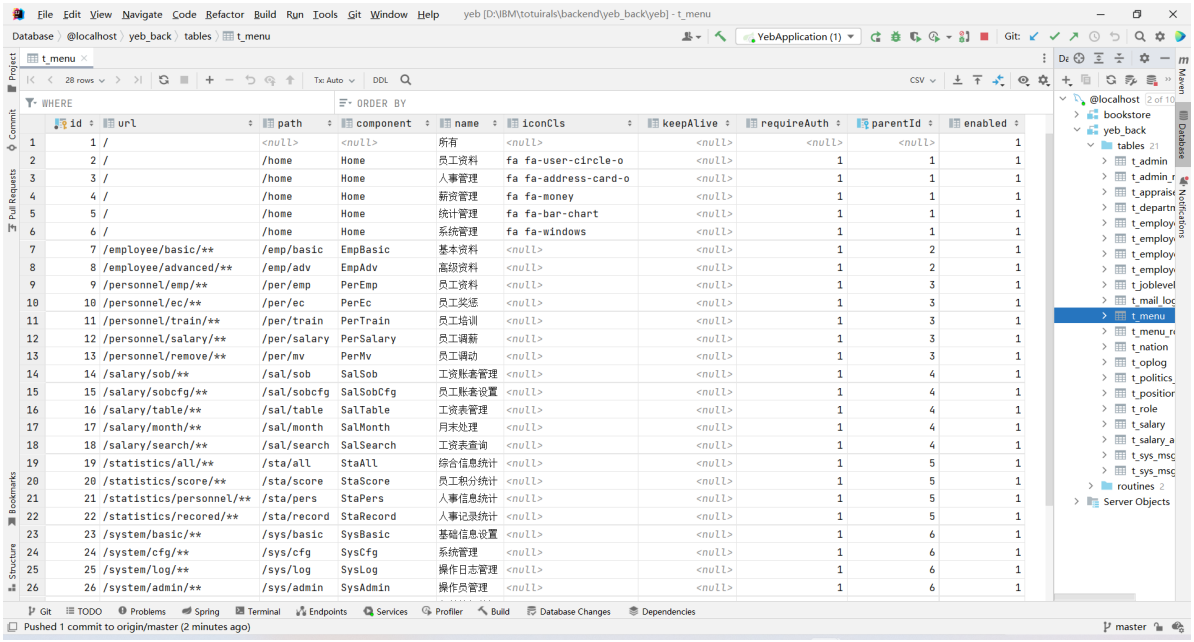






14. 菜单功能

14.1 表结构和数据



id	url	path	component	name	iconCls	keepAlive	requireAuth	parentId	enabled
1	/	<null>	<null>	所有	<null>	<null>	<null>	<null>	1
2	/	/home	Home	员工资料	fa fa-user-circle-o	<null>	1	1	1
3	/	/home	Home	人事管理	fa fa-address-card-o	<null>	1	1	1
4	/	/home	Home	薪资管理	fa fa-money	<null>	1	1	1
5	/	/home	Home	统计管理	fa fa-bar-chart	<null>	1	1	1
6	/	/home	Home	系统管理	fa fa-windows	<null>	1	1	1
7	/employee/basic/**	/emp/basic	EmpBasic	基本资料	<null>	<null>	1	2	1
8	/employee/advanced/**	/emp/adv	EmpAdv	高级资料	<null>	<null>	1	2	1
9	/personnel/emp/**	/per/emp	PerEmp	员工资料	<null>	<null>	1	3	1
10	/personnel/ec/**	/per/ec	PerEc	员工奖惩	<null>	<null>	1	3	1
11	/personnel/train/**	/per/train	PerTrain	员工培训	<null>	<null>	1	3	1
12	/personnel/salary/**	/per/salary	PerSalary	员工调薪	<null>	<null>	1	3	1
13	/personnel/remove/**	/per/mv	PerMv	员工调动	<null>	<null>	1	3	1
14	/salary/sob/**	/sal/sob	SalSob	工资账套管理	<null>	<null>	1	4	1
15	/salary/sobcfg/**	/sal/sobcfg	SalSobCfg	员工账套设置	<null>	<null>	1	4	1
16	/salary/table/**	/sal/table	SalTable	工资表管理	<null>	<null>	1	4	1
17	/salary/month/**	/sal/month	SalMonth	月末处理	<null>	<null>	1	4	1
18	/salary/search/**	/sal/search	SalSearch	工资表查询	<null>	<null>	1	4	1
19	/statistics/all/**	/sta/all	StaAll	综合信息统计	<null>	<null>	1	5	1
20	/statistics/score/**	/sta/score	StaScore	员工积分统计	<null>	<null>	1	5	1
21	/statistics/personnel/**	/sta/pers	StaPers	人事信息统计	<null>	<null>	1	5	1
22	/statistics/recored/**	/sta/record	StaRecord	人事记录统计	<null>	<null>	1	5	1
23	/system/basic/**	/sys/basic	SysBasic	基础信息设置	<null>	<null>	1	6	1
24	/system/cfg/**	/sys/cfg	SysCfg	系统管理	<null>	<null>	1	6	1
25	/system/log/**	/sys/log	SysLog	操作日志管理	<null>	<null>	1	6	1
26	/system/admin/**	/sys/admin	SysAdmin	操作员管理	<null>	<null>	1	6	1

14.2 修改菜单类

- [ ] 菜单分两级，一级菜单下面有子菜单

```
@ApiModelProperty(value = "子菜单")
@TableField(exist = false)
private List<Menu> children;
```

Menu

```
@Data
@EqualsAndHashCode(callSuper = false)
@Accessors(chain = true)
@TableName("t_menu")
@ApiModel(value = "Menu对象", description = "")
public class Menu implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "url")
    private String url;

    @ApiModelProperty(value = "path")
    private String path;

    @ApiModelProperty(value = "组件")
    private String component;

    @ApiModelProperty(value = "菜单名")
    private String name;

    @ApiModelProperty(value = "图标")
```

```
    private String iconCls;

    @ApiModelProperty(value = "是否保持激活")
    private Boolean keepAlive;

    @ApiModelProperty(value = "是否要求权限")
    private Boolean requireAuth;

    @ApiModelProperty(value = "父id")
    private Integer parentId;

    @ApiModelProperty(value = "是否启用")
    private Boolean enabled;

    @ApiModelProperty(value = "子菜单")
    @TableField(exist = false)
    private List<Menu> children;

}
```

MenuController

```
@RestController
@RequestMapping("/system/config")
public class MenuController {

    @Resource
    private IMenuService menuService;

    @ApiOperation(value = "根据用户id查询菜单列表")
    @GetMapping("/menu")
    public List<Menu> getMenuByAdminId() {
        return menuService.getMenuByAdminId();
    }

}
```

MenuService

```
public interface IMenuService extends IService<Menu> {

    /**
     * @return java.util.List<com.xxxx.server.pojo.Menu>
     * @Author James
     * @Description //TODO  根据用户id查询菜单列表
     * @Date 2021/7/14
     * @Param []
     **/
    List<Menu> getMenuByAdminId();

}
```

MenuServiceImpl

```
@Service
public class MenuServiceImpl extends ServiceImpl<MenuMapper, Menu> implements
IMenuService {

    @Resource
    private MenuMapper menuMapper;

    @Override
    public List<Menu> getMenuByAdminId() {
        return menuMapper.getMenuByAdminId(((Admin) SecurityContextHolder
                .getContext()
                .getAuthentication()
                .getPrincipal())
                .getId());
    }
}
```

MenuMapper

```
public interface MenuMapper extends BaseMapper<Menu> {

    /**
     * @return java.util.List<com.xxxx.server.pojo.Menu>
     * @Author James
     * @Description //TODO   根据用户id查询菜单列表
     * @Date 2021/7/14
     * @Param [id]
     **/
    List<Menu> getMenuByAdminId(Integer id);

}
```

MenuMapper.xml

- [ ] collection： 关联关系，是实现一对多的关键
- [ ] property： javabean中容器对应字段名
- [ ] ofType： 指定集合中元素的对象类型
- [ ] select： 使用另一个查询封装的结果
- [ ] column： 数据库中的列名，与 select 配合使用

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.MenuMapper">
```

```
    <!-- 通用查询映射结果 -->
    <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.Menu">
        <id column="id" property="id"/>
        <result column="url" property="url"/>
        <result column="path" property="path"/>
        <result column="component" property="component"/>
        <result column="name" property="name"/>
        <result column="iconCls" property="iconCls"/>
        <result column="keepAlive" property="keepAlive"/>
        <result column="requireAuth" property="requireAuth"/>
        <result column="parentId" property="parentId"/>
        <result column="enabled" property="enabled"/>
    </resultMap>

    <resultMap id="Menus" type="com.xxxx.server.pojo.Menu" extends="BaseResultMap">
        <collection property="children" ofType="com.xxxx.server.pojo.Menu">
            <id column="id2" property="id"/>
            <result column="url2" property="url"/>
            <result column="path2" property="path"/>
            <result column="component2" property="component"/>
            <result column="name2" property="name"/>
            <result column="iconCls2" property="iconCls"/>
            <result column="keepAlive2" property="keepAlive"/>
            <result column="requireAuth2" property="requireAuth"/>
            <result column="parentId2" property="parentId"/>
            <result column="enabled2" property="enabled"/>
        </collection>
    </resultMap>

    <!-- 通用查询结果列 -->
    <sql id="Base_Column_List">
        id, url, path, component, name, iconCls, keepAlive, requireAuth, parentId,
enabled
    </sql>
    <select id="getMenuByAdminId" resultMap="Menus">
        SELECT DISTINCT
            m1.*,
            m2.id AS id2,
            m2.component AS component2,
            m2.enabled AS enabled2,
            m2.iconCls AS iconCls2,
            m2.keepAlive AS keepAlive2,
            m2.NAME AS name2,
            m2.parentId AS parentId2,
            m2.requireAuth AS requireAuth2,
            m2.path AS path2
        FROM
            t_menu m1,
            t_menu m2,
            t_admin_role ar,
            t_menu_role mr
        WHERE
            m1.id = m2.parentId
            AND m2.id = mr.mid
            AND mr.rid = ar.rid
            AND ar.adminId =#{id}
            AND m2.enabled=TRUE
        ORDER BY
```

```
            m1.id,
            m2.id
    </select>
</mapper>
```

14.3 Redis优化菜单

> ▢ 菜单大部分情况下不会出现变化，我们可以将其放入 Redis 加快加载速度

14.3.1 添加pom依赖包(★)

```
<!-- spring data redis 依赖 -->
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
<!-- commons-pool2 对象池依赖 -->
<dependency>
   <groupId>org.apache.commons</groupId>
   <artifactId>commons-pool2</artifactId>
</dependency>
```

14.3.2 Docker 安装 Redis

https://blog.csdn.net/qq_51076413/article/details/123462701

```
docker run \
-p 6379:6379 \
--name redis \
-v /docker/redis/redis.conf:/etc/redis/redis.conf \
-v /docker/redis/data:/data \
--restart=always \
-d redis:7.0.4 redis-server /etc/redis/redis.conf
```

开启redis持久化

```
# 进入conf目录

cd /mydata/redis/conf
vi redis.conf

# 进入redis.conf文件中,按【i】键，进行输入，输入以下命令，开启持久化【开启的时AOF模式的持久化】
# 持久化
appendonly yes
# 允许外网访问  yes-不运许外网访问 no-允许
protected-mode no
# 允许后台运行 yes-运行后台运行  no-不运许
# daemonize yes
# 监听访问ip,指定的ip才能访问(注释掉或指定IP)
# bind 127.0.0.1
# redis访问密码
```

```
requirepass root
```

14.3.3 添加Redis配置

```
redis:
    #超时时间
    timeout: 10000ms
    #服务器地址
    host: localhost
    #服务器端口
    port: 6379
    #数据库
    database: 0
    #密码
    password: root
    lettuce:
      pool:
        #最大连接数，默认8
        max-active: 1024
        #最大连接阻塞等待时间，默认-1
        max-wait: 10000ms
        #最大空闲连接
        max-idle: 200
        #最小空闲连接
        min-idle: 5
```

14.3.4 配置Redis(★★★)

```
@Configuration
public class RedisConfig {
   @Bean
   public RedisTemplate<String,Object> redisTemplate(LettuceConnectionFactory
redisConnectionFactory){
      RedisTemplate<String,Object> redisTemplate = new RedisTemplate<>();
      //为string类型key设置序列器
      redisTemplate.setKeySerializer(new StringRedisSerializer());
      //为string类型value设置序列器
      redisTemplate.setValueSerializer(new
GenericJackson2JsonRedisSerializer());
      //为hash类型key设置序列器
      redisTemplate.setHashKeySerializer(new StringRedisSerializer());
      //为hash类型value设置序列器
      redisTemplate.setHashValueSerializer(new
GenericJackson2JsonRedisSerializer());
      redisTemplate.setConnectionFactory(redisConnectionFactory);
      return redisTemplate;
   }
}
```

14.3.5 读取redis缓存(★)

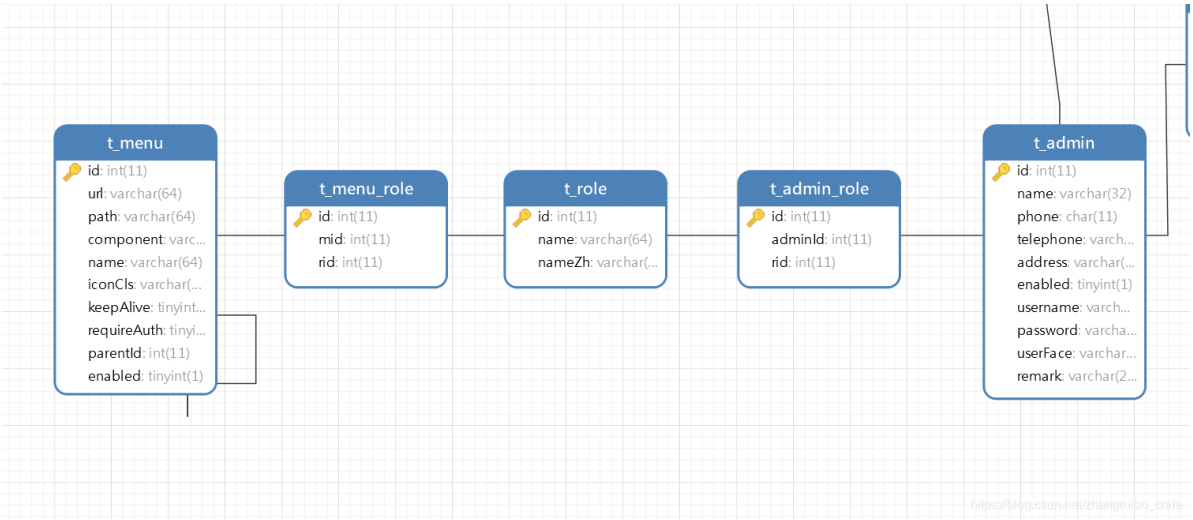
```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.mapper.MenuMapper;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.Menu;
import com.xxxx.server.service.IMenuService;
import java.util.Collections;
import java.util.List;
import javax.annotation.Resource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.core.ValueOperations;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import org.springframework.util.CollectionUtils;
import org.springframework.util.StringUtils;

/**
 * <p>
 * 服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-19
 */
@Service
public class MenuServiceImpl extends ServiceImpl<MenuMapper, Menu> implements
IMenuService {

  @Autowired
  @Resource
  private MenuMapper menuMapper;

  @Autowired
  private RedisTemplate redisTemplate;

  /**
   * 根据用户id获取菜单列表
   *
   * @return
   */
  @Override
  public List<Menu> getMenusByAdminId() {
    //获取用户ID
    Integer adminId = ((Admin)
SecurityContextHolder.getContext().getAuthentication().getPrincipal()).getId();

    ValueOperations<String, Object> valueOperations = redisTemplate.opsForValue();
    //从redis读取菜单数据
    List<Menu> menus = (List<Menu>) valueOperations.get("menu_" + adminId);
    //若redis缓存为空,则从数据库获取菜单数据
    if (CollectionUtils.isEmpty(menus)) {
```

```
            menus = menuMapper.getMenusByAdminId(adminId);
            //将菜单数据设置在redis中
            valueOperations.set("menu_" + adminId,menus);
        }
        //返回结果
        return menus;
    }
}
```

15. 权限管理RBAC

- RBAC是基于角色的访问控制（Role-Based Access Control）在RBAC中，权限与角色相关联，用户通过扮演适当的角色从而得到这些角色的权限。这样管理都是层级相互依赖的，权限赋予给角色，角色又赋予用户，这样的权限设计很清楚，管理起来很方便。
  RBAC授权实际上是 Who 、 What 、 How 三元组之间的关系，也就是 Who 对 What 进行 How 的操作，简单说明就是谁对什么资源做了怎样的操作

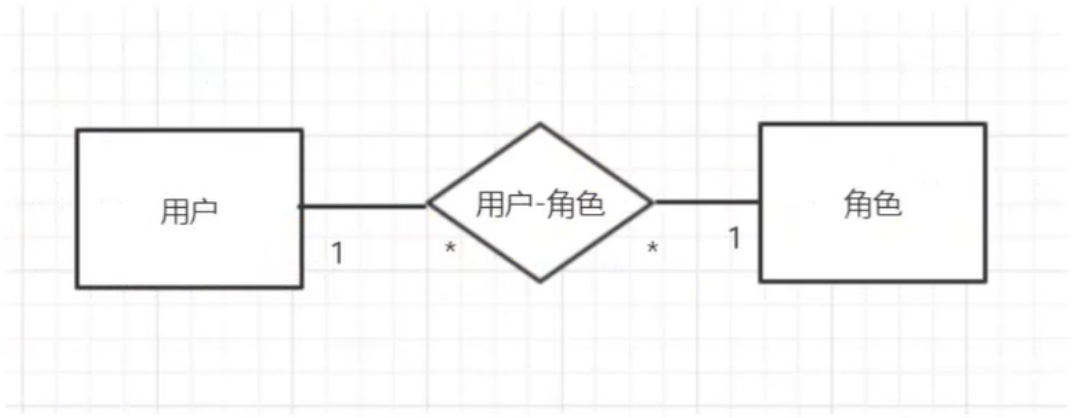
- 从实体对应关系分析,权限表设计为五张表结构:用户表(admin),角色表(role),用户角色表(admin_role),菜单表(menu),菜单权(menu_role)

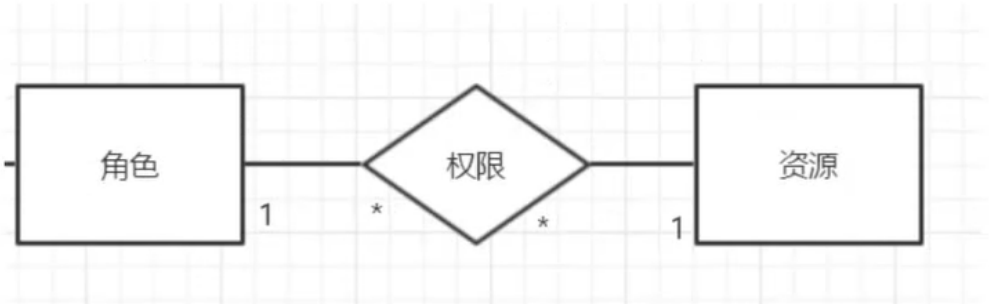


8.2. RBAC表结构设计

8.2.1. 实体对应关系

用户-角色-资源实体间对应关系图分析如下





15.1 根据请求的url判断角色

15.1.1 Menu添加角色 (★)

在菜单类里添加角色列表属性

```
@ApiModelProperty(value = "角色列表")
@TableField(exist = false)
private List<Role> roles;
```

```
@Data
@EqualsAndHashCode(callSuper = false)
@Accessors(chain = true)
@TableName("t_menu")
@ApiModel(value="Menu对象", description="")
public class Menu implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "url")
    private String url;
```

```
    @ApiModelProperty(value = "path")
    private String path;

    @ApiModelProperty(value = "组件")
    private String component;

    @ApiModelProperty(value = "菜单名")
    private String name;

    @ApiModelProperty(value = "图标")
    private String iconCls;

    @ApiModelProperty(value = "是否保持激活")
    private Boolean keepAlive;

    @ApiModelProperty(value = "是否要求权限")
    private Boolean requireAuth;

    @ApiModelProperty(value = "父id")
    private Integer parentId;

    @ApiModelProperty(value = "是否启用")
    private Boolean enabled;

    @ApiModelProperty(value = "子菜单")
    @TableField(exist = false)
    private List<Menu> children;

    @ApiModelProperty(value = "角色列表")
    @TableField(exist = false)
    private List<Role> roles;
}
```

15.1.2 IMenuService

```
public interface IMenuService extends IService<Menu> {
  /**
   * @Description //TODO 根据角色获取菜单列表
   * @Author lizongzai
   * @Date 2023/01/06 14:01
   * @return
   */
  List<Menu> getMenusWithRole();
}
```

15.1.3 MenuServiceImpl

```
public class MenuServiceImpl extends ServiceImpl<MenuMapper, Menu> implements
IMenuService {

  @Autowired
  private MenuMapper menuMapper;
  /**
```

```
     * @Description //TODO 根据角色获取菜单列表
     * @Author lizongzai
     * @Date 2023/01/06 14:01
     * @return
     */
    @Override
    public List<Menu> getMenusWithRole() {
        return menuMapper.getMenusWithRole();
    }
}
```

15.1.4 MenuMapper

```
public interface MenuMapper extends BaseMapper<Menu> {
    /**
     * @Description //TODO 根据角色获取菜单列表
     * @Author lizongzai
     * @Date 2023/01/06 14:01
     * @return
     */
    List<Menu> getMenusWithRole();
}
```

15.2.5 MenuMapper.xml

```
    <!-- 根据角色获取菜单列表 -->
    <resultMap id="MenusWithRole" type="com.xxxx.server.pojo.Menu"
extends="BaseResultMap">
        <collection property="roles" ofType="com.xxxx.server.pojo.Role">
            <id column="rid" property="id"/>
            <result column="rname" property="name"/>
            <result column="rnameZh" property="nameZh"/>
        </collection>
    </resultMap>

    <!-- 根据角色获取菜单列表 -->
    <select id="getMenusWithRole" resultMap="MenusWithRole">
        SELECT
            m.*,
            r.id AS rid,
            r.`name` AS rname,
            r.nameZh AS rnameZh
        FROM
            t_menu m,
            t_menu_role mr,
            t_role r
        WHERE
            m.id = mr.mid
            AND r.id = mr.rid
        ORDER BY
            m.id
    </select>
```

15.2.6 CustomFilter (★★★)

> 添加过滤器根据url获取所需角色

```
/**
 * 权限控制
 * 根据请求url分析请求所需的角色
 *
 * @author zhoubin
 * @since 1.0.0
 */
@Component
public class CustomFilter implements FilterInvocationSecurityMetadataSource {

    @Autowired
    private IMenuService menuService;

    AntPathMatcher antPathMatcher = new AntPathMatcher();

    @Override
    public Collection<ConfigAttribute> getAttributes(Object object) throws
IllegalArgumentException {
        //获取请求的url
        String requestUrl = ((FilterInvocation) object).getRequestUrl();
        List<Menu> menus = menuService.getMenusWithRole();
        for (Menu menu : menus) {
            //判断请求url与菜单角色是否匹配
            if (antPathMatcher.match(menu.getUrl(),requestUrl)){
                String[] str =
menu.getRoles().stream().map(Role::getName).toArray(String[]::new);
                return SecurityConfig.createList(str);
            }
        }
        //没匹配的url默认登录即可访问
        return SecurityConfig.createList("ROLE_LOGIN");
    }

    @Override
    public Collection<ConfigAttribute> getAllConfigAttributes() {
        return null;
    }

    @Override
    public boolean supports(Class<?> clazz) {
        return true;
    }
}
```

15.2 判断用户的角色

15.2.1 Admin添加角色和修改权限 (★)

- [ ] 在用户 里添加角色列表属性，并且可以获取到当前用户的角色

```
@ApiModelProperty(value = "角色")
@TableField(exist = false)
private List<Role> roles;
```

- [ ] 修改权限

```
@Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        List<SimpleGrantedAuthority> authorities = roles
                .stream()
                .map(role -> new SimpleGrantedAuthority(role.getName()))
                .collect(Collectors.toList());
        return authorities;
    }
```

```
@Data
@EqualsAndHashCode(callSuper = false)
@Accessors(chain = true)
@TableName("t_admin")
@ApiModel(value="Admin对象", description="")
public class Admin implements Serializable, UserDetails {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "姓名")
    private String name;

    @ApiModelProperty(value = "手机号码")
    private String phone;

    @ApiModelProperty(value = "住宅电话")
    private String telephone;

    @ApiModelProperty(value = "联系地址")
    private String address;

    @ApiModelProperty(value = "是否启用")
    @Getter(AccessLevel.NONE)
    private Boolean enabled;

    @ApiModelProperty(value = "用户名")
    private String username;

    @ApiModelProperty(value = "密码")
    private String password;
```

```
    @ApiModelProperty(value = "用户头像")
    private String userFace;

    @ApiModelProperty(value = "备注")
    private String remark;

    @ApiModelProperty(value = "角色")
    @TableField(exist = false)
    private List<Role> roles;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        List<SimpleGrantedAuthority> authorities = roles
                .stream()
                .map(role -> new SimpleGrantedAuthority(role.getName()))
                .collect(Collectors.toList());
        return authorities;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return enabled;
    }
}
```

15.2.2 IAdminService

```
public interface IAdminService extends IService<Admin> {
  /**
   * @Description //TODO 根据用户ID查询角色列表
   * @param adminId
   * @return
   */
  List<Role> getRolesByAdminId(Integer adminId);
}
```

15.2.3 AdminServiceImpl

```
public class AdminServiceImpl extends ServiceImpl<AdminMapper, Admin> implements
IAdminService {
  @Autowired
  private RoleMapper roleMapper;

   /**
   * @Description //TODO 根据用户ID查询角色列表
   * @param adminId
   * @return
   */
  @Override
  public List<Role> getRolesByAdminId(Integer adminId) {
    return roleMapper.getRolesByAdminId(adminId);
  }
}
```

15.2.4 RoleMapper

```
public interface RoleMapper extends BaseMapper<Role> {

  /**
   * @Description //TODO 根据用户ID查询角色列表
   * @param adminId
   * @return
   */
  List<Role> getRolesByAdminId(Integer adminId);
}
```

15.2.5 RoleMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.RoleMapper">

    <!-- 通用查询映射结果 -->
    <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.Role">
        <id column="id" property="id" />
        <result column="name" property="name" />
        <result column="nameZh" property="nameZh" />
    </resultMap>

    <!-- 通用查询结果列 -->
    <sql id="Base_Column_List">
        id, name, nameZh
    </sql>

    <!--根据用户ID查询角色列表-->
    <select id="getRolesByAdminId" resultType="com.xxxx.server.pojo.Role">
        SELECT
            distinct
            r.id,
```

```
            r.`name`,
            r.nameZh
        FROM
            t_role AS r
                LEFT JOIN t_admin_role AS ar ON ar.rid = r.id
        WHERE
            ar.adminId = #{adminId}

    </select>

</mapper>
```

15.2.6 LoginController

> ☐ 在获取用户信息和登录方法中添加该方法，获取用户信息时能得到角色列表

```
@ApiOperation(value = "获取当前登录用户的信息")
  @GetMapping("/admin/info")
  public Admin getAdminInfo(Principal principal) {
    if (principal == null) {
      return null;
    }
    String username = principal.getName();
    Admin admin = adminService.getAdminByUserName(username);
    admin.setPassword(null);
    admin.setRoles(adminService.getRolesByAdminId(admin.getId()));
    return admin;
  }
```

15.2.7 SecurityConfig

```
@Bean
public UserDetailsService userDetailsService() {
    return username -> {
      Admin admin = adminService.getAdminByUserName(username);
      if (admin != null) {
        admin.setRoles(adminService.getRolesByAdminId(admin.getId()));
        return admin;
      }
      throw new UsernameNotFoundException("用户名和密码不正确!");
    };
  }
```

15.28 添加过滤器判断用户的角色 (CustomUrlDecisionManager) (★★★)

```
@Component
public class CustomUrlDecisionManager implements AccessDecisionManager {

    @Override
    public void decide(Authentication authentication, Object object,
Collection<ConfigAttribute> configAttributes) throws AccessDeniedException,
InsufficientAuthenticationException {
        for (ConfigAttribute configAttribute : configAttributes) {
            //当前url所需角色
            String needRole = configAttribute.getAttribute();
            //判断角色是否登录即可访问的角色，此角色在CustomFilter中设置
            if ("ROLE_LOGIN".equals(needRole)){
                //判断是否登录
                if (authentication instanceof AnonymousAuthenticationToken){
                    throw new AccessDeniedException("尚未登录，请登录！");
                }else {
                    return;
                }
            }
            //判断用户角色是否为url所需角色
            Collection<? extends GrantedAuthority> authorities =
authentication.getAuthorities();
            for (GrantedAuthority authority : authorities) {
                if (authority.getAuthority().equals(needRole)){
                    return;
                }
            }
        }
        throw new AccessDeniedException("权限不足，请联系管理员!");
    }

    @Override
    public boolean supports(ConfigAttribute attribute) {
        return true;
    }

    @Override
    public boolean supports(Class<?> clazz) {
        return true;
    }
}
```

15.2.9 配置动态权限 (SecurityConfig) (★★★)

```
@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private IAdminService adminService;
    @Autowired
    private RestAuthorizationEntryPoint restAuthorizationEntryPoint;
    @Autowired
    private RestfulAccessDeniedHandler restfulAccessDeniedHandler;
```

```
    @Autowired
    private CustomFilter customFilter;
    @Autowired
    private CustomUrlDecisionManager customUrlDecisionManager;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService()).passwordEncoder(passwordEncoder());
    }

    @Override
    public void configure(WebSecurity web) throws Exception {
        web.ignoring().antMatchers(
                "/login",
                "/logout",
                "/css/**",
                "/js/**",
                "/index.html",
                "favicon.ico",
                "/doc.html",
                "/webjars/**",
                "/swagger-resources/**",
                "/v2/api-docs/**",
                "/captcha",
                "/ws/**"
        );
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        //使用JWT，不需要csrf
        http.csrf()
                .disable()
                //基于token，不需要session
                .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and()
                .authorizeRequests()
                //所有请求都要求认证
                .anyRequest()
                .authenticated()
                //动态权限配置
                .withObjectPostProcessor(new
ObjectPostProcessor<FilterSecurityInterceptor>() {
                    @Override
                    public <O extends FilterSecurityInterceptor> O postProcess(O
object) {
                        object.setAccessDecisionManager(customUrlDecisionManager);
                        object.setSecurityMetadataSource(customFilter);
                        return object;
                    }
                })
                .and()
                //禁用缓存
                .headers()
                .cacheControl();
        //添加jwt 登录授权过滤器
```

```
        http.addFilterBefore(jwtAuthencationTokenFilter(),
UsernamePasswordAuthenticationFilter.class);
        //添加自定义未授权和未登录结果返回
        http.exceptionHandling()
                .accessDeniedHandler(restfulAccessDeniedHandler)
                .authenticationEntryPoint(restAuthorizationEntryPoint);
    }

    @Override
    @Bean
    public UserDetailsService userDetailsService(){
        return username -> {
            Admin admin = adminService.getAdminByUserName(username);
            if (null!=admin){
                admin.setRoles(adminService.getRoles(admin.getId()));
                return admin;
            }
            throw new UsernameNotFoundException("用户名或密码不正确");
        };
    }


    @Bean
    public PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
    }

    @Bean
    public JwtAuthencationTokenFilter jwtAuthencationTokenFilter(){
        return new JwtAuthencationTokenFilter();
    }

}
```

15.2.10 准备两个测试类

准备两个测试类

```
@GetMapping("/employee/base/hello")
public String hello2(){
 return "/employee/base/hello";
}

@GetMapping("/employee/advanced/hello")
public String hello3(){
 return "/employee/advanced/hello";
}
```

```
package com.xxxx.server.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
```

```java
import org.springframework.web.bind.annotation.RestController;

/**
 * 测试接口
 *
 * @author lizongzai
 * @since 1.0.0
 */
@RestController
public class HelloController {

  @PostMapping("/hello")
  public String hello(){
    return "hello";
  }

  @GetMapping("/employee/base/hello")
  public String hello2(){
    return "/employee/base/hello";
  }

  @GetMapping("/employee/advanced/hello")
  public String hello3(){
    return "/employee/advanced/hello";
  }

}
```

16. 职位管理 RESTful接口

- REST 描述了一个**架构**样式的网络系统，指的是一组架构约束条件和原则
- RESTful 指的是满足这些**约束**条件和原则的应用程序或设计

> 提供职位的常用操作，例如查询职位，添加职位，更新职位，删除职位等方法

16.1 t_position 表

	id	name	createDate	enabled
1	1	技术总监	2020-03-31 16:20:34	1
2	2	运营总监	2020-03-31 16:20:34	1
3	3	市场总监	2020-03-31 16:20:34	1
4	4	研发工程师	2020-03-31 16:20:34	1
5	5	运维工程师	2020-03-31 16:20:34	1
6	17	DevOps工程师	2023-01-30 15:40:28	1

16.2 自定义日期格式(★)

> ▢ 自定义创建日期格式
>
> ```
> @JsonFormat(pattern = "yyyy-MM-dd",timezone = "Asia/Shanghai")
> ```

`Position`

```
@Data
@EqualsAndHashCode(callSuper = false)
@Accessors(chain = true)
@TableName("t_position")
@ApiModel(value="Position对象", description="")
public class Position implements Serializable {
    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "职位")
    private String name;

    @ApiModelProperty(value = "创建时间")
    @JsonFormat(pattern = "yyyy-MM-dd",timezone = "Asia/Shanghai")
    private LocalDateTime createDate;

    @ApiModelProperty(value = "是否启用")
    private Boolean enabled;
}
```

16.3 功能实现接口

`PositionController`

```
@RestController
@RequestMapping("/system/basic/pos")
public class PositionController {
 @Autowired
 private IPositionService positionService;

 @ApiOperation(value = "获取所有职位信息")
 @GetMapping("/")
 public List<Position> getAllPositions() {
    return positionService.list();
 }

 @ApiOperation(value = "添加职位信息")
 @PostMapping("/")
 public RespBean addPosition(@RequestBody Position position) {
    position.setCreateDate(LocalDateTime.now());
     if (positionService.save(position)) {
        return RespBean.success("添加成功!");
     }
    return RespBean.error("添加失败!");
```

```
    }

    @ApiOperation(value = "更新职位信息")
    @PutMapping("/")
    public RespBean updatePosition(@RequestBody Position position) {
        if (positionService.updateById(position)) {
            return RespBean.success("更新成功!");
        }
        return RespBean.error("更新失败! ");
    }

    @ApiOperation(value = "删除职位信息")
    @DeleteMapping("/{id}")
    public RespBean deletePosition(@PathVariable Integer id) {
        if (positionService.removeById(id)) {
            return RespBean.success("删除成功!");
         }
        return RespBean.error("删除失败!");
    }

    @ApiOperation(value = "批量删除职位信息")
    @DeleteMapping("/")
    public RespBean deletePositionByIds(Integer[] ids) {
        if (positionService.removeByIds(Arrays.asList(ids))) {
            return RespBean.success("删除成功!");
        }
        return RespBean.error("删除失败!");
    }
}
```

17. 全局异常处理

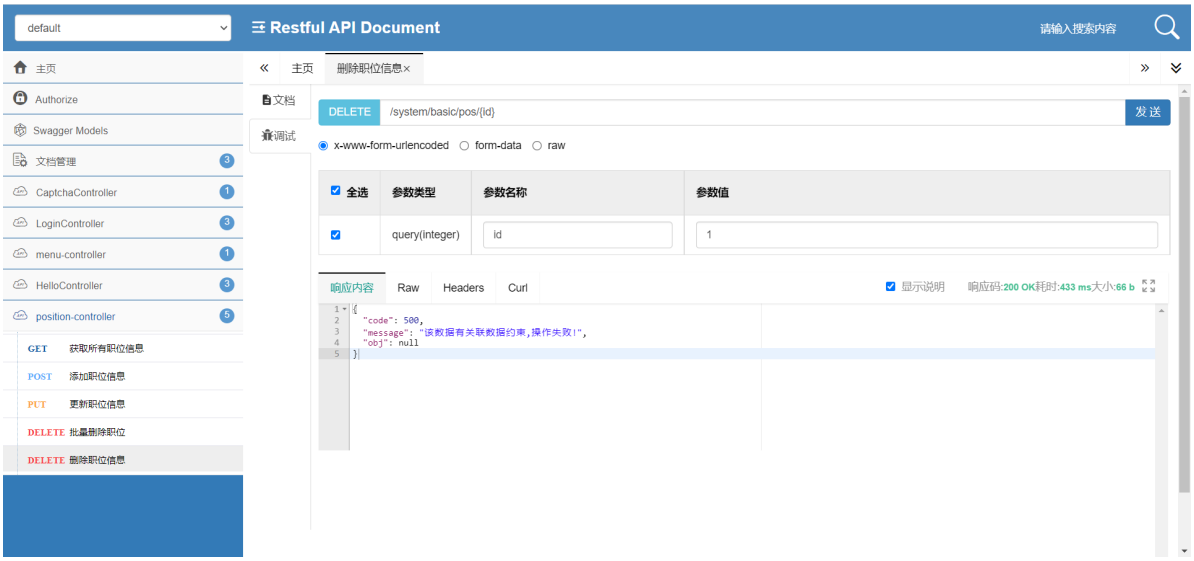
17.1 数据库操作异常(★)

> ☐ 创建已存在信息，可能会提示数据库约束错误信息:
>
> ```
> ### Error updating database.  Cause:
> java.sql.SQLIntegrityConstraintViolationException: Cannot delete or
> update a parent row: a foreign key constraint fails
> ( yeb_back . t_employee , CONSTRAINT t_employee_ibfk_3 FOREIGN KEY ( posId )
> REFERENCES t_position ( id ) ON DELETE RESTRICT ON UPDATE RESTRICT)
> ```

```
package com.xxxx.server.config.exception;

import com.xxxx.server.pojo.common.RespBean;
import java.sql.SQLException;
import java.sql.SQLIntegrityConstraintViolationException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;
```

```java
/**
 * @Description //TODO 全局异常处理
 * @author lizongzai
 * @Date 2023/01/07 13:32
 * @Return
 */
@RestControllerAdvice
public class GlobalException {

  @ExceptionHandler(SQLException.class)
  public RespBean mySqlException(SQLException e) {
    if(e instanceof SQLIntegrityConstraintViolationException) {
      return RespBean.error("该数据有关联数据约束, 操作失败!");
    }
    return RespBean.error("数据库异常, 操作失败!");
  }
}
```

17.2 测试全局异常



18. 职称管理Restful接口

18.1 t_joblevel表

	id	name	titleLevel	createDate	enabled
1	1	教授	正高级	2020-03-31 16:20:34	1
2	2	副教授	副高级	2020-03-31 16:20:34	1
3	3	助教	初级	2020-03-31 16:20:34	1
4	4	讲师	中级	2020-03-31 16:20:34	0
5	5	初级工程师	初级	2020-03-31 16:20:34	1
6	6	中级工程师	中级	2020-03-31 16:20:34	1
7	7	高级工程师	副高级	2020-03-31 16:20:34	1
8	8	骨灰级工程师	正高级	2020-03-31 16:20:34	1

18.2 自定义日期格式

`Joblevel`

```
package com.xxxx.server.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.annotation.JsonFormat;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import java.time.LocalDateTime;
import lombok.Data;
import lombok.EqualsAndHashCode;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_joblevel")
@ApiModel(value="Joblevel对象", description="")
public class Joblevel implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "职称名称")
    private String name;

    @ApiModelProperty(value = "职称等级")
    private String titleLevel;

    @ApiModelProperty(value = "创建时间")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    private LocalDateTime createDate;

    @ApiModelProperty(value = "是否启用")
    private Boolean enabled;
}
```

18.3 职称Restful接口

`JoblevelController`

```
package com.xxxx.server.controller;

import com.xxxx.server.pojo.Joblevel;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IJoblevelService;
import io.swagger.annotations.ApiOperation;
import java.time.LocalDateTime;
import java.util.Arrays;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/system/basic/joblevel")
public class JoblevelController {

  @Autowired
  private IJoblevelService joblevelService;

  @ApiOperation(value = "获取所有职称")
  @GetMapping("/")
  public List<Joblevel> getAllJobLevel() {
    return joblevelService.list();
  }

  @ApiOperation(value = "添加职称")
  @PostMapping("/")
  public RespBean addJobLevel(@RequestBody Joblevel joblevel) {
    joblevel.setCreateDate(LocalDateTime.now());
    if (joblevelService.save(joblevel)) {
      return RespBean.success("添加成功!");
    }
    return RespBean.error("添加失败!");
  }

  @ApiOperation(value = "更新职称信息")
  @PutMapping("/")
  public RespBean updateJobLevel(@RequestBody Joblevel joblevel) {
    if (joblevelService.updateById(joblevel)) {
```

```
        return RespBean.success("更新成功");
      }
      return RespBean.error("更新失败!");
    }

    @ApiOperation(value = "删除职称")
    @DeleteMapping("/{id}")
    public RespBean deleteJobLevel(@PathVariable Integer id) {
      if (joblevelService.removeById(id)) {
        return RespBean.success("删除成功!");
      }
      return RespBean.error("删除失败!");
    }

    @ApiOperation(value = "批量删除职称")
    @DeleteMapping("/")
    public RespBean deleteJobLevelByIds(Integer[] ids) {
      if (joblevelService.removeByIds(Arrays.asList(ids))) {
        return RespBean.success("删除成功!");
      }
      return RespBean.error("删除失败!");
    }
}
```

19. 权限组角色功能

19.1 t_role 表

	id	name	nameZh
1	1	ROLE_manager	部门经理
2	2	ROLE_personnel	人事专员
3	3	ROLE_recruiter	招聘主管
4	4	ROLE_train	培训主管
5	5	ROLE_performance	薪酬绩效主管
6	6	ROLE_admin	系统管理员
7	8	ROLE_test	测试角色

19.2 Role 实体

Role

```
package com.xxxx.server.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import io.swagger.annotations.ApiModel;
```

```
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import lombok.Data;
import lombok.EqualsAndHashCode;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_role")
@ApiModel(value="Role对象", description="")
public class Role implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "名称")
    private String name;

    @ApiModelProperty(value = "角色名称")
    private String nameZh;

}
```

19.3 Role Restful接口(★)

`PermissionController`

```
package com.xxxx.server.controller;

import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IRoleService;
import io.swagger.annotations.ApiModelProperty;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(value = "/system/basic/permission")
```

```
public class PermissionController {

  @Autowired
  private IRoleService roleService;

  @ApiOperation(value = "查询所有角色")
  @GetMapping("/")
  public List<Role> getAllRoles() {
    return roleService.list();
  }

  @ApiOperation(value = "添加角色")
  @PostMapping("/")
  public RespBean addRole(@RequestBody Role role) {

    //添加前缀"ROLE_",这是springframework-security安全框架要求规范
    if (!role.getName().startsWith("ROLE_")) {
      role.setName("ROLE_" + role.getName());
    }
    if (roleService.save(role)) {
      return RespBean.success("添加成功!");
    }
    return RespBean.error("添加失败!");
  }

  @ApiOperation(value = "删除角色")
  @DeleteMapping("/role/{rid}")
  public RespBean deleteRole(@PathVariable Integer rid) {
    if (roleService.removeById(rid)) {
      return RespBean.success("删除成功!");
    }
    return RespBean.error("删除失败!");
  }

}
```

20. 权限组菜单查询功能

20.1 Restful接口

`PermissionController`

```
package com.xxxx.server.controller;

import com.xxxx.server.pojo.Menu;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IMenuRoleService;
import com.xxxx.server.service.IMenuService;
import com.xxxx.server.service.IRoleService;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
```

```java
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(value = "/system/basic/permission")
public class PermissionController {

  @Autowired
  private IRoleService roleService;
  @Autowired
  private IMenuService menuService;
  @Autowired
  private IMenuRoleService menuRoleService;

  @ApiOperation(value = "查询所有菜单")
  @GetMapping("/menus")
  public List<Menu> getAllMenus() {
    return menuService.getAllMenus();
  }

}
```

20.2 IMenuService

```java
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Menu;
import java.util.List;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IMenuService extends IService<Menu> {
  /**
   * @Discription //TODO 查询所有菜单
   * @Author lizongzai
   * @Date 2023/01/08 14:10
   * @return
   */
  List<Menu> getAllMenus();
}
```

20.3 MenuServiceImpl

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.mapper.MenuMapper;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.Menu;
import com.xxxx.server.service.IMenuService;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.core.ValueOperations;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import org.springframework.util.CollectionUtils;

/**
 * <p>
 * 服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class MenuServiceImpl extends ServiceImpl<MenuMapper, Menu> implements
IMenuService {
  @Autowired
  private MenuMapper menuMapper;

  /**
   * @Description //TODO 查询所有菜单
   * @Author lizongzai
   * @Date 2023/01/08 14:10
   * @return
   */
  @Override
  public List<Menu> getAllMenus() {
    return menuMapper.getAllMenus();
  }
}
```

20.4 MenuMapper

```
package com.xxxx.server.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.xxxx.server.pojo.Menu;
import java.util.List;

/**
 * <p>
 *  Mapper 接口
 * </p>
 *
```

```
 * @author lizongzai
 * @since 2023-01-04
 */
public interface MenuMapper extends BaseMapper<Menu> {

  /**
   * @Description //TODO 查询所有菜单
   * @Author lizongzai
   * @Date 2023/01/08 14:10
   * @return
   */
  List<Menu> getAllMenus();
}
```

20.5 SQL查询语句

```
select
    m1.id as id1,
    m1.name as name1,
    m2.id as id2,
    m2.name as name2,
    m3.id as id3,
    m3.name as name3
from
    t_menu m1,
    t_menu m2,
    t_menu m3
where
      m1.id = m2.parentId
  and m2.id = m3.parentId
  and m3.enabled = true
order by
    m1.id,m2.id,m3.id
```

	id1	name1	id2	name2	id3	name3
1	1	所有	2	员工资料	7	基本资料
2	1	所有	2	员工资料	8	高级资料
3	1	所有	3	人事管理	9	员工资料
4	1	所有	3	人事管理	10	员工奖惩
5	1	所有	3	人事管理	11	员工培训
6	1	所有	3	人事管理	12	员工调薪
7	1	所有	3	人事管理	13	员工调动
8	1	所有	4	薪资管理	14	工资账套管理
9	1	所有	4	薪资管理	15	员工账套设置
10	1	所有	4	薪资管理	16	工资表管理
11	1	所有	4	薪资管理	17	月末处理
12	1	所有	4	薪资管理	18	工资表查询
13	1	所有	5	统计管理	19	综合信息统计
14	1	所有	5	统计管理	20	员工积分统计
15	1	所有	5	统计管理	21	人事信息统计
16	1	所有	5	统计管理	22	人事记录统计
17	1	所有	6	系统管理	23	基础信息设置
18	1	所有	6	系统管理	24	系统管理
19	1	所有	6	系统管理	25	操作日志管理
20	1	所有	6	系统管理	26	操作员管理
21	1	所有	6	系统管理	27	备份恢复数据库
22	1	所有	6	系统管理	28	初始化数据库

20.6 `MenuMapper.xml` (★)

- [ ] 方法一、使用嵌套结果集实现
  - [ ] 假设目前菜单只有三级级，MyBatis 语句如下。其原理是通过关联查询，一次性将数据查询出来，然后根据 resultMap 的配置进行转换，构建目标实体类。
  - [ ] 优点：只由于该方法需要访问一次数据库就可以了，不会造成严重的数据库访问消耗
- [ ] 方法二：使用递归查询实现
  - [ ] 请阅览获取部门(★★★★★)

```
<!--查询所有菜单-->
    <resultMap id="MenusWithChildren" type="com.xxxx.server.pojo.Menu"
extends="BaseResultMap">
        <id column="id1" property="id"/>
        <result column="name1" property="name"/>
        <collection property="children" ofType="com.xxxx.server.pojo.Menu">
            <id column="id2" property="id"/>
            <result column="name2" property="name"/>
            <collection property="children" ofType="com.xxxx.server.pojo.Menu">
                <id column="id3" property="id"/>
                <result column="name3" property="name"/>
            </collection>
        </collection>
    </resultMap>

<!--查询所有菜单-->
```

```
    <select id="getAllMenus" resultMap="MenusWithChildren">
        select
            m1.id as id1,
            m1.name as name1,
            m2.id as id2,
            m2.name as name2,
            m3.id as id3,
            m3.name as name3
        from
            t_menu m1,
            t_menu m2,
            t_menu m3
        where
            m1.id = m2.parentId
          and m2.id = m3.parentId
          and m3.enabled = true
        order by
            m1.id,m2.id,m3.id
    </select>
```

21. 根据角色ID查询菜单ID功能

21.1 Restful接口(★)

```
package com.xxxx.server.controller;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.xxxx.server.pojo.Menu;
import com.xxxx.server.pojo.MenuRole;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IMenuRoleService;
import com.xxxx.server.service.IMenuService;
import com.xxxx.server.service.IRoleService;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import java.util.stream.Collectors;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(value = "/system/basic/permission")
public class PermissionController {

  @Autowired
  private IMenuRoleService menuRoleService;

  @ApiOperation(value = "根据角色id查询菜单id")
```

```
  @GetMapping("/mid/{rid}")
  public List<Integer> getMidByRid(@PathVariable Integer rid) {
    return menuRoleService.list(new QueryWrapper<MenuRole>().eq("rid", rid)).stream()
        .map(MenuRole::getMid).collect(
            Collectors.toList());
  }
}
```

22. 权限组更新菜单功能

22.1 PermissionController

```
package com.xxxx.server.controller;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.xxxx.server.pojo.Menu;
import com.xxxx.server.pojo.MenuRole;
import com.xxxx.server.pojo.RespBean;
import com.xxxx.server.service.IMenuRoleService;
import com.xxxx.server.service.IMenuService;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiModelProperty;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import java.util.stream.Collectors;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * 权限控制
 */
@RestController
@Api(tags = "PermissionController")
@RequestMapping("/system/basic/permission")
public class PermissionController {

  @Autowired
  private IMenuService menuService;
  @Autowired
  private IMenuRoleService menuRoleService;

  @ApiOperation(value = "查询所有菜单")
  @GetMapping("/menus")
  public List<Menu> getAllMenus() {
    return menuService.getAllMenus();
  }

  @ApiOperation(value = "根据角色ID查询菜单ID")
  @GetMapping("/mid/rid")
  public List<Integer> getMidByRid(Integer rid) {
    return menuRoleService.list(new QueryWrapper<MenuRole>().eq("rid", rid)).stream()
        .map(MenuRole::getMid).collect( //根据角色ID查询菜单ID,并转成数组
```

```
            Collectors.toList());
    }

    @ApiOperation(value = "更新角色菜单")
    @PutMapping("/role")
    public RespBean updateMenuRole(Integer rid,Integer[] mids) {
        return menuRoleService.updateMenuRole(rid,mids);
    }

}
```

22.2 IMenuRoleService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.MenuRole;
import com.xxxx.server.pojo.common.RespBean;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IMenuRoleService extends IService<MenuRole> {

    /**
     * @Descrition //TODO 更新角色菜单
     * @Author lizongzai
     * @Date 2023/01/08 14:42
     * @param rid
     * @param mids
     * @return
     */
    RespBean updateMenuRole(Integer rid, Integer[] mids);
}
```

22.3 MenuRoleServiceImpl (★★)

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.mapper.MenuRoleMapper;
import com.xxxx.server.pojo.MenuRole;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IMenuRoleService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
```

```java
/**
 * <p>
 *  服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class MenuRoleServiceImpl extends ServiceImpl<MenuRoleMapper, MenuRole>
implements
    IMenuRoleService {

  @Autowired
  private MenuRoleMapper menuRoleMapper;

  /**
   * @Descrition //TODO 更新角色菜单,先删除角色下的菜单删除，然后重新添加菜单
   * @Author lizongzai
   * @Date 2023/01/08 14:42
   * @param rid
   * @param mids
   * @return
   */
  @Override
  @Transactional //添加事务注解
  public RespBean updateMenuRole(Integer rid, Integer[] mids) {
    // 先删除角色下的菜单删除，然后重新添加菜单
    menuRoleMapper.delete(new QueryWrapper<MenuRole>().eq("rid", rid));
    if (mids == null || mids.length == 0) {
      return RespBean.success("删除成功!");
    }
    // 添加菜单记录
    Integer result = menuRoleMapper.insertRecord(rid, mids);
    if (mids.length == result) {
      return RespBean.success("更新成功!");
    }
    return RespBean.error("更新失败!");
  }
}
```

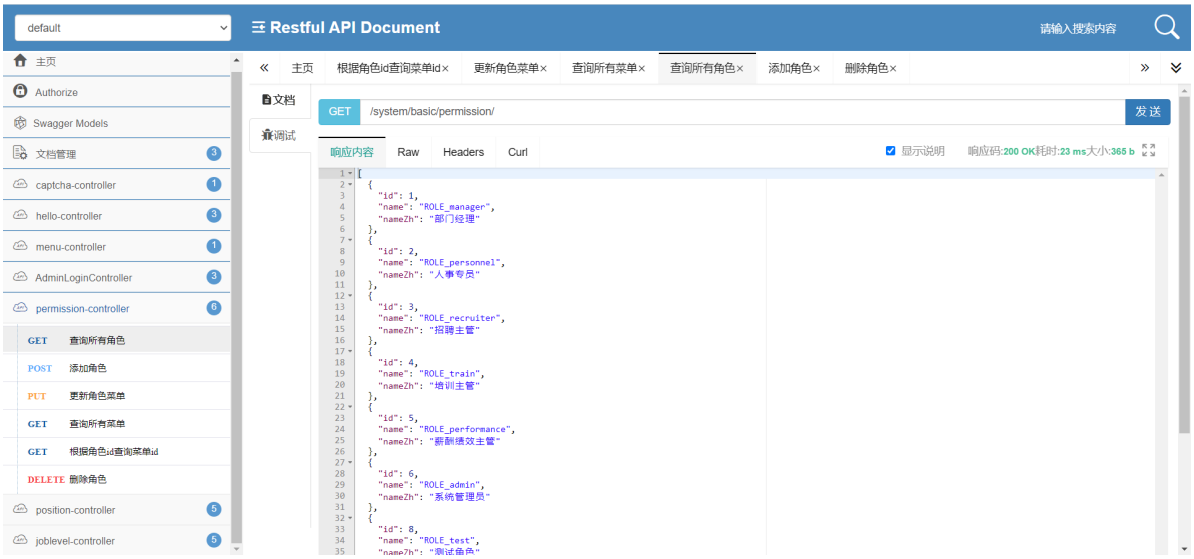
22.4 MenuRoleMapper

```java
package com.xxxx.server.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.xxxx.server.pojo.MenuRole;
import io.lettuce.core.dynamic.annotation.Param;

/**
 * <p>
 *  Mapper 接口
 * </p>
 *
 * @author lizongzai
```

```
 * @since 2023-01-04
 */
public interface MenuRoleMapper extends BaseMapper<MenuRole> {

  /**
   * @Descrition //TODO 更新角色菜单,先删除角色下的菜单删除，然后重新添加菜单
   * @Author lizongzai
   * @Date 2023/01/08 14:42
   * @param rid
   * @param mids
   * @return
   */
  Integer insertRecord(@Param("rid") Integer rid, @Param("mids") Integer[] mids);
}
```

22.5 MenuRoleMapper.xml (★)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.MenuRoleMapper">

    <!-- 通用查询映射结果 -->
    <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.MenuRole">
        <id column="id" property="id" />
        <result column="mid" property="mid" />
        <result column="rid" property="rid" />
    </resultMap>

    <!-- 通用查询结果列 -->
    <sql id="Base_Column_List">
        id, mid, rid
    </sql>

    <!-- 更新角色菜单-->
    <insert id="insertRecord">
        insert into t_menu_role(mid, rid) values
        <foreach collection="mids" item="mid" separator=",">
            (#{mid},#{rid})
        </foreach>
    </insert>

</mapper>
```



23. 获取所有部门(★★★★★)

- [ ] 特别注意结合部门表实现的处理逻辑
- [ ] 存储过程用于 添加部门、删除部门
- [ ] 调用添加和删除存储过程的脚本，实现添加部门和删除部门方法

23.1 创建存储过程(★)

```
create
    definer = root@`%` procedure addDep(IN depName varchar(32), IN parentId int,
                                        IN enabled tinyint(1), OUT result int, OUT
result2 int)
begin
    declare did int;
    declare pDepPath varchar(64);
    insert into t_department set name=depName,parentId=parentId,enabled=enabled;
    select row_count() into result;
    select last_insert_id() into did;
    set result2=did;
    select depPath into pDepPath from t_department where id=parentId;
    update t_department set depPath=concat(pDepPath,'.',did) where id=did;
    update t_department set isParent=true where id=parentId;
end;
```

```
create
    definer = root@`%` procedure deleteDep(IN did int, OUT result int)
begin
    declare ecount int;
    declare pid int;
    declare pcount int;
    declare a int;
    select count(*) into a from t_department where id=did and isParent=false;
    if a=0 then set result=-2;
    else
        select count(*) into ecount from t_employee where departmentId=did;
```

```
        if ecount>0 then set result=-1;
        else
            select parentId into pid from t_department where id=did;
            delete from t_department where id=did and isParent=false;
            select row_count() into result;
            select count(*) into pcount from t_department where parentId=pid;
            if pcount=0 then update t_department set isParent=false where id=pid;
            end if;
        end if;
    end if;
end;
```

23.2 Department实体(★)

- [ ] 添加子部门列表

```
@ApiModelProperty(value = "子部门列表")
@TableField(exist = false)
private List<Department> children;
```

- [ ] 添加返回结果，仅供存储过程

```
@ApiModelProperty(value = "返回结果，仅供存储过程使用")
@TableField(exist = false)
private Integer result;
```

```
package com.xxxx.server.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import io.swagger.annotations.ApiOperation;
import java.io.Serializable;
import java.util.List;
import lombok.Data;
import lombok.EqualsAndHashCode;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_department")
```

```java
@ApiModel(value="Department对象", description="")
public class Department implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "部门名称")
    private String name;

    @ApiModelProperty(value = "父id")
    private Integer parentId;

    @ApiModelProperty(value = "路径")
    private String depPath;

    @ApiModelProperty(value = "是否启用")
    private Boolean enabled;

    @ApiModelProperty(value = "是否上级")
    private Boolean isParent;

    @ApiModelProperty(value = "子部门列表")
    @TableField(exist = false)
    private List<Department> children;

    @ApiModelProperty(value = "返回结果, 仅供存储过程使用")
    @TableField(exist = false)
    private Integer result;

}
```

23.3 DepartmentController

```java
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Department;
import com.xxxx.server.service.IDepartmentService;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
```

```
 */
@RestController
@RequestMapping("/system/basic/department")
public class DepartmentController {

  @Autowired
  private IDepartmentService departmentService;

  @ApiOperation(value = "获取所有部门")
  @GetMapping("/")
  public List<Department> getAllDepartments() {
    return departmentService.getAllDepartments();
  }

}
```

23.4 IDepartmentService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Department;
import java.util.List;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IDepartmentService extends IService<Department> {

  /**
   * @Description //TODO 获取所有部门
   * @Author lizongzai
   * @Date 2023/01/08 16:50
   * @return
   */
  List<Department> getAllDepartments();
}
```

23.5 DepartmentServiceImpl

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import com.xxxx.server.mapper.DepartmentMapper;
import com.xxxx.server.pojo.Department;
import com.xxxx.server.service.IDepartmentService;
```

```java
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * <p>
 *  服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class DepartmentServiceImpl extends ServiceImpl<DepartmentMapper, Department>
implements
    IDepartmentService {

  @Autowired
  private DepartmentMapper departmentMapper;

  /**
   * @Description //TODO 获取所有部门
   * @Author lizongzai
   * @Date 2023/01/08 16:50
   * @return
   */
  @Override
  public List<Department> getAllDepartments() {
    return departmentMapper.getAllDepartments(-1);
  }
}
```

23.6 DepartmentMapper

```java
package com.xxxx.server.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.xxxx.server.pojo.Department;
import java.util.List;

/**
 * <p>
 *  Mapper 接口
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface DepartmentMapper extends BaseMapper<Department> {

  /**
   * @Description //TODO 获取所有部门
   * @Author lizongzai
   * @Date 2023/01/08 16:50
   * @return
```

```
    */
    List<Department> getAllDepartments(Integer parentId);

}
```

23.7 DepartmentMapper.xml(★★★)

> ☐ 获取所有部门, 使用mybatis提供的递归方式
>
> ```
>   <resultMap id="DepartmentWithChildren"
> type="com.xxxx.server.pojo.Department"
>     extends="BaseResultMap">
>     <collection property="children" ofType="com.xxxx.server.pojo.Department"
>       select="com.xxxx.server.mapper.DepartmentMapper.getAllDepartments"
> column="id">
>     </collection>
>   </resultMap>
> ```
>
> ☐ 这个在 MyBatis 查询数据库的sql中经常会出现。它的在上面已经定义，作用相当于 `*`，Base_Column_List是固定的几个字段，而用*号的话会降低查询效率，因为后期数据库的字段会不断增加

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.DepartmentMapper">

  <!-- 通用查询映射结果 -->
  <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.Department">
    <id column="id" property="id"/>
    <result column="name" property="name"/>
    <result column="parentId" property="parentId"/>
    <result column="depPath" property="depPath"/>
    <result column="enabled" property="enabled"/>
    <result column="isParent" property="isParent"/>
  </resultMap>

  <!-- 获取所有部门, 使用mybatis提供的递归方式 -->
  <resultMap id="DepartmentWithChildren" type="com.xxxx.server.pojo.Department"
    extends="BaseResultMap">
    <collection property="children" ofType="com.xxxx.server.pojo.Department"
      select="com.xxxx.server.mapper.DepartmentMapper.getAllDepartments" column="id">
    </collection>
  </resultMap>

  <!-- 通用查询结果列 -->
  <sql id="Base_Column_List">
    id, name, parentId, depPath, enabled, isParent
  </sql>

  <!-- 获取所有部门 -->
  <select id="getAllDepartments" resultMap="DepartmentWithChildren">
    select
    <include refid="Base_Column_List"/>
```

```
    from t_department
    where parentId = #{parentId}
  </select>

</mapper>
```

24. 添加部门

24.1 DepartmentController

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Department;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IDepartmentService;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/system/basic/department")
public class DepartmentController {

  @Autowired
  private IDepartmentService departmentService;

  @ApiOperation(value = "添加部门")
  @PostMapping("/")
  public RespBean addDep(@RequestBody Department dep) {
    return departmentService.addDep(dep);
  }

}
```

24.2 IDepartmentService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Department;
import com.xxxx.server.pojo.common.RespBean;
import java.util.List;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IDepartmentService extends IService<Department> {

  /**
   * @Description //TODO 添加部门
   * @Param department
   * @Author lizongzai
   * @Date 2023/01/09 10.31
   * @return
   */
  RespBean addDep(Department dep);
}
```

24.3 DepartmentServiceImpl(★)

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import com.xxxx.server.mapper.DepartmentMapper;
import com.xxxx.server.pojo.Department;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IDepartmentService;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * <p>
 *  服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class DepartmentServiceImpl extends ServiceImpl<DepartmentMapper, Department>
implements
    IDepartmentService {
```

```
  @Autowired
  private DepartmentMapper departmentMapper;

  /**
   * @Description //TODO 添加部门
   * @Param department
   * @Author lizongzai
   * @Date 2023/01/09 10.31
   * @return
   */
  @Override
  public RespBean addDep(Department dep) {
    dep.setEnabled(true);
    departmentMapper.addDep(dep);
    if (1 == dep.getResult()) {
      return RespBean.success("添加成功!", dep);
    }
    return RespBean.error("添加失败!");
  }
}
```

23.4 DepartmentMapper

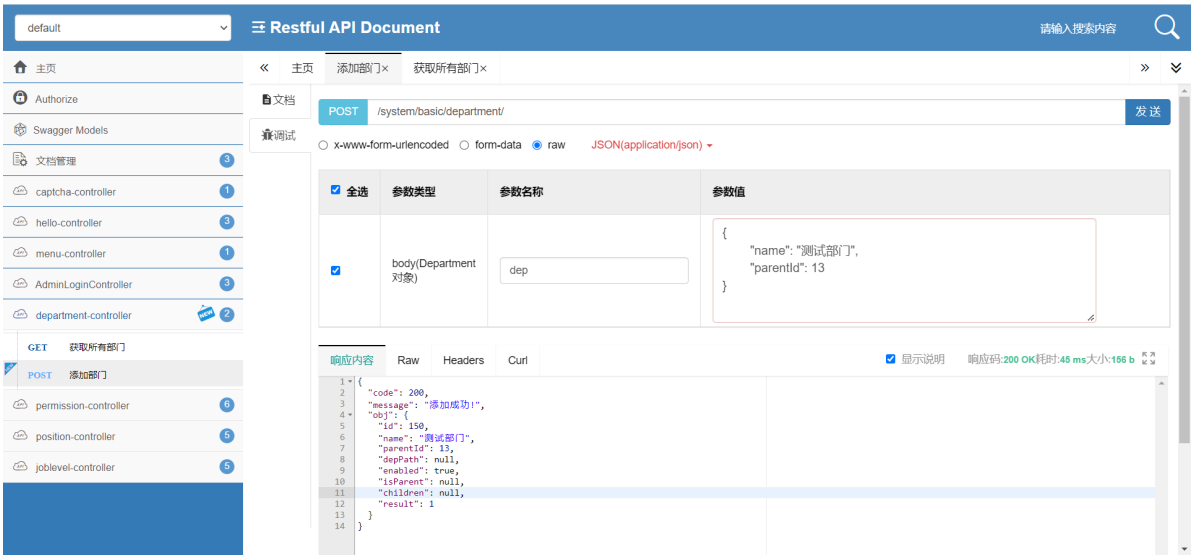
```
package com.xxxx.server.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.xxxx.server.pojo.Department;
import java.util.List;

/**
 * <p>
 *  Mapper 接口
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface DepartmentMapper extends BaseMapper<Department> {

  /**
   * @Description //TODO 添加部门
   * @Param department
   * @Author lizongzai
   * @Date 2023/01/09 10.31
   * @return
   */
  void addDep(Department dep);
}
```

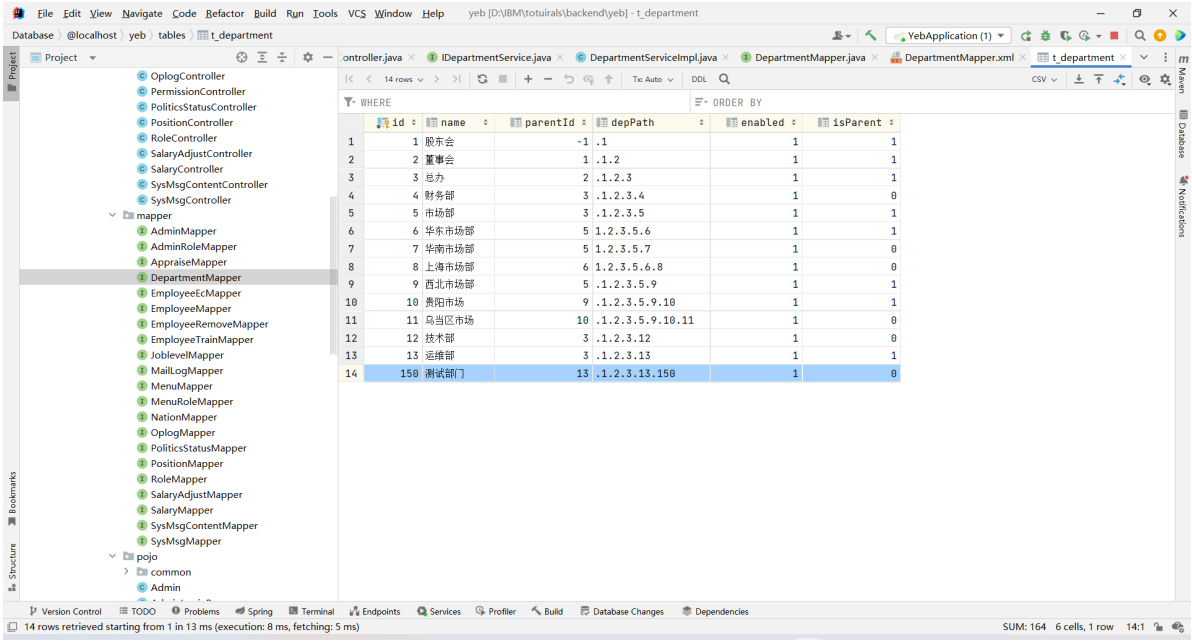
24.5 DepartmentMapper.xml(★★★)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.DepartmentMapper">

  <!-- 通用查询映射结果 -->
  <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.Department">
    <id column="id" property="id"/>
    <result column="name" property="name"/>
    <result column="parentId" property="parentId"/>
    <result column="depPath" property="depPath"/>
    <result column="enabled" property="enabled"/>
    <result column="isParent" property="isParent"/>
  </resultMap>

  <!-- 通用查询结果列 -->
  <sql id="Base_Column_List">
    id
    , name, parentId, depPath, enabled, isParent
  </sql>

  <!-- 添加部门 -->
  <select id="addDep" statementType="CALLABLE">
    call addDep(#{name,mode=IN,jdbcType=VARCHAR},
        #{parentId,mode=IN,jdbcType=INTEGER},
        #{enabled,mode=IN,jdbcType=BOOLEAN},
        #{result,mode=OUT,jdbcType=INTEGER},
        #{id,mode=OUT,jdbcType=INTEGER})
  </select>

</mapper>
```

24.6 Open Restful API and test

```
{
    "name": "测试部门",
    "parentId": 13
}
```





25. 删除部门

> ☐ 首先开发者需要了解具体业务，比如要删除当前部门，那么需要分析当前部门下面其它组织架构和人员。
>
> ☐ 其次，有其它组织架构和人员不能删除。给予优雅的提示信息。

25.1 DepartmentController

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Department;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IDepartmentService;
import io.swagger.annotations.ApiOperation;
```

```java
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/system/basic/department")
public class DepartmentController {

  @Autowired
  private IDepartmentService departmentService;

  @ApiOperation(value = "删除部门")
  @DeleteMapping("/{id}")
  public RespBean deleteDep(@PathVariable Integer id) {
    return departmentService.deleteDep(id);
  }
}
```

25.2 IDepartmentService

```java
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Department;
import com.xxxx.server.pojo.common.RespBean;
import java.util.List;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IDepartmentService extends IService<Department> {

  /**
   * @Description //TODO 删除部门
   * @Param department
   * @Author lizongzai
   * @Date 2023/01/09 10:31
```

```
     * @param id
     * @return
     */
    RespBean deleteDep(Integer id);
}
```

25.3 DepartmentServiceImpl(★)

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import com.xxxx.server.mapper.DepartmentMapper;
import com.xxxx.server.pojo.Department;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IDepartmentService;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * <p>
 * 服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class DepartmentServiceImpl extends ServiceImpl<DepartmentMapper, Department>
implements
    IDepartmentService {

  @Autowired
  private DepartmentMapper departmentMapper;

  /**
   * @param id
   * @return
   * @Description //TODO 删除部门
   * @Param dep
   * @Author lizongzai
   * @Date 2023/01/09 10:31
   */
  @Override
  public RespBean deleteDep(Integer id) {
    Department department = new Department();
    department.setId(id);
    departmentMapper.deleteDep(department);
    if (-2 == department.getResult()) {
      return RespBean.error("该部门下还有子部门, 删除失败!");
    } else if (-1 == department.getResult()) {
      return RespBean.error("该部门下还有员工,删除失败!");
    } else if (1 == department.getResult()) {
      return RespBean.success("删除成功!");
```

```
    }
    return RespBean.error("删除失败!");
  }
}
```

25.4 DepartmentMapper

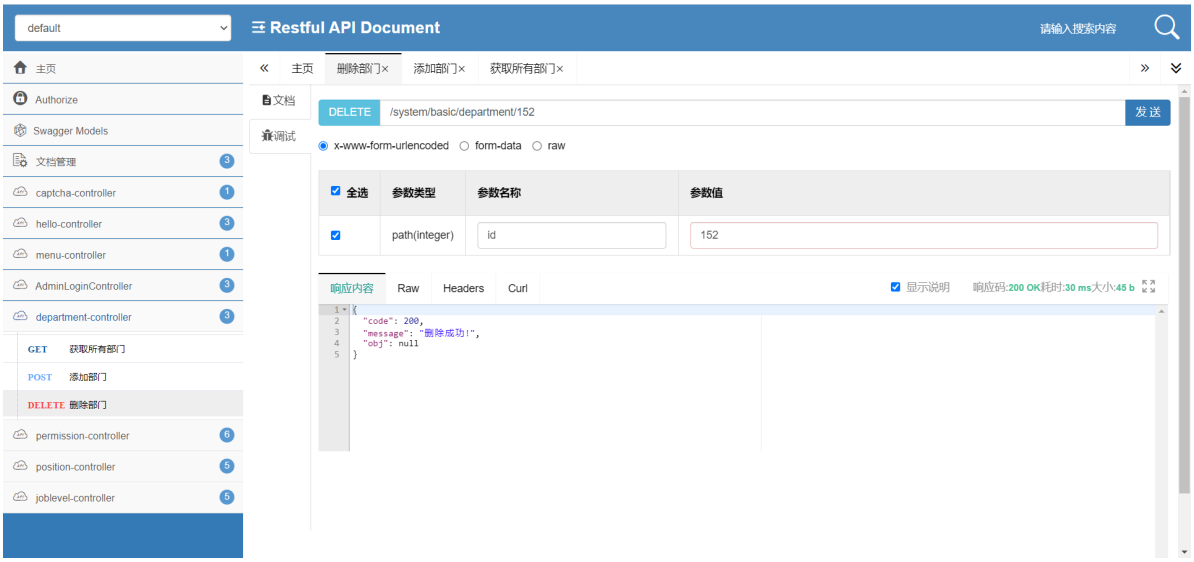
```
package com.xxxx.server.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.xxxx.server.pojo.Department;
import java.util.List;

/**
 * <p>
 *  Mapper 接口
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface DepartmentMapper extends BaseMapper<Department> {

  /**
   * @Description //TODO 删除部门
   * @Param department
   * @Author lizongzai
   * @Date 2023/01/09 10:31
   * @param dep
   * @return
   */
  void deleteDep(Department dep);
}
```

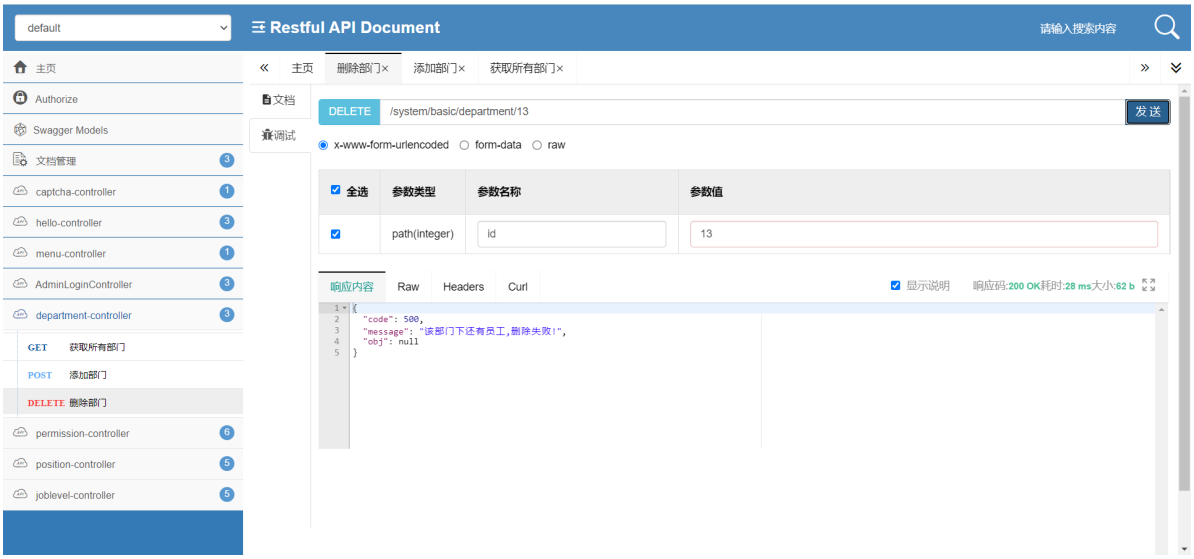
25.5 DepartmentMapper.xml(★★★)

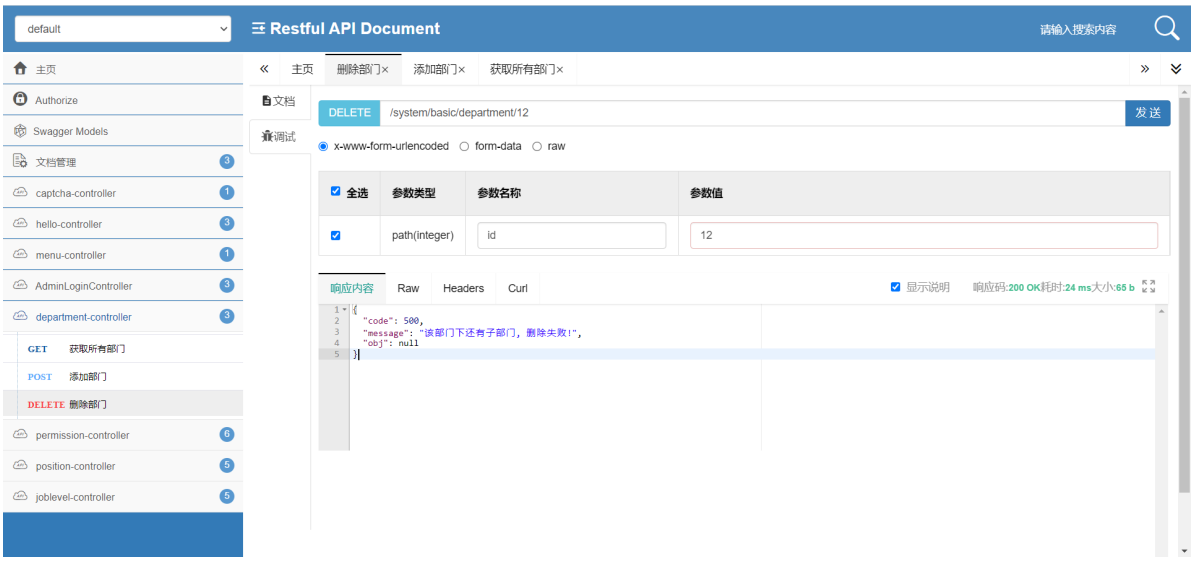
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.DepartmentMapper">

  <!-- 通用查询映射结果 -->
  <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.Department">
    <id column="id" property="id"/>
    <result column="name" property="name"/>
    <result column="parentId" property="parentId"/>
    <result column="depPath" property="depPath"/>
    <result column="enabled" property="enabled"/>
    <result column="isParent" property="isParent"/>
  </resultMap>

  <!--    获取所有部门-->
  <!--  使用mybatis提供的递归方式查询结果-->
  <resultMap id="DepartmentWithChildren" type="com.xxxx.server.pojo.Department"
```

```
    extends="BaseResultMap">
    <collection property="children" ofType="com.xxxx.server.pojo.Department"
      select="com.xxxx.server.mapper.DepartmentMapper.getAllDepartments" column="id">
    </collection>
  </resultMap>

  <!-- 通用查询结果列 -->
  <sql id="Base_Column_List">
    id
    , name, parentId, depPath, enabled, isParent
  </sql>

  <!--    获取所有部门-->
  <select id="getAllDepartments" resultMap="DepartmentWithChildren">
    select
    <include refid="Base_Column_List"></include>
    from t_department
    where parentId = #{parentId}
  </select>

  <!--  添加部门-->
  <insert id="addDepartment" statementType="CALLABLE">
    call addDep(#{name,mode=IN,jdbcType=VARCHAR},
      #{parentId,mode=IN,jdbcType=INTEGER},
      #{enabled,mode=IN,jdbcType=BOOLEAN},
      #{result,mode=OUT,jdbcType=INTEGER},
      #{id,mode=OUT,jdbcType=INTEGER})
  </insert>

  <!--  删除部门-->
  <delete id="deleteDepartment" statementType="CALLABLE">
    call deleteDep(#{id,mode=IN,jdbcType=INTEGER},#{result,mode=OUT,jdbcType=INTEGER})
  </delete>

</mapper>
```

25.6 Open Restful API and Test







26. 获取所有操作员

> ☐ 获取操纵员功能，那么需要有搜索的功能。也就是前端传送登录的用户ID和搜索的关键字。

26.1 AdminController

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Admin;
import com.xxxx.server.service.IAdminService;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
```

```
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/system/admin")
public class AdminController {

  @Autowired
  private IAdminService adminService;

  @ApiOperation(value = "获取所有操作员")
  @GetMapping("/")
  public List<Admin> getAllAdmins(String keywords){
    return adminService.getAllAdmins(keywords);
  }
}
```

26.2 IAdminService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import java.util.List;
import javax.servlet.http.HttpServletRequest;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IAdminService extends IService<Admin> {

  /**
   * @Description //TODO 获取所有操作员
   * @param keywords
   * @return
   */
  List<Admin> getAllAdmins(String keywords);
}
```

26.3 AdminServiceImpl

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.config.jwt.JwtTokenUtil;
import com.xxxx.server.mapper.AdminMapper;
import com.xxxx.server.mapper.RoleMapper;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IAdminService;
import com.xxxx.server.utils.AdminUtils;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;

@Service
public class AdminServiceImpl extends ServiceImpl<AdminMapper, Admin> implements
IAdminService {

  @Autowired
  private UserDetailsService userDetailsService;
  @Autowired
  private PasswordEncoder passwordEncoder;
  @Autowired
  private JwtTokenUtil jwtTokenUtil;
  @Value("${jwt.tokenHead}")
  private String tokenHead;
  @Autowired
  private AdminMapper adminMapper;

  /**
   * @param keywords
   * @return
   * @Description //TODO 获取所有操作员
   */
  @Override
  public List<Admin> getAllAdmins(String keywords) {
    //获取当前登录用户ID
    //Integer adminId = ((Admin)
SecurityContextHolder.getContext().getAuthentication().getPrincipal()).getId();
    return adminMapper.getAllAdmins((AdminUtils.getCurrentUtils()).getId(), keywords);
  }
}
```

26.4 AdminMapper

```
package com.xxxx.server.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.Menu;
import java.util.List;

/**
 * <p>
 *  Mapper 接口
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface AdminMapper extends BaseMapper<Admin> {

  /**
   * 获取所有操作员
   * @param id
   * @param keywords
   * @return
   */
  List<Admin> getAllAdmins(Integer id, String keywords);
}
```

26.5 AdminMapper.xml(★)

- [ ] mybatis模糊查询的三种方法
  - [ ] 方式一：`%${值}%`，注意是一定不能使用`#{}`没有拼接字符串的作用，会直接当作为`%#{值}%`一个字符串，所以会报错

    ```
    select * from t_user where username like '%${username}%';
    ```

  - [ ] 方式二：`concat('%',#{值},'%')`，使用`concat`函数，来拼接字符串

    ```
    select * from t_user where username like concat('%',#{username},'%');
    ```

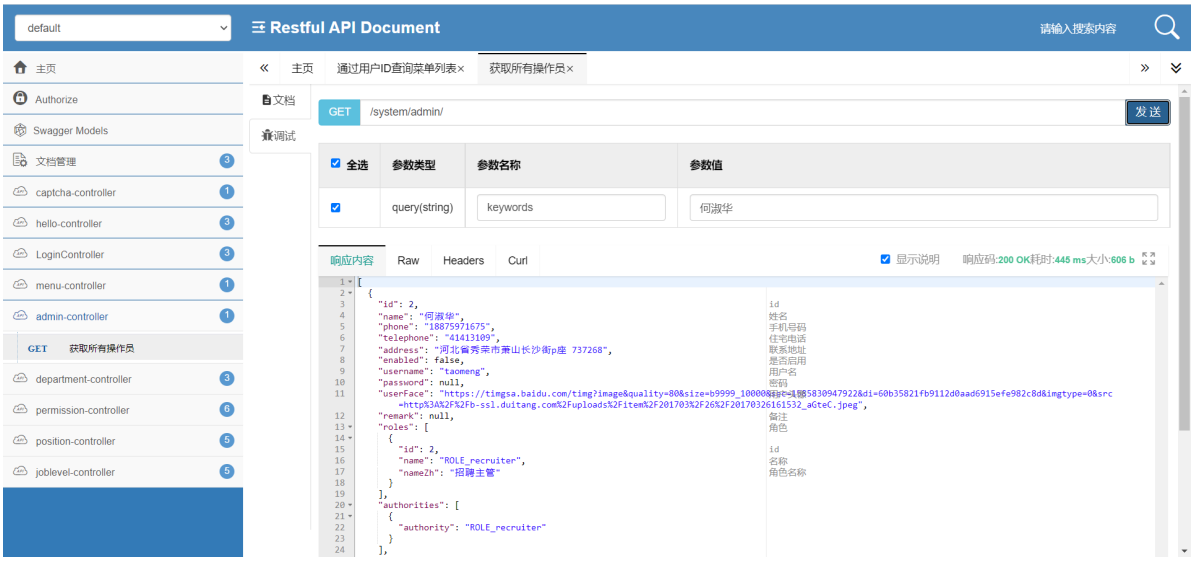
  - [ ] 方式三：`"%"#{值}"%"`【推荐使用这种方式】

    ```
    select * from t_user where username like "%"#{username}"%";
    ```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.AdminMapper">
```

```
    <!-- 通用查询映射结果 -->
    <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.Admin">
        <id column="id" property="id" />
        <result column="name" property="name" />
        <result column="phone" property="phone" />
        <result column="telephone" property="telephone" />
        <result column="address" property="address" />
        <result column="enabled" property="enabled" />
        <result column="username" property="username" />
        <result column="password" property="password" />
        <result column="userFace" property="userFace" />
        <result column="remark" property="remark" />
    </resultMap>

    <!-- 通用查询结果列 -->
    <sql id="Base_Column_List">
        id, name, phone, telephone, address, enabled, username, password, userFace,
remark
    </sql>

    <!-- 获取所有操作员 -->
    <resultMap id="AdminWithRole" type="com.xxxx.server.pojo.Admin"
extends="BaseResultMap">
        <collection property="roles" ofType="com.xxxx.server.pojo.Role">
            <id column="id" property="id" />
            <result column="rname" property="name" />
            <result column="rnameZh" property="nameZh" />
        </collection>
    </resultMap>

    <!-- 获取所有操作员 -->
    <select id="getAllAdmins" resultMap="AdminWithRole">
        select
            a.id,
            a.enabled,
            a.name,
            a.address,
            a.phone,
            a.remark,
            a.telephone,
            a.userFace,
            a.username,
            r.id as rid,
            r.name as rname,
            r.nameZh as rnameZh
        from
            t_admin as a
            LEFT JOIN t_admin_role as ar on a.id = ar.adminId
            LEFT JOIN t_role as r on ar.rid = r.id
        where a.id != #{id}
            <if test="null != keywords and '' != keywords">
                and a.name like concat('%',#{keywords},'%')
            </if>
        order by
            a.id
    </select>
</mapper>
```

26.6 Open Restful API and test



27. 更新和删除操作员

27.1 Admin

> ☐ 禁用@data注解自动生成get方法
>
> 1.由于@Data注解自动生成了@Getter方法.
> 2.同时Admin实现了UserDetails，并且重写isEnabled方法。此时程序不知道使用哪个方法.
> 3.解决方法：使用@Getter(AccessLevel.NONE)注解来禁用lombok自动生成的Getter()方法

```
package com.xxxx.server.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import java.util.Collection;
import java.util.List;
import java.util.stream.Collectors;
import lombok.AccessLevel;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
```

```java
/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_admin")
@ApiModel(value = "Admin对象", description = "")
public class Admin implements Serializable, UserDetails {

  private static final long serialVersionUID = 1L;

  @ApiModelProperty(value = "id")
  @TableId(value = "id", type = IdType.AUTO)
  private Integer id;

  @ApiModelProperty(value = "姓名")
  private String name;

  @ApiModelProperty(value = "手机号码")
  private String phone;

  @ApiModelProperty(value = "住宅电话")
  private String telephone;

  @ApiModelProperty(value = "联系地址")
  private String address;

  @ApiModelProperty(value = "是否启用")
  @Getter(AccessLevel.NONE)
  private Boolean enabled;

  @ApiModelProperty(value = "用户名")
  private String username;

  @ApiModelProperty(value = "密码")
  private String password;

  @ApiModelProperty(value = "用户头像")
  private String userFace;

  @ApiModelProperty(value = "备注")
  private String remark;

  @ApiModelProperty(value = "角色")
  @TableField(exist = false)
  private List<Role> roles;

  @Override
  public Collection<? extends GrantedAuthority> getAuthorities() {
    List<SimpleGrantedAuthority> authorities =
        roles.stream()
            .map(role -> new SimpleGrantedAuthority(role.getName()))
            .collect(Collectors.toList());
```

```
    return authorities;
  }

  @Override
  public boolean isAccountNonExpired() {
    return true;
  }

  @Override
  public boolean isAccountNonLocked() {
    return true;
  }

  @Override
  public boolean isCredentialsNonExpired() {
    return true;
  }

  @Override
  public boolean isEnabled() {
    return enabled;
  }
}
```

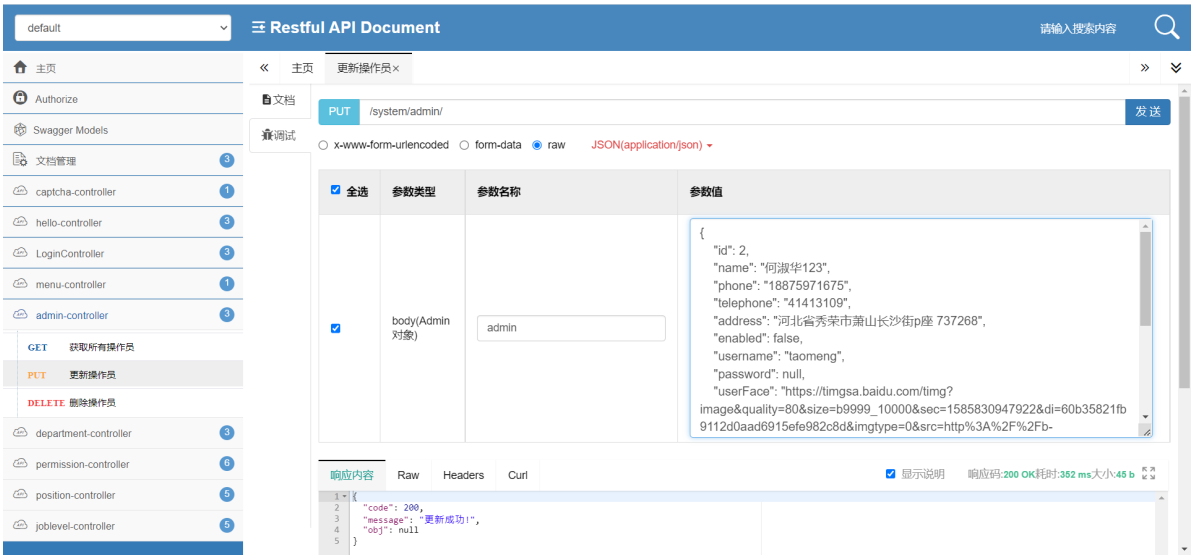
27.2 AdminController

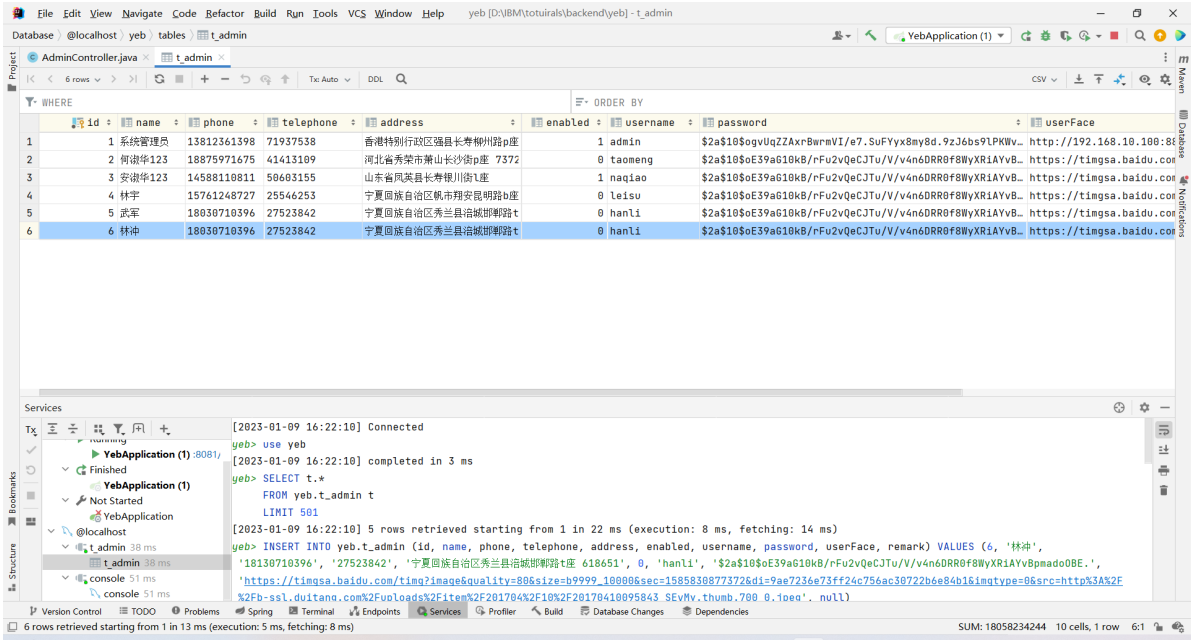
```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IAdminService;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/system/admin")
public class AdminController {
```

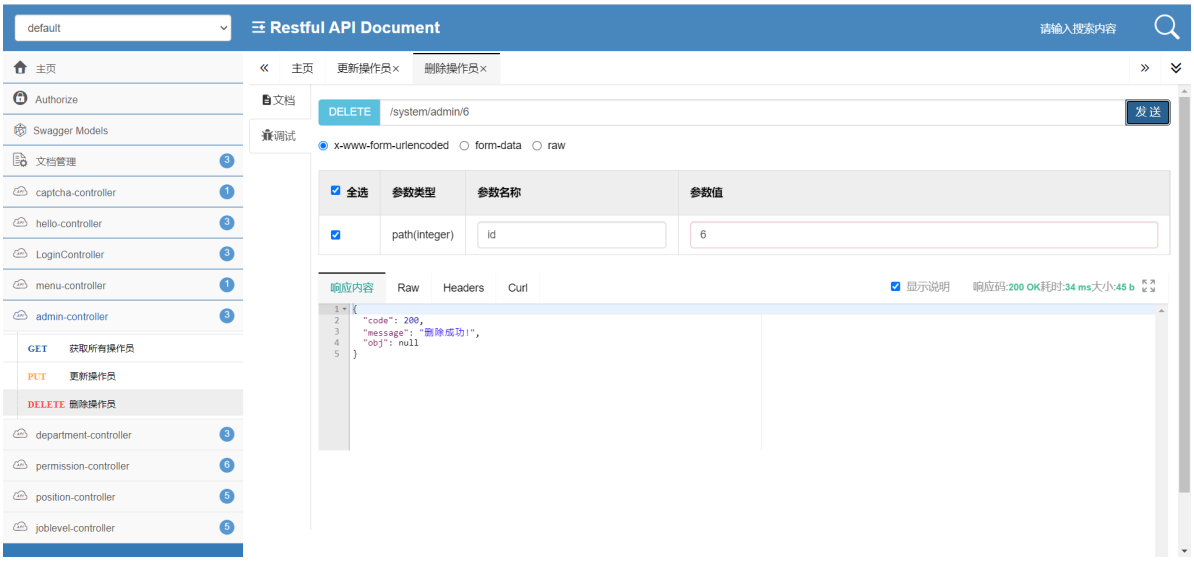
```
  @Autowired
  private IAdminService adminService;

  @ApiOperation(value = "获取所有操作员")
  @GetMapping("/")
  public List<Admin> getAllAdmins(String keywords){
    return adminService.getAllAdmins(keywords);
  }

  @ApiOperation(value = "更新操作员")
  @PutMapping("/")
  public RespBean updateAdmin(@RequestBody Admin admin) {
    if (adminService.updateById(admin)) {
      return RespBean.success("更新成功!");
    }
    return RespBean.error("删除失败!");
  }

  @ApiOperation(value = "删除操作员")
  @DeleteMapping("/{id}")
  public RespBean deleteAdmin(@PathVariable Integer id) {
    if (adminService.removeById(id)) {
      return RespBean.success("删除成功!");
    }
    return RespBean.error("删除失败!");
  }
}
```

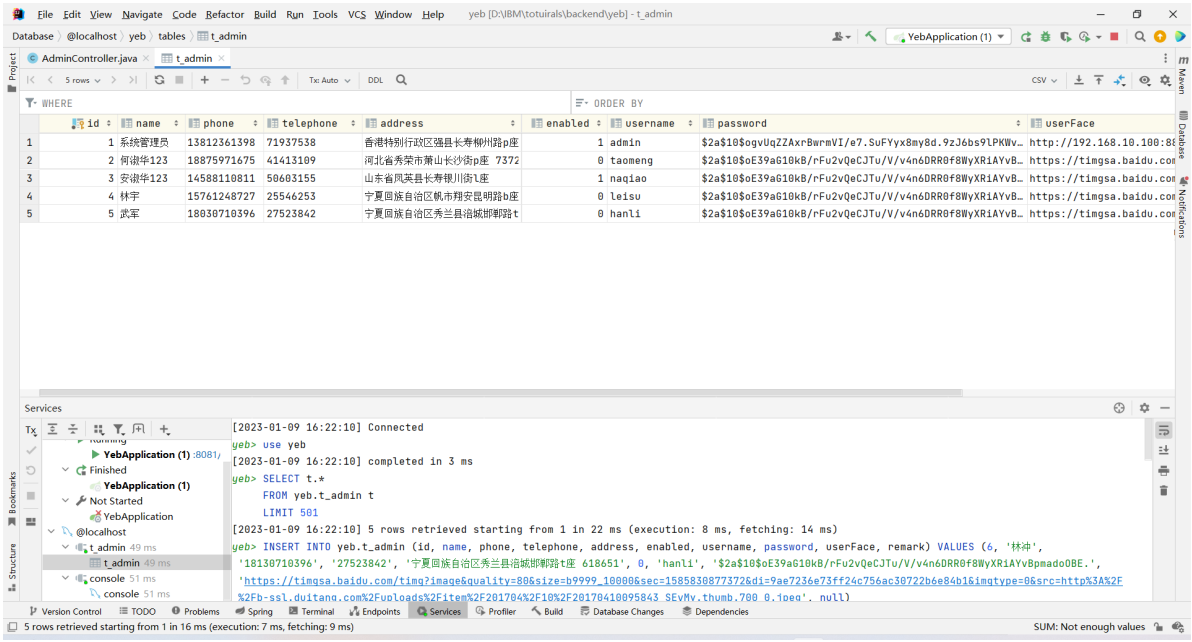
27.3 Open Restful API and test

```
{
    "id": 2,
    "name": "何淑华123",
    "phone": "18875971675",
    "telephone": "41413109",
    "address": "河北省秀荣市萧山长沙街p座 737268",
    "enabled": false,
    "username": "taomeng",
    "password": null,
    "userFace": "https://timgsa.baidu.com/timg?
image&quality=80&size=b9999_10000&sec=1585830947922&di=60b35821fb9112d0aad6915efe982c8
d&imgtype=0&src=http%3A%2F%2Fb-
ssl.duitang.com%2Fuploads%2Fitem%2F201703%2F26%2F20170326161532_aGteC.jpeg",
    "remark": null,
    "roles": [
      {
        "id": 2,
        "name": "ROLE_recruiter",
        "nameZh": "招聘主管"
      }
    ]
  }
```


default
Restful API Document
请输入搜索内容
主页
Authorize
Swagger Models
文档管理
captcha-controller
hello-controller
LoginController
menu-controller
admin-controller
GET 获取所有操作员
PUT 更新操作员
DELETE 删除操作员
department-controller
permission-controller
position-controller
joblevel-controller
主页
更新操作员×
文档
调试
PUT /system/admin/
发送
x-www-form-urlencoded form-data raw JSON(application/json)
全选 参数类型 参数名称 参数值
body(Admin 对象)
admin
{
"id": 2,
"name": "何淑华123",
"phone": "18875971675",
"telephone": "41413109",
"address": "河北省秀荣市萧山长沙街p座 737268",
"enabled": false,
"username": "taomeng",
"password": null,
"userFace": "https://timgsa.baidu.com/timg?
image&quality=80&size=b9999_10000&sec=1585830947922&di=60b35821fb
9112d0aad6915efe982c8d&imgtype=0&src=http%3A%2F%2Fb-
响应内容 Raw Headers Curl
显示说明
响应码:200 OK耗时:352 ms大小:45 b
{
"code": 200,
"message": "更新成功!",
"obj": null
}


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
yeb [D:\IBM\totuirals\backend\yeb] - t_admin
Database @localhost yeb tables t_admin
YebApplication (1)
AdminController.java
t_admin
WHERE
ORDER BY
id name phone telephone address enabled username password userFace
1 1 系统管理员 13812361398 71937538 香港特别行政区强县长寿柳州路p座 1 admin $2a$10$qgvUqZZAxrBwrmVI/e7.SuFYyx8my8d.9zJ6bs9lPKWv… http://192.168.10.100:8
2 2 何淑华123 18875971675 41413109 河北省秀荣市萧山长沙街p座 7372 0 taomeng $2a$10$oE39aG10kB/rFu2vQeCJTu/V/v4n6DRR0f8WyXRiAYvB… https://timgsa.baidu.com
3 3 安淑华123 14588110811 50603155 山东省凤英县长寿银川街t座 1 naqiao $2a$10$oE39aG10kB/rFu2vQeCJTu/V/v4n6DRR0f8WyXRiAYvB… https://timgsa.baidu.com
4 4 林宇 15761248727 25546253 宁夏回族自治区帆市翔安昆明路b座 0 leisu $2a$10$oE39aG10kB/rFu2vQeCJTu/V/v4n6DRR0f8WyXRiAYvB… https://timgsa.baidu.com
5 5 武军 18030710396 27523842 宁夏回族自治区秀兰县洛城邯郸路t 0 hanli $2a$10$oE39aG10kB/rFu2vQeCJTu/V/v4n6DRR0f8WyXRiAYvB… https://timgsa.baidu.com
6 6 林冲 18030710396 27523842 宁夏回族自治区秀兰县洛城邯郸路t 0 hanli $2a$10$oE39aG10kB/rFu2vQeCJTu/V/v4n6DRR0f8WyXRiAYvB… https://timgsa.baidu.com
Services
YebApplication (1) :8081/
Finished
YebApplication (1)
Not Started
YebApplication
@localhost
t_admin 38 ms
t_admin 38 ms
console 51 ms
console 51 ms
[2023-01-09 16:22:10] Connected
yeb> use yeb
[2023-01-09 16:22:10] completed in 3 ms
yeb> SELECT t.*
FROM yeb.t_admin t
LIMIT 501
[2023-01-09 16:22:10] 5 rows retrieved starting from 1 in 22 ms (execution: 8 ms, fetching: 14 ms)
yeb> INSERT INTO yeb.t_admin (id, name, phone, telephone, address, enabled, username, password, userFace, remark) VALUES (6, '林冲',
'18130710396', '27523842', '宁夏回族自治区秀兰县洛城邯郸路t座 618651', 0, 'hanli', '$2a$10$oE39aG10kB/rFu2vQeCJTu/V/v4n6DRR0f8WyXRiAYvBpmadoOBE.',
'https://timgsa.baidu.com/timg?image&quality=80&size=b9999_10000&sec=1585830877372&di=9ae7236e73ff24c756ac30722b6e84b1&imgtype=0&src=http%3A%2F
%2Fb-ssl.duitang.com%2Fuploads%2Fitem%2F201704%2F10%2F20170410095843_SEvMv.thumb.700_0.jpeg', null)
Version Control TODO Problems Spring Terminal Endpoints Services Profiler Build Database Changes Dependencies
6 rows retrieved starting from 1 in 13 ms (execution: 5 ms, fetching: 8 ms)
SUM: 18058234244 10 cells, 1 row 6:1


default
Restful API Document
请输入搜索内容
主页
Authorize
Swagger Models
文档管理
captcha-controller
hello-controller
LoginController
menu-controller
admin-controller
GET 获取所有操作员
PUT 更新操作员
DELETE 删除操作员
department-controller
permission-controller
position-controller
joblevel-controller
主页
更新操作员×
删除操作员×
文档
调试
DELETE /system/admin/6
发送
x-www-form-urlencoded form-data raw
全选 参数类型 参数名称 参数值
path(integer)
id
6
响应内容 Raw Headers Curl
显示说明
响应码:200 OK耗时:34 ms大小:45 b
{
"code": 200,
"message": "删除成功!",
"obj": null
}



28. 更新操作员角色

- [ ] Spring boot & idea 中 mapper注入 红色警告提示怎么解决，什么情况下会出现?
  - [ ] 解决的方案
    - **1、为@Autowired注解设置request=false**

```
@Autowired(request=false)
private UserDao userDao;
```

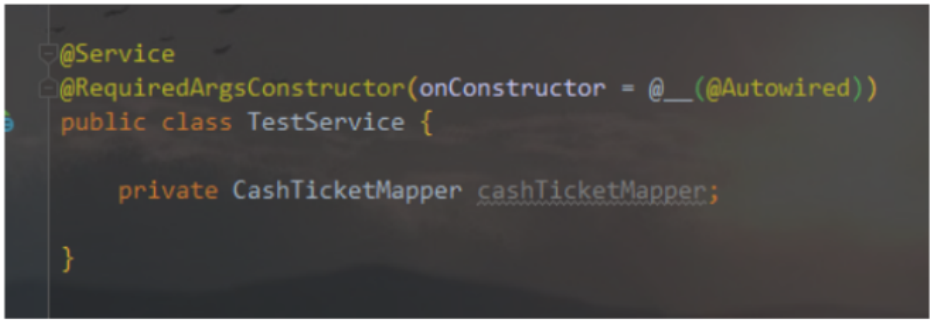
    - **2、用@Resource代替@Autowired**

```
@Resource
private UserDao userDao;
```

    - **3、在Dao层（或Mapper层）接口上加上@Repository注解**

```
@Repository
pubulic interface UserDao
```

    - **4、使用Lombox**



Lombok生成的代码是这样的：

```
@Service
public class TestService {
   private CashTicketMapper cashTicketMapper;
   @Autowired
   public TestService(CashTicketMapper cashTicketMapper;) {
      this.cashTicketMapper= cashTicketMapper;
   }
}
```


28.1 AdminController

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IAdminService;
import com.xxxx.server.service.IRoleService;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
```

```
@RequestMapping("/system/admin")
public class AdminController {

  @Autowired
  private IAdminService adminService;

  @Autowired
  private IRoleService roleService;


  @ApiOperation(value = "更新操作员角色")
  @PutMapping("/role")
  public RespBean updateAdminRole(Integer adminId, Integer[] rids) {
    return adminService.updateAdminRole(adminId, rids);
  }
}
```

28.2 IAdminService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import java.util.List;
import javax.servlet.http.HttpServletRequest;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IAdminService extends IService<Admin> {
  /**
   * @param adminId
   * @param rids
   * @return
   * @Description //TODO 更新操作员角色
   */
  RespBean updateAdminRole(Integer adminId, Integer[] rids);
}
```

28.3 AdminServiceImpl

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.config.jwt.JwtTokenUtil;
```

```java
import com.xxxx.server.mapper.AdminMapper;
import com.xxxx.server.mapper.AdminRoleMapper;
import com.xxxx.server.mapper.RoleMapper;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.AdminRole;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IAdminService;
import com.xxxx.server.utils.AdminUtils;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;

@Service
public class AdminServiceImpl extends ServiceImpl<AdminMapper, Admin> implements
IAdminService {

  @Autowired
  private UserDetailsService userDetailsService;
  @Autowired
  private PasswordEncoder passwordEncoder;
  @Autowired
  private JwtTokenUtil jwtTokenUtil;
  @Value("${jwt.tokenHead}")
  private String tokenHead;
  @Autowired
  private AdminMapper adminMapper;
  @Autowired
  private RoleMapper roleMapper;
  @Autowired
  private AdminRoleMapper adminRoleMapper;

  /**
   * @Description //TODO 更新操作员角色
   * @param adminId
   * @param rids
   * @return
   */
  @Override
  public RespBean updateAdminRole(Integer adminId, Integer[] rids) {

    //在更新之前，先删除操作员角色
    adminRoleMapper.delete(new QueryWrapper<AdminRole>().eq("adminId", adminId));

    //添加操作员角色
    Integer result = adminRoleMapper.addAdminRole(adminId,rids);
    if (result == rids.length) {
```

```
        return RespBean.success("添加成功!");
      }
      return RespBean.error("添加失败!");
    }
}
```

28.4 AdminRoleMapper(★)

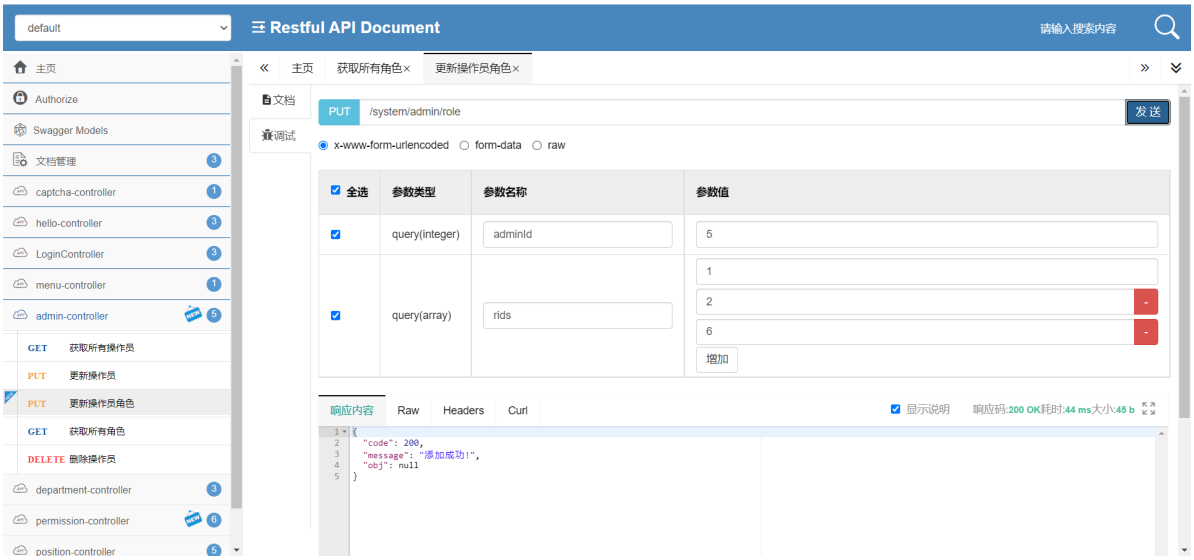
```
package com.xxxx.server.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.xxxx.server.pojo.AdminRole;
import com.xxxx.server.pojo.common.RespBean;
import io.lettuce.core.dynamic.annotation.Param;

/**
 * <p>
 *  Mapper 接口
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface AdminRoleMapper extends BaseMapper<AdminRole> {

    /**
     * @Description //TODO 更新操作员角色
     * @param adminId
     * @param rids
     * @return
     */
    Integer addAdminRole(@Param("adminId") Integer adminId, @Param("rids") Integer[]
rids);
}
```

28.5 AdminRoleMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.AdminRoleMapper">

    <!-- 通用查询映射结果 -->
    <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.AdminRole">
        <id column="id" property="id" />
        <result column="adminId" property="adminId" />
        <result column="rid" property="rid" />
    </resultMap>

    <!-- 通用查询结果列 -->
    <sql id="Base_Column_List">
        id, adminId, rid
```

```
    </sql>

    <!-- 更新操作员角色 -->
    <insert id="addAdminRole">
        insert into t_admin_role(adminId, rid) values
        <foreach collection="rids" item="rid" separator=",">
            (#{adminId},#{rid})
        </foreach>
    </insert>

</mapper>
```

28.6 Open Restful API and test



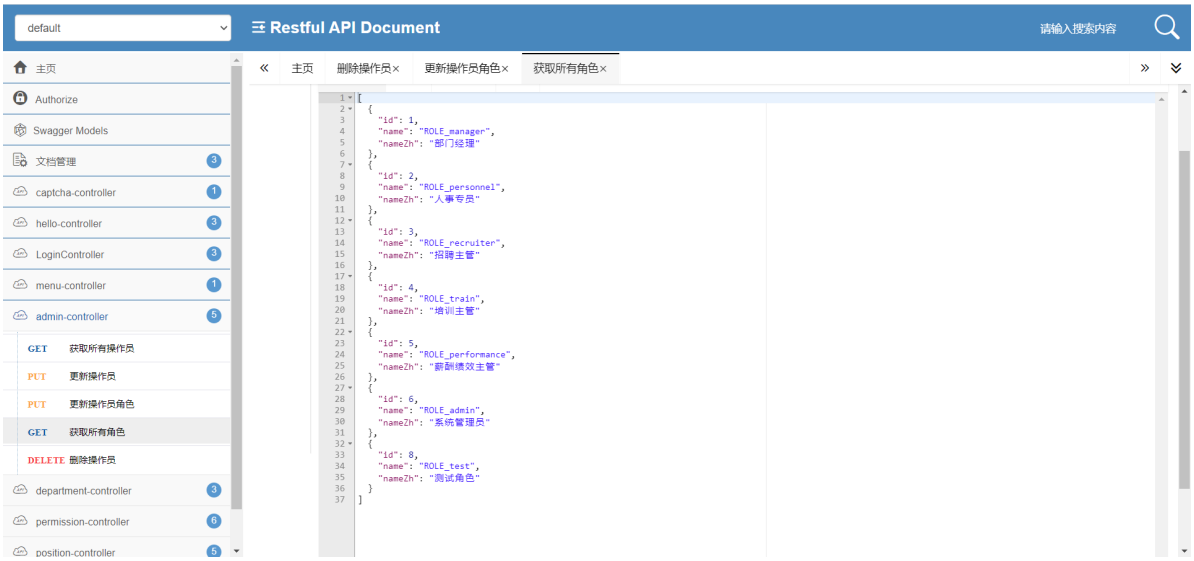
29. 获取所有角色

29.1 AdminController

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IAdminService;
import com.xxxx.server.service.IRoleService;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
```

```java
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/system/admin")
public class AdminController {

  @Autowired
  private IAdminService adminService;

  @Autowired
  private IRoleService roleService;

  @ApiOperation(value = "获取所有角色")
  @GetMapping("/roles")
  public List<Role> getAllRoles() {
    return roleService.list();
  }

}
```

2.Open Restful API and test



30. 获取所有员工(分页查询)

- [ ] 分页插件配置 `MybatisPlusConfig`
- [ ] 分页公共返回对象 `RespPageBean`
- [ ] 全局日期转换配置DateConverter类
- [ ] Employee实体格式转换为 `"yyyy-MM-dd"` ,时区 `"Asia/Shanghai"` 。

```
@JsonFormat(pattern = "yyyy-MM-dd",timezone = "Asia/Shanghai")
```

- [ ] Employee表无法外键ID，需要处理这部分对象。

```
    xxxxxxxxxx @ApiModelProperty(value = "民族")@TableField(exist =
false)private Nation nation;@ApiModelProperty(value = "政治面貌")
 @TableField(exist = false)  private PoliticsStatus politicsStatus;
 @ApiModelProperty(value = "部门")  @TableField(exist = false)  private
Department department;  @ApiModelProperty(value = "职位")  @TableField(exist
= false)  private Position  position;  @ApiModelProperty(value = "职称")
 @TableField(exist = false)  private Joblevel joblevel;java
```

30.1 Employee

```
package com.xxxx.server.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.annotation.JsonFormat;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import java.time.LocalDate;
import lombok.Data;
import lombok.EqualsAndHashCode;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_employee")
@ApiModel(value="Employee对象", description="")
public class Employee implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "员工编号")
```

```java
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "员工姓名")
    private String name;

    @ApiModelProperty(value = "性别")
    private String gender;

    @ApiModelProperty(value = "出生日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    private LocalDate birthday;

    @ApiModelProperty(value = "身份证号")
    private String idCard;

    @ApiModelProperty(value = "婚姻状况")
    private String wedlock;

    @ApiModelProperty(value = "民族")
    private Integer nationId;

    @ApiModelProperty(value = "籍贯")
    private String nativePlace;

    @ApiModelProperty(value = "政治面貌")
    private Integer politicId;

    @ApiModelProperty(value = "邮箱")
    private String email;

    @ApiModelProperty(value = "电话号码")
    private String phone;

    @ApiModelProperty(value = "联系地址")
    private String address;

    @ApiModelProperty(value = "所属部门")
    private Integer departmentId;

    @ApiModelProperty(value = "职称ID")
    private Integer jobLevelId;

    @ApiModelProperty(value = "职位ID")
    private Integer posId;

    @ApiModelProperty(value = "聘用形式")
    private String engageForm;

    @ApiModelProperty(value = "最高学历")
    private String tiptopDegree;

    @ApiModelProperty(value = "所属专业")
    private String specialty;

    @ApiModelProperty(value = "毕业院校")
    private String school;
```

```
    @ApiModelProperty(value = "入职日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    private LocalDate beginDate;

    @ApiModelProperty(value = "在职状态")
    private String workState;

    @ApiModelProperty(value = "工号")
    private String workID;

    @ApiModelProperty(value = "合同期限")
    private Double contractTerm;

    @ApiModelProperty(value = "转正日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    private LocalDate conversionTime;

    @ApiModelProperty(value = "离职日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    private LocalDate notWorkDate;

    @ApiModelProperty(value = "合同起始日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    private LocalDate beginContract;

    @ApiModelProperty(value = "合同终止日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    private LocalDate endContract;

    @ApiModelProperty(value = "工龄")
    private Integer workAge;

    @ApiModelProperty(value = "工资账套ID")
    private Integer salaryId;

    @ApiModelProperty(value = "民族")
    @TableField(exist = false)
    private Nation nation;

    @ApiModelProperty(value = "政治面貌")
    @TableField(exist = false)
    private PoliticsStatus politicsStatus;

    @ApiModelProperty(value = "部门")
    @TableField(exist = false)
    private Department department;

    @ApiModelProperty(value = "职称")
    @TableField(exist = false)
    private Joblevel joblevel;

    @ApiModelProperty(value = "职位")
    @TableField(exist = false)
    private Position position;
}
```

30.2 DateConverter

```
package com.xxxx.server.config.converter;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import org.springframework.core.convert.converter.Converter;

/**
 * @Description //TODO 日期转换
 */
public class DateConverter implements Converter<String,LocalDate> {

  @Override
  public LocalDate convert(String source) {
    try {
      return LocalDate.parse(source, DateTimeFormatter.ofPattern("yyyy-MM-dd"));
    } catch (Exception e) {
      e.printStackTrace();
    }
    return null;
  }
}
```

30.3 RespPageBean

```
package com.xxxx.server.pojo.common;


import io.swagger.annotations.ApiModel;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * @Description //TODO 分页公共返回对象
 * @author lizongzai
 * @since 1.0.0
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@ApiModel(value = "RespPageBean对象", description = "分页对象")
public class RespPageBean {

  //总条数
  private Long total;
  //数据list
  private List<?> data;

}
```

30.4 MybatisPlusConfig

```
package com.xxxx.server.config.page;

import com.baomidou.mybatisplus.extension.plugins.MybatisPlusInterceptor;
import com.baomidou.mybatisplus.extension.plugins.inner.PaginationInnerInterceptor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * @Description //TODO 分页配置
 * @author lizongzai
 * @since 1.0.0
 */
@Configuration
public class MybatisPlusConfig {

  @Bean
  public MybatisPlusInterceptor paginationInnerInterceptor() {
    // 1、定义MP拦截器
    MybatisPlusInterceptor interceptor = new MybatisPlusInterceptor();
    // 2、添加具体的拦截器
    interceptor.addInnerInterceptor(new PaginationInnerInterceptor());
    return interceptor;
  }
}
```

30.5 EmployeeController

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IEmployeeService;
import io.swagger.annotations.ApiOperation;
import java.time.LocalDate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 * 前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/employee/basic")
public class EmployeeController {
```

```
  @Autowired
  private IEmployeeService employeeService;

  @ApiOperation(value = "获取所有员工(分页查询)")
  @GetMapping("/")
  public RespPageBean getAllEmployee(@RequestParam(defaultValue = "1") Integer
currentPage,
                                     @RequestParam(defaultValue = "10") Integer
pageSize,
                                     Employee employee,
                                     LocalDate beginDateScope) {
    return
employeeService.getAllEmployeeByPage(currentPage,pageSize,employee,beginDateScope);

  }

}
```

30.6 IEmployeeService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespPageBean;
import java.time.LocalDate;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IEmployeeService extends IService<Employee> {

  /**
   * @Description //TODO 获取所有员工(分页查询)
   * @param currentPage
   * @param pageSize
   * @param employee
   * @param beginDateScope
   * @return
   */
  RespPageBean getAllEmployeeByPage(Integer currentPage, Integer pageSize, Employee
employee, LocalDate beginDateScope);
}
```

30.7 EmployeeServiceImpl(★)

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.mapper.EmployeeMapper;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IEmployeeService;
import java.time.LocalDate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * <p>
 * 服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class EmployeeServiceImpl extends ServiceImpl<EmployeeMapper, Employee>
implements
    IEmployeeService {

  @Autowired
  private EmployeeMapper employeeMapper;

  /**
   * @param currentPage
   * @param pageSize
   * @param employee
   * @param beginDateScope
   * @return
   * @Description //TODO 获取所有员工(分页查询)
   */
  @Override
  public RespPageBean getAllEmployeeByPage(Integer currentPage, Integer pageSize,
Employee employee,
      LocalDate beginDateScope) {
    //开启分页
    Page<Employee> employeePage = new Page<>();
    IPage<Employee> employeeByPage =
employeeMapper.getAllEmployeeByPage(employeePage,employee,beginDateScope);
    RespPageBean respPageBean = new RespPageBean(employeeByPage.getTotal(),
employeeByPage.getRecords());
    return respPageBean;
  }
}
```

30.8 EmployeeMapper

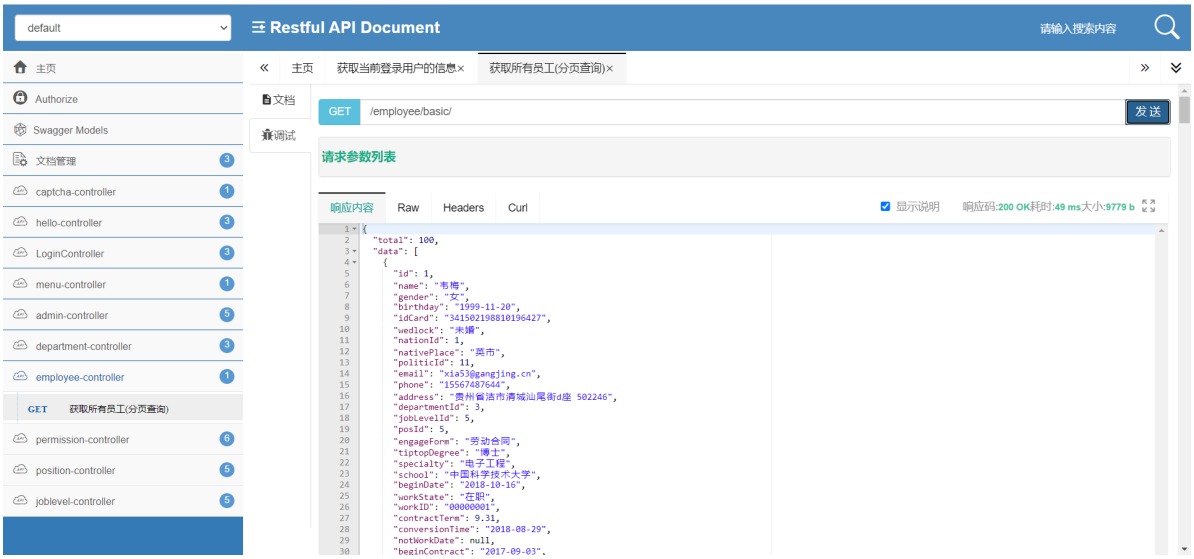
```
package com.xxxx.server.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.xxxx.server.pojo.Employee;
import java.time.LocalDate;
import org.springframework.data.repository.query.Param;

/**
 * <p>
 * Mapper 接口
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface EmployeeMapper extends BaseMapper<Employee> {

  /**
   * TODO 获取所有员工(分页查询)
   * @param employeePage
   * @param employee
   * @param beginDateScope
   * @return
   */
  IPage<Employee> getAllEmployeeByPage(Page<Employee> employeePage,
      @Param("employee") Employee employee, @Param("beginDateScope") LocalDate
beginDateScope);
}
```

30.9 EmployeeMapper.xml(★★)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.EmployeeMapper">

    <!-- 通用查询映射结果 -->
    <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.Employee">
        <id column="id" property="id" />
        <result column="name" property="name" />
        <result column="gender" property="gender" />
        <result column="birthday" property="birthday" />
        <result column="idCard" property="idCard" />
        <result column="wedlock" property="wedlock" />
        <result column="nationId" property="nationId" />
        <result column="nativePlace" property="nativePlace" />
        <result column="politicId" property="politicId" />
        <result column="email" property="email" />
        <result column="phone" property="phone" />
        <result column="address" property="address" />
        <result column="departmentId" property="departmentId" />
```

```
        <result column="jobLevelId" property="jobLevelId" />
        <result column="posId" property="posId" />
        <result column="engageForm" property="engageForm" />
        <result column="tiptopDegree" property="tiptopDegree" />
        <result column="specialty" property="specialty" />
        <result column="school" property="school" />
        <result column="beginDate" property="beginDate" />
        <result column="workState" property="workState" />
        <result column="workID" property="workID" />
        <result column="contractTerm" property="contractTerm" />
        <result column="conversionTime" property="conversionTime" />
        <result column="notWorkDate" property="notWorkDate" />
        <result column="beginContract" property="beginContract" />
        <result column="endContract" property="endContract" />
        <result column="workAge" property="workAge" />
        <result column="salaryId" property="salaryId" />
    </resultMap>

    <resultMap id="EmployeeInfo" type="com.xxxx.server.pojo.Employee"
extends="BaseResultMap">
        <!-- 民族 -->
        <association property="nation" javaType="com.xxxx.server.pojo.Nation">
            <id column="nid" property="id"/>
            <result column="nname" property="name"/>
        </association>
        <!-- 政治面貌 -->
        <association property="politicsStatus"
javaType="com.xxxx.server.pojo.PoliticsStatus">
            <id column="pid" property="id"/>
            <result column="pname" property="name"/>
        </association>
        <!-- 部门 -->
        <association property="department" javaType="com.xxxx.server.pojo.Department">
            <id column="did" property="id"/>
            <result column="dname" property="name"/>
        </association>
        <!-- 职称 -->
        <association property="joblevel" javaType="com.xxxx.server.pojo.Joblevel">
            <id column="jid" property="id"/>
            <result column="jname" property="name"/>
        </association>
        <!-- 职位 -->
        <association property="position" javaType="com.xxxx.server.pojo.Position">
            <id column="posid" property="id"/>
            <result column="posname" property="name"/>
        </association>
    </resultMap>

    <!-- 通用查询结果列 -->
    <sql id="Base_Column_List">
        id, name, gender, birthday, idCard, wedlock, nationId, nativePlace, politicId,
email, phone, address, departmentId, jobLevelId, posId, engageForm, tiptopDegree,
specialty, school, beginDate, workState, workID, contractTerm, conversionTime,
notWorkDate, beginContract, endContract, workAge, salaryId
    </sql>

    <!-- 获取所有员工(分页查询) -->
    <select id="getAllEmployeeByPage" resultMap="EmployeeInfo">
```

```
        select
            e.*,
            n.id as nid,
            n.name as nname,
            p.id as pid,
            p.name as pname,
            d.id as did,
            d.name as dname,
            j.id as jid,
            j.name as jname,
            pos.id as posid,
            pos.name as posname
        from
            t_employee e,
            t_nation n,
            t_politics_status p,
            t_department d,
            t_joblevel j,
            t_position pos
        where
            e.nationId = n.id
          and e.politicId = p.id
          and e.jobLevelId = j.id
          and e.departmentId = d.id
          and e.posId = pos.id
          <if test="employee.name != null and '' != employee.name">
              AND e.name like concat('%',#{employee.name},'%')
          </if>
          <if test="employee.politicId != null">
              AND e.politicId like concat('%',#{employee.politicId},'%')
          </if>
          <if test="employee.nationId != null">
              AND e.nationId like concat('%',#{employee.nationId},'%')
          </if>
          <if test="employee.jobLevelId != null">
              AND e.jobLevelId like concat('%',#{employee.jobLevelId},'%')
          </if>
          <if test="employee.posId != null">
              AND e.posId like concat('%',#{employee.posId},'%')
          </if>
          <if test="employee.departmentId != null">
              AND e.departmentId like concat('%',#{employee.departmentId},'%')
          </if>
          <if test="employee.engageForm != null and '' != employee.engageForm">
              AND e.engageForm like concat('%',#{employee.engageForm},'%')
          </if>
        <if test="beginDateScope != null and beginDateScope.length == 2">
            AND e.beginDate between #{beginDateScope[0]} and beginDateScope[1]
        </if>
        ORDER BY e.id
    </select>

</mapper>
```



31. 获取所有政治面貌

31.1 Restful 接口

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.PoliticsStatus;
import com.xxxx.server.pojo.RespBean;
import com.xxxx.server.service.IPoliticsStatusService;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-19
 */
@RestController
@Api(tags = "PoliticsStatusController")
@RequestMapping("/employee/basic")
public class PoliticsStatusController {

  @Autowired
  private IPoliticsStatusService politicsStatusService;

  @ApiOperation(value = "获取所有政治面貌")
  @GetMapping("/politicsStatus")
  public List<PoliticsStatus> getPoliticsStatus() {
```

```
    return politicsStatusService.list();
  }
}
```

32. 获取所有职称

32.1 Restful 接口

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Joblevel;
import com.xxxx.server.pojo.RespBean;
import com.xxxx.server.service.IJoblevelService;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import java.time.LocalDateTime;
import java.util.Arrays;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 * 前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-19
 */
@RestController
@Api(tags = "JoblevelController")
@RequestMapping("/system/basic/joblevel")
public class JoblevelController {

  @Autowired
  private IJoblevelService joblevelService;

  @ApiOperation(value = "获取所有职称信息")
  @GetMapping("/")
  public List<Joblevel> getAllJobLevels() {
    return joblevelService.list();
  }
}
```

33. 获取所有民族

33.1 Restful 接口

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Nation;
import com.xxxx.server.service.INationService;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-19
 */
@RestController
@Api(tags = "NationController")
@RequestMapping("/employee/basic")
public class NationController {

  @Autowired
  private INationService nationService;

  @ApiOperation(value = "获取所有民族")
  @GetMapping("/nation")
  public List<Nation> getAllNations() {
    return nationService.list();
  }

}
```

34. 获取工号

34.1 EmployeeController

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.RespBean;
import com.xxxx.server.pojo.RespPageBean;
import com.xxxx.server.service.IEmployeeService;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
```

```java
import java.time.LocalDate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 * 前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-19
 */
@RestController
@Api(tags = "EmployeeController")
@RequestMapping("/employee/basic")
public class EmployeeController {

  @Autowired
  private IEmployeeService employeeService;

  @ApiOperation(value = "获取工号")
  @GetMapping("/maxWorkId")
  public RespBean getMaxWorkId() {
    return employeeService.getMaxWorkId();
  }

}
```

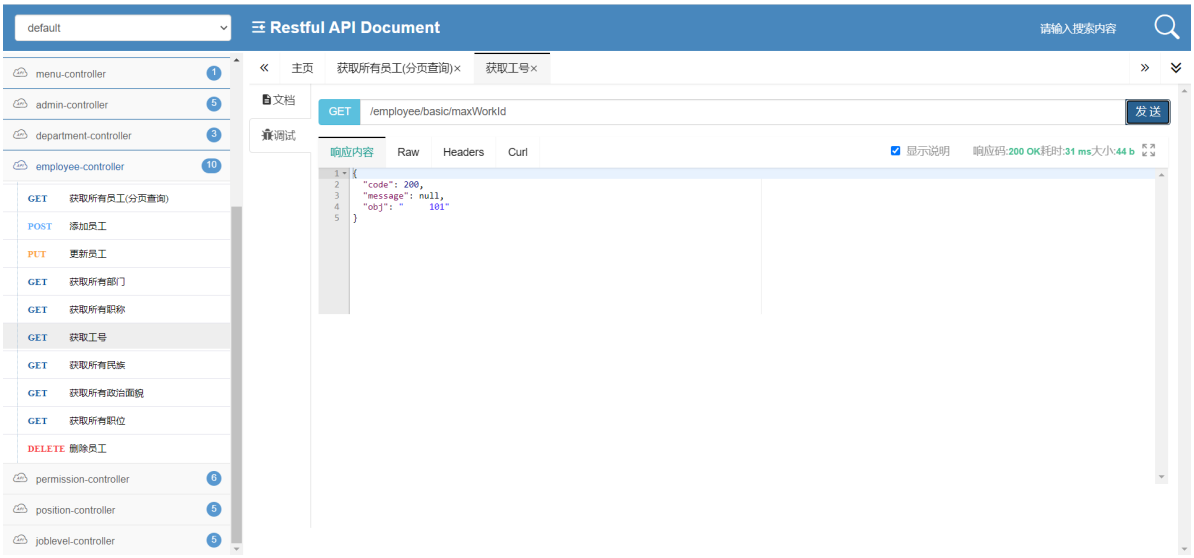
34.2 IEmployeeService

```java
  /**
   * @Description //TODO 获取工号
   * @Author lizongzai
   * @Date 2023/01/11 13:06
   * @return
   */
  RespBean getMaxWorkId();
```

34.3 EmployeeServiceImpl(★)

```java
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.mapper.EmployeeMapper;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IEmployeeService;
```

```java
import java.text.DecimalFormat;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.Map;
import javax.annotation.Resource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * <p>
 * 服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class EmployeeServiceImpl extends ServiceImpl<EmployeeMapper, Employee>
implements
    IEmployeeService {

  @Autowired
  @Resource
  private EmployeeMapper employeeMapper;
  /**
   * @Description //TODO 获取工号
   * @Author lizongzai
   * @Date 2023/01/11 13:06
   * @return
   */
  @Override
  public RespBean getMaxWorkId() {
    //获取员工Key和value
    List<Map<String,Object>> maps = employeeMapper.selectMaps(new
QueryWrapper<Employee>().select("max(workId)"));
    //System.out.println("maps = " + maps);
    //maps = [{max(workId)=00000100}]
    //只有一个最大值，那么get(0).get("max(workId")即可获取它的值
    //工号强制转换整型，然后再加1
    //maps.get(0).get("max(workId)"是一个Object对象，先将其转String
    //%8d，表示不足长度时，自动左补位，补位数为0
    return RespBean.success(null,
String.format("%8d",Integer.parseInt(maps.get(0).get("max(workId)").toString()) + 1));
  }
}
```



35. 添加员工

35.1 EmployeeController

```
@ApiOperation(value = "添加员工")
@PostMapping("/")
public RespBean addEmp(@RequestBody Employee employee) {
  return employeeService.addEmp(employee);
}
```

35.2 IEmployeeService

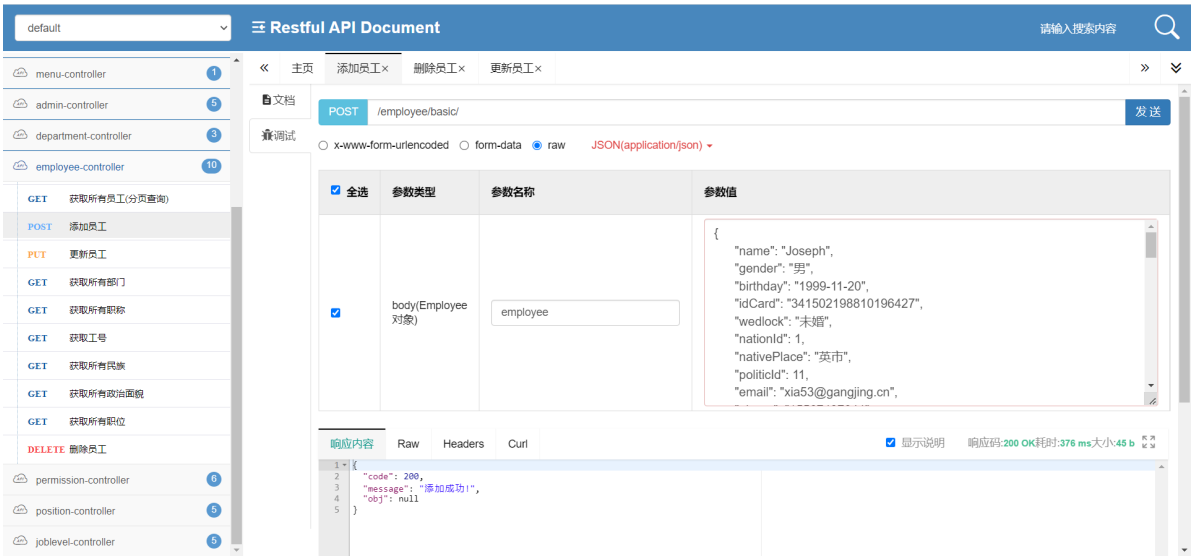
```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import java.time.LocalDate;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IEmployeeService extends IService<Employee> {

  /**
   * @Description //TODO 添加员工
   * @param employee
   * @return
```

```
    */
    RespBean addEmp(Employee employee);

}
```

35.3 EmployeeServiceImpl(★)

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.mapper.EmployeeMapper;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IEmployeeService;
import java.text.DecimalFormat;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.Map;
import javax.annotation.Resource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * <p>
 * 服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class EmployeeServiceImpl extends ServiceImpl<EmployeeMapper, Employee>
implements
    IEmployeeService {

  @Autowired
  @Resource
  private EmployeeMapper employeeMapper;

  @Override
  public RespBean addEmp(Employee employee) {
    //处理合同期限，保留两位小数
    LocalDate beginContract = employee.getBeginContract();
    LocalDate endContract = employee.getEndContract();
    long days = beginContract.until(endContract, ChronoUnit.DAYS); //计算合同的天数
    DecimalFormat decimalFormat = new DecimalFormat("##.00"); //设置保留两位小数
    employee.setContractTerm(Double.parseDouble(decimalFormat.format(days/365.00)));
//算计年数，保留两位小数点
    //System.out.println("合同年数 = " + employee.getContractTerm());
```

```
        //若添加成功，则返回一条记录。表示添加成功
        if (1==employeeMapper.insert(employee)) {
            return RespBean.success("添加成功!");
        }
        return RespBean.error("添加失败!");
    }
}
```



36. 更新员工

36. 1Resful接口

```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.RespBean;
import com.xxxx.server.pojo.RespPageBean;
import com.xxxx.server.service.IEmployeeService;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import java.time.LocalDate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
```

```
 * 前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-19
 */
@RestController
@Api(tags = "EmployeeController")
@RequestMapping("/employee/basic")
public class EmployeeController {

  @Autowired
  private IEmployeeService employeeService;

  @ApiOperation(value = "获取所有员工(分页查询)")
  @GetMapping("/")
  public RespPageBean getEmployeeByPage(@RequestParam(defaultValue = "1") Integer
currentPage,
      @RequestParam(defaultValue = "10") Integer size, Employee employee,
      LocalDate beginDateScope) {

    return employeeService.getEmployeeByPage(currentPage, size, employee,
beginDateScope);
  }

  @ApiOperation(value = "获取工号")
  @GetMapping("/maxWorkId")
  public RespBean getMaxWorkId() {
    return employeeService.getMaxWorkId();
  }

  @ApiOperation(value = "添加员工")
  @PostMapping("/")
  public RespBean addEmployee(@RequestBody Employee employee) {
    return employeeService.addEmployee(employee);
  }

  @ApiOperation(value = "更新员工")
  @PutMapping("/")
  public RespBean updateEmployee(@RequestBody Employee employee) {
    if (employeeService.updateById(employee)) {
      return RespBean.success("更新成功!");
    }
    return RespBean.error("更新失败!");
  }

  @ApiOperation(value = "删除员工")
  @DeleteMapping("/{id}")
  public RespBean removeEmployeeById(@PathVariable Integer id) {
    if (employeeService.removeById(id)) {
      return RespBean.success("删除成功!");
    }
    return RespBean.error("删除失败!");
  }


}
```

37. 员工数据导出

- [ ] 添加pom依赖包
- [ ] 使用 `@Excel(name = "")` 注解和 `@ExcelEntity(name = "")`

37.1 导入pom依赖包

```
<!--导数导入数据 依赖-->
<!--easy poi-->
<dependency>
    <groupId>cn.afterturn</groupId>
    <artifactId>easypoi-spring-boot-starter</artifactId>
    <version>4.1.3</version>
</dependency>
```

37.2 Employee实体

- [ ] @Excel(name = "婚姻状况")
- [ ] @Excel(name = "出生日期",width = 20, format = "yyyy-MM-dd")
- [ ] @Excel(name = "身份证号", width = 30)
- [ ] @Excel(name = "合同期限",suffix = "年")
- [ ] @ExcelEntity(name = "民族"), 属于 对象

```
package com.xxxx.server.pojo;

import cn.afterturn.easypoi.excel.annotation.Excel;
import cn.afterturn.easypoi.excel.annotation.ExcelEntity;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.annotation.JsonFormat;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import java.time.LocalDate;
import lombok.Data;
import lombok.EqualsAndHashCode;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
```

```
 * @since 2023-01-04
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_employee")
@ApiModel(value="Employee对象", description="")
public class Employee implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "员工编号")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "员工姓名")
    @Excel(name = "员工姓名")
    private String name;

    @ApiModelProperty(value = "性别")
    @Excel(name = "性别")
    private String gender;

    @ApiModelProperty(value = "出生日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "出生日期",width = 20, format = "yyyy-MM-dd")
    private LocalDate birthday;

    @ApiModelProperty(value = "身份证号")
    @Excel(name = "身份证号", width = 30)
    private String idCard;

    @ApiModelProperty(value = "婚姻状况")
    @Excel(name = "婚姻状况")
    private String wedlock;

    @ApiModelProperty(value = "民族")
    private Integer nationId;

    @ApiModelProperty(value = "籍贯")
    @Excel(name = "籍贯")
    private String nativePlace;

    @ApiModelProperty(value = "政治面貌")
    private Integer politicId;

    @ApiModelProperty(value = "邮箱")
    @Excel(name = "邮箱", width = 30)
    private String email;

    @ApiModelProperty(value = "电话号码")
    @Excel(name = "电话号码", width = 15)
    private String phone;

    @ApiModelProperty(value = "联系地址")
    @Excel(name = "联系地址", width = 40)
    private String address;

    @ApiModelProperty(value = "所属部门")
```

```
    private Integer departmentId;

    @ApiModelProperty(value = "职称ID")
    private Integer jobLevelId;

    @ApiModelProperty(value = "职位ID")
    private Integer posId;

    @ApiModelProperty(value = "聘用形式")
    @Excel(name = "聘用形式")
    private String engageForm;

    @ApiModelProperty(value = "最高学历")
    @Excel(name = "最高学历")
    private String tiptopDegree;

    @ApiModelProperty(value = "所属专业")
    @Excel(name = "所属专业", width = 20)
    private String specialty;

    @ApiModelProperty(value = "毕业院校")
    @Excel(name = "毕业院校", width = 20)
    private String school;

    @ApiModelProperty(value = "入职日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "入职日期", width = 20, format = "yyyy-MM-dd")
    private LocalDate beginDate;

    @ApiModelProperty(value = "在职状态")
    @Excel(name = "在职状态")
    private String workState;

    @ApiModelProperty(value = "工号")
    private String workID;

    @ApiModelProperty(value = "合同期限")
    @Excel(name = "合同期限",suffix = "年")
    private Double contractTerm;

    @ApiModelProperty(value = "转正日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "转正日期", width = 20, format = "yyyy-MM-dd")
    private LocalDate conversionTime;

    @ApiModelProperty(value = "离职日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "离职日期", width = 20, format = "yyyy-MM-dd")
    private LocalDate notWorkDate;

    @ApiModelProperty(value = "合同起始日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "合同起始日期", width = 20, format = "yyyy-MM-dd")
    private LocalDate beginContract;

    @ApiModelProperty(value = "合同终止日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "合同终止日期", width = 20, format = "yyyy-MM-dd")
```

```
    private LocalDate endContract;

    @ApiModelProperty(value = "工龄")
    @Excel(name = "工龄")
    private Integer workAge;

    @ApiModelProperty(value = "工资账套ID")
    private Integer salaryId;

    @ApiModelProperty(value = "民族")
    @TableField(exist = false)
    @ExcelEntity(name = "民族")
    private Nation nation;

    @ApiModelProperty(value = "政治面貌")
    @TableField(exist = false)
    @ExcelEntity(name = "政治面貌")
    private PoliticsStatus politicsStatus;

    @ApiModelProperty(value = "部门")
    @TableField(exist = false)
    @ExcelEntity(name = "部门")
    private Department department;

    @ApiModelProperty(value = "职称")
    @TableField(exist = false)
    @ExcelEntity(name = "职称")
    private Joblevel joblevel;

    @ApiModelProperty(value = "职位")
    @TableField(exist = false)
    @ExcelEntity(name = "职位")
    private Position position;

}
```

```
package com.xxxx.server.pojo;

import cn.afterturn.easypoi.excel.annotation.Excel;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import lombok.Data;
import lombok.EqualsAndHashCode;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
```

```
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_nation")
@ApiModel(value="Nation对象", description="")
public class Nation implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "民族")
    @Excel(name = "民族")
    private String name;
}
```

```
package com.xxxx.server.pojo;

import cn.afterturn.easypoi.excel.annotation.Excel;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import lombok.Data;
import lombok.EqualsAndHashCode;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_politics_status")
@ApiModel(value="PoliticsStatus对象", description="")
public class PoliticsStatus implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "政治面貌")
    @Excel(name = "政治面貌")
    private String name;
}
```

37.3 EmployeeController

```
package com.xxxx.server.controller;


import cn.afterturn.easypoi.excel.ExcelExportUtil;
import cn.afterturn.easypoi.excel.entity.ExportParams;
import cn.afterturn.easypoi.excel.entity.enmus.ExcelType;
import com.xxxx.server.pojo.Department;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.Joblevel;
import com.xxxx.server.pojo.Nation;
import com.xxxx.server.pojo.PoliticsStatus;
import com.xxxx.server.pojo.Position;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IDepartmentService;
import com.xxxx.server.service.IEmployeeService;
import com.xxxx.server.service.IJoblevelService;
import com.xxxx.server.service.INationService;
import com.xxxx.server.service.IPoliticsStatusService;
import com.xxxx.server.service.IPositionService;
import io.swagger.annotations.ApiOperation;
import java.io.IOException;
import java.net.URLEncoder;
import java.time.LocalDate;
import java.util.List;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletResponse;
import org.apache.poi.ss.usermodel.Workbook;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 * 前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/employee/basic")
public class EmployeeController {

  @Autowired
  private IEmployeeService employeeService;
  @Autowired
```

```
  private IPoliticsStatusService politicsStatusService;
  @Autowired
  IJoblevelService joblevelService;
  @Autowired
  private IPositionService positionService;
  @Autowired
  private INationService nationService;
  @Autowired
  private IDepartmentService departmentService;

  /**
   * @Description //TODO 由于使用stream流形式导出数，所以使用HttpServletResponse
   */

  @ApiOperation(value = "导出员工数据")
  @GetMapping(value = "/export", produces = "application/octet-stream")
  public void exportEmployee(HttpServletResponse response) {
    //查询数据
    List<Employee> employeeList = employeeService.getEmployee(null);
    //实例化,HSSF表示excel版本为2003
    ExportParams params = new ExportParams("员工表", "员工表", ExcelType.HSSF);
    //使用ExcelExportUtil工具导出,类型为工作簿
    Workbook workbook = ExcelExportUtil.exportExcel(params, Employee.class,
employeeList);
    //使用流导出文件
    ServletOutputStream outputStream = null;
    try {
      //输出流形式
      response.setHeader("content-type","application/octet-stream");
      //防止中文乱码
      response.setHeader("content-disposition","attachment;filename=" +
URLEncoder.encode("员工表.xls", "UTF-8"));
      //获取OutputStream输出流
      outputStream = response.getOutputStream();
      //保存文件,仅供下载使用
      workbook.write(outputStream);
    } catch (IOException e) {
      e.printStackTrace();
    } finally {
      if (outputStream != null) {
        try {
          //关闭输出流
          outputStream.flush();
          outputStream.close();
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
    }
  }

}
```

37.4 IEmployeeService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import java.time.LocalDate;
import java.util.List;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IEmployeeService extends IService<Employee> {
  /**
   * @Description //TODO 导出员工数据
   * @param id
   * @return
   */
  List<Employee> getEmployee(Integer id);
}
```

37.5 EmployeeServiceImpl(★★)

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.mapper.EmployeeMapper;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IEmployeeService;
import java.text.DecimalFormat;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.Map;
import javax.annotation.Resource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * <p>
 * 服务实现类
 * </p>
 *
```

```
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class EmployeeServiceImpl extends ServiceImpl<EmployeeMapper, Employee>
implements
    IEmployeeService {

  @Autowired
  private EmployeeMapper employeeMapper;

  /**
   * @Description //TODO 导出员工数据
   * @param id
   * @return
   */
  @Override
  public List<Employee> getEmployee(Integer id) {
    return employeeMapper.getEmployee(id);
  }
}
```

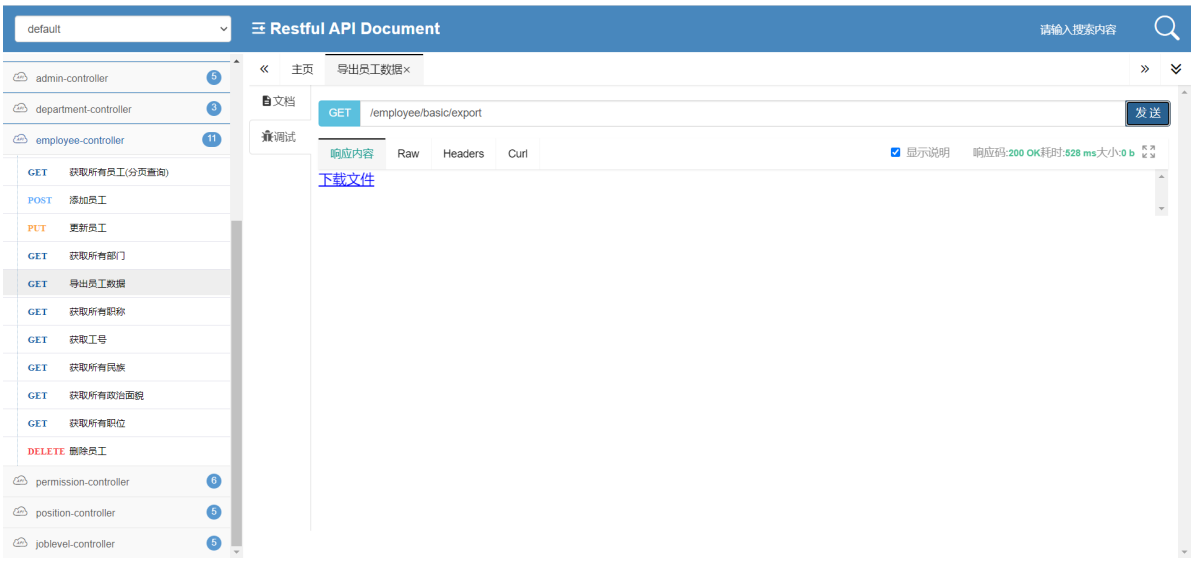
37.6 EmployeeMapper

```
package com.xxxx.server.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.xxxx.server.pojo.Employee;
import java.time.LocalDate;
import java.util.List;
import org.springframework.data.repository.query.Param;

/**
 * <p>
 * Mapper 接口
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface EmployeeMapper extends BaseMapper<Employee> {

  /**
   * @Description //TODO 导出员工数据
   * @param id
   * @return
   */
  List<Employee> getEmployee(Integer id);
}
```

37.7 EmployeeMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.EmployeeMapper">

    <!-- 通用查询映射结果 -->
    <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.Employee">
        <id column="id" property="id" />
        <result column="name" property="name" />
        <result column="gender" property="gender" />
        <result column="birthday" property="birthday" />
        <result column="idCard" property="idCard" />
        <result column="wedlock" property="wedlock" />
        <result column="nationId" property="nationId" />
        <result column="nativePlace" property="nativePlace" />
        <result column="politicId" property="politicId" />
        <result column="email" property="email" />
        <result column="phone" property="phone" />
        <result column="address" property="address" />
        <result column="departmentId" property="departmentId" />
        <result column="jobLevelId" property="jobLevelId" />
        <result column="posId" property="posId" />
        <result column="engageForm" property="engageForm" />
        <result column="tiptopDegree" property="tiptopDegree" />
        <result column="specialty" property="specialty" />
        <result column="school" property="school" />
        <result column="beginDate" property="beginDate" />
        <result column="workState" property="workState" />
        <result column="workID" property="workID" />
        <result column="contractTerm" property="contractTerm" />
        <result column="conversionTime" property="conversionTime" />
        <result column="notWorkDate" property="notWorkDate" />
        <result column="beginContract" property="beginContract" />
        <result column="endContract" property="endContract" />
        <result column="workAge" property="workAge" />
        <result column="salaryId" property="salaryId" />
    </resultMap>

    <resultMap id="EmployeeInfo" type="com.xxxx.server.pojo.Employee"
extends="BaseResultMap">
        <!-- 民族 -->
        <association property="nation" javaType="com.xxxx.server.pojo.Nation">
            <id column="nid" property="id"/>
            <result column="nname" property="name"/>
        </association>
        <!-- 政治面貌 -->
        <association property="politicsStatus"
javaType="com.xxxx.server.pojo.PoliticsStatus">
            <id column="pid" property="id"/>
            <result column="pname" property="name"/>
        </association>
        <!-- 部门 -->
        <association property="department" javaType="com.xxxx.server.pojo.Department">
            <id column="did" property="id"/>
            <result column="dname" property="name"/>
```

```
        </association>
        <!-- 职称 -->
        <association property="joblevel" javaType="com.xxxx.server.pojo.Joblevel">
            <id column="jid" property="id"/>
            <result column="jname" property="name"/>
        </association>
        <!-- 职位 -->
        <association property="position" javaType="com.xxxx.server.pojo.Position">
            <id column="posid" property="id"/>
            <result column="posname" property="name"/>
        </association>
    </resultMap>

    <!-- 通用查询结果列 -->
    <sql id="Base_Column_List">
        id, name, gender, birthday, idCard, wedlock, nationId, nativePlace, politicId,
email, phone, address, departmentId, jobLevelId, posId, engageForm, tiptopDegree,
specialty, school, beginDate, workState, workID, contractTerm, conversionTime,
notWorkDate, beginContract, endContract, workAge, salaryId
    </sql>

    <!--导出员工数据-->
    <select id="getEmployee" resultMap="EmployeeInfo">
        select
            e.*,
            n.id as nid,
            n.name as nname,
            p.id as pid,
            p.name as pname,
            d.id as did,
            d.name as dname,
            j.id as jid,
            j.name as jname,
            pos.id as posid,
            pos.name as posname
        from
            t_employee e,
            t_nation n,
            t_politics_status p,
            t_department d,
            t_joblevel j,
            t_position pos
        where
            e.nationId = n.id
          and e.politicId = p.id
          and e.jobLevelId = j.id
          and e.departmentId = d.id
          and e.posId = pos.id
          -- 如果传入id的值为空，那么就查询全部。否则就根据传入id编号进行查询
          <if test="id != null">
            and e.id = #{id}
          </if>
    </select>

</mapper>
```



38. 员工数据导入

38.1 实体对象

> 前提条件，实体对象需要添加如下注解即可:
>
> - [ ] @NoArgsConstructor //该注解表示添加无参构造方法
> - [ ] @RequiredArgsConstructor //该注解表示添加有参构造方法
> - [ ] @EqualsAndHashCode(callSuper = false, of = "name") //of = "name"表示重写Equals和HashCode
> - [ ] @NonNull //表示非空

38.1.1 Nation实体

```
package com.xxxx.server.pojo;

import cn.afterturn.easypoi.excel.annotation.Excel;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

/**
 * <p>
 *
 * </p>
```

```
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@NoArgsConstructor //该注解表示添加无参构造方法
@RequiredArgsConstructor //该注解表示添加有参构造方法
@EqualsAndHashCode(callSuper = false, of = "name") //of = "name"表示重写Equals和HashCode
@TableName("t_nation")
@ApiModel(value="Nation对象", description="")
public class Nation implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "民族")
    @Excel(name = "民族")
    @NonNull //表示非空
    private String name;
}
```

38.1.2 PoliticsStatus实体

```
package com.xxxx.server.pojo;

import cn.afterturn.easypoi.excel.annotation.Excel;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@NoArgsConstructor //该注解表示添加无参构造方法
@RequiredArgsConstructor //该注解表示添加有参构造方法
@EqualsAndHashCode(callSuper = false, of = "name")
@TableName("t_politics_status")
@ApiModel(value="PoliticsStatus对象", description="")
```

```
public class PoliticsStatus implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "政治面貌")
    @Excel(name = "政治面貌")
    @NonNull
    private String name;


}
```

38.1.3 Department实体

```
package com.xxxx.server.pojo;

import cn.afterturn.easypoi.excel.annotation.Excel;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import io.swagger.annotations.ApiOperation;
import java.io.Serializable;
import java.util.List;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@NoArgsConstructor //该注解表示添加无参构造方法
@RequiredArgsConstructor //该注解表示添加有参构造方法
@EqualsAndHashCode(callSuper = false, of = "name")
@TableName("t_department")
@ApiModel(value="Department对象", description="")
public class Department implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
```

```
    private Integer id;

    @ApiModelProperty(value = "部门名称")
    @Excel(name = "部门名称")
    @NonNull
    private String name;

    @ApiModelProperty(value = "父id")
    private Integer parentId;

    @ApiModelProperty(value = "路径")
    private String depPath;

    @ApiModelProperty(value = "是否启用")
    private Boolean enabled;

    @ApiModelProperty(value = "是否上级")
    private Boolean isParent;

    @ApiModelProperty(value = "子部门列表")
    @TableField(exist = false)
    private List<Department> children;

    @ApiModelProperty(value = "返回结果，仅供存储过程使用")
    @TableField(exist = false)
    private Integer result;

}
```

38.1.4 Joblevel实体

```
package com.xxxx.server.pojo;

import cn.afterturn.easypoi.excel.annotation.Excel;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.annotation.JsonFormat;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import java.time.LocalDateTime;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
```

```
@Data
@NoArgsConstructor //该注解表示添加无参构造方法
@RequiredArgsConstructor //该注解表示添加有参构造方法
@EqualsAndHashCode(callSuper = false,of = "name")
@TableName("t_joblevel")
@ApiModel(value="Joblevel对象", description="")
public class Joblevel implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "职称名称")
    @Excel(name = "职称")
    @NonNull
    private String name;

    @ApiModelProperty(value = "职称等级")
    @Excel(name = "职称等级")
    private String titleLevel;

    @ApiModelProperty(value = "创建时间")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    private LocalDateTime createDate;

    @ApiModelProperty(value = "是否启用")
    private Boolean enabled;


}
```

38.1.5 Position实体

```
package com.xxxx.server.pojo;

import cn.afterturn.easypoi.excel.annotation.Excel;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.annotation.JsonFormat;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import java.time.LocalDateTime;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

/**
 * <p>
 *
 * </p>
```

```
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@NoArgsConstructor //该注解表示添加无参构造方法
@RequiredArgsConstructor //该注解表示添加有参构造方法
@EqualsAndHashCode(callSuper = false, of = "name")
@TableName("t_position")
@ApiModel(value="Position对象", description="")
public class Position implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "职位")
    @Excel(name = "职位")
    @NonNull
    private String name;

    @ApiModelProperty(value = "创建时间")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    private LocalDateTime createDate;

    @ApiModelProperty(value = "是否启用")
    private Boolean enabled;


}
```

38.2 EmployeeController(★)

```
    @ApiOperation(value = "导入员工数据")
    @PostMapping("/import")
    public RespBean importEmployeeInfo(MultipartFile file) {

      //准备导入参数
      ImportParams params = new ImportParams();

      //删除title行
      params.setTitleRows(1);
      List<Nation> nations = nationService.list();

      try {
        //使用流导入数据
        List<Employee> employeeList =
  ExcelImportUtil.importExcel(file.getInputStream(),
            Employee.class, params);
        employeeList.forEach(employee -> {
          employee.setNationId(nations.indexOf(new
  Nation(employee.getNation().getName())).getId()));
```

```
        });

        if (employeeService.saveBatch(employeeList)) {
          return RespBean.success("导入成功!");
        }

      } catch (Exception e) {
        e.printStackTrace();
      }
      return RespBean.error("导入失败!");
    }
```

```
package com.xxxx.server.controller;


import cn.afterturn.easypoi.excel.ExcelExportUtil;
import cn.afterturn.easypoi.excel.ExcelImportUtil;
import cn.afterturn.easypoi.excel.entity.ExportParams;
import cn.afterturn.easypoi.excel.entity.ImportParams;
import cn.afterturn.easypoi.excel.entity.enmus.ExcelType;
import com.xxxx.server.pojo.Department;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.Joblevel;
import com.xxxx.server.pojo.Nation;
import com.xxxx.server.pojo.PoliticsStatus;
import com.xxxx.server.pojo.Position;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IDepartmentService;
import com.xxxx.server.service.IEmployeeService;
import com.xxxx.server.service.IJoblevelService;
import com.xxxx.server.service.INationService;
import com.xxxx.server.service.IPoliticsStatusService;
import com.xxxx.server.service.IPositionService;
import io.swagger.annotations.ApiOperation;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.URLEncoder;
import java.time.LocalDate;
import java.util.List;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletResponse;
import org.apache.poi.ss.formula.functions.Na;
import org.apache.poi.ss.usermodel.Workbook;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;
```

```java
/**
 * <p>
 * 前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/employee/basic")
public class EmployeeController {

  @Autowired
  private IEmployeeService employeeService;
  @Autowired
  private IPoliticsStatusService politicsStatusService;
  @Autowired
  IJoblevelService joblevelService;
  @Autowired
  private IPositionService positionService;
  @Autowired
  private INationService nationService;
  @Autowired
  private IDepartmentService departmentService;

  @ApiOperation(value = "导入员工数据")
  @PostMapping("/import")
  public RespBean importEmployee(MultipartFile file) {
    //准备导入参数
    ImportParams params = new ImportParams();
    //去掉标题行
    params.setTitleRows(1);
    //民族
    List<Nation> nations = nationService.list();
    //政治面貌
    List<PoliticsStatus> politicsStatuses = politicsStatusService.list();
    //部门
    List<Department> departments = departmentService.list();
    //职位
    List<Position> positions = positionService.list();
    //职称
    List<Joblevel> joblevels = joblevelService.list();
    try {
      //使用流形式导入数据
      List<Employee> list = ExcelImportUtil.importExcel(file.getInputStream(),
Employee.class, params);
      //使用循环导入多行
      list.forEach(employee -> {
        //民族id
        employee.setNationId(nations.get(nations.indexOf(new
Nation(employee.getNation().getName()))).getId());
        //政治面貌id
        employee.setPoliticId(politicsStatuses.get(politicsStatuses.indexOf(new
PoliticsStatus(employee.getPoliticsStatus().getName()))).getId());
        //部门id
        employee.setDepartmentId(departments.get(departments.indexOf(new
Department(employee.getDepartment().getName()))).getId());
```

```
            //职位id
            employee.setPosId(positions.get(positions.indexOf(new
Position(employee.getPosition().getName()))).getId());
            //职称id
            employee.setJobLevelId(joblevels.get(joblevels.indexOf(new
Joblevel(employee.getJoblevel().getName()))).getId());
        });
        //导入数据
        if (employeeService.saveBatch(list)) {
          return RespBean.success("导入成功!");
        }

      } catch (Exception e) {
        e.printStackTrace();
      }
      return RespBean.error("导入失败!");
    }
}
```

39. 邮件服务

- [ ] RabbitMQ安装和配置
- [ ] 邮箱开启SMTP权限服务
- [ ] 导入pom依赖包
- [ ] 配置application.yml文件
- [ ] 在启动类配置监听消息队列
- [ ] 消息发送者
- [ ] 消失接收者

39.1 RabbitMQ 安装及配置

- [ ] 本项目使用容器化的RabbitMQ安装方式，具体操作命令请阅读 九、容器化部署 。

https://blog.csdn.net/sepnineth/article/details/126235964

一、安装 erlang

```
# 安装 erlang
sudo apt-get install erlang-nox

# 查看relang语言版本，成功执行则说明relang安装成功
erl
```

二、添加公钥

```
wget -O- https://www.rabbitmq.com/rabbitmq-release-signing-key.asc | sudo apt-key add
-
```

三、更新软件包（可选）

```
sudo apt-get update
```

四、安装 RabbitMQ

```
# 安装成功自动启动
sudo apt-get install rabbitmq-server
```

五、查看 RabbitMQ 状态

```
# Active: active (running) 说明处于运行状态
systemctl status rabbitmq-server
# 或
service rabbitmq-server status

# --------- 常用命令 ---------

# 启动 | 停止 | 重启
sudo service rabbitmq-server start
sudo service rabbitmq-server stop
sudo service rabbitmq-server restart
```

六、安装 web 插件

```
# 启用插件
sudo rabbitmq-plugins enable rabbitmq_management

# 重启
sudo service rabbitmq-server restart

# 至此，http://localhost:15672 应该可以访问了，默认账户是 guest/guest
```

七、远程登录（web端）

```
# 查看用户
sudo rabbitmqctl list_users

# 添加用户
sudo rabbitmqctl add_user username password
```

```
# 分配管理员角色后，才能远程 web 访问（https://ip:15672）
# 确认服务器的安全组开放 15672 端口
sudo rabbitmqctl set_user_tags username administrator

$ sudo rabbitmqctl add_user lizongzai password
$ sudo rabbitmqctl set_user_tags lizongzai administrator

# 设置用户权限
格式：set_permissions [-p <vhostpath>] <user> <conf> <write> <read>
sudo rabbitmqctl set_permissions -p '/' ?lizongzai '.' '.' '.'
```

八、开通防火墙端口号

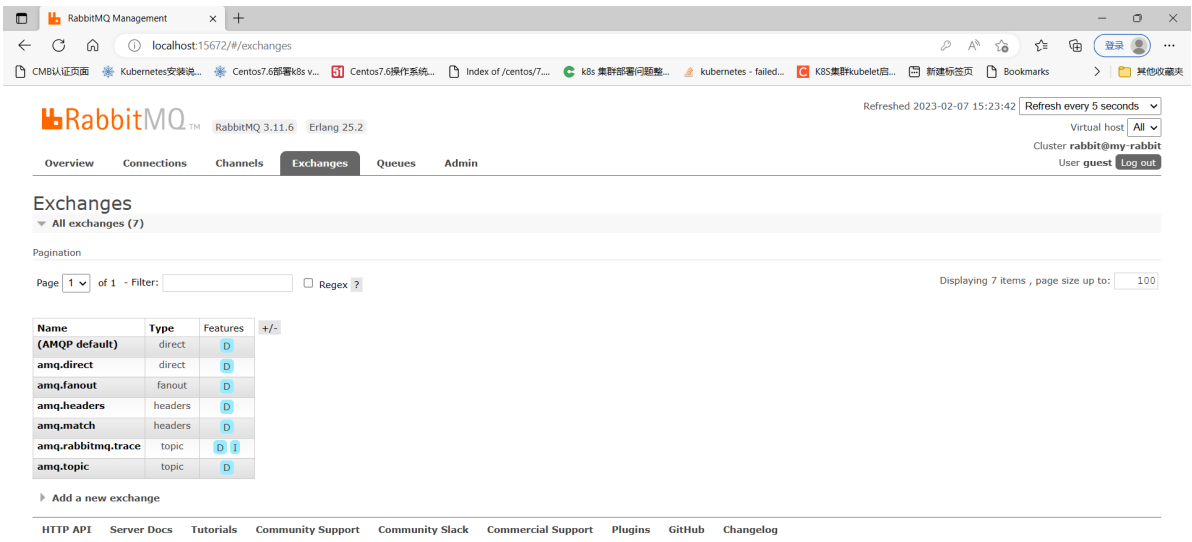
```
$ sudo ufw allow 15672
$ sudo ufw status
```

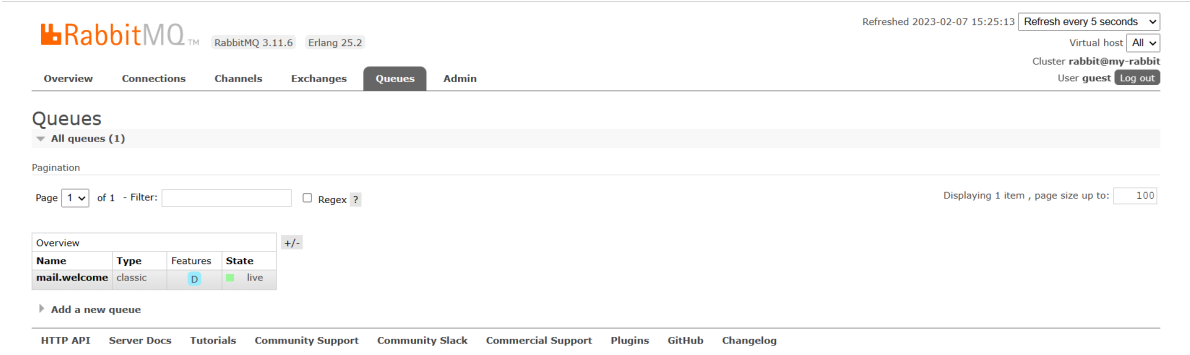
九、容器化部署(★)

https://blog.csdn.net/qq_45502336/article/details/118699251

```
# 需要注意的是-p 5673:5672 解释：-p 外网端口：docker的内部端口 ，你们可以改成自己的外网端口号，我这里映射的外网端口是5673那么程序连接端口就是用5673
docker run -d --hostname my-rabbit --name rabbitmq -p 15672:15672 -p 5672:5672 rabbitmq

# 通过docker ps -a查看部署的mq容器id，在通过 docker exec -it 容器id /bin/bash 进入容器内部在运行：rabbitmq-plugins enable rabbitmq_management
运行：rabbitmqctl status
```

http://localhost:15672/

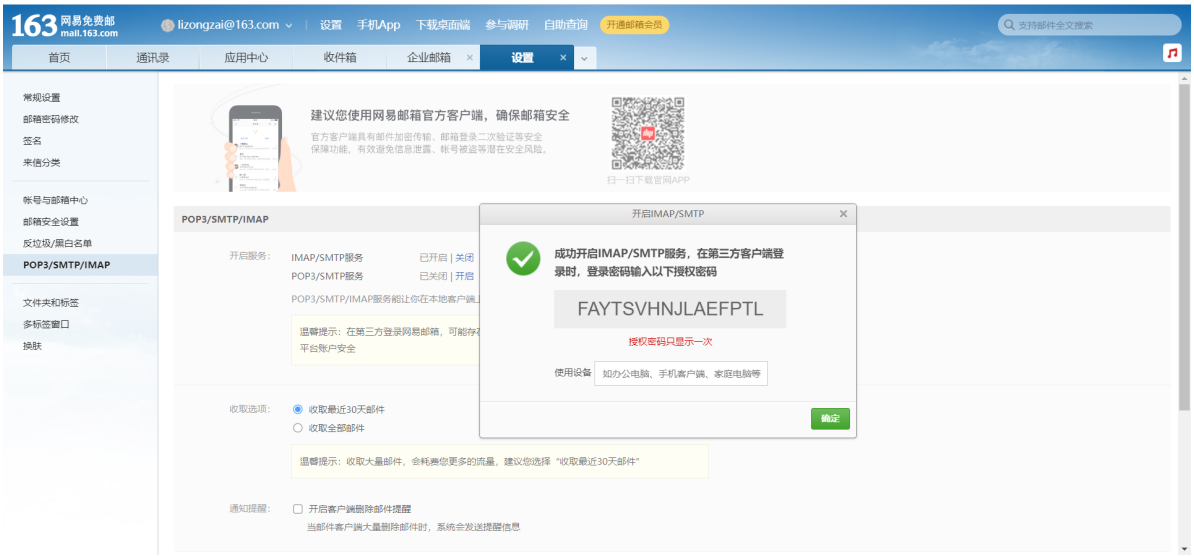




39.2 163邮箱开启SMTP权限(★)

https://blog.csdn.net/qq_39933045/article/details/126957074

授权码: `FAYTSVHNJLAEFPTL`



39.3. Maven依赖包

39.3.1 `pom.xml`

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <!--引入父工程依赖-->
  <parent>
    <artifactId>yeb</artifactId>
    <groupId>com.xxxx</groupId>
    <version>0.0.1-SNAPSHOT</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>
```

```
  <artifactId>yeb-mail</artifactId>
  <packaging>jar</packaging>

  <name>yeb-mail</name>
  <url>http://maven.apache.org</url>

  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <!--rabbitmq 依赖-->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-amqp</artifactId>
    </dependency>

    <!--mail 依赖-->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-mail</artifactId>
    </dependency>

    <!--thymeleaf 依赖-->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>

    <!--server 依赖-->
    <dependency>
      <groupId>com.xxxx</groupId>
      <artifactId>yeb-server</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <!--<scope>compile</scope>-->
    </dependency>
  </dependencies>
</project>
```

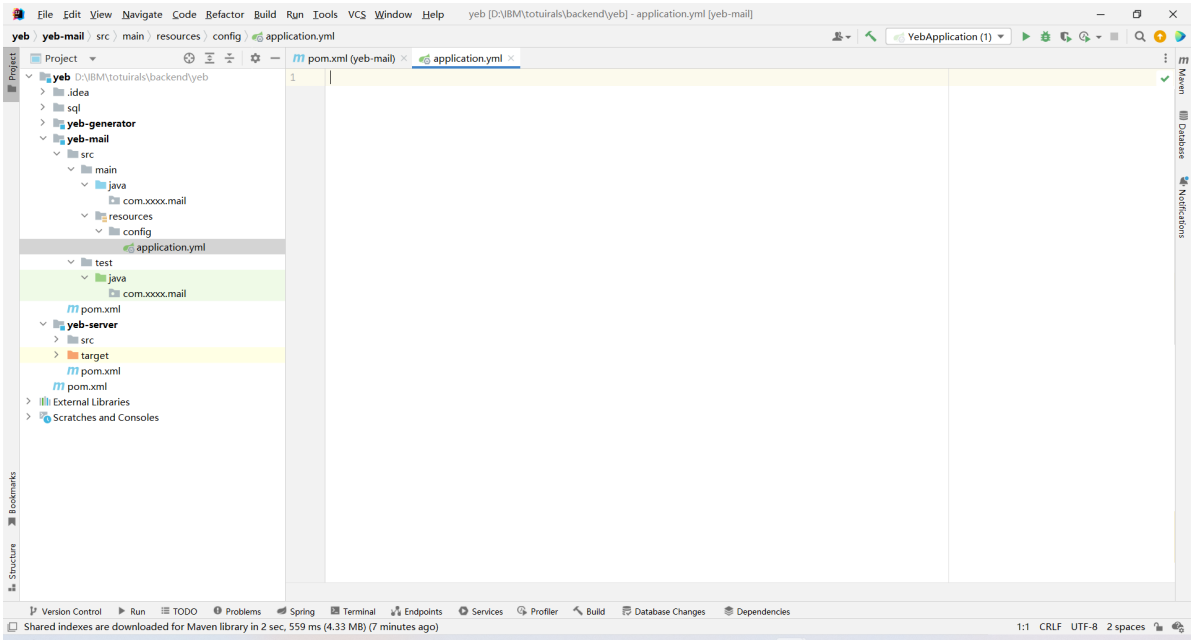
39.3.2 准备配置application.yml

```
server:
  port: 8082

spring:
  # 邮件配置
  mail:
    # 邮件服务器地址
    host: smtp.163.com
    # 协议
    protocol: smtp
    # 编码格式
    default-encoding: UTF-8
    # 授权码(在邮箱开通服务时,系统自动提供)
```

```
    password: FAYTSVHNJLAEFPTL
    # 发送者邮箱地址
    username: lizongzai@163.com
    # 端口号(不同邮箱端口号不同)
    port: 25

  # rabbitmq消息队列配置
  rabbitmq:
    # 用户名
    username: guest
    # 密码
    password: guest
    # 服务器地址
    host: localhost
    # 端口号
    port: 5672
    listener:
      simple:
        acknowledge-mode: manual
  redis:
    timeout: 10000ms
    host: localhost
    port: 6379
    database: 0 # 选择哪个库，默认0库
    lettuce:
      pool:
        max-active: 1024 # 最大连接数，默认 8
        max-wait: 10000ms # 最大连接阻塞等待时间，单位毫秒，默认 -1
        max-idle: 200 # 最大空闲连接，默认 8
        min-idle: 5
```

39.3.3 添加项目目录和配置

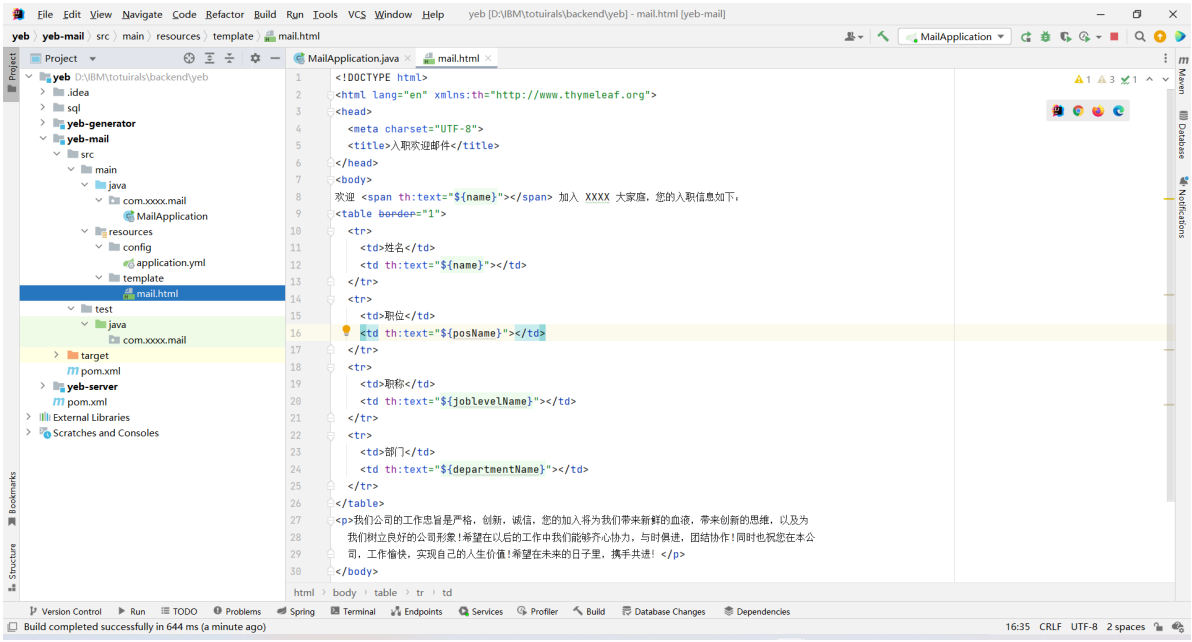
- [ ] src/main/java/com.xxxx/mail
- [ ] src/main/resources/config/application.yml
- [ ] src/main/resources/templates/mail.html
- [ ] src/mian/test/java/com/xxxx/mail



39.3.4 创建启动类MailApplication

```
package com.xxxx.mail;

import org.springframework.amqp.core.Queue;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
import org.springframework.context.annotation.Bean;

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})
public class MailApplication {

  public static void main(String[] args) {
    SpringApplication.run(MailApplication.class,args);
  }

  /**
   * @Description //TODO 监听消息队列
   * @Author lizongzai
   * @Date 2023/01/13 10:29
   * @return
   */
  @Bean
  public Queue queue() {
    return new Queue("mail.welcome");
  }
}
```

39.3.5 建立邮件模板(★)




```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>入职欢迎邮件</title>
</head>
<body>
欢迎 <span th:text="${name}"></span> 加入 XXXX 大家庭，您的入职信息如下：
<table border="1">
  <tr>
    <td>姓名</td>
    <td th:text="${name}"></td>
  </tr>
  <tr>
    <td>职位</td>
    <td th:text="${posName}"></td>
  </tr>
  <tr>
    <td>职称</td>
    <td th:text="${joblevelName}"></td>
  </tr>
  <tr>
    <td>部门</td>
    <td th:text="${departmentName}"></td>
  </tr>
</table>
<p>我们公司的工作忠旨是严格，创新，诚信，您的加入将为我们带来新鲜的血液，带来创新的思维，以及为
  我们树立良好的公司形象!希望在以后的工作中我们能够齐心协力，与时俱进，团结协作!同时也祝您在本公
  司，工作愉快，实现自己的人生价值!希望在未来的日子里，携手共进！</p>
</body>
</html>
```

39.3.6 配置yeb-server依赖包(★)

- [ ] 将RabbitMQ配置到yeb-server pom文件中

```
<!--rabbitmq 依赖-->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

`application.yml`

- [ ] 消息失败回调

```
publisher-returns: true
```

- [ ] 消息确认回调

```
publisher-confirm-type: correlated
```

```
server:
  port: 8081

spring:
  main:
    allow-circular-references: true
  mvc:
    pathmatch:
      matching-strategy: ANT_PATH_MATCHER
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: "jdbc:mysql://localhost:3306/yeb?useUnicode=true&characterEncoding=UTF-
8&serverTimezone=Asia/Shanghai"
    username: root
    password: password
    redis:
      timeout: 10000ms
      host: localhost
      port: 6379
      database: 0 # 选择哪个库，默认0库
      password: root
      lettuce:
        pool:
          max-active: 1024 # 最大连接数，默认 8
          max-wait: 10000ms # 最大连接阻塞等待时间，单位毫秒，默认 -1
          max-idle: 200 # 最大空闲连接，默认 8
          min-idle: 5
    # rabbitmq配置
    rabbitmq:
      # 用户名
      username: guest
```

```yaml
      # 密码
      password: guest
      # 服务器地址
      host: localhost
      # 端口
      port: 5672
      # 消息失败回调
      publisher-returns: true
      # 消息确认回调
      publisher-confirm-type: correlated
    hikari:
      # 连接池名
      pool-name: DateHikariCP
      # 最小空闲连接数
      minimum-idle: 5
      # 空闲连接存活最大时间，默认值为6000000
      idle-timeout: 180000
      # 最大连接数,默认10
      maximum-pool-size: 10
      # 从连接池返回的连接的自动提交
      auto-commit: true
      # 连接最大存活时间，0表示永久存活，默认1800000(30分钟)
      max-lifetime: 1800000
      # 连接超时时间, 默认30000(30秒)
      connection-timeout: 30000
      # 测试连接是否可用的查询语句
      connection-test-query: SELECT 1

# Mybatis-plus配置
mybatis-plus:
  # 配置Mapper映射文件
  mapper-locations: classpath*:/mapper/*Mapper.xml
  #配置mybatis数据返回类型别名
  type-aliases-package: com.ibm.server.pojo
  configuration:
    # 自动驼峰命名
    map-underscore-to-camel-case: false

# Mybatis SQL打印(方法接口所在的包，不是Mapper.xml所在的包)
logging:
  level:
    com.xxxx.server.mapper: debug

jwt:
  # Jwt存储的请求头
  tokenHeader: Authorization
  # Jwt加密秘钥
  secret: yeb-secret
  # Jwt 的超期限时间（60*60）*24
  expiration: 604800
  # Jwt负载中拿到开头
  tokenHead: Bearer
```

39.4 消息发送者

39.4.1 EmployeeController

```
package com.xxxx.server.controller;


import cn.afterturn.easypoi.excel.ExcelExportUtil;
import cn.afterturn.easypoi.excel.ExcelImportUtil;
import cn.afterturn.easypoi.excel.entity.ExportParams;
import cn.afterturn.easypoi.excel.entity.ImportParams;
import cn.afterturn.easypoi.excel.entity.enmus.ExcelType;
import com.xxxx.server.pojo.Department;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.Joblevel;
import com.xxxx.server.pojo.Nation;
import com.xxxx.server.pojo.PoliticsStatus;
import com.xxxx.server.pojo.Position;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IDepartmentService;
import com.xxxx.server.service.IEmployeeService;
import com.xxxx.server.service.IJoblevelService;
import com.xxxx.server.service.INationService;
import com.xxxx.server.service.IPoliticsStatusService;
import com.xxxx.server.service.IPositionService;
import io.swagger.annotations.ApiOperation;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.URLEncoder;
import java.time.LocalDate;
import java.util.List;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletResponse;
import org.apache.poi.ss.formula.functions.Na;
import org.apache.poi.ss.usermodel.Workbook;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

/**
 * <p>
 * 前端控制器
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/employee/basic")
```

```
public class EmployeeController {

  @Autowired
  private IEmployeeService employeeService;

  @ApiOperation(value = "添加员工")
  @PostMapping("/")
  public RespBean addEmp(@RequestBody Employee employee) {
    return employeeService.addEmp(employee);
  }
}
```

39.4.2 IEmployeeService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import java.time.LocalDate;
import java.util.List;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IEmployeeService extends IService<Employee> {
  /**
   * @Description //TODO 添加员工
   * @param employee
   * @return
   */
  RespBean addEmp(Employee employee);
}
```

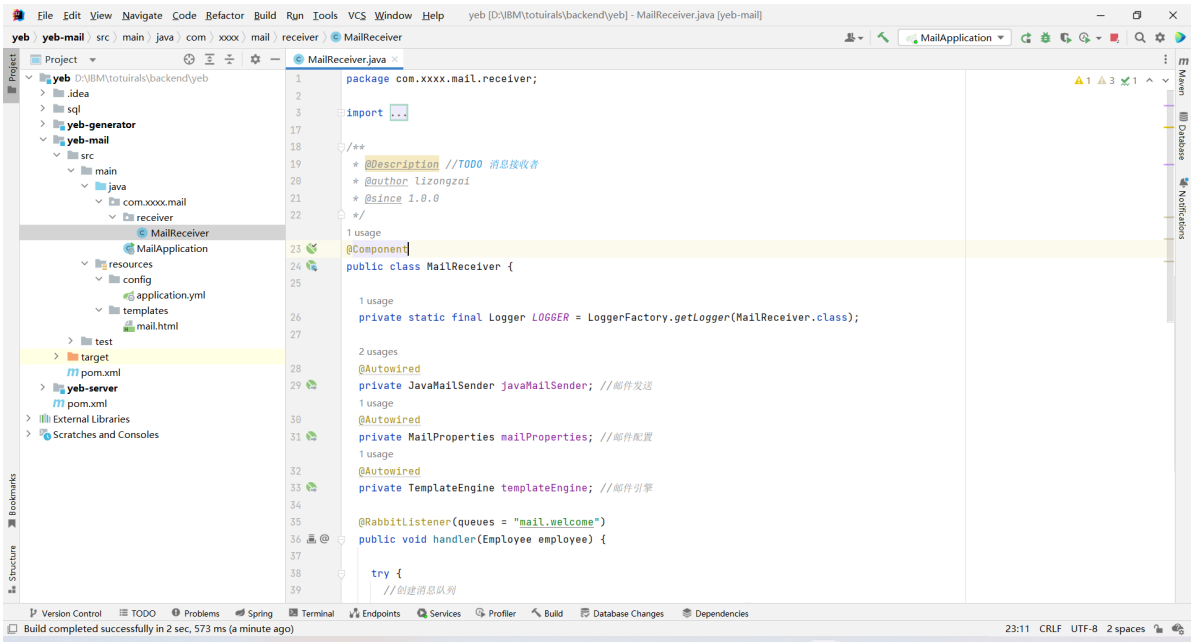
39.4.3 EmployeeServiceImpl(★)

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.mapper.EmployeeMapper;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
```

```java
import com.xxxx.server.service.IEmployeeService;
import java.text.DecimalFormat;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.Map;
import javax.annotation.Resource;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * <p>
 * 服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class EmployeeServiceImpl extends ServiceImpl<EmployeeMapper, Employee>
implements
    IEmployeeService {

  @Autowired
  @Resource
  private EmployeeMapper employeeMapper;
  @Autowired
  private RabbitTemplate rabbitTemplate;

  @Override
  public RespBean addEmp(Employee employee) {
    //处理合同期限，保留两位小数
    LocalDate beginContract = employee.getBeginContract();
    LocalDate endContract = employee.getEndContract();
    //计算合同的天数
    long days = beginContract.until(endContract, ChronoUnit.DAYS);
    //设置保留两位小数
    DecimalFormat decimalFormat = new DecimalFormat("##.00");
    //算计年数，保留两位小数点
    employee.setContractTerm(Double.parseDouble(decimalFormat.format(days/365.00)));
    //System.out.println("合同年数 = " + employee.getContractTerm());

    //若添加成功，则返回一条记录。表示添加成功
    if (1==employeeMapper.insert(employee)) {
      //通过员工id获取员工对象
      Employee emp = employeeMapper.getEmployee(employee.getId()).get(0);
      //发送邮件
      rabbitTemplate.convertAndSend("mail.welcome",emp);
      return RespBean.success("添加成功!");
    }
    return RespBean.error("添加失败!");
  }
}
```

39.5 消息接收者

39.5.1 MailReceiver(★★★)



参考连接 https://blog.csdn.net/qq3892997/article/details/85691807

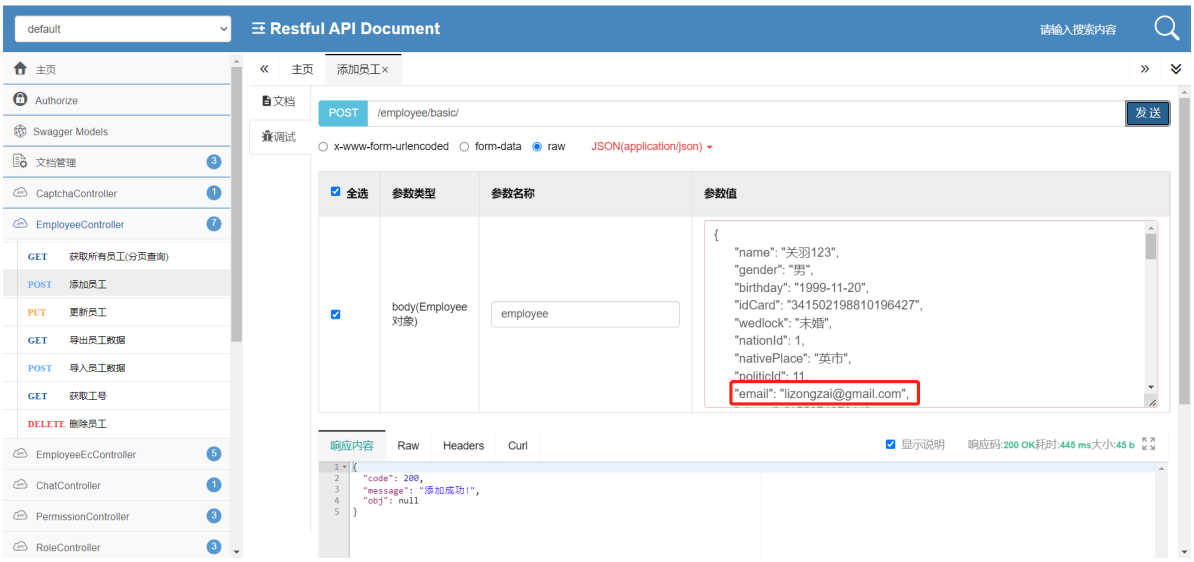
```
package com.xxxx.mail;


import com.rabbitmq.client.Channel;
import com.xxxx.server.config.constant.MailConstants;
import com.xxxx.server.pojo.Employee;
import java.util.Date;
import javax.mail.MessagingException;
import javax.mail.internet.MimeMessage;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.mail.MailProperties;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.messaging.Message;
import org.springframework.stereotype.Component;
import org.thymeleaf.TemplateEngine;
import org.thymeleaf.context.Context;

/**
 * 消息接收者
 *
 * @author lizongzai
 * @since 1.0.0
 */
@Component
public class MailReceiver {
```

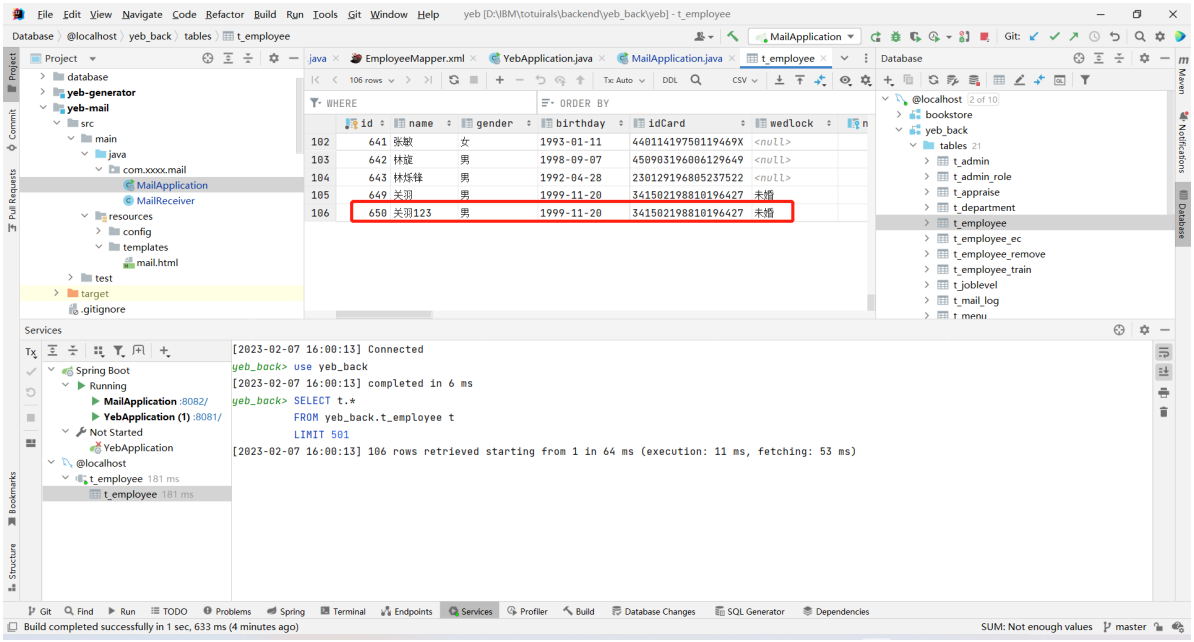
```
    private static final Logger LOGGER = LoggerFactory.getLogger(MailReceiver.class);
    //邮件发送
    @Autowired
    private JavaMailSender javaMailSender;
    //邮件配置
    @Autowired
    private MailProperties mailProperties;
    //邮件引擎
    @Autowired
    private TemplateEngine templateEngine;
    //Redis缓存数据库
    private RedisTemplate redisTemplate;

    //接收者监听
    @RabbitListener(queues = MailConstants.MAIL_QUEUE_NAME)
    public void handler(Employee employee) {
        //创建邮件消息队列
        MimeMessage message = javaMailSender.createMimeMessage();
        //设置utf-8或GBK编码，否则邮件会有乱码，true表示为multipart邮件
        //MimeMessageHelper helper = new MimeMessageHelper(message,true,"utf-8");
        MimeMessageHelper helper = new MimeMessageHelper(message);
        try {
            //发件人
            helper.setFrom(mailProperties.getUsername());
            //收件人
            helper.setTo(employee.getEmail());
            //主题
            helper.setSubject("入职欢迎邮件");
            //发送日期
            helper.setSentDate(new Date());
            //发送内容
            Context context = new Context();
            //员工名称
            context.setVariable("name", employee.getName());
            //员工职位
            context.setVariable("posName", employee.getPosition().getName());
            //员工职称
            context.setVariable("joblevelName", employee.getJoblevel().getName());
            //所属部门
            context.setVariable("departmentName", employee.getDepartment().getName());
            //通过templateEngine获取mail模板，并将邮件放入模板，最后生成mail
            String mail = templateEngine.process("mail", context);
            //确认邮件模板样式
            helper.setText(mail, true);
            //发送邮件
            javaMailSender.send(message);
        } catch (Exception e) {
            LOGGER.error("MailReceiver + 邮件发送失败========{}", e.getMessage());
        }
    }
}
```

39.5.2 消息队列常量

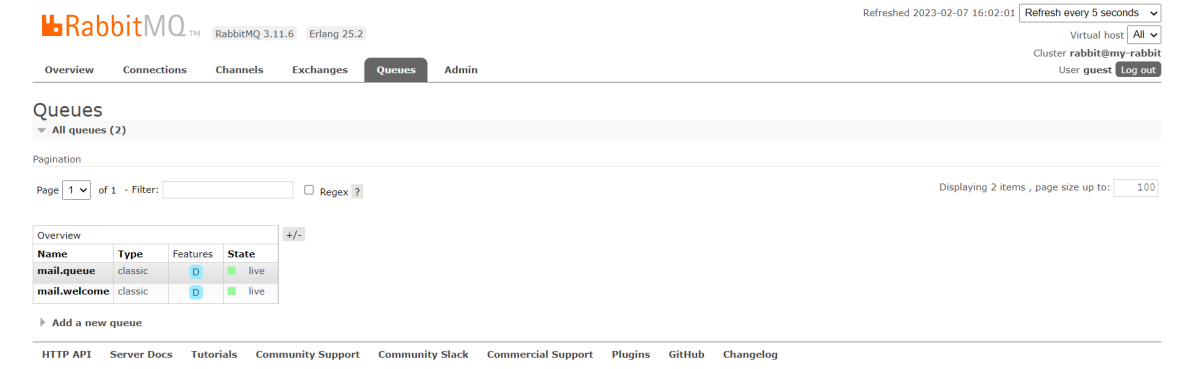
```
package com.xxxx.server.config.constant;

/**
 * 消息状态
 *
 * @Author lizongzai
 * @Since 1.0.0
 */
public class MailConstants {

  //消息投递中
  public static final Integer DELIVERING = 0;
  //消息投递成功
  public static final Integer SUCCESS = 1;
  //消息投递失败
  public static final Integer FAILURE = 2;
  //最大重试次数
  public static final Integer MAX_TRY_COUNT = 3;
  //消息超时时间
  public static final Integer MSG_TIMEOUT = 1;
  //队列
  public static final String MAIL_QUEUE_NAME = "mail.queue";
  //交换机
  public static final String MAIL_EXCHANGE_NAME = "mail.exchange";
  //路由键
  public static final String MAIL_ROUTING_KEY_NAME = "mail.routing.key";

}
```

39.5.3 Restful Api Test(★)




File Edit View Navigate Code Refactor Build Run Tools Git Window Help
yeb [D:\IBM\totuirals\backend\yeb_back\yeb] - t_employee
Database @localhost yeb_back tables t_employee
MailApplication
Project
database
yeb-generator
yeb-mail
src
main
java
com.xxxx.mail
MailApplication
MailReceiver
resources
config
templates
mail.html
test
target
.gitignore
java EmployeeMapper.xml YebApplication.java MailApplication.java t_employee
106 rows
WHERE
ORDER BY
id name gender birthday idCard wedlock
102 641 张敏 女 1993-01-11 440114197501194697X <null>
103 642 林旋 男 1998-09-07 450903196806129649 <null>
104 643 林烁锋 男 1992-04-28 230129196805237522 <null>
105 649 关羽 男 1999-11-20 341502198810196427 未婚
106 650 关羽123 男 1999-11-20 341502198810196427 未婚
Database
@localhost 2 of 10
bookstore
yeb_back
tables 21
t_admin
t_admin_role
t_appraise
t_department
t_employee
t_employee_ec
t_employee_remove
t_employee_train
t_joblevel
t_mail_log
t_menu
Services
Spring Boot
Running
MailApplication :8082/
YebApplication (1) :8081/
Not Started
YebApplication
@localhost
t_employee 181 ms
t_employee 181 ms
[2023-02-07 16:00:13] Connected
yeb_back> use yeb_back
[2023-02-07 16:00:13] completed in 6 ms
yeb_back> SELECT t.*
FROM yeb_back.t_employee t
LIMIT 501
[2023-02-07 16:00:13] 106 rows retrieved starting from 1 in 64 ms (execution: 11 ms, fetching: 53 ms)
Git Find Run TODO Problems Spring Terminal Endpoints Services Profiler Build Database Changes SQL Generator Dependencies
Build completed successfully in 1 sec, 633 ms (4 minutes ago)
SUM: Not enough values
master


Gmail
搜索邮件
写邮件
收件箱 408
已加星标
已延后
重要邮件
已发邮件
草稿 18
类别
社交 2
动态 204
论坛 1
推广 452
显示更多标签
标签
[Imap]/Sent
[Imap]/Trash
旅行相关
私人邮件
第 2 个会话，共 3,428 个
入职欢迎邮件
收件箱
lizongzai@163.com
发送至 我
15:45 (15分钟前)
欢迎 关羽 加入 XXXX 大家庭，您的入职信息如下：
姓名 关羽
职位 运维工程师
职称 初级工程师
部门 总办
我们公司的工作思旨是严格，创新，诚信，您的加入将为我们带来新鲜的血液，带来创新的思维，以及为 我们树立良好的公司形象!希望在以后的工作中我们能够齐心协力，与时俱进，团结协作!同时也祝您在本公 司，工作愉快，实现自己的人生价值!希望在未来的日子里，携手共进!
回复
转发

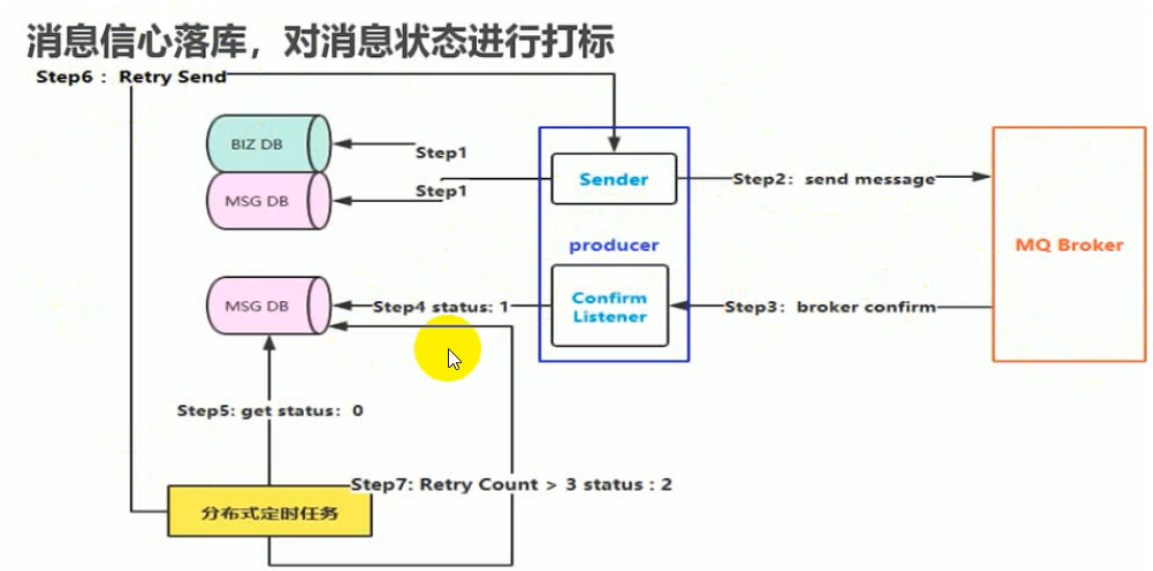

RabbitMQ
RabbitMQ 3.11.6 Erlang 25.2
Refreshed 2023-02-07 16:02:01
Refresh every 5 seconds
Virtual host All
Cluster rabbit@my-rabbit
User guest Log out
Overview Connections Channels Exchanges Queues Admin
Queues
All queues (2)
Pagination
Page 1 of 1 - Filter: Regex ?
Displaying 2 items , page size up to: 100
Overview +/-
Name Type Features State
mail.queue classic D live
mail.welcome classic D live
Add a new queue
HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins GitHub Changelog

39.6 可靠投递方案(★★★★★)

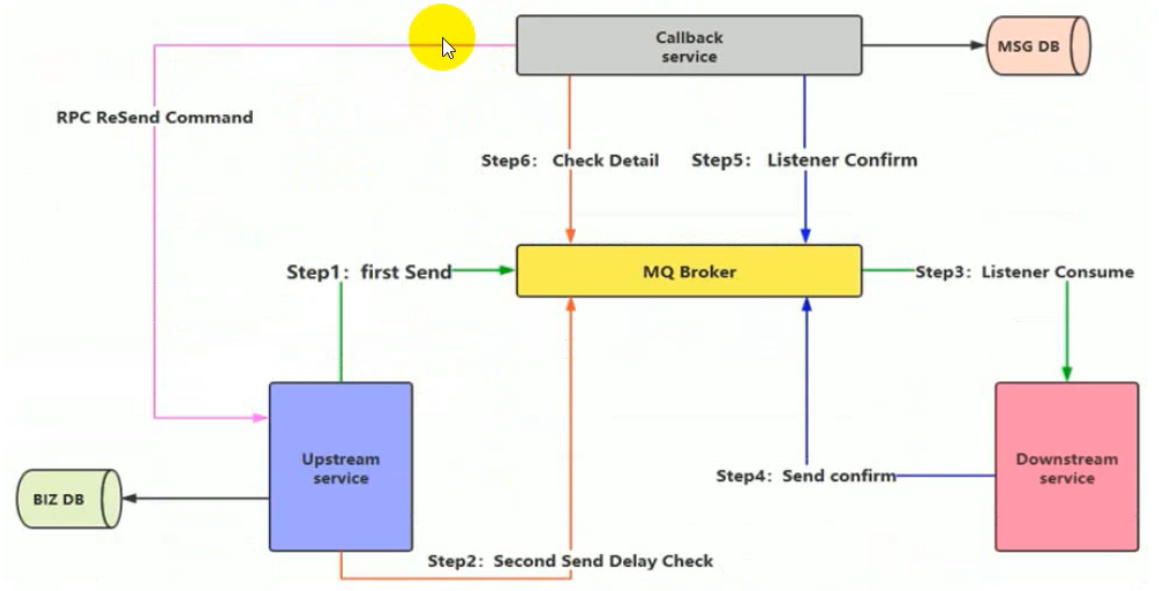
39.6.1 投递流程图

方案:

1. 消息落库，对消息状态进行打标



2. 消息延迟投递，做二次确认，回调检查



项目实现流程:

1.发送消息时，将当前消息存入数据库，投递状态设置为`消息投递中`。

2.开启消息确认回调机制。确认成功，更新投递状态为`消息投递成功`。

3.开启定时任务，重新投递失败的信息。重试超过三次，设置更新投递状态为`投递失败`。

39.7 消息发送者(优化)

39.7.1 t_mail_log 表

```
CREATE TABLE `t_mail_log` (
  `msgId` varchar(64) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL COMMENT '消息id',
  `eid` int DEFAULT NULL COMMENT '接收员工id',
  `status` int DEFAULT NULL COMMENT '状态（0:消息投递中 1:投递成功 2:投递失败）',
  `routeKey` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL COMMENT '路由键',
  `exchange` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL COMMENT '交换机',
  `count` int DEFAULT NULL COMMENT '重试次数',
  `tryTime` datetime DEFAULT NULL COMMENT '重试时间',
  `createTime` datetime DEFAULT NULL COMMENT '创建时间',
  `updateTime` datetime DEFAULT NULL COMMENT '更新时间',
  UNIQUE KEY `msgId` (`msgId`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin ROW_FORMAT=DYNAMIC
```

39.7.2 MailConstants.java(★)

```
package com.xxxx.server.constant;

/**
 * 消失状态
 *
 * @Author lizongzai
 * @Since 1.0.0
 * @Date 2023/01/13 15:05
 */
public class MailConstants {

  //消息投递中
  public static final Integer DELIVERING = 0;
  //消息投递成功
  public static final Integer SUCCESS = 1;
  //消息投递失败
  public static final Integer FAILURE = 2;
  //最大重试次数
  public static final Integer MAX_TRY_COUNT = 3;
  //消息超时时间
  public static final Integer MSG_TIMEOUT = 1;
  //队列
  public static final String MAIL_QUEUE_NAME = "mail.queue";
  //交换机
  public static final String MAIL_EXCHANGE_NAME = "mail.exchange";
  //路由键
  public static final String MAIL_ROUTING_KEY_NAME = "mail.routing.key";

}
```

93.7.3 EmployeeServiceImpl(★★★)

```
package com.xxxx.server.service.impl;

import com.xxxx.server.mapper.MailLogMapper;
import com.xxxx.server.constant.MailConstants;
import java.time.LocalDateTime;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.mapper.EmployeeMapper;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.MailLog;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IEmployeeService;
import java.text.DecimalFormat;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import javax.annotation.Resource;
import org.springframework.amqp.rabbit.connection.CorrelationData;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * <p>
 * 服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class EmployeeServiceImpl extends ServiceImpl<EmployeeMapper, Employee>
implements
    IEmployeeService {

  @Autowired
  @Resource
  private EmployeeMapper employeeMapper;
  @Autowired
  private RabbitTemplate rabbitTemplate;
  @Autowired
  @Resource
  private MailLogMapper mailLogMapper;

  /**
   * @param currentPage
   * @param pageSize
   * @param employee
   * @param beginDateScope
   * @return
```

```
     * @Description //TODO 获取所有员工(分页查询)
     */
    @Override
    public RespPageBean getAllEmployeeByPage(Integer currentPage, Integer pageSize,
Employee employee,
            LocalDate beginDateScope) {
        //开启分页
        Page<Employee> employeePage = new Page<>();
        IPage<Employee> employeeByPage = employeeMapper.getAllEmployeeByPage(employeePage,
employee,
                beginDateScope);
        RespPageBean respPageBean = new RespPageBean(employeeByPage.getTotal(),
                employeeByPage.getRecords());
        return respPageBean;
    }

    /**
     * @return
     * @Description //TODO 获取工号
     * @Author lizongzai
     * @Date 2023/01/11 13:06
     */
    @Override
    public RespBean getMaxWorkId() {
        //获取员工Key和value
        List<Map<String, Object>> maps = employeeMapper.selectMaps(
                new QueryWrapper<Employee>().select("max(workId)"));
        //System.out.println("maps = " + maps);
        //maps = [{max(workId)=00000100}]
        //只有一个最大值，那么get(0).get("max(workId")即可获取它的值
        //工号强制转换整型，然后再加1
        //maps.get(0).get("max(workId)"是一个Object对象，先将其转String
        //%8d，表示不足长度时，自动左补位，补位数为0
        return RespBean.success(null,
                String.format("%8d",
Integer.parseInt(maps.get(0).get("max(workId)").toString()) + 1));
    }

    /**
     * @param employee
     * @return
     * @Description //TODO 添加员工
     */
    @Override
    public RespBean addEmp(Employee employee) {
        //处理合同期限，保留两位小数
        LocalDate beginContract = employee.getBeginContract();
        LocalDate endContract = employee.getEndContract();
        long days = beginContract.until(endContract, ChronoUnit.DAYS); //计算合同的天数
        DecimalFormat decimalFormat = new DecimalFormat("##.00"); //设置保留两位小数
        employee.setContractTerm(
                Double.parseDouble(decimalFormat.format(days / 365.00))); //算计年数，保留两位小数
点
        //System.out.println("合同年数 = " + employee.getContractTerm());
        //若添加成功，则返回一条记录。表示添加成功
        if (1 == employeeMapper.insert(employee)) {
            //通过员工id获取员工对象
            Employee emp = employeeMapper.getEmployee(employee.getId()).get(0);
```

```
        //数据库记录发送消息
        String msgId = UUID.randomUUID().toString();
        MailLog mailLog = new MailLog();
        mailLog.setMsgId(msgId); //消息id
        mailLog.setEid(employee.getId()); //员工id
        mailLog.setStatus(0); //投递状态 （0:消息投递中 1:投递成功 2:投递失败）
        mailLog.setRouteKey(MailConstants.MAIL_ROUTING_KEY_NAME); //路由键
        mailLog.setExchange(MailConstants.MAIL_EXCHANGE_NAME); //交换机
        mailLog.setCount(0); //重试次数
        mailLog.setTryTime(LocalDateTime.now().plusMinutes(MailConstants.MSG_TIMEOUT));
//重试时间+1分钟
        mailLog.setCreateTime(LocalDateTime.now()); //创建时间
        mailLog.setUpdateTime(LocalDateTime.now()); //更新时间
        //消息落库
        mailLogMapper.insert(mailLog);
        //发送邮件
        rabbitTemplate.convertAndSend(MailConstants.MAIL_QUEUE_NAME,
            MailConstants.MAIL_ROUTING_KEY_NAME, emp, new CorrelationData(msgId));
        return RespBean.success("添加成功!");
      }
      return RespBean.error("添加失败!");
    }

    /**
     * @param id
     * @return
     * @Description //TODO 导出员工数据
     */
    @Override
    public List<Employee> getEmployee(Integer id) {
      return employeeMapper.getEmployee(id);
    }

}
```

39.8 消息接收者(优化)

39.9.1 MailApplication

> ☐ 监听消息队列,使用常量的方式
>
> ```
> MailConstants.MAIL_QUEUE_NAME
> ```

```
package com.xxxx.mail;

import com.xxxx.server.pojo.MailConstants;
import org.springframework.amqp.core.Queue;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
import org.springframework.context.annotation.Bean;
```

```
@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})
public class MailApplication {

  public static void main(String[] args) {
    SpringApplication.run(MailApplication.class,args);
  }

  /**
   * @Description //TODO 监听消息队列
   * @Author lizongzai
   * @Date 2023/01/13 10:29
   * @return
   */
  @Bean
  public Queue queue() {
    return new Queue(MailConstants.MAIL_QUEUE_NAME);
  }

}
```

39.8.2 MailReceiver

> @RabbitListener(queues = MailConstants.MAIL_QUEUE_NAME)

```
package com.xxxx.mail.receiver;

import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.MailConstants;
import java.io.File;
import java.io.UnsupportedEncodingException;
import java.util.Date;
import javax.mail.MessagingException;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeUtility;
import org.slf4j.LoggerFactory;
import org.slf4j.Logger;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.mail.MailProperties;
import org.springframework.core.io.FileSystemResource;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Component;
import org.thymeleaf.TemplateEngine;
import org.thymeleaf.context.Context;

/**
 * @Description //TODO 消息接收者
 * @author lizongzai
 * @since 1.0.0
 */
@Component
public class MailReceiver {
```

```
    private static final Logger LOGGER = LoggerFactory.getLogger(MailReceiver.class);
    @Autowired
    private JavaMailSender javaMailSender; //邮件发送
    @Autowired
    private MailProperties mailProperties; //邮件配置
    @Autowired
    private TemplateEngine templateEngine; //邮件引擎

    @RabbitListener(queues = MailConstants.MAIL_QUEUE_NAME)
    public void handler(Employee employee) {
        try {
            //创建消息队列
            MimeMessage mimeMessage = javaMailSender.createMimeMessage();
            //      mimeMessage.setHost("服务器名");
            //      mimeMessage.setPort(3306);
            //      mimeMessage.setUsername("用户名");
            //      mimeMessage.setPassword("密码");
            //设置utf-8或GBK编码，否则邮件会有乱码，true表示为multipart邮件
            MimeMessageHelper helper = new MimeMessageHelper(mimeMessage,true,"utf-8");
            //发件人
            helper.setFrom(mailProperties.getUsername());
            //收件人
            helper.setTo(employee.getEmail());
            //主题
            helper.setSubject("入职欢迎邮件");
            //发送日期
            helper.setSentDate(new Date());
            //邮件内容
            Context context = new Context();
            //员工名称
            context.setVariable("name",employee.getName());
            //员工职位
            context.setVariable("posName",employee.getPosition().getName());
            //员工职称
            context.setVariable("joblevelName",employee.getJoblevel().getName());
            //所属部门
            context.setVariable("departmentName",employee.getDepartment().getName());
            //通过templateEngine获取mail模板，并将邮件放入模板，最后生成mail
            String mail = templateEngine.process("mail", context);
            //设置邮件内容，注意加参数true，表示启用html格式
            helper.setText(mail,true);
            //发送带有附件的可以省略...参数一：读取word文档，参数二：
            //      try {
            //        helper.addAttachment(MimeUtility.encodeWord("Word文件名"),
            //            new FileSystemResource(new File("文件地址")));
            //      } catch (UnsupportedEncodingException e) {
            //        e.printStackTrace();
            //      }
            //第一个参数附件名，第二个参数附件
            //发送邮件
            javaMailSender.send(mimeMessage);

        } catch (MessagingException e) {
            LOGGER.error("邮件发送失败=============={}",e.getMessage());
        }
    }
}
```

39.9 RabbitMQ配置

39.9.1 RabbitMQConfig(★★★)

```
package com.xxxx.server.config.rabbitMQ;

import com.baomidou.mybatisplus.core.conditions.update.UpdateWrapper;
import com.xxxx.server.constant.MailConstants;
import com.xxxx.server.pojo.MailLog;
import com.xxxx.server.service.IMailLogService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.amqp.core.Binding;
import org.springframework.amqp.core.BindingBuilder;
import org.springframework.amqp.core.DirectExchange;
import org.springframework.amqp.core.Queue;
import org.springframework.amqp.rabbit.connection.CachingConnectionFactory;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * @Description //TODO RabbitMQ配置
 * 1.准备消息队列和交换机，并通过路由键将消息队列与交换机进行绑定
 * 2.注入缓存连接工厂，并将缓存工厂放入RabbitTemplate模板
 * 3.消息确认回调，确认消息是否达到broker
 * 4.消息失败回调，比如router不到Queue队列时回调
 * @Date 2023/01/13 16:42
 * @Author lizongzai
 * @Since 1.0.0
 */
@Configuration
public class RabbitMQConfig {

  private static final Logger LOGGER = LoggerFactory.getLogger(RabbitMQConfig.class);
  @Autowired
  private CachingConnectionFactory cachingConnectionFactory;
  @Autowired
  private IMailLogService mailLogService;

  @Bean
  public RabbitTemplate rabbitTemplate() {

    //开启confirm模式
    cachingConnectionFactory.setPublisherConfirms(true);

    //将缓存连接工厂注入RabbitMQ模板
    RabbitTemplate rabbitTemplate = new RabbitTemplate(cachingConnectionFactory);

    /**
     * 消息确认回调，确认消息是否达到broker
     * data:消息唯一标识
     * ack:确认结果
```

```
     * cause:失败原因
     * @return
     */
    rabbitTemplate.setConfirmCallback((data, ack, cause) -> {
      String msgId = data.getId();
      System.out.println("RabbitMQConfig: msgId = " + msgId);
      if (ack) {
        LOGGER.info("{}==========>消息发送成功", msgId);
        mailLogService.update(new UpdateWrapper<MailLog>().set("status",
1).eq("msgId", msgId));
      } else {
        LOGGER.info("{}==========>消息发送失败", msgId);
      }
    });

    /**
     * 消息失败回调，比如router不到Queue队列时回调
     * msg:消息主题
     * repCode:响应码
     * repText:响应描述
     * exchange:交换机
     * routingKey:路由键
     */
    rabbitTemplate.setReturnCallback((msg, repCode, repText, exchange, routingKey) ->
{
      LOGGER.error("{}==================消息发送到Queue队列时失败", msg.getBody());
    });

    return rabbitTemplate;
  }

  /**
   * 消息队列
   *
   * @return
   */
  @Bean
  public Queue queue() {
    return new Queue(MailConstants.MAIL_QUEUE_NAME, true, false, false);
  }

  /**
   * 交换机
   *
   * @return
   */
  @Bean
  public DirectExchange directExchange() {
    return new DirectExchange(MailConstants.MAIL_EXCHANGE_NAME, true, false);
  }

  /**
   * 使用路由键将消息队列与交换机进行绑定
   *
   * @return
   */
  @Bean
  public Binding binding() {
```

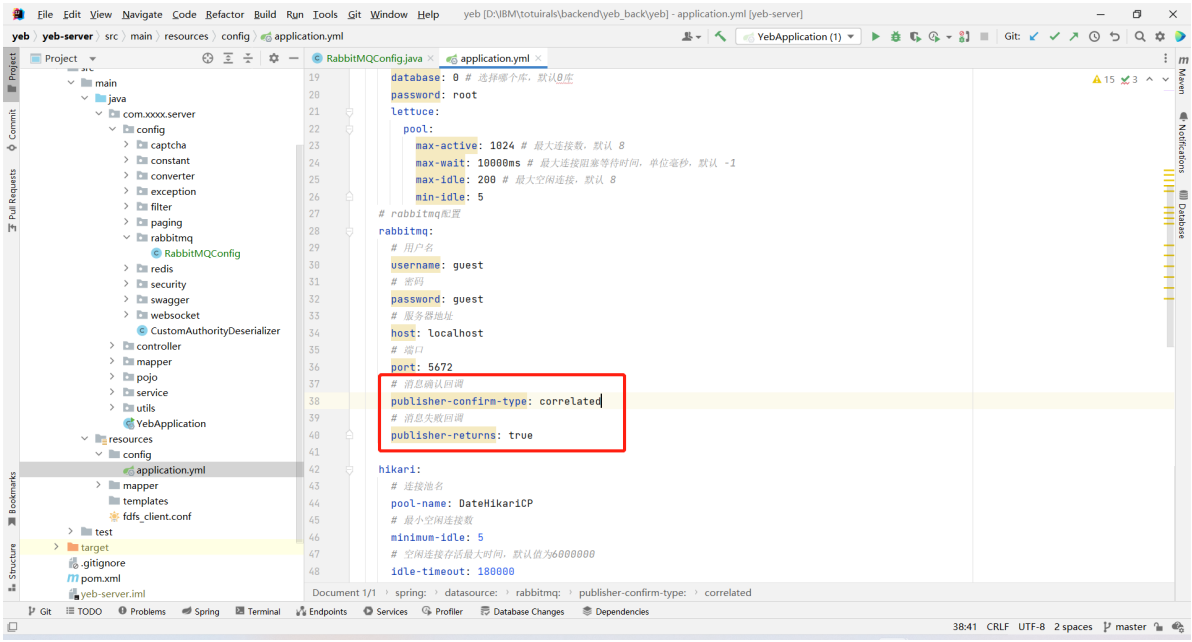
```
    return
BindingBuilder.bind(queue()).to(directExchange()).with(MailConstants.MAIL_ROUTING_KEY_
NAME);
  }

}
```

39.9.2 配置application.yml

```
# 消息确认回调
publisher-confirm-type: correlated
# 消息失败回调
publisher-returns: true
```

```
server:
  port: 8081

spring:
  main:
    allow-circular-references: true
  mvc:
    pathmatch:
      matching-strategy: ANT_PATH_MATCHER
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: "jdbc:mysql://localhost:3306/yeb?useUnicode=true&characterEncoding=UTF-
8&serverTimezone=Asia/Shanghai"
    username: root
    password: password
    redis:
      timeout: 10000ms
      host: localhost
      port: 6379
      database: 0 # 选择哪个库，默认0库
      password: root
      lettuce:
        pool:
          max-active: 1024 # 最大连接数，默认 8
          max-wait: 10000ms # 最大连接阻塞等待时间，单位毫秒，默认 -1
          max-idle: 200 # 最大空闲连接，默认 8
          min-idle: 5
    # rabbitmq配置
    rabbitmq:
      # 用户名
      username: guest
      # 密码
      password: guest
      # 服务器地址
      host: localhost
      # 端口
      port: 5672
      # 消息确认回调
      publisher-confirm-type: correlated
      # 消息失败回调
```

```yaml
      publisher-returns: true

    hikari:
      # 连接池名
      pool-name: DateHikariCP
      # 最小空闲连接数
      minimum-idle: 5
      # 空闲连接存活最大时间，默认值为6000000
      idle-timeout: 180000
      # 最大连接数,默认10
      maximum-pool-size: 10
      # 从连接池返回的连接的自动提交
      auto-commit: true
      # 连接最大存活时间，0表示永久存活，默认1800000(30分钟)
      max-lifetime: 1800000
      # 连接超时时间，默认30000(30秒)
      connection-timeout: 30000
      # 测试连接是否可用的查询语句
      connection-test-query: SELECT 1

# Mybatis-plus配置
mybatis-plus:
  # 配置Mapper映射文件
  mapper-locations: classpath*:/mapper/*Mapper.xml
  #配置mybatis数据返回类型别名
  type-aliases-package: com.ibm.server.pojo
  configuration:
    # 自动驼峰命名
    map-underscore-to-camel-case: false

# Mybatis SQL打印(方法接口所在的包，不是Mapper.xml所在的包)
logging:
  level:
    com.xxxx.server.mapper: debug

jwt:
  # Jwt存储的请求头
  tokenHeader: Authorization
  # Jwt加密秘钥
  secret: yeb-secret
  # Jwt 的超期限时间（60*60）*24
  expiration: 604800
  # Jwt负载中拿到开头
  tokenHead: Bearer
```



39.10 邮件发送定时任

39.10.1 MailTask(★★★)

```
package com.xxxx.server.task;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.conditions.update.UpdateWrapper;
import com.xxxx.server.constant.MailConstants;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.MailLog;
import com.xxxx.server.service.IEmployeeService;
import com.xxxx.server.service.IMailLogService;
import java.time.LocalDateTime;
import java.util.List;
import org.springframework.amqp.rabbit.connection.CorrelationData;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

/**
 * @Description //TODO 邮件发送定时任务
 * 1.先查询状态为投递失败"0",并且重试时间小于当前时间的信息
 * 2.若重试次数超过3次,则更新状态为投递失败”2“并且不在更新
 * 3.重试3次失败之后,更新基本信息(重试次数、更新时间、重试时间)
 * 4.通过员工id获取员工对象 5.发送消息
 * @Author lizongzai
 * @Date 2023/01/13 17:05
 * @Since 1.0.0
 */
@Component
public class MailTask {

  @Autowired
  private IMailLogService mailLogService;
```

```
  @Autowired
  private IEmployeeService employeeService;
  @Autowired
  private RabbitTemplate rabbitTemplate;

  /**
   * 邮件发送定时任务 10秒执行一次
   */
  @Scheduled(cron = "0/10 * * * * ?")
  public void mailTask() {
    //查询状态为投递失败"0",并且重试时间小于当前时间的信息
    List<MailLog> list = mailLogService.list(
        new QueryWrapper<MailLog>().eq("status", 0).lt("tryTime",
LocalDateTime.now()));

    list.forEach(mailLog -> {

      //若重试次数超过3次,则更新状态为投递失败并且不在更新
      if (3 <= mailLog.getCount()) {
        mailLogService.update(
            new UpdateWrapper<MailLog>().set("status", 2).eq("msgId",
mailLog.getMsgId()));
      }

      //重试3次失败之后,更新基本信息(重试次数、更新时间、重试时间)
      mailLogService.update(new UpdateWrapper<MailLog>()
          .set("count", mailLog.getCount() + 1) //重试次数
          .set("updateTime", LocalDateTime.now())   //更新时间
          .set("tryTime", LocalDateTime.now().plusMinutes(MailConstants.MSG_TIMEOUT))
//重试时间
          .eq("msgId", mailLog.getMsgId())); //消息id

      //通过员工id获取员工对象
      Employee employee = employeeService.getEmployee(mailLog.getEid()).get(0);
      //      System.out.println("MailTask Employee = " + employee);
      //      System.out.println("MailTask msgId = " + mailLog.getMsgId());

      //发送消息
      rabbitTemplate.convertAndSend(MailConstants.MAIL_EXCHANGE_NAME,
          MailConstants.MAIL_ROUTING_KEY_NAME, employee, new
CorrelationData(mailLog.getMsgId()));
    });
  }
}
```

39.11 MailApplication启动类

```
@Bean
  public Queue queue() {
    return new Queue(MailConstants.MAIL_QUEUE_NAME);
  }
```

```java
package com.xxxx.mail;

import com.xxxx.server.constant.MailConstants;
import org.springframework.amqp.core.Queue;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
import org.springframework.context.annotation.Bean;

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})
public class MailApplication {

  public static void main(String[] args) {
    SpringApplication.run(MailApplication.class,args);
  }

  /**
   * @Description //TODO 监听消息队列
   * @Author lizongzai
   * @Date 2023/01/13 10:29
   * @return
   */
  @Bean
  public Queue queue() {
    return new Queue(MailConstants.MAIL_QUEUE_NAME);
  }
}
```

```java
@EnableScheduling
```

```java
package com.xxxx.server;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;

/**
 * 启动类
 * @author lizongzai
 * @since 1.0.0
 */
@SpringBootApplication
@MapperScan("com.xxxx.server.mapper")
@EnableScheduling
public class YebApplication {
  public static void main(String[] args) {
    SpringApplication.run(YebApplication.class,args);
  }

}
```

39.12.yeb-mail配置

```yaml
server:
  port: 8082

spring:
  # 邮件配置
  mail:
    # 邮件服务器地址
    host: smtp.163.com
    # 协议
    protocol: smtp
    # 编码格式
    default-encoding: UTF-8
    # 授权码(在邮箱开通服务时,系统自动提供)
    password: FAYTSVHNJLAEFPTL
    # 发送者邮箱地址
    username: lizongzai@163.com
    # 端口号(不同邮箱端口号不同)
    port: 25

  # rabbitmq消息队列配置
  rabbitmq:
    # 用户名
    username: guest
    # 密码
    password: guest
    # 服务器地址
    host: localhost
    # 端口号
    port: 5672
    listener:
      simple:
        acknowledge-mode: manual
  redis:
    timeout: 10000ms
    host: localhost
    port: 6379
    database: 0 # 选择哪个库，默认0库
    lettuce:
      pool:
        max-active: 1024 # 最大连接数，默认 8
        max-wait: 10000ms # 最大连接阻塞等待时间，单位毫秒，默认 -1
        max-idle: 200 # 最大空闲连接，默认 8
        min-idle: 5
```

40.工资账套

40.1 Salarys实体

> ☐ 在创建时间添加时间格式
>
> ```
> @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
> ```

```
package com.xxxx.server.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.annotation.JsonFormat;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import java.time.LocalDateTime;
import lombok.Data;
import lombok.EqualsAndHashCode;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_salary")
@ApiModel(value="Salary对象", description="")
public class Salary implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "id")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "基本工资")
    private Integer basicSalary;

    @ApiModelProperty(value = "奖金")
    private Integer bonus;

    @ApiModelProperty(value = "午餐补助")
    private Integer lunchSalary;

    @ApiModelProperty(value = "交通补助")
    private Integer trafficSalary;

    @ApiModelProperty(value = "应发工资")
    private Integer allSalary;
```

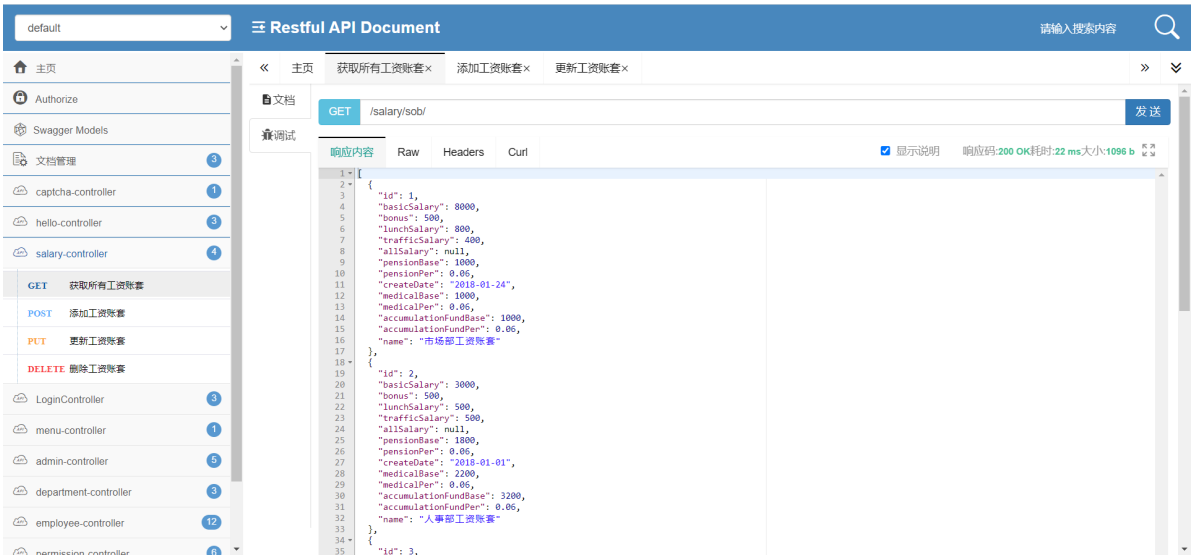
```
    @ApiModelProperty(value = "养老金基数")
    private Integer pensionBase;

    @ApiModelProperty(value = "养老金比率")
    private Float pensionPer;

    @ApiModelProperty(value = "启用时间")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    private LocalDateTime createDate;

    @ApiModelProperty(value = "医疗基数")
    private Integer medicalBase;

    @ApiModelProperty(value = "医疗保险比率")
    private Float medicalPer;

    @ApiModelProperty(value = "公积金基数")
    private Integer accumulationFundBase;

    @ApiModelProperty(value = "公积金比率")
    private Float accumulationFundPer;

    @ApiModelProperty(value = "名称")
    private String name;


}
```

40.2 SalaryController

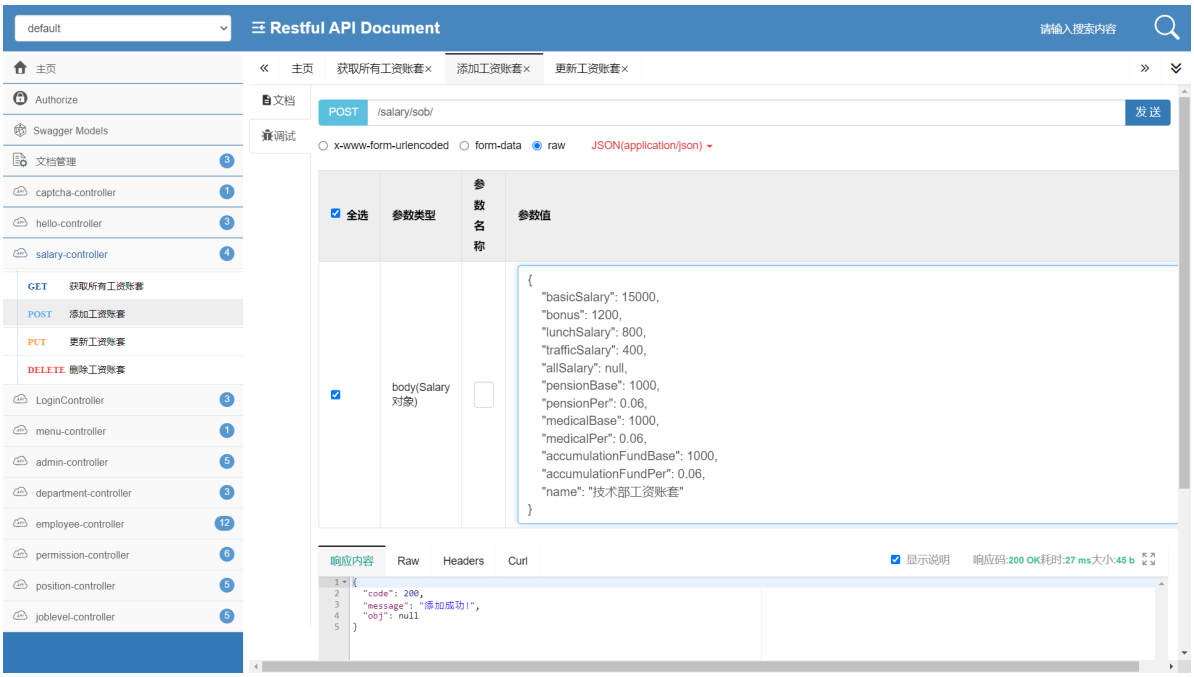
```
package com.xxxx.server.controller;


import com.xxxx.server.pojo.Salary;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.ISalaryService;
import io.swagger.annotations.ApiOperation;
import java.time.LocalDateTime;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author lizongzai
```

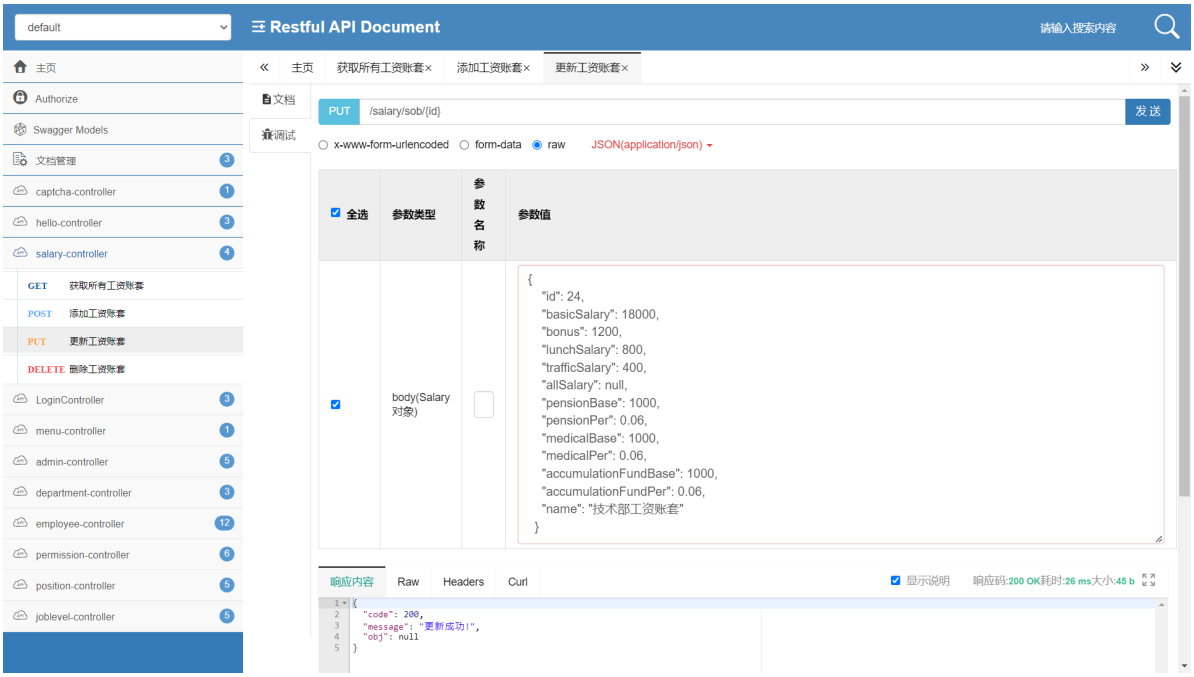
```
 * @since 2023-01-04
 */
@RestController
@RequestMapping("/salary/sob")
public class SalaryController {

  @Autowired
  private ISalaryService salaryService;

  @ApiOperation(value = "获取所有工资账套")
  @GetMapping("/")
  public List<Salary> getAllSalaries() {
    return salaryService.list();
  }

  @ApiOperation(value ="添加工资账套")
  @PostMapping("/")
  public RespBean addSalary(@RequestBody Salary salary) {
    salary.setCreateDate(LocalDateTime.now());
    if (salaryService.save((salary))) {
      return RespBean.success("添加成功!");
    }
    return RespBean.error("添加失败!");
  }

  @ApiOperation(value = "更新工资账套")
  @PutMapping("/{id}")
  public RespBean updateSalary(@RequestBody Salary salary) {
    if (salaryService.updateById(salary)) {
      return RespBean.success("更新成功!");
    }
    return RespBean.error("更新失败!");
  }

  @ApiOperation(value = "删除工资账套")
  @DeleteMapping("/{id}")
  public RespBean deleteSalary(@PathVariable Integer id) {
    if (salaryService.removeById(id)) {
      return RespBean.success("删除成功!");
    }
    return RespBean.error("删除失败!");
  }

}
```

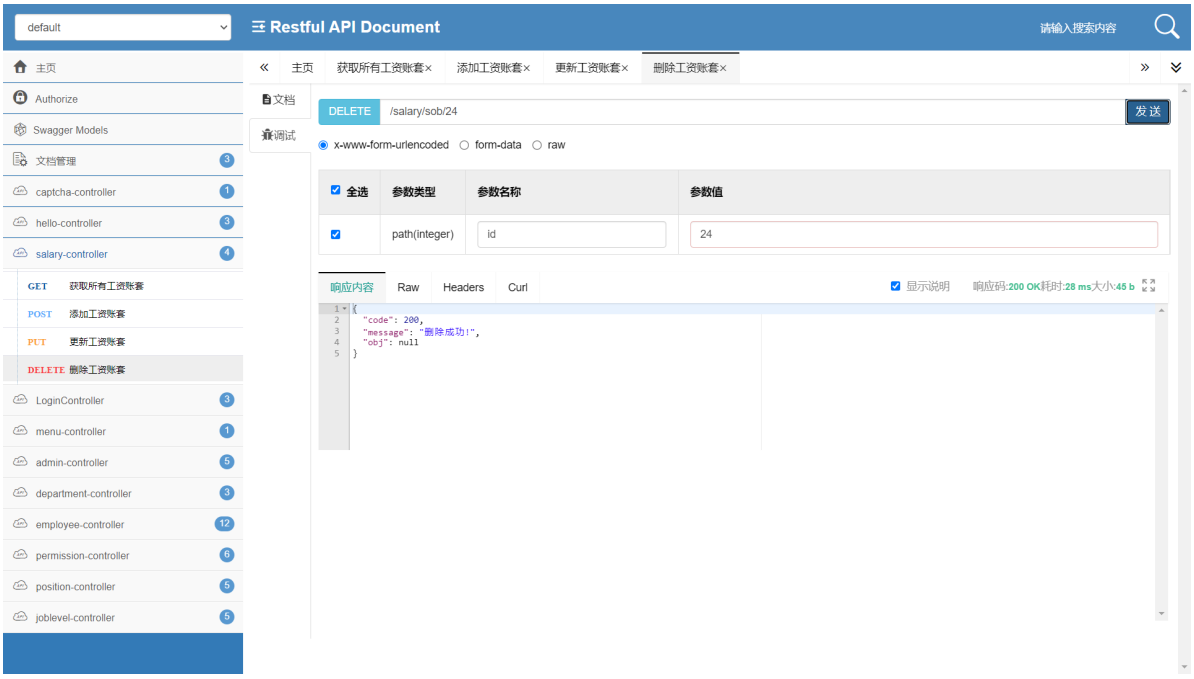
40.3 Restful API test




```
{
    "id": 24,
    "basicSalary": 15000,
    "bonus": 1200,
    "lunchSalary": 800,
    "trafficSalary": 400,
    "allSalary": null,
    "pensionBase": 1000,
    "pensionPer": 0.06,
    "createDate": "2023-01-15",
    "medicalBase": 1000,
    "medicalPer": 0.06,
    "accumulationFundBase": 1000,
    "accumulationFundPer": 0.06,
    "name": "技术部工资账套"
}
```





41. 员工账套

41.1 Employee

☐ 在employee员工实体添加salary工资账套对象

```
@ApiModelProperty(value = "工资账套")
@TableField(exist = false)
private Salary salary;
```

```java
package com.xxxx.server.pojo;

import cn.afterturn.easypoi.excel.annotation.Excel;
import cn.afterturn.easypoi.excel.annotation.ExcelEntity;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.annotation.JsonFormat;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import java.time.LocalDate;
import lombok.Data;
import lombok.EqualsAndHashCode;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_employee")
@ApiModel(value="Employee对象", description="")
public class Employee implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "员工编号")
    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    @ApiModelProperty(value = "员工姓名")
    @Excel(name = "员工姓名")
    private String name;

    @ApiModelProperty(value = "性别")
    @Excel(name = "性别")
    private String gender;

    @ApiModelProperty(value = "出生日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "出生日期",width = 20, format = "yyyy-MM-dd")
    private LocalDate birthday;

    @ApiModelProperty(value = "身份证号")
    @Excel(name = "身份证号", width = 30)
    private String idCard;

    @ApiModelProperty(value = "婚姻状况")
    @Excel(name = "婚姻状况")
    private String wedlock;
```

```java
    @ApiModelProperty(value = "民族")
    private Integer nationId;

    @ApiModelProperty(value = "籍贯")
    @Excel(name = "籍贯")
    private String nativePlace;

    @ApiModelProperty(value = "政治面貌")
    private Integer politicId;

    @ApiModelProperty(value = "邮箱")
    @Excel(name = "邮箱", width = 30)
    private String email;

    @ApiModelProperty(value = "电话号码")
    @Excel(name = "电话号码", width = 15)
    private String phone;

    @ApiModelProperty(value = "联系地址")
    @Excel(name = "联系地址", width = 40)
    private String address;

    @ApiModelProperty(value = "所属部门")
    private Integer departmentId;

    @ApiModelProperty(value = "职称ID")
    private Integer jobLevelId;

    @ApiModelProperty(value = "职位ID")
    private Integer posId;

    @ApiModelProperty(value = "聘用形式")
    @Excel(name = "聘用形式")
    private String engageForm;

    @ApiModelProperty(value = "最高学历")
    @Excel(name = "最高学历")
    private String tiptopDegree;

    @ApiModelProperty(value = "所属专业")
    @Excel(name = "所属专业", width = 20)
    private String specialty;

    @ApiModelProperty(value = "毕业院校")
    @Excel(name = "毕业院校", width = 20)
    private String school;

    @ApiModelProperty(value = "入职日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "入职日期", width = 20, format = "yyyy-MM-dd")
    private LocalDate beginDate;

    @ApiModelProperty(value = "在职状态")
    @Excel(name = "在职状态")
    private String workState;

    @ApiModelProperty(value = "工号")
    private String workID;
```

```
    @ApiModelProperty(value = "合同期限")
    @Excel(name = "合同期限",suffix = "年")
    private Double contractTerm;

    @ApiModelProperty(value = "转正日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "转正日期", width = 20, format = "yyyy-MM-dd")
    private LocalDate conversionTime;

    @ApiModelProperty(value = "离职日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "离职日期", width = 20, format = "yyyy-MM-dd")
    private LocalDate notWorkDate;

    @ApiModelProperty(value = "合同起始日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "合同起始日期", width = 20, format = "yyyy-MM-dd")
    private LocalDate beginContract;

    @ApiModelProperty(value = "合同终止日期")
    @JsonFormat(pattern = "yyyy-MM-dd", timezone = "Asia/Shanghai")
    @Excel(name = "合同终止日期", width = 20, format = "yyyy-MM-dd")
    private LocalDate endContract;

    @ApiModelProperty(value = "工龄")
    @Excel(name = "工龄")
    private Integer workAge;

    @ApiModelProperty(value = "工资账套ID")
    private Integer salaryId;

    @ApiModelProperty(value = "民族")
    @TableField(exist = false)
    @ExcelEntity(name = "民族")
    private Nation nation;

    @ApiModelProperty(value = "政治面貌")
    @TableField(exist = false)
    @ExcelEntity(name = "政治面貌")
    private PoliticsStatus politicsStatus;

    @ApiModelProperty(value = "部门")
    @TableField(exist = false)
    @ExcelEntity(name = "部门")
    private Department department;

    @ApiModelProperty(value = "职称")
    @TableField(exist = false)
    @ExcelEntity(name = "职称")
    private Joblevel joblevel;

    @ApiModelProperty(value = "职位")
    @TableField(exist = false)
    @ExcelEntity(name = "职位")
    private Position position;

    @ApiModelProperty(value = "工资账套")
```

```
    @TableField(exist = false)
    private Salary salary;

}
```

41.2 SalarySobCfgController

```
package com.xxxx.server.controller;

import com.baomidou.mybatisplus.core.conditions.update.UpdateWrapper;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.Salary;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IEmployeeService;
import com.xxxx.server.service.ISalaryService;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/salary/sobcfg")
public class SalarySobCfgController {

  @Autowired
  private ISalaryService salaryService;
  @Autowired
  private IEmployeeService employeeService;

  @ApiOperation(value = "获取所有工资账套")
  @GetMapping("/salaries")
  public List<Salary> getAllSalaries() {
    return salaryService.list();
  }

  @ApiOperation(value = "获取所有员工账套")
  @GetMapping("/")
  public RespPageBean getEmployeeWithSalary(@RequestParam(defaultValue = "1") Integer
currentPage,
                                            @RequestParam(defaultValue = "10") Integer
size) {
    return employeeService.getEmployeeWithSalary(currentPage, size);
  }

  @ApiOperation(value = "更新员工账套")
  @PutMapping("/")
  public RespBean updateEmployeeSalary(Integer eid, Integer sid) {
    if (employeeService.update(new UpdateWrapper<Employee>
().set("salaryId",sid).eq("id",eid))) {
      return RespBean.success("更新成功!");
```

```
    }
    return RespBean.error("更新失败!");
  }
}
```

41.3 IEmployeeService

```
/**
   * 获取所有员工账套
   * @param currentPage
   * @param size
   * @return
   */
RespPageBean getEmployeeWithSalary(Integer currentPage, Integer size);
```

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import java.time.LocalDate;
import java.util.List;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IEmployeeService extends IService<Employee> {

  /**
   * @Description //TODO 获取所有员工(分页查询)
   * @param currentPage
   * @param pageSize
   * @param employee
   * @param beginDateScope
   * @return
   */
  RespPageBean getAllEmployeeByPage(Integer currentPage, Integer pageSize, Employee
employee, LocalDate beginDateScope);

  /**
   * @Description //TODO 获取工号
   * @Author lizongzai
   * @Date 2023/01/11 13:06
   * @return
   */
```

```
    RespBean getMaxWorkId();

    /**
     * @Description //TODO 添加员工
     * @param employee
     * @return
     */
    RespBean addEmp(Employee employee);

    /**
     * @Description //TODO 导出员工数据
     * @param id
     * @return
     */
    List<Employee> getEmployee(Integer id);

    /**
     * 获取所有员工账套
     * @param currentPage
     * @param size
     * @return
     */
    RespPageBean getEmployeeWithSalary(Integer currentPage, Integer size);
}
```

41.4 EmployeeServiceImpl

```
    /**
     * 获取所有员工账套
     * @param currentPage
     * @param size
     * @return
     */
    @Override
    public RespPageBean getEmployeeWithSalary(Integer currentPage, Integer size) {
        //开启分页
        Page<Employee> page = new Page<>(currentPage,size);
        IPage<Employee> employeeIPage = employeeMapper.getEmployeeWithSalary(page);
        RespPageBean respPageBean = new
    RespPageBean(employeeIPage.getTotal(),employeeIPage.getRecords());
        return respPageBean;
    }
```

```
package com.xxxx.server.service.impl;

import com.xxxx.server.mapper.MailLogMapper;
import com.xxxx.server.constant.MailConstants;
import java.time.LocalDateTime;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
```

```java
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.mapper.EmployeeMapper;
import com.xxxx.server.pojo.Employee;
import com.xxxx.server.pojo.MailLog;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.pojo.common.RespPageBean;
import com.xxxx.server.service.IEmployeeService;
import java.text.DecimalFormat;
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import javax.annotation.Resource;
import org.springframework.amqp.rabbit.connection.CorrelationData;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * <p>
 * 服务实现类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Service
public class EmployeeServiceImpl extends ServiceImpl<EmployeeMapper, Employee>
implements
    IEmployeeService {

  @Autowired
  @Resource
  private EmployeeMapper employeeMapper;
  @Autowired
  private RabbitTemplate rabbitTemplate;
  @Autowired
  @Resource
  private MailLogMapper mailLogMapper;

  /**
   * @param currentPage
   * @param pageSize
   * @param employee
   * @param beginDateScope
   * @return
   * @Description //TODO 获取所有员工(分页查询)
   */
  @Override
  public RespPageBean getAllEmployeeByPage(Integer currentPage, Integer pageSize,
Employee employee,
      LocalDate beginDateScope) {
    //开启分页
    Page<Employee> employeePage = new Page<>();
    IPage<Employee> employeeByPage = employeeMapper.getAllEmployeeByPage(employeePage,
employee,
```

```
            beginDateScope);
        RespPageBean respPageBean = new RespPageBean(employeeByPage.getTotal(),
            employeeByPage.getRecords());
        return respPageBean;
    }

    /**
     * @return
     * @Description //TODO 获取工号
     * @Author lizongzai
     * @Date 2023/01/11 13:06
     */
    @Override
    public RespBean getMaxWorkId() {
        //获取员工Key和value
        List<Map<String, Object>> maps = employeeMapper.selectMaps(
            new QueryWrapper<Employee>().select("max(workId)"));
        //System.out.println("maps = " + maps);
        //maps = [{max(workId)=00000100}]
        //只有一个最大值，那么get(0).get("max(workId")即可获取它的值
        //工号强制转换整型，然后再加1
        //maps.get(0).get("max(workId)"是一个Object对象，先将其转String
        //%8d，表示不足长度时，自动左补位，补位数为0
        return RespBean.success(null,
            String.format("%8d",
Integer.parseInt(maps.get(0).get("max(workId)").toString()) + 1));
    }

    /**
     * @param employee
     * @return
     * @Description //TODO 添加员工
     */
    @Override
    public RespBean addEmp(Employee employee) {
        //处理合同期限，保留两位小数
        LocalDate beginContract = employee.getBeginContract();
        LocalDate endContract = employee.getEndContract();
        long days = beginContract.until(endContract, ChronoUnit.DAYS); //计算合同的天数
        DecimalFormat decimalFormat = new DecimalFormat("##.00"); //设置保留两位小数
        employee.setContractTerm(
            Double.parseDouble(decimalFormat.format(days / 365.00))); //算计年数，保留两位小数
点
        //System.out.println("合同年数 = " + employee.getContractTerm());
        //若添加成功，则返回一条记录。表示添加成功
        if (1 == employeeMapper.insert(employee)) {
            //通过员工id获取员工对象
            Employee emp = employeeMapper.getEmployee(employee.getId()).get(0);
            //数据库记录发送消息
            String msgId = UUID.randomUUID().toString();
            MailLog mailLog = new MailLog();
            mailLog.setMsgId(msgId); //消息id
            mailLog.setEid(employee.getId()); //员工id
            mailLog.setStatus(0); //投递状态 （0:消息投递中 1:投递成功 2:投递失败）
            mailLog.setRouteKey(MailConstants.MAIL_ROUTING_KEY_NAME); //路由键
            mailLog.setExchange(MailConstants.MAIL_EXCHANGE_NAME); //交换机
            mailLog.setCount(0); //重试次数
```

```
        mailLog.setTryTime(LocalDateTime.now().plusMinutes(MailConstants.MSG_TIMEOUT));
//重试时间+1分钟
        mailLog.setCreateTime(LocalDateTime.now()); //创建时间
        mailLog.setUpdateTime(LocalDateTime.now()); //更新时间
        //消息落库
        mailLogMapper.insert(mailLog);
        //发送邮件
        rabbitTemplate.convertAndSend(MailConstants.MAIL_QUEUE_NAME,
            MailConstants.MAIL_ROUTING_KEY_NAME, emp, new CorrelationData(msgId));
        return RespBean.success("添加成功!");
      }
      return RespBean.error("添加失败!");
    }

    /**
     * @param id
     * @return
     * @Description //TODO 导出员工数据
     */
    @Override
    public List<Employee> getEmployee(Integer id) {
        return employeeMapper.getEmployee(id);
    }

    /**
     * 获取所有员工账套
     * @param currentPage
     * @param size
     * @return
     */
    @Override
    public RespPageBean getEmployeeWithSalary(Integer currentPage, Integer size) {
      //开启分页
      Page<Employee> page = new Page<>(currentPage,size);
      IPage<Employee> employeeIPage = employeeMapper.getEmployeeWithSalary(page);
      RespPageBean respPageBean = new
RespPageBean(employeeIPage.getTotal(),employeeIPage.getRecords());
      return respPageBean;
    }

}
```

41.5 EmployeeMapper

```
package com.xxxx.server.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.xxxx.server.pojo.Employee;
import java.time.LocalDate;
import java.util.List;
import org.springframework.data.repository.query.Param;

/**
```

```
 * <p>
 * Mapper 接口
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface EmployeeMapper extends BaseMapper<Employee> {

  /**
   * TODO 获取所有员工(分页查询)
   * @param employeePage
   * @param employee
   * @param beginDateScope
   * @return
   */
  IPage<Employee> getAllEmployeeByPage(Page<Employee> employeePage,
      @Param("employee") Employee employee, @Param("beginDateScope") LocalDate
beginDateScope);

  /**
   * @Description //TODO 导出员工数据
   * @param id
   * @return
   */
  List<Employee> getEmployee(Integer id);

  /**
   * 获取所有员工账套
   * @param page
   * @return
   */
  IPage<Employee> getEmployeeWithSalary(Page<Employee> page);
}
```

41.6 EmployeeMapper

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.EmployeeMapper">

  <!-- 通用查询映射结果 -->
  <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.Employee">
    <id column="id" property="id"/>
    <result column="name" property="name"/>
    <result column="gender" property="gender"/>
    <result column="birthday" property="birthday"/>
    <result column="idCard" property="idCard"/>
    <result column="wedlock" property="wedlock"/>
    <result column="nationId" property="nationId"/>
    <result column="nativePlace" property="nativePlace"/>
    <result column="politicId" property="politicId"/>
    <result column="email" property="email"/>
    <result column="phone" property="phone"/>
```

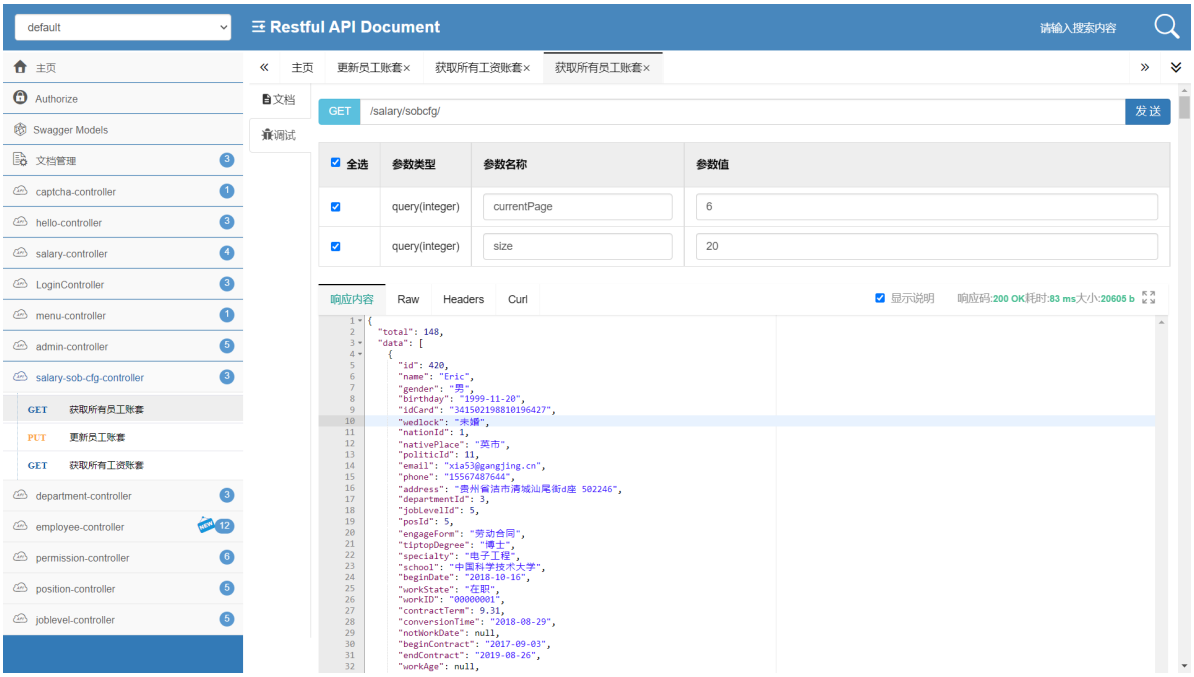
```
    <result column="address" property="address"/>
    <result column="departmentId" property="departmentId"/>
    <result column="jobLevelId" property="jobLevelId"/>
    <result column="posId" property="posId"/>
    <result column="engageForm" property="engageForm"/>
    <result column="tiptopDegree" property="tiptopDegree"/>
    <result column="specialty" property="specialty"/>
    <result column="school" property="school"/>
    <result column="beginDate" property="beginDate"/>
    <result column="workState" property="workState"/>
    <result column="workID" property="workID"/>
    <result column="contractTerm" property="contractTerm"/>
    <result column="conversionTime" property="conversionTime"/>
    <result column="notWorkDate" property="notWorkDate"/>
    <result column="beginContract" property="beginContract"/>
    <result column="endContract" property="endContract"/>
    <result column="workAge" property="workAge"/>
    <result column="salaryId" property="salaryId"/>
  </resultMap>

  <!-- 通用查询结果列 -->
  <sql id="Base_Column_List">
    id
    , name, gender, birthday, idCard, wedlock, nationId, nativePlace, politicId,
email, phone, address, departmentId, jobLevelId, posId, engageForm, tiptopDegree,
specialty, school, beginDate, workState, workID, contractTerm, conversionTime,
notWorkDate, beginContract, endContract, workAge, salaryId
  </sql>

  <resultMap id="EmployeeInfo" type="com.xxxx.server.pojo.Employee"
extends="BaseResultMap">
    <!-- 民族 -->
    <association property="nation" javaType="com.xxxx.server.pojo.Nation">
      <id column="nid" property="id"/>
      <result column="nname" property="name"/>
    </association>
    <!-- 政治面貌 -->
    <association property="politicsStatus"
javaType="com.xxxx.server.pojo.PoliticsStatus">
      <id column="pid" property="id"/>
      <result column="pname" property="name"/>
    </association>
    <!-- 部门 -->
    <association property="department" javaType="com.xxxx.server.pojo.Department">
      <id column="did" property="id"/>
      <result column="dname" property="name"/>
    </association>
    <!-- 职称 -->
    <association property="joblevel" javaType="com.xxxx.server.pojo.Joblevel">
      <id column="jid" property="id"/>
      <result column="jname" property="name"/>
    </association>
    <!-- 职位 -->
    <association property="position" javaType="com.xxxx.server.pojo.Position">
      <id column="posid" property="id"/>
      <result column="posname" property="name"/>
    </association>
  </resultMap>
```

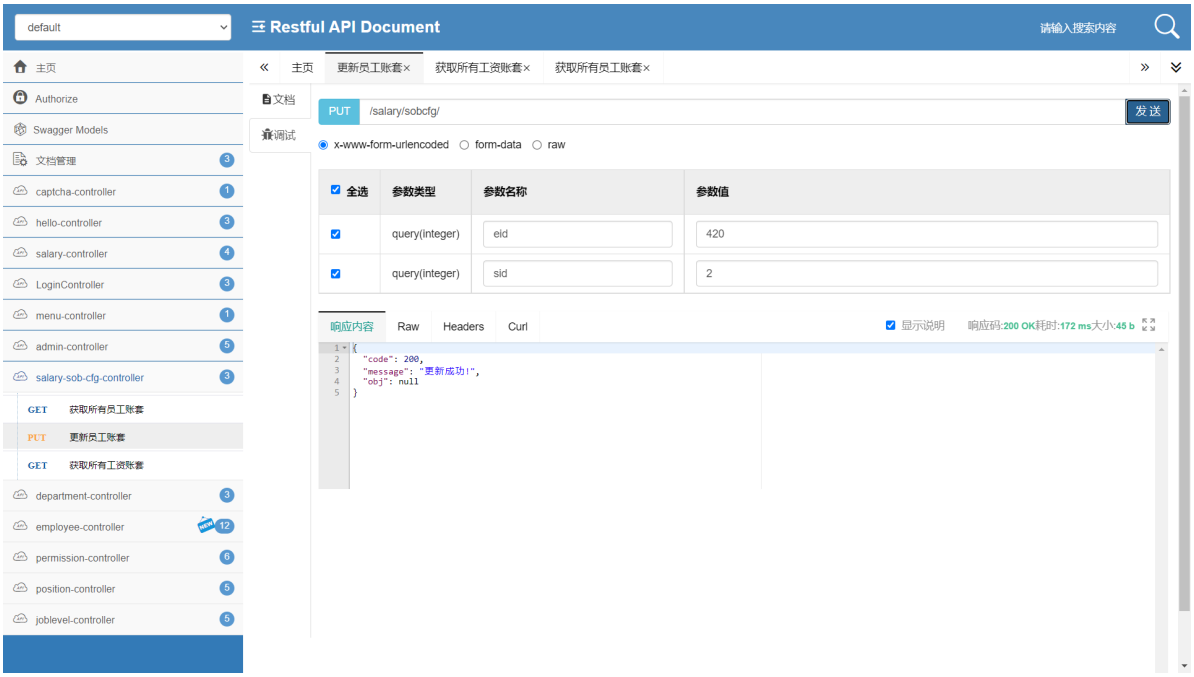
```
  <!-- 获取所有员工(分页查询)-->
  <select id="getEmployeeByPage" resultMap="EmployeeInfo">
    select
    e.*,
    n.id as nid,
    n.name as nname,
    p.id as pid,
    p.name as pname,
    d.id as did,
    d.name as dname,
    j.id as jid,
    j.name as jname,
    pos.id as posid,
    pos.name as posname
    from
    t_employee e,
    t_nation n,
    t_politics_status p,
    t_department d,
    t_joblevel j,
    t_position pos
    where
    e.nationId = n.id
    and e.politicId = p.id
    and e.jobLevelId = j.id
    and e.departmentId = d.id
    and e.posId = pos.id
    <if test="employee.name != null and '' != employee.name">
      AND e.name like concat('%',#{employee.name},'%')
    </if>
    <if test="employee.politicId != null">
      AND e.politicId like concat('%',#{employee.politicId},'%')
    </if>
    <if test="employee.nationId != null">
      AND e.nationId like concat('%',#{employee.nationId},'%')
    </if>
    <if test="employee.jobLevelId != null">
      AND e.jobLevelId like concat('%',#{employee.jobLevelId},'%')
    </if>
    <if test="employee.posId != null">
      AND e.posId like concat('%',#{employee.posId},'%')
    </if>
    <if test="employee.departmentId != null">
      AND e.departmentId like concat('%',#{employee.departmentId},'%')
    </if>
    <if test="employee.engageForm != null and '' != employee.engageForm">
      AND e.engageForm like concat('%',#{employee.engageForm},'%')
    </if>
    <if test="beginDateScope != null and beginDateScope.length == 2">
      AND e.beginDate between #{beginDateScope[0]} and beginDateScope[1]
    </if>
    ORDER BY e.id
  </select>

  <!--  获取所有员工工资账套-->
  <resultMap id="EmployeeWithSalary" type="com.xxxx.server.pojo.Employee"
extends="BaseResultMap">
```

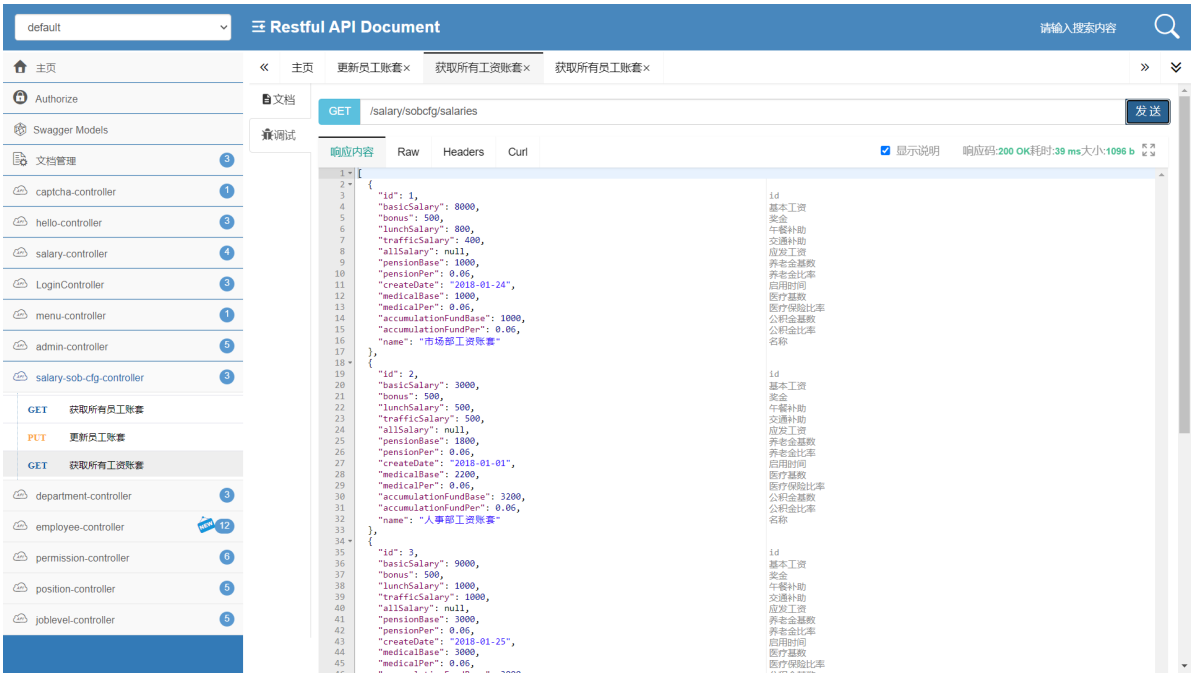
```
    <association property="salary" javaType="com.xxxx.server.pojo.Salary">
      <id column="sid" property="id"/>
      <result column="sbasicSalary" property="basicSalary"/>
      <result column="sbonus" property="bonus"/>
      <result column="slunchSalary" property="lunchSalary"/>
      <result column="strafficSalary" property="trafficSalary"/>
      <result column="sallSalary" property="allSalary"/>
      <result column="spensionBase" property="pensionBase"/>
      <result column="spensionPer" property="pensionPer"/>
      <result column="smedicalBase" property="medicalBase"/>
      <result column="smedicalPer" property="medicalPer"/>
      <result column="saccumulationFundBase"
        property="accumulationFundBase"/>
      <result column="saccumulationFundPer"
        property="accumulationFundPer"/>
      <result column="sname" property="name"/>
    </association>
    <association property="department" javaType="com.xxxx.server.pojo.Department">
      <result column="dname" property="name"/>
    </association>
  </resultMap>

  <!--  导出员工数据-->
  <select id="getEmployeeInfo" resultMap="EmployeeInfo">
    select
    e.*,
    n.id as nid,
    n.name as nname,
    p.id as pid,
    p.name as pname,
    d.id as did,
    d.name as dname,
    j.id as jid,
    j.name as jname,
    pos.id as posid,
    pos.name as posname
    from
    t_employee e,
    t_nation n,
    t_politics_status p,
    t_department d,
    t_joblevel j,
    t_position pos
    where
    e.nationId = n.id
    and e.politicId = p.id
    and e.jobLevelId = j.id
    and e.departmentId = d.id
    and e.posId = pos.id
    <if test="id != null ">
      e.id = #{id}
    </if>
  </select>


  <!--  获取所有员工工资账套-->
  <select id="getEmployeeWithSalary" resultMap="EmployeeWithSalary">
    SELECT e.*,
```

```
            td.`name`              as dname,
            s.id                   AS sid,
            s.`name`               AS sname,
            s.basicSalary          AS sbasicSalary,
            s.trafficSalary        AS strafficSalary,
            s.lunchSalary          AS slunchSalary,
            s.bonus                AS sbonus,
            s.allSalary            AS sallSalary,
            s.pensionPer           AS spensionPer,
            s.pensionBase          AS spensionBase,
            s.medicalPer           AS smedicalPer,
            s.medicalBase          AS smedicalBase,
            s.accumulationFundPer  AS saccumulationFundPer,
            s.accumulationFundBase AS saccumulationFundBase
    FROM t_employee as e
            LEFT JOIN t_salary as s ON e.salaryId = s.id
            LEFT JOIN t_department as td ON e.departmentId = td.id
    ORDER BY e.id
  </select>

</mapper>
```

41.7 Restful API Test







41.8 EmployeeMapper

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxxx.server.mapper.EmployeeMapper">

    <!-- 通用查询映射结果 -->
    <resultMap id="BaseResultMap" type="com.xxxx.server.pojo.Employee">
        <id column="id" property="id" />
        <result column="name" property="name" />
        <result column="gender" property="gender" />
        <result column="birthday" property="birthday" />
        <result column="idCard" property="idCard" />
```

```
        <result column="wedlock" property="wedlock" />
        <result column="nationId" property="nationId" />
        <result column="nativePlace" property="nativePlace" />
        <result column="politicId" property="politicId" />
        <result column="email" property="email" />
        <result column="phone" property="phone" />
        <result column="address" property="address" />
        <result column="departmentId" property="departmentId" />
        <result column="jobLevelId" property="jobLevelId" />
        <result column="posId" property="posId" />
        <result column="engageForm" property="engageForm" />
        <result column="tiptopDegree" property="tiptopDegree" />
        <result column="specialty" property="specialty" />
        <result column="school" property="school" />
        <result column="beginDate" property="beginDate" />
        <result column="workState" property="workState" />
        <result column="workID" property="workID" />
        <result column="contractTerm" property="contractTerm" />
        <result column="conversionTime" property="conversionTime" />
        <result column="notWorkDate" property="notWorkDate" />
        <result column="beginContract" property="beginContract" />
        <result column="endContract" property="endContract" />
        <result column="workAge" property="workAge" />
        <result column="salaryId" property="salaryId" />
    </resultMap>

    <resultMap id="EmployeeInfo" type="com.xxxx.server.pojo.Employee"
extends="BaseResultMap">
        <!-- 民族 -->
        <association property="nation" javaType="com.xxxx.server.pojo.Nation">
            <id column="nid" property="id"/>
            <result column="nname" property="name"/>
        </association>
        <!-- 政治面貌 -->
        <association property="politicsStatus"
javaType="com.xxxx.server.pojo.PoliticsStatus">
            <id column="pid" property="id"/>
            <result column="pname" property="name"/>
        </association>
        <!-- 部门 -->
        <association property="department" javaType="com.xxxx.server.pojo.Department">
            <id column="did" property="id"/>
            <result column="dname" property="name"/>
        </association>
        <!-- 职称 -->
        <association property="joblevel" javaType="com.xxxx.server.pojo.Joblevel">
            <id column="jid" property="id"/>
            <result column="jname" property="name"/>
        </association>
        <!-- 职位 -->
        <association property="position" javaType="com.xxxx.server.pojo.Position">
            <id column="posid" property="id"/>
            <result column="posname" property="name"/>
        </association>
    </resultMap>

    <resultMap id="EmployeeWithSalary" type="com.xxxx.server.pojo.Employee"
extends="BaseResultMap">
```

```
        <association property="salary" javaType="com.xxxx.server.pojo.Salary">
            <id column="sid" property="id"/>
            <result column="sbasicSalary" property="basicSalary"/>
            <result column="sbonus" property="bonus"/>
            <result column="slunchSalary" property="lunchSalary"/>
            <result column="strafficSalary" property="trafficSalary"/>
            <result column="sallSalary" property="allSalary"/>
            <result column="spensionBase" property="pensionBase"/>
            <result column="spensionPer" property="pensionPer"/>
            <result column="smedicalBase" property="medicalBase"/>
            <result column="smedicalPer" property="medicalPer"/>
            <result column="saccumulationFundBase"
              property="accumulationFundBase"/>
            <result column="saccumulationFundPer"
              property="accumulationFundPer"/>
            <result column="sname" property="name"/>
        </association>
        <association property="department" javaType="com.xxxx.server.pojo.Department">
            <result column="dname" property="name"/>
        </association>
    </resultMap>

    <!-- 通用查询结果列 -->
    <sql id="Base_Column_List">
        id, name, gender, birthday, idCard, wedlock, nationId, nativePlace, politicId,
email, phone, address, departmentId, jobLevelId, posId, engageForm, tiptopDegree,
specialty, school, beginDate, workState, workID, contractTerm, conversionTime,
notWorkDate, beginContract, endContract, workAge, salaryId
    </sql>

    <!-- 获取所有员工(分页查询) -->
    <select id="getAllEmployeeByPage" resultMap="EmployeeInfo">
        select
            e.*,
            n.id as nid,
            n.name as nname,
            p.id as pid,
            p.name as pname,
            d.id as did,
            d.name as dname,
            j.id as jid,
            j.name as jname,
            pos.id as posid,
            pos.name as posname
        from
            t_employee e,
            t_nation n,
            t_politics_status p,
            t_department d,
            t_joblevel j,
            t_position pos
        where
            e.nationId = n.id
          and e.politicId = p.id
          and e.jobLevelId = j.id
          and e.departmentId = d.id
          and e.posId = pos.id
          <if test="employee.name != null and '' != employee.name">
```

```
            AND e.name like concat('%',#{employee.name},'%')
        </if>
        <if test="employee.politicId != null">
            AND e.politicId like concat('%',#{employee.politicId},'%')
        </if>
        <if test="employee.nationId != null">
            AND e.nationId like concat('%',#{employee.nationId},'%')
        </if>
        <if test="employee.jobLevelId != null">
            AND e.jobLevelId like concat('%',#{employee.jobLevelId},'%')
        </if>
        <if test="employee.posId != null">
            AND e.posId like concat('%',#{employee.posId},'%')
        </if>
        <if test="employee.departmentId != null">
            AND e.departmentId like concat('%',#{employee.departmentId},'%')
        </if>
        <if test="employee.engageForm != null and '' != employee.name">
            AND e.engageForm like concat('%',#{employee.engageForm},'%')
        </if>
    <if test="beginDateScope != null and beginDateScope.length == 2">
        AND e.beginDate between #{beginDateScope[0]} and beginDateScope[1]
    </if>

    ORDER BY e.id
</select>

<!--导出员工数据-->
<select id="getEmployee" resultMap="EmployeeInfo">
    select
        e.*,
        n.id as nid,
        n.name as nname,
        p.id as pid,
        p.name as pname,
        d.id as did,
        d.name as dname,
        j.id as jid,
        j.name as jname,
        pos.id as posid,
        pos.name as posname
    from
        t_employee e,
        t_nation n,
        t_politics_status p,
        t_department d,
        t_joblevel j,
        t_position pos
    where
        e.nationId = n.id
      and e.politicId = p.id
      and e.jobLevelId = j.id
      and e.departmentId = d.id
      and e.posId = pos.id
      -- 如果传入id的值为空，那么就查询全部。否则就根据传入id编号进行查询
      <if test="id != null">
        and e.id = #{id}
      </if>
```

```
    </select>

  <!-- 获取所有员工账套 -->
  <select id="getEmployeeWithSalary" resultMap="EmployeeWithSalary">
      SELECT
          e.*,
          td.`name` as dname,
          s.id AS sid,
          s.`name` AS sname,
          s.basicSalary AS sbasicSalary,
          s.trafficSalary AS strafficSalary,
          s.lunchSalary AS slunchSalary,
          s.bonus AS sbonus,
          s.allSalary AS sallSalary,
          s.pensionPer AS spensionPer,
          s.pensionBase AS spensionBase,
          s.medicalPer AS smedicalPer,
          s.medicalBase AS smedicalBase,
          s.accumulationFundPer AS saccumulationFundPer,
          s.accumulationFundBase AS saccumulationFundBase
      FROM
          t_employee as e
          LEFT JOIN t_salary as s ON e.salaryId = s.id
          LEFT JOIN t_department as td ON e.departmentId = td.id
      ORDER BY e.id

  </select>

</mapper>
```

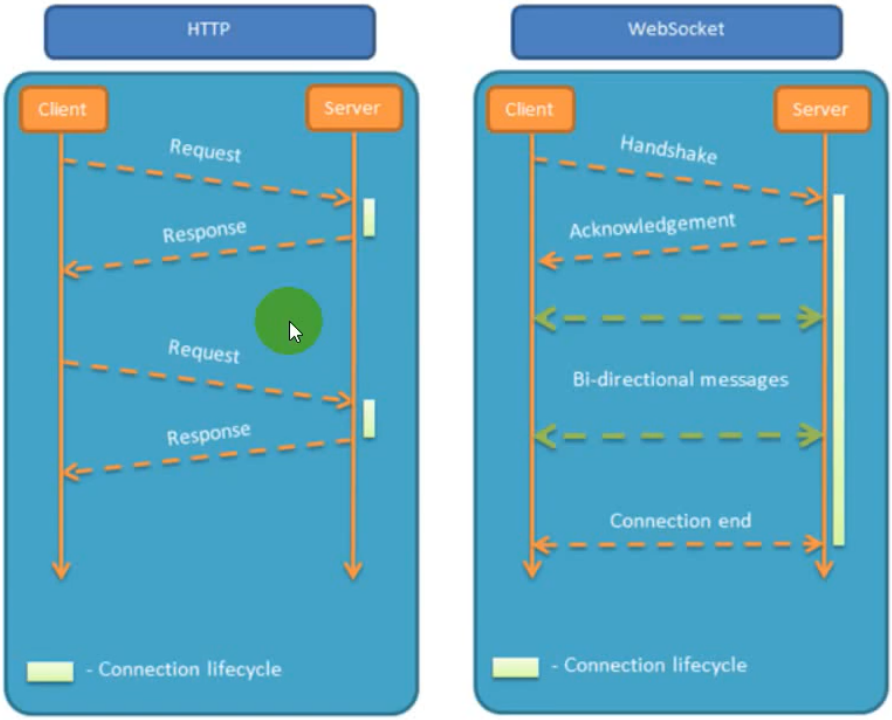
42. WebSocket配置

42.1 什么是WebSocket?

WebSocket 是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通讯的协议。

WebSocket 使得客户端和服务器之间的数据交换变得更加简单，允许服务端主动向客户端推送数据。在 WebSocket API 中，浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。

它的最大特点就是，服务器可以主动向客户端推送信息，客户端也可以主动向服务器发送信息，是真正的双向平等对话，属于服务器推送技术的一种。



其它特点包括:

- 较少的控制开销
- 更强的实时性
- 保持连接状态
- 更好的二进制支持
- 可以支持扩展
- 更好的压缩效果

42.2 Websocker依赖包

```
<!--websocket 依赖-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>
```

42.3 Websocket配置类

参考信息： https://vimsky.com/examples/detail/java-class-org.springframework.messaging.simp.stomp.StompHeaderAccessor.html

```
package com.xxxx.server.config.websocket;

import com.xxxx.server.config.jwt.JwtTokenUtil;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.messaging.Message;
import org.springframework.messaging.MessageChannel;
import org.springframework.messaging.simp.config.ChannelRegistration;
import org.springframework.messaging.simp.config.MessageBrokerRegistry;
import org.springframework.messaging.simp.stomp.StompCommand;
import org.springframework.messaging.simp.stomp.StompHeaderAccessor;
import org.springframework.messaging.support.ChannelInterceptor;
import org.springframework.messaging.support.MessageHeaderAccessor;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;
import org.springframework.web.socket.config.annotation.StompEndpointRegistry;
import
org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer;

/**
 * Websocket配置
 *
 * @author lizongzai
 * @since 1.0.0
 */
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

  @Value("${jwt.tokenHead}")
  private String tokenHead;
  @Autowired
  private JwtTokenUtil jwtTokenUtil;
  @Autowired
  private UserDetailsService userDetailsService;

  /**
   * 1.添加endpoint，这样在网页里面可以通过websocket连接上后端服务
   * 2.也就是我们配置websocket服务地址，并且可以指定是否可以使用socketJS
   *
   * @param registry
   */
  @Override
  public void registerStompEndpoints(StompEndpointRegistry registry) {
    WebSocketMessageBrokerConfigurer.super.registerStompEndpoints(registry);

    /**
```

```
     * 1.将"/ws/ep"路径注册为端点, 用户连接了这个端点就可以进行websocket通讯,并支持websocketJS
     * 2.setAllowedOrigins("*"): 允许跨域
     * 3.withSocketJS(): 支持socketJS连接
     */
    registry.addEndpoint("/ws/ep").setAllowedOrigins("*").withSockJS();
  }

  /**
   * 输入通道参数配置
   *
   * @param registration
   */
  @Override
  public void configureClientInboundChannel(ChannelRegistration registration) {

 WebSocketMessageBrokerConfigurer.super.configureClientInboundChannel(registration);

    registration.interceptors(new ChannelInterceptor() {
      //预发送配置
      @Override
      public Message<?> preSend(Message<?> message, MessageChannel channel) {

        StompHeaderAccessor accessor = MessageHeaderAccessor.getAccessor(message,
StompHeaderAccessor.class);
        //判断是否为连接, 若是连接状态, 则需要token,并且设置用户对象
        if (StompCommand.CONNECT.equals(accessor.getCommand()) && accessor != null) {
          //获取token,此外参数"Auth-Token"是前端传递过来值
          String token = accessor.getFirstNativeHeader("Auth-Token");
          //判断token是否为空
          if (!StringUtils.isEmpty(token)) {
            //获取授权token
            String authToken = token.substring(tokenHead.length());
            //获取用户名
            String username = jwtTokenUtil.getUserNameFromToken(token);
            if (!StringUtils.isEmpty(username)) {
              //获取用户信息, 表示登录
              UserDetails userDetails =
userDetailsService.loadUserByUsername(username);
              //验证token是否有效,重新设置用户对象
              if (jwtTokenUtil.validateToken(authToken, userDetails)) {
                UsernamePasswordAuthenticationToken authenticationToken = new
UsernamePasswordAuthenticationToken(
                    userDetails, null, userDetails.getAuthorities());

 SecurityContextHolder.getContext().setAuthentication(authenticationToken);
                accessor.setUser(authenticationToken);
              }
            }
          }

        }
        return message;
        //return ChannelInterceptor.super.preSend(message, channel);
      }
    });
  }

  /**
```

```
     * 配置消息代理
     *
     * @param registry
     */
    @Override
    public void configureMessageBroker(MessageBrokerRegistry registry) {
        WebSocketMessageBrokerConfigurer.super.configureMessageBroker(registry);

        //配置代理域, 可以配置多个,配置代理目的地前缀为"/queue",可以在配置域上向客户端推送消息
        registry.enableSimpleBroker("/queue");
    }
}
```

42.4 ChatMsg实体

```
package com.xxxx.server.pojo;

import java.time.LocalDateTime;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.experimental.Accessors;

/**
 * 消息
 * @author lizongzai
 * @since 1.0.0
 */
@Data
@EqualsAndHashCode(callSuper = false) //排除父类字段
@Accessors(chain = true) //开启链式编程
public class ChatMsg {

    //发送端
    private String from;
    //接收端
    private String to;
    //发送内容
    private String content;
    //发送时间
    private LocalDateTime date;
    //昵称
    private String fromNickName;

}
```

42.5 WsController

> 知识点： 注意注解为@MessageMapping(“路由路径”)

```
package com.xxxx.server.controller;
```

```
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.ChatMsg;
import java.time.LocalDateTime;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.messaging.simp.SimpMessagingTemplate;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;

/**
 * websocket
 * @author lizongzai
 * @since 1.0.0
 */
@Controller
public class WsController {
  @Autowired
  private SimpMessagingTemplate simpMessagingTemplate;

  @MessageMapping("/ws/chat")
  public void handleMsg(Authentication authentication, ChatMsg chatMsg) {
    //获取当前用户对象
    Admin admin = ((Admin) authentication.getPrincipal());
    //当前用户名
    chatMsg.setFrom(admin.getUsername());
    //显示用户名
    chatMsg.setFrom(admin.getName());
    //发送时间
    chatMsg.setDate(LocalDateTime.now());
    //参数：消息接收者,队列,发送内容
    simpMessagingTemplate.convertAndSendToUser(chatMsg.getTo(),"/queue/chat",chatMsg);
  }

}
```

42.6 ChatController

```
package com.xxxx.server.controller;

import com.xxxx.server.pojo.Admin;
import com.xxxx.server.service.IAdminService;
import io.swagger.annotations.ApiOperation;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * 在线聊天
 * @author lizongzai
 * @since 1.0.0
 */
@RestController
@RequestMapping("/chat")
public class ChatController {
```

```
  @Autowired
  private IAdminService adminService;

  @ApiOperation(value = "获取所有操作员")
  @GetMapping("/admin")
  public List<Admin> getAllAdmins(String keywords) {
    return adminService.getAllAdmins(keywords);
  }
}
```

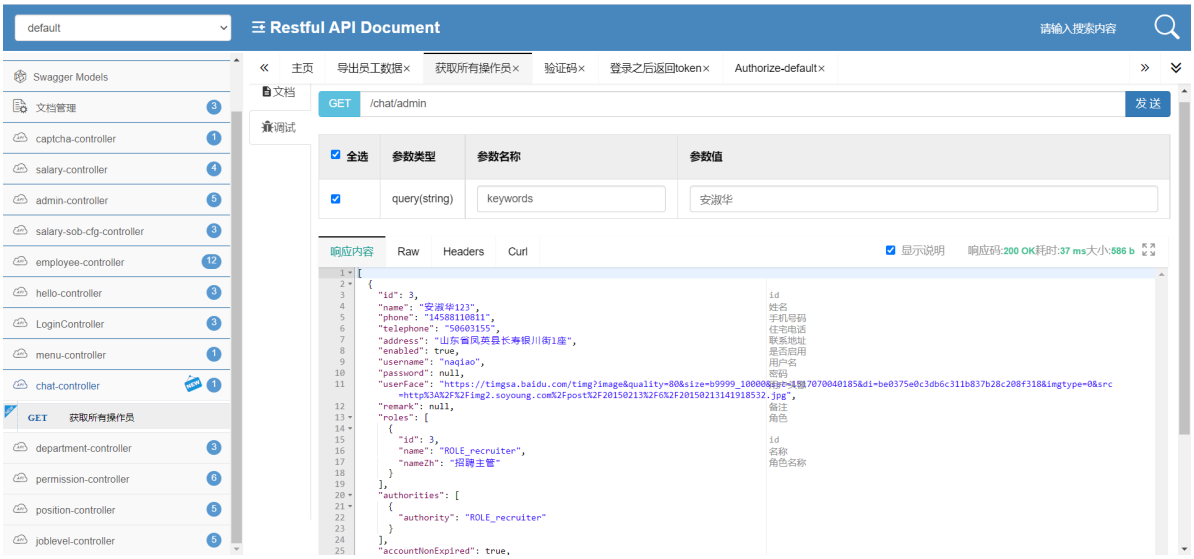
42.7 SecurityConfig

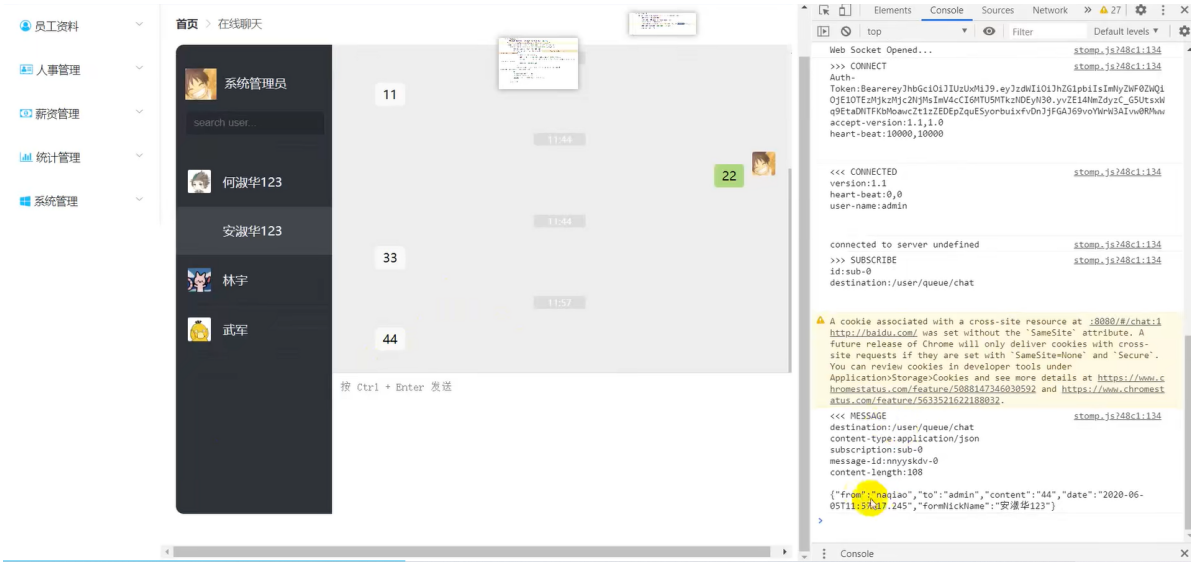
> ☐ 配置放行 `"/ws/**"`

```
package com.xxxx.server.config.security;

import com.xxxx.server.config.filter.CustomFilter;
import com.xxxx.server.config.filter.CustomUrlDecisionManager;
import com.xxxx.server.config.jwt.JwtAuthenticationTokenFilter;
import com.xxxx.server.config.jwt.RestAuthorizationEntryPoint;
import com.xxxx.server.config.jwt.RestfulAccessDeniedHandler;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.service.IAdminService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.ObjectPostProcessor;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationM
anagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurer
Adapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.access.intercept.FilterSecurityInterceptor;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {
  @Autowired
  private IAdminService adminService;
  @Autowired
  private RestAuthorizationEntryPoint restAuthorizationEntryPoint;
  @Autowired
```

```
    private RestfulAccessDeniedHandler restfulAccessDeniedHandler;
    @Autowired
    private CustomUrlDecisionManager customUrlDecisionManager;
    @Autowired
    private CustomFilter customFilter;

    @Override
    public void configure(WebSecurity web) throws Exception {
      web.ignoring().antMatchers(
          "/login",
          "/logout",
          "/ws/**",
          "/css/**",
          "/js/**",
          "/index.html",
          "/img/**",
          "/fonts/**",
          "favicon.ico",
          "/doc.html",
          "/webjars/**",
          "/swagger-resources/**",
          "/v2/api-docs/**",
          "/captcha",
          "/ws/**"
      );
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
      //使用JWT，不需要csrf
      http.csrf()
          .disable()
          //基于token,不需要session
          .sessionManagement()
          .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
          .and()
          .authorizeRequests()
          //.antMatchers(HttpMethod.OPTIONS) //跨域请求先进行一次options请求
          //.permitAll()
          //需要认证
          .anyRequest()
          .authenticated()
          //动态权限配置
          .withObjectPostProcessor(new ObjectPostProcessor<FilterSecurityInterceptor>()
{
            @Override
            public <O extends FilterSecurityInterceptor> O postProcess(O object) {
              object.setAccessDecisionManager(customUrlDecisionManager);
              object.setSecurityMetadataSource(customFilter);
              return object;
            }
          })
          .and()
          .headers()
          //禁用缓存
          .cacheControl();
      //添加jwt登录授权过滤器或拦截器
```

```
    http.addFilterBefore(jwtAuthenticationTokenFilter(),
UsernamePasswordAuthenticationFilter.class);
    //添加自定义未授权和未登录结果返回
    http.exceptionHandling()
        .accessDeniedHandler(restfulAccessDeniedHandler)
        .authenticationEntryPoint(restAuthorizationEntryPoint);
  }

  /**
   * 设置执行自定义认证登录
   *
   * @param auth
   * @throws Exception
   */
  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService()).passwordEncoder(passwordEncoder());
  }

  /**
   * 重写 UserDetailsService
   *
   * @return
   * @throws Exception
   */
  @Override
  @Bean
  public UserDetailsService userDetailsService() {
    return username -> {
      Admin admin = adminService.getAdminByUserName(username);
      if (admin != null) {
        admin.setRoles(adminService.getRolesByAdminId(admin.getId()));
        return admin;
      }
      throw new UsernameNotFoundException("用户名和密码不正确!");
    };
  }

  @Bean
  public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
  }

  @Bean
  public JwtAuthenticationTokenFilter jwtAuthenticationTokenFilter() {
    return new JwtAuthenticationTokenFilter();
  }

}
```





43.个人中心

43.1 更新当前用户信息(★)

```
package com.xxxx.server.controller;

import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IAdminService;
import io.swagger.annotations.ApiOperation;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
```

```
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

/**
 * 个人中心
 *
 * @author lizongzai
 * @since 1.0.0
 */
@RestController
public class AdminInfoController {

  @Autowired
  private IAdminService adminService;

  @ApiOperation(value = "更新当前用户信息")
  @PutMapping("/admin/info")
  public RespBean updateAdminInfo(@RequestBody Admin admin, Authentication
authentication) {
    if (adminService.updateById(admin)) {
      //获取认证token
      UsernamePasswordAuthenticationToken authenticationToken = new
UsernamePasswordAuthenticationToken(admin, null, authentication.getAuthorities());
      //设置全局上下文
      SecurityContextHolder.getContext().setAuthentication(authenticationToken);
      return RespBean.success("更新成功!");
    }
    return RespBean.error("更新失败!");
  }
}
```

43.2 更新用户密码

43.2.1 AdminInfoController(★)

```
package com.xxxx.server.controller;

import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IAdminService;
import io.swagger.annotations.ApiOperation;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

/**
 * 个人中心
 *
```

```
 * @author lizongzai
 * @since 1.0.0
 */
@RestController
public class AdminInfoController {

  @Autowired
  private IAdminService adminService;

  @ApiOperation(value = "更用用户密码")
  @PutMapping("/admin/pass")
  public RespBean updateAdminPassword(@RequestBody Map<String,Object> info) {
    String oldPass = info.get("oldPass").toString();
    String pass = info.get("pass").toString();
    Integer adminId = (Integer) info.get("adminId");
    return adminService.updateAdminPassowrd(oldPass,pass,adminId);
  }
}
```

43.2.2 IAdminService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import java.util.List;
import javax.servlet.http.HttpServletRequest;

/**
 * <p>
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IAdminService extends IService<Admin> {
  /**
   * 更用用户密码
   * @param oldPass
   * @param pass
   * @param adminId
   * @return
   */
  RespBean updateAdminPassowrd(String oldPass, String pass, Integer adminId);
}
```

43.2.3 AdminServiceImpl(★)

```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.config.jwt.JwtTokenUtil;
import com.xxxx.server.mapper.AdminMapper;
import com.xxxx.server.mapper.AdminRoleMapper;
import com.xxxx.server.mapper.RoleMapper;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.AdminRole;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IAdminService;
import com.xxxx.server.utils.AdminUtils;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;

@Service
public class AdminServiceImpl extends ServiceImpl<AdminMapper, Admin> implements
IAdminService {

  @Autowired
  private UserDetailsService userDetailsService;
  @Autowired
  private PasswordEncoder passwordEncoder;
  @Autowired
  private JwtTokenUtil jwtTokenUtil;
  @Value("${jwt.tokenHead}")
  private String tokenHead;
  @Autowired
  private AdminMapper adminMapper;

  /**
   * 更用用户密码
   * @param oldPass
   * @param pass
   * @param adminId
   * @return
   */
  @Override
  public RespBean updateAdminPassowrd(String oldPass, String pass, Integer adminId) {
    Admin admin = adminMapper.selectById(adminId); //通过adminId查询当前需要用户对象
    BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
```

```
    //判断旧密码是否正确,若正确则更新新的密码
    if (encoder.matches(oldPass, admin.getPassword())) {
      admin.setPassword(encoder.encode(pass)); //设置新密码
      int result = adminMapper.updateById(admin);
      //判断密码是否更新成功
      if (result == 1) {
        return RespBean.success("更新成功!");
      }
    }
    return RespBean.error("更新失败!");
  }
}
```

43.2.4 CustomAuthorityDeserializer(★★)

```
package com.xxxx.server.config;

import com.fasterxml.jackson.core.JacksonException;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.DeserializationContext;
import com.fasterxml.jackson.databind.JsonDeserializer;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.io.IOException;
import java.util.Iterator;
import java.util.List;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;

/**
 * 自定义Authority反序列化
 * @author lizongzai
 * @since 1.0.0
 */
public class CustomAuthorityDeserializer extends JsonDeserializer {

  @Override
  public Object deserialize(JsonParser jsonParser, DeserializationContext
deserializationContext) throws IOException, JacksonException {
    ObjectMapper objectMapper = ((ObjectMapper) jsonParser.getCodec());
    JsonNode jsonNode = objectMapper.readTree(jsonParser);
    List<GrantedAuthority> grantedAuthorities =null;
    Iterator<JsonNode> elements = jsonNode.elements();
    while (elements.hasNext()) {
      JsonNode next = elements.next();
      JsonNode authority = next.get("authority");
      grantedAuthorities.add(new SimpleGrantedAuthority(authority.asText()));
    }
    return grantedAuthorities;
  }
}
```

```
{ "id": 1, "name": "系统管理员", "phone": "13812361398", "telephone": "71937538", "address": "香港特别行政区强县长寿柳州路p座123", "enabled": true, "username": "admin", "password": null, "userFace": "http://192.168.10.100:8888/group1/M00/00/00/wKgKZF6oHzuAXnw9AABaLsrkrQQ148.jpg", "remark": null, "roles": [ { "id": 6, "name": "ROLE_admin", "nameZh": "系统管理员" } ], "accountNonExpired": true, "credentialsNonExpired": true, "authorities": [ { "authority": "ROLE_admin" } ], "accountNonLocked": true}xml
```

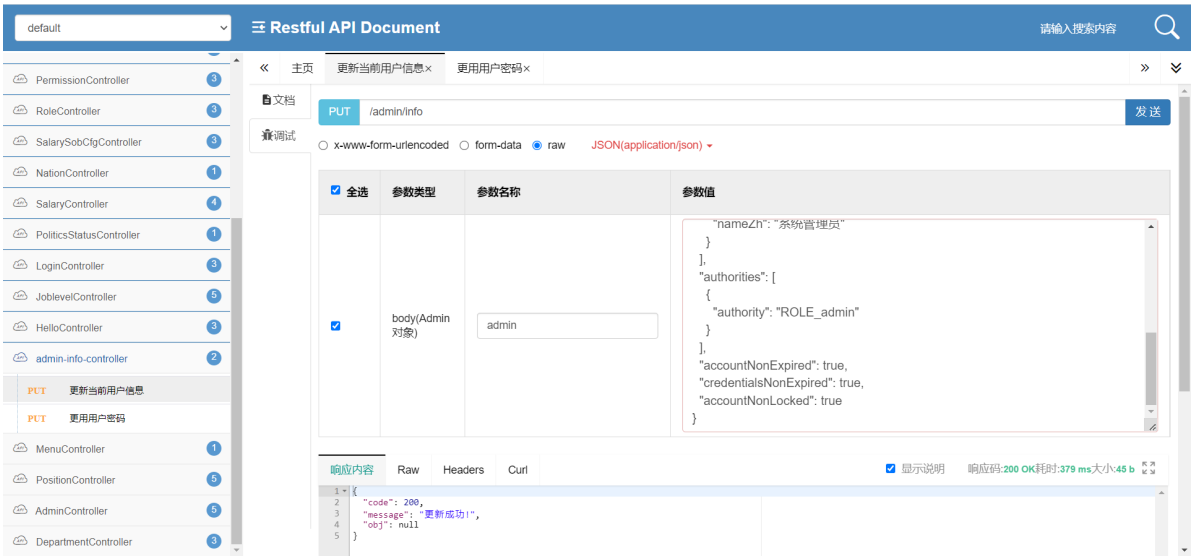

43.2.5 Admin(★)

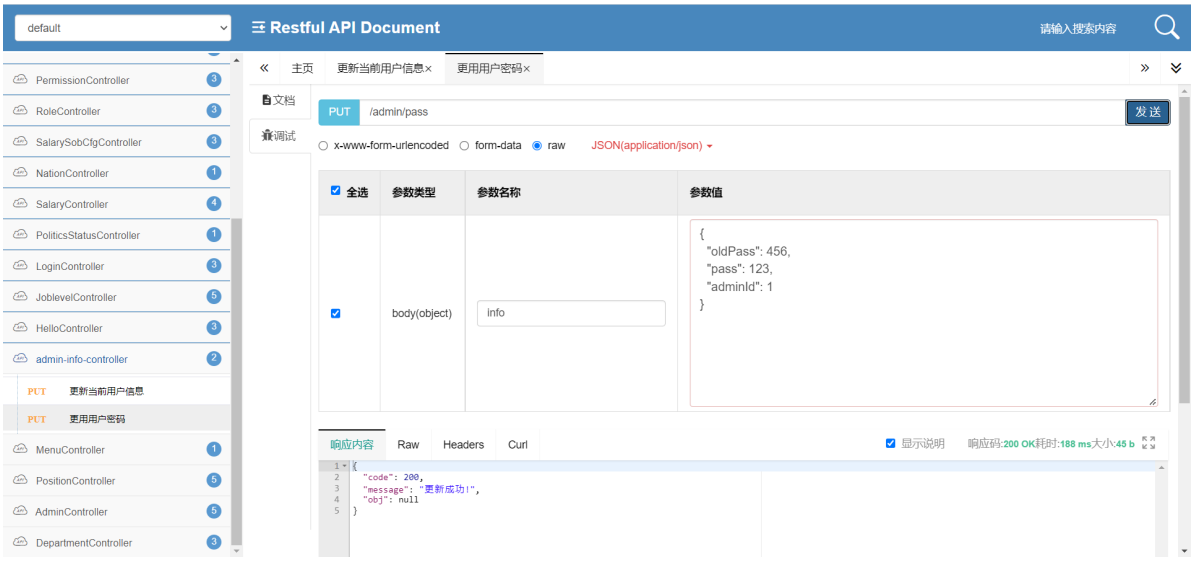
> ☐ 使用自定义Authority，引入到admin实体中 `@JsonDeserialize(using = CustomAuthorityDeserializer.class)`

```java
package com.xxxx.server.pojo;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.xxxx.server.config.CustomAuthorityDeserializer;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import java.io.Serializable;
import java.util.Collection;
import java.util.List;
import java.util.stream.Collectors;
import lombok.AccessLevel;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

/**
 * <p>
 *
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
@Data
@EqualsAndHashCode(callSuper = false)
@TableName("t_admin")
@ApiModel(value = "Admin对象", description = "")
public class Admin implements Serializable, UserDetails {

  private static final long serialVersionUID = 1L;

  @ApiModelProperty(value = "id")
  @TableId(value = "id", type = IdType.AUTO)
```

```
    private Integer id;

    @ApiModelProperty(value = "姓名")
    private String name;

    @ApiModelProperty(value = "手机号码")
    private String phone;

    @ApiModelProperty(value = "住宅电话")
    private String telephone;

    @ApiModelProperty(value = "联系地址")
    private String address;

    @ApiModelProperty(value = "是否启用")
    @Getter(AccessLevel.NONE)
    private Boolean enabled;

    @ApiModelProperty(value = "用户名")
    private String username;

    @ApiModelProperty(value = "密码")
    private String password;

    @ApiModelProperty(value = "用户头像")
    private String userFace;

    @ApiModelProperty(value = "备注")
    private String remark;

    @ApiModelProperty(value = "角色")
    @TableField(exist = false)
    private List<Role> roles;

    @Override
    @JsonDeserialize(using = CustomAuthorityDeserializer.class)
    public Collection<? extends GrantedAuthority> getAuthorities() {
      List<SimpleGrantedAuthority> authorities =
          roles.stream()
              .map(role -> new SimpleGrantedAuthority(role.getName()))
              .collect(Collectors.toList());
      return authorities;
    }

    @Override
    public boolean isAccountNonExpired() {
      return true;
    }

    @Override
    public boolean isAccountNonLocked() {
      return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
      return true;
    }
```

```
    @Override
    public boolean isEnabled() {
        return enabled;
    }
}
```





44. Docker 安装 fastdfs

https://www.cnblogs.com/javakhbd/p/13392090.html

44.1 先去docker hub寻找镜像文件

```
$ docker search fastdfs
```

44.2 拉取镜像文件

```
$ docker pull delron/fastdfs
```

44.3 通过Docker命令来创建Tracker服务

```
# 执行docker命令
docker run -it -d --name tracker --network=host -v /mydata/fastdfs/tracker:/var/fdfs
delron/fastdfs tracker
# 注意:tracker服务默认的端口为22122
```

44.4 通过Docker命令构建构建Storage服务

```
# 执行命令
docker run -d --name storage --network=host  -e TRACKER_SERVER=192.168.126.57:22122 -v
/mydata/fastdfs/storage:/var/fdfs -e GROUP_NAME=group1 delron/fastdfs storage
# 注意:其中TRACKER_SERVER中的ip要修改为你的Tracker服务所在的服务IP地址
```

默认情况下在Storage服务中是帮我们安装了Nginx服务的，相关的端口为：
服务 默认端口
tracker 22122
storage 23000
Nginx 8888

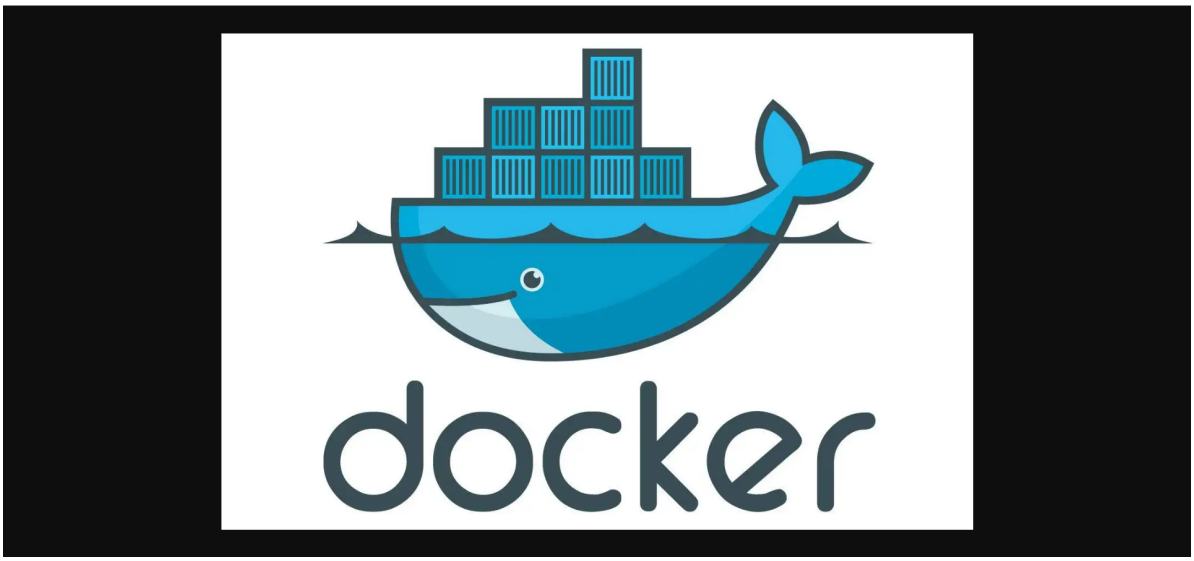
44.5 配置文件的查看&根据要求自行修改(比如端口冲突)

```
[root@VM-4-9-centos ~]# docker exec -it storage /bin/bash
[root@VM-4-9-centos nginx-1.12.2]# ls
CHANGES  CHANGES.ru  LICENSE  Makefile  README  auto  conf  configure  contrib  html
man  objs  src
[root@VM-4-9-centos nginx-1.12.2]# cd /
[root@VM-4-9-centos /]# ls
anaconda-post.log  bin  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run
sbin  srv  sys  tmp  usr  var
[root@VM-4-9-centos /]# cd etc/fdfs/
[root@VM-4-9-centos fdfs]# ls
client.conf         http.conf   mod_fastdfs.conf  storage.conf.sample
storage_ids.conf.sample  tracker.conf.sample
client.conf.sample  mime.types  storage.conf      storage_ids.conf     tracker.conf
[root@VM-4-9-centos fdfs]# cat storage.conf
```

44.6 验证图片服务器的服务

```
docker exec -it storage bash
ls
cd /var/fdfs
/usr/bin/fdfs_upload_file /etc/fdfs/client.conf docker.png
group1/M00/00/00/wKh-OWPGafGAdE91AABZ-siszsY418.png
```

http://192.168.126.57:8888/group1/M00/00/00/wKh-OWPGafGAdE91AABZ-siszsY418.png



45. 更新头像

45.1 引入FastDFS依赖

```
<!--FastDFS依赖-->
<!-- https://mvnrepository.com/artifact/org.csource/fastdfs-client-java -->
<dependency>
    <groupId>org.csource</groupId>
    <artifactId>fastdfs-client-java</artifactId>
    <version>1.27-RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/com.github.tobato/fastdfs-client -->
<dependency>
    <groupId>com.github.tobato</groupId>
    <artifactId>fastdfs-client</artifactId>
    <version>1.27.2</version>
</dependency>

<!--FastDFS依赖-->
    <!-- https://mvnrepository.com/artifact/com.roncoo/fastdfs-client-java -->
    <dependency>
      <groupId>com.roncoo</groupId>
      <artifactId>fastdfs-client-java</artifactId>
      <version>1.29</version>
    </dependency>
```

fdfs_client.conf配置

```
#连接超时
connect_timeout = 2
#网络超时
network_timeout = 30
#编码格式
charset = UTF-8
#tracker端口号
http.tracker_http_port = 8888
#防盗链功能
http.anti_steal_token = no
#秘钥
http.secret_key = FastDFS1234567890
#tracker ip：端口号
tracker_server = 192.168.126.57:22122
#连接池配置
connection_pool.enabled = true
connection_pool.max_count_per_entry = 500
connection_pool.max_idle_time = 3600
connection_pool.max_wait_time_in_ms = 1000
```

45.2 FastDFS工具类配置(★★★)

```
package com.xxxx.server.utils;

import com.roncoo.common.MyException;
import com.roncoo.fastdfs.ClientGlobal;
import com.roncoo.fastdfs.FileInfo;
import com.roncoo.fastdfs.StorageClient;
import com.roncoo.fastdfs.StorageServer;
import com.roncoo.fastdfs.TrackerClient;
import com.roncoo.fastdfs.TrackerServer;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.core.io.ClassPathResource;
import org.springframework.web.multipart.MultipartFile;

/**
 * FastDFS工具类
 *
 * @author lizongzai
 * @since 1.0.0
 */
public class FastDFSUtils {

  private static final Logger LOGGER = LoggerFactory.getLogger(FastDFSUtils.class);

  /**
   * 初始化客户端
   * ClientGlobal.init(filePath): 读取配置文件,并初始化对应的属性
   */
  static {
```

```
    try {
      //获取配置文件路径
      String filePath = new
ClassPathResource("fdfs_client.conf").getFile().getAbsolutePath();
      ClientGlobal.init(filePath);
    } catch (Exception e) {
      LOGGER.error("初始化FastDFS失败!", e);
    }
  }

  /**
   * 上传文件
   *
   * @param file
   * @return
   */
  public static String[] upload(MultipartFile file) {
    String filename = file.getOriginalFilename();
    LOGGER.info("文件名: " + filename);
    StorageClient storageClient = null;
    String[] uploadResults = null;
    try {
      //获取StorageClient客户端
      storageClient = getStorageClient();
      //上传文件: 第一个参数表示字节码,第二参数表示文件后缀名,第三个参数表示描述内容，但是这里为Null
      uploadResults =
storageClient.upload_file(file.getBytes(),filename.substring(filename.lastIndexOf(".")
+1),null);
    } catch (Exception e) {
      LOGGER.error("上传文件失败", e.getMessage());
    }
    //验证上传结果
    if (uploadResults == null && storageClient != null) {
      LOGGER.error("上传文件失败", storageClient.getErrorCode());
    }
    //上传成功返回groupName
    LOGGER.info("upload file successfully!!!" + "group_name:" + uploadResults[0] + ",
remoteFileName:" + " " + uploadResults[1]);

    return uploadResults;
  }

  /**
   * 下载
   *
   * @param groupName
   * @param remoteFileName
   * @return
   */
  public static InputStream downFile(String groupName, String remoteFileName) {
    try {
      StorageClient storageClient = getStorageClient();
      byte[] bytes = storageClient.download_file(groupName, remoteFileName);
      InputStream inputStream = new ByteArrayInputStream(bytes);
      return inputStream;
    } catch (Exception e) {
      LOGGER.error("文件下载失败", e.getMessage());
    }
```

```
    return null;

  }

  /**
   * 删除文件
   * @param groupName
   * @param remoteFileName
   * @throws Exception
   */
  public static void deleteFile(String groupName, String remoteFileName) {
    StorageClient storageClient = null;
    try {
      storageClient = getStorageClient();
      int i = storageClient.delete_file(groupName, remoteFileName);
    } catch (Exception e) {
      LOGGER.info("文件删除失败", e.getMessage());
    }
  }

  /**
   * 获取文件信息
   *
   * @param groupName
   * @return
   */
  public static FileInfo getFileInfo(String groupName, String remoteFileName) {
    StorageClient storageClient = null;
    try {
      storageClient = getStorageClient();
      FileInfo fileInfo = storageClient.get_file_info(groupName, remoteFileName);
      return fileInfo;
    } catch (Exception e) {
      LOGGER.error("文件信息获取失败", e.getMessage());
    }
    return null;
  }

  /**
   * 生成StorageClient客户端
   *
   * @return
   * @throws IOException
   */
  private static StorageClient getStorageClient() throws IOException {
    TrackerServer trackerServer = getTrackerServer();
    StorageClient storageClient = new StorageClient(trackerServer, null);
    return storageClient;
  }

  /**
   * 生成tracker服务器
   *
   * @return
   */
  private static TrackerServer getTrackerServer() throws IOException {
    TrackerClient trackerClient = new TrackerClient();
    TrackerServer trackerServer = trackerClient.getTrackerServer();
```

```
    return trackerServer;
  }

  /**
   * 获取文件路径
   * @return
   */
  public static String getTrackerUrl(){
    TrackerClient trackerClient = new TrackerClient();
    TrackerServer trackerServer = null;
    StorageServer storageServer = null;
    try {
      trackerServer = trackerClient.getTrackerServer();
      storageServer = trackerClient.getStoreStorage(trackerServer);
    } catch (Exception e) {
      LOGGER.error("文件路径获取失败", e.getMessage());
    }
    return "http://"+storageServer.getInetSocketAddress().getHostString() + ":8888/";
  }

}
```

45.3 Restful API's接口

```
package com.xxxx.server.controller;

import static com.xxxx.server.utils.FastDFSUtils.upload;

import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IAdminService;
import com.xxxx.server.utils.FastDFSUtils;
import io.swagger.annotations.ApiImplicitParam;
import io.swagger.annotations.ApiImplicitParams;
import io.swagger.annotations.ApiOperation;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

/**
 * 个人中心
 *
 * @author lizongzai
 * @since 1.0.0
 */
@RestController
public class AdminInfoController {

  @Autowired
```

```
    private IAdminService adminService;

    @ApiOperation(value = "更新当前用户信息")
    @PutMapping("/admin/info")
    public RespBean updateAdminInfo(@RequestBody Admin admin, Authentication
authentication) {
        if (adminService.updateById(admin)) {
            //获取认证token
            UsernamePasswordAuthenticationToken authenticationToken = new
UsernamePasswordAuthenticationToken(admin, null, authentication.getAuthorities());
            //设置全局上下文
            SecurityContextHolder.getContext().setAuthentication(authenticationToken);
            return RespBean.success("更新成功!");
        }
        return RespBean.error("更新失败!");
    }

    @ApiOperation(value = "更用用户密码")
    @PutMapping("/admin/pass")
    public RespBean updateAdminPassword(@RequestBody Map<String,Object> info) {
        String oldPass = info.get("oldPass").toString();
        String pass = info.get("pass").toString();
        Integer adminId = (Integer) info.get("adminId");
        return adminService.updateAdminPassowrd(oldPass,pass,adminId);
    }

    @ApiOperation(value = "更新用户头像")
    @ApiImplicitParams({@ApiImplicitParam(name = "file", value = "头像", dataType =
"MultipartFile")})
    @PutMapping("/admin/userface")
    public RespBean updateUserFace(MultipartFile file, Integer id, Authentication
authentication) {
        //获取文件上传地址
        String[] uploadPath = FastDFSUtils.upload(file);
        //System.out.println("groupName = " + uploadPath[0]);
        //System.out.println("remoteFileName = " + uploadPath[1]);
        String url = FastDFSUtils.getTrackerUrl() + uploadPath[0] + "/" + uploadPath[1];
        //System.out.println("url = " + url);
        return adminService.updateAdminUserFace(url,id,authentication);
    }
}
```

45.4 IAdminService

```
package com.xxxx.server.service;

import com.baomidou.mybatisplus.extension.service.IService;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import org.springframework.security.core.Authentication;

/**
 * <p>
```

```
 *  服务类
 * </p>
 *
 * @author lizongzai
 * @since 2023-01-04
 */
public interface IAdminService extends IService<Admin> {

  /**
   * 更用用户密码
   * @param oldPass
   * @param pass
   * @param adminId
   * @return
   */
  RespBean updateAdminPassowrd(String oldPass, String pass, Integer adminId);

  /**
   * 更新用户头像
   * @param url
   * @param id
   * @param authentication
   * @return
   */
  RespBean updateAdminUserFace(String url, Integer id, Authentication authentication);
}
```

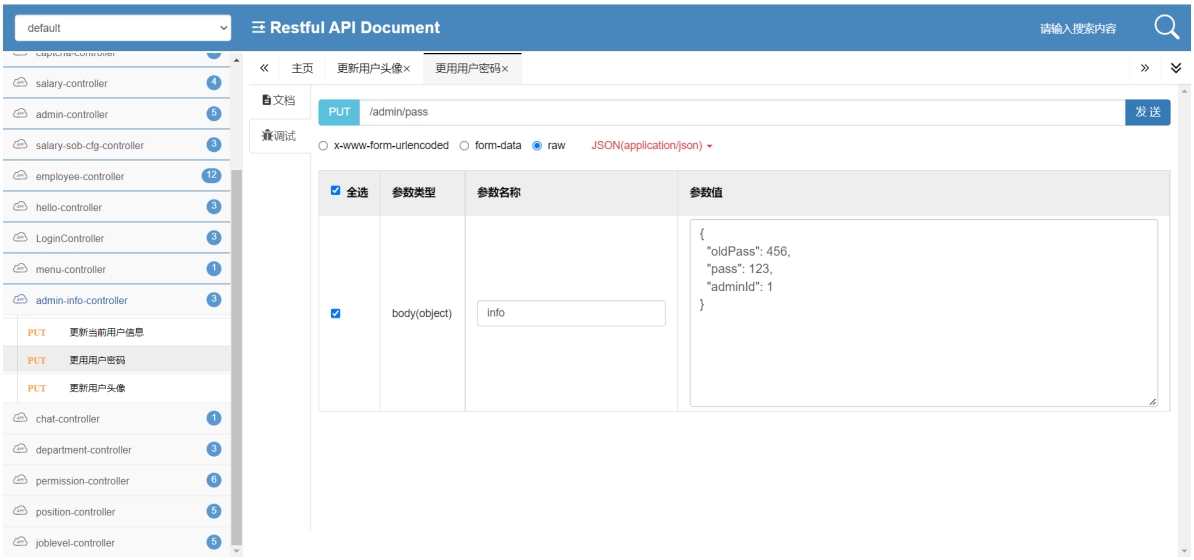
45.6 AdminServiceImpl(★)

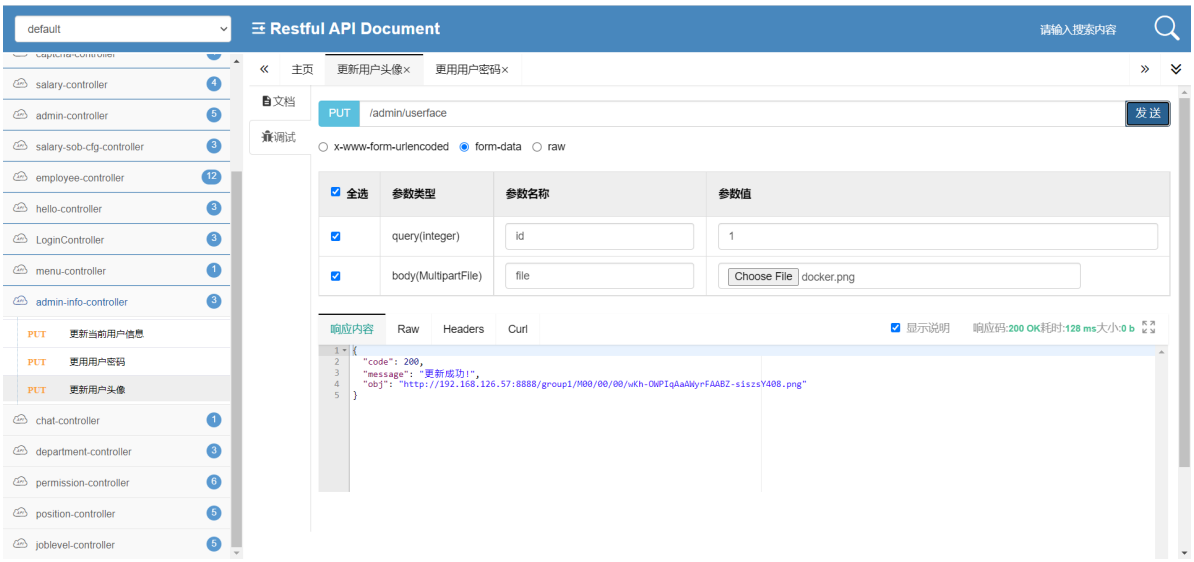
```
package com.xxxx.server.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.xxxx.server.config.jwt.JwtTokenUtil;
import com.xxxx.server.mapper.AdminMapper;
import com.xxxx.server.mapper.AdminRoleMapper;
import com.xxxx.server.mapper.RoleMapper;
import com.xxxx.server.pojo.Admin;
import com.xxxx.server.pojo.AdminRole;
import com.xxxx.server.pojo.Role;
import com.xxxx.server.pojo.common.RespBean;
import com.xxxx.server.service.IAdminService;
import com.xxxx.server.utils.AdminUtils;
import io.swagger.models.auth.In;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
```

```java
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;

@Service
public class AdminServiceImpl extends ServiceImpl<AdminMapper, Admin> implements
IAdminService {

  @Autowired
  private UserDetailsService userDetailsService;
  @Autowired
  private PasswordEncoder passwordEncoder;
  @Autowired
  private JwtTokenUtil jwtTokenUtil;
  @Value("${jwt.tokenHead}")
  private String tokenHead;
  @Autowired
  private AdminMapper adminMapper;
  @Autowired
  private RoleMapper roleMapper;
  @Autowired
  private AdminRoleMapper adminRoleMapper;

  /**
   * 更用用户密码
   * @param oldPass
   * @param pass
   * @param adminId
   * @return
   */
  @Override
  public RespBean updateAdminPassowrd(String oldPass, String pass, Integer adminId) {
    Admin admin = adminMapper.selectById(adminId); //通过adminId查询当前需要用户对象
    BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
//BCryptPasswordEncoder 加密算法解析
    if (encoder.matches(oldPass, admin.getPassword())) { //判断旧密码是否正确,若正确则更新新
的密码
      admin.setPassword(encoder.encode(pass)); //设置新密码
      int result = adminMapper.updateById(admin); //更新结果,并返回一条记录
      //判断密码是否更新成功
      if (result == 1) {
        return RespBean.success("更新成功!");
      }
    }
    return RespBean.error("更新失败!");
  }

  /**
   * 更新用户头像
   * @param url
   * @param id
   * @param authentication
   * @return
   */
  @Override
```

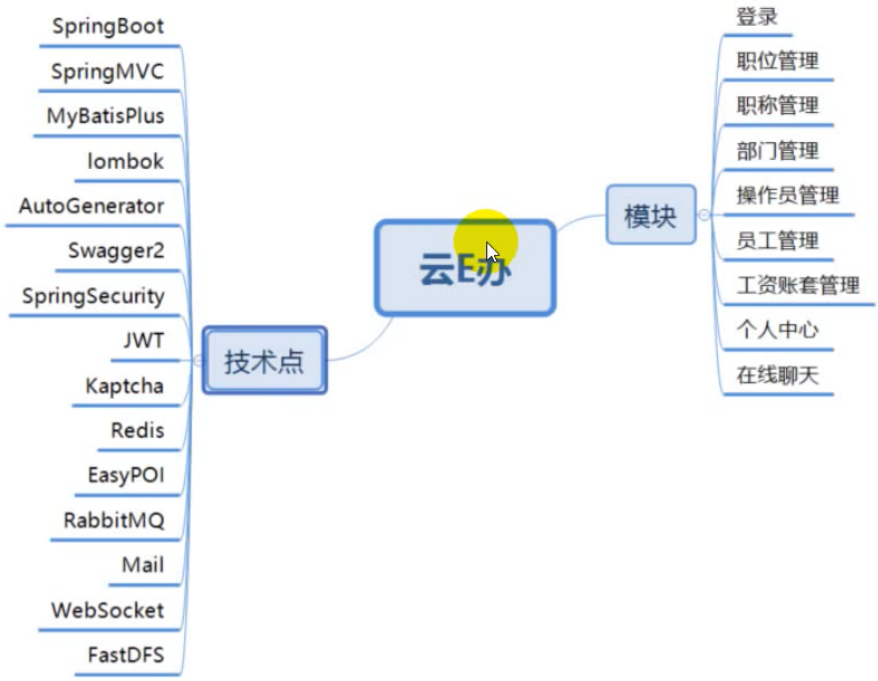
```
  public RespBean updateAdminUserFace(String url, Integer id, Authentication
authentication) {
    //获取用户对象
    Admin admin = adminMapper.selectById(id);
    //为用户设置头像
    admin.setUserFace(url);
    //更新头像
    int result = adminMapper.updateById(admin);
    if (result == 1) {
      //用户对象
      Admin principal = (Admin) authentication.getPrincipal();
      //为用户设置头像
      principal.setUserFace(url);
      //设置全局上下文
      SecurityContextHolder.getContext().setAuthentication(new
UsernamePasswordAuthenticationToken(admin, null, authentication.getAuthorities()));
      return RespBean.success("更新成功!",url);
    }
    return RespBean.error("更新失败!");
  }
}
```

45.7 Restful API's Test





项目总结



Java附件内容

https://blog.51cto.com/u_15127644/4564712

手册主要是将Java程序员按照年限来进行分层，清晰的标注着Java程序员应该按照怎样的路线来提升自己，需要去学习哪些技术点。

0-1年入门：

- Java基础复盘 （面向对象+Java的超类+Java的反射机制+异常处理+集合+泛型+基础IO操作+多线程+网络编程+JDK新特性）
- Web编程初探 （Servlet+MySQL数据库+商品管理系统实战）
- SSM从入门到精通 （Spring+SpringMVC+Mybatis+商品管理系统实战-SSM版）
- SpringBoot快速上手 （SpringBoot+基于SpringBoot的商品管理系统实战）
- 零距离互联网项目实战 （Linux+Redis+双十一秒杀实战系统）

1-3年高工：

- 并发编程进阶 （并发工具类实战+CAS+显示锁解析+线程池内部机制+性能优化）
- JVM深度剖析 （理解运行时数据区+堆外内存解读+JDK+内存泄漏问题排查+Arthas+GC算法和垃圾回收器+类加载机制）
- MySQL深度进阶
- 深入Tomcat底层 （线程模型+性能调优）

3-5年资深：

- 数据库（调优+事务+锁+集群+主从+缓存等）
- Linux（命令+生产环境+日志等）
- 中间件&分布式 （dubbo+MQ/kafka、ElasticSearch、SpringCloud等组件）

5-7年架构：

- 开源框架 （Spring5源码+SpringMVC源码+Mybatis源码）
- 分布式架构 （Zk实战+RabbitMQ+RocketMQ+Kafka）
- 高效存储 （Redis+mongoDB+MySQL高可用+Mycat+Sharing-Sphere）
- 微服务架构(RPC+SpringBoot+SpringCloud+Netflix+SpringCloudAlibaba+docker+k8s)

Mybatis-plus

学习参考: https://blog.csdn.net/m0_46313726/article/details/124187527

MyBatis-Plus多表联查的实现方法(动态查询和静态查询)

https://www.jb51.net/article/241979.htm