

## 目录

目录	1
导航	8
开始	8
系统运行及部署	8
框架功能	8
1、框架相关	8
2、基础功能	8
3、扩展功能	8
4、功能说明	8
5、前端文档	8
扩展环境搭建	8
常见问题	9
其他	9
项目简介	10
平台简介(公测)	10
软件架构图	11
贡献代码	11
其他	11
捐献作者	12
业务功能	12
更新日志	13
更新日志	13
v1.4.0 - 2022-12-01	13
重大更新	13
依赖升级	13
功能更新	13
新功能	14
问题修复	14
v1.3.0 - 2022-09-29	14
重大更新	14
依赖升级	15
功能更新	15
新功能	15
问题修复	15
v1.2.0 - 2022-08-09	16
重大更新	16
依赖升级	16
功能更新	16
新功能	16
问题修复	16
v1.1.0 - 2022-07-18	17
重大更新	17
依赖升级	17
功能更新	17
新功能	17
问题修复	17
v1.0.0 - 2022-06-20	18
新增/优化 工程模块	18
代码依赖改动	18

后续/进行中计划	18
系统演示	20
微服务版本与分布式集群版本 功能基本一致	20
专栏与视频	21
粉丝整理 欢迎投稿	21
快速开始	22
项目初始化	22
项目分支说明	22
项目必备环境	22
需勾选 maven 对应环境	22
默认 JDK1.8 如有变动 需更改以下配置	22
sql导入	23
框架 >= 1.3.0 可直接使用内置 ruoyi-naocs 服务	23
自建 Nacos 或 框架 < 1.3.0 配置 Nacos	24
更改 Nacos 自定义配置	26
idea环境配置	28
配置项目编码	28
配置运行看板	28
配置spring与docker看板	28
配置服务器SSH连接	29
配置服务器FTP连接	30
配置Docker连接	31
可操作远程docker与构建上传docker镜像(代替原来maven docker插件)	31
应用部署	33
版本 >= 1.3.0	33
请优先阅读 idea环境配置	33
手动部署	33
docker 后端部署	33
请优先阅读 idea环境配置	33
将配置使用FTP上传到根目录	33
给docker分配文件夹权限	33
构建应用镜像	34
创建基础服务	35
创建中心服务(需要先构建服务镜像)	35
创建业务服务(需要先构建服务镜像)	35
docker其他操作(idea的docker插件 推荐使用)	35
前端部署	36
更改后端代理路径或者后端ip地址	36
(旧)应用部署	37
手动部署	37
docker 后端部署	37
将源码内 docker 文件夹上传到服务器(注意: 不要放到根目录)	37
开放外网防火墙端口(内网服务无需开启)	37
放置挂载文件(切勿多次执行)	37
分配文件夹权限	37
启动基础服务	37
启动可视化服务	37
启动与停止业务服务(需要先构建服务镜像)	37
停止所有服务	37
删除所有容器	38

删除所有空版本镜像	38
构建服务镜像	38
前端部署	38
Seata服务搭建	40
版本 >= 0.12.0 源码集成 Seata 1.5.X 服务端	40
框架功能	41
项目结构	41
目录结构	41
创建新服务	43
最简单的方式	43
多团队开发	45
功能介绍	45
重点说明	45
分页功能	46
重点说明	46
代码用法	46
事务相关	48
多数据源	49
关于多数据源	49
用法参考demo模块	49
用法	49
更多用法	49
OSS功能	50
关于OSS模块使用	50
重点注意事项	50
代码使用	50
功能配置	50
配置OSS	50
切换OSS	54
扩展分类	55
上传图片或文件	55
列表展示	57
删除功能	60
下载功能	61
数据权限	63
关于数据权限	63
使用教程	63
数据权限功能:	63
数据权限相关代码	63
使用方式 参考demo模块	63
创建角色 test1 为 本部门及以下	63
创建角色 test2 为 仅本人	63
将其分配给用户 test	64
编写列表查询(注意: 数据权限注解只能在 Mapper 层使用)	64
编写数据权限模板	65
测试代码	65
自定义SQL模板	67
mybatis-plus 原生方法 增加数据权限过滤	69
支持类标注	69
防重幂等	71

功能介绍	71
美团GTIS系统流程图	71
使用方法	71
数据脱敏	73
功能说明	73
使用教程	73
脱敏逻辑修改	74
国际化	75
国际化方案	75
获取 code 对应国际化内容	75
使用 Validator 框架校验 controller 参数返回国际化	76
使用 Validator 框架校验 Bean 返回国际化	78
短信模块	81
配置功能	81
功能使用	81
重点须知	82
邮件功能	83
配置功能	83
功能使用	83
网关路由与放行	85
新增路由	85
放行使用方式	85
代码生成	87
功能介绍	87
数据源配置	87
导入数据表	87
编辑表生成结构	89
生成条件影响	90
树表配置	91
主子表说明	91
预览功能	92
代码结构同步	92
接口文档	93
版本 >= 1.2.0	93
说明	93
文档工具使用	93
Swagger升级SpringDoc指南	93
建议使用 Apifox	93
接入框架	93
(旧)接口文档	98
版本 <= 1.1.0	98
访问方式	98
前端访问 系统工具 -> 系统接口	98
文档配置	98
ruoyi-doc 文档服务配置说明	99
内网鉴权	101
功能介绍	101
开启/关闭内网鉴权	101
放行内网鉴权	101
修改包名	102

关于修改包名	102
主键使用说明	103
关于如何使用分布式id或雪花id	103
重点说明	103
修改应用路径	104
修改访问后端接口路径	104
修改前端页面访问路径	104
单元测试	106
参考文章	106
参考代码(1.4.0新增)	106
关于多表查询	107
建议单表查询	107
前端相关文档	109
扩展功能	110
ELK搭建	110
环境搭建	110
运行命令	110
参考文章	110
项目内配置	110
ES搜索引擎	111
环境搭建(如果已经搭建了ELK则跳过)	111
运行命令	111
Easy-ES 文档	111
用法	111
RabbitMQ搭建	112
环境搭建	112
用法参考	112
RocketMQ搭建	113
环境搭建	113
用法参考	113
Kafka搭建	114
环境搭建	114
用法参考	114
Nacos集群搭建	115
集群搭建两种方式	115
文件寻址集群	115
地址服务器寻址集群(推荐)	115
集群路由代理设置	115
SkyWalking搭建与集成	116
服务搭建	116
本地开发使用	116
docker部署使用	116
对接日志推送(不推荐 建议使用ELK收集日志)	117
Prometheus+Grafana搭建	118
基础搭建	118
框架内扩展	118
应用配置	118
导入框架特制模板	119
选择查看监控	120
常见问题	122

Lombok注解爆红	122
如何使用Tomcat	123
关于如何使用Tomcat	123
更改 ruoyi-common-web 模块依赖 使用 springboot 默认依赖 即为 tomcat	123
更改 application.yml 文件, 将 undertow 配置改为 tomcat 配置	123
如何使用druid连接池	125
为何移除druid	125
性能对比图	125
包大小对比图	125
为何使用hikari(中文: 光)	125
参考提交记录反向操作即可	125
导入excel实体类为空	126
如何同步项目更新	127
ParseException SQL解析异常	128
异常内容	128
异常由来	129
解决方案	129
swagger相关问题	131
空指针问题	131
参数不显示问题	131
实体bean为空问题	132
问题排查	132
原因	132
Redis 报错 Permission denied	133
此报错为无权限	133
关于RDB报错 /etc 无权限问题	133
关于HTTPS配置	134
后端 HTTPS 改造	134
监控中心 与 任务调度中心 改造	134
Minio https 改造	134
放行接口提示认证失败	135
可能的原因	135
解决方案	135
打包jar运行报错问题	136
打包jar运行报错问题	136
解决方案	136
如何指定dubbo注册ip	137
重点说明	137
在nacos指定协议IP地址(全局生效)	137
docker指定dubbo环境变量(单服务生效)	137
Sentinel页面404问题	138
原因	138
解决方案	138
无法读取nacos配置	139
检查 group 与 namespace 是否一致	139
检查是否手动改过 nacos 数据库数据	139
不支持ST请求	140
问题原因	140
解决方案	140
Only one connection receive subscriber allowed	141

问题原因	141
解决方案	141
nacos 报错 The Raft Group [naming_instance_metadata]	142
Nacos 服务下线报错问题	142
unable to read meta-data for class xxx	143
问题原因	143
解决方案	143
扩展项目	144
基于 RuoYi-Cloud-Plus 的扩展项目列表	144
精品PR 欢迎投稿	144
欢迎投稿 项目介绍+项目地址	144
加群方式	145
贡献代码	145
VIP群(提供 问题解答 技术支持 技术分享)(捐献100元提供订单号方可入群 开源不易 赚点饭钱)	145
关注作者	145
使用者登记	146
使用本开源项目的公司或者组织	146
捐献作者	147
捐献作者	147

# 导航

## 开始

- 项目简介、技术选型、软件架构、业务功能
- 项目结构
- 演示环境
- 视频与专栏
- 更新日志

## 系统运行及部署

- 项目初始化
- idea环境配置
- 应用部署
- (旧)应用部署
- Seata服务搭建

## 框架功能

### 1、框架相关

- 创建新服务
- 修改包名
- 接口文档
- (旧)接口文档
- 修改应用路径
- 系统国际化
- 多团队开发
- 内网鉴权

### 2、基础功能

- 代码生成
- 分页功能
- OSS功能
- 数据权限
- 网关路由与放行

### 3、扩展功能

- 多数据源
- 短信模块
- 邮件功能
- 防重幕等
- 数据脱敏

### 4、功能说明

- 事务相关
- 单元测试
- 主键使用说明
- 关于多表查询

### 5、前端文档

- 前端相关文档

## 扩展环境搭建

- ELK搭建
- ES搜索引擎
- RabbitMQ搭建
- RocketMQ搭建
- Kafka搭建
- Nacos集群搭建
- SkyWalking搭建与集成
- Prometheus+Grafana搭建

## 常见问题

- Lombok注解爆红
- 如何使用Tomcat
- 如何使用druid连接池
- 导入excel实体类为空
- 如何同步项目更新
- ParseException SQL解析异常
- swagger相关问题
- 实体bean为空问题
- Redis 报错 Permission denied
- 关于HTTPS配置
- 放行接口提示认证失败
- 打包jar运行报错
- 如何指定dubbo注册ip
- Sentinel页面404问题
- 无法读取nacos配置
- 不支持ST请求
- Only one connection receive subscriber allowed
- unable to read meta-data for class xxx

## 其他

- 扩展项目&精品PR
- 加群方式
- 使用者登记
- 捐献作者

# 项目简介

## 平台简介(公测)

1.2K Stars 48 License MIT IntelliJ IDEA 提供支持 RuoYi Cloud Plus 1.2.0 Spring Boot 2.7 JDK 8 JDK 11

RuoYi-Cloud-Plus 微服务通用权限管理系统 重写 RuoYi-Cloud 全方位升级(不兼容原框架)

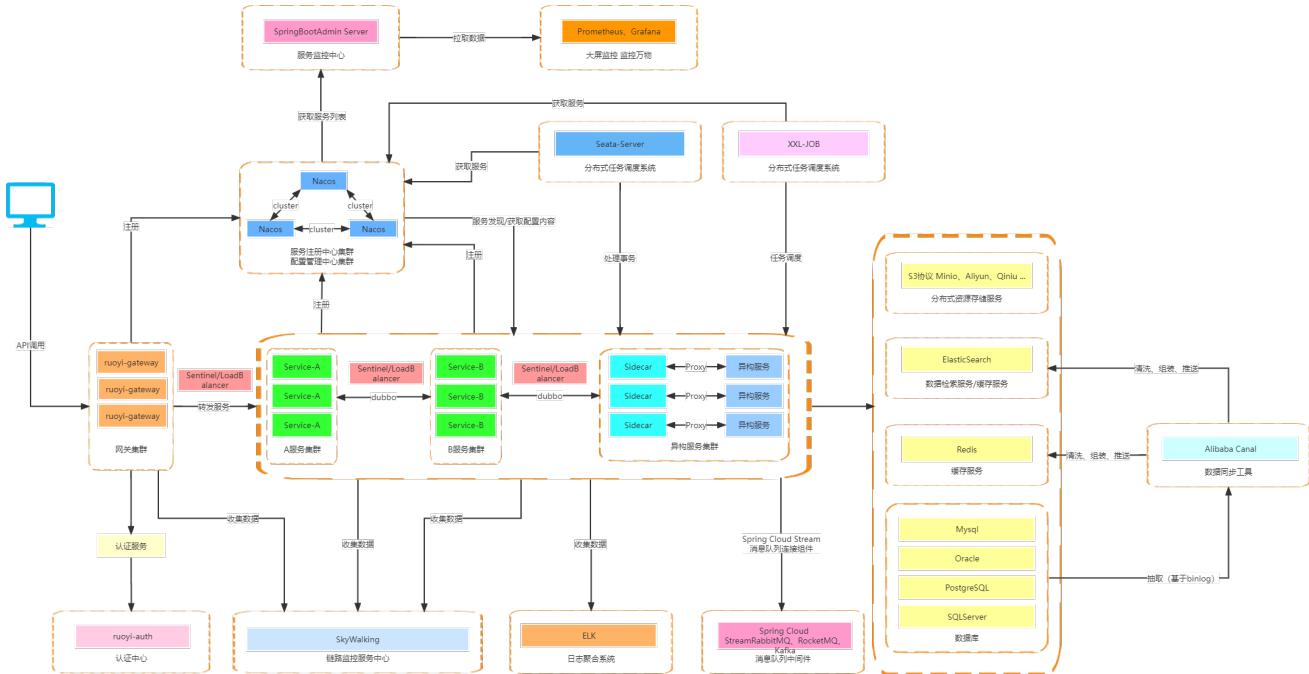
项目代码、文档 均开源免费可商用 遵循开源协议在项目中保留开源协议文件即可 活到老写到老 为兴趣而开源 为学习而开源 为让大家真正可以学到技术而开源

系统演示: [传送门](#) 分布式集群版本(功能一致)

功能介绍	使用技术	文档地址	特性注意事项
微服务权限管理系统	RuoYi-Cloud-Plus	<a href="#">RuoYi-Cloud-Plus官网</a>	重写 RuoYi-Cloud 全方位升级(不兼容原框架)
分布式集群分支	RuoYi-Vue-Plus	<a href="#">RuoYi-Vue-Plus官网</a>	重写 RuoYi-Vue (不兼容原框架)
Vue3分支	RuoYi-Cloud-Plus-UI	UI地址	由于组件还未完善 仅供学习
前端开发框架	Vue、Element UI	<a href="#">Element UI官网</a>	
后端开发框架	SpringBoot	<a href="#">SpringBoot官网</a>	
微服务开发框架	SpringCloud	<a href="#">SpringCloud官网</a>	
微服务开发框架	SpringCloudAlibaba	<a href="#">SpringCloudAlibaba官网</a>	
容器框架	Undertow	<a href="#">Undertow官网</a>	基于 XNIO 的高性能容器
权限认证框架	Sa-Token、Jwt	<a href="#">Sa-Token官网</a>	强解耦、强扩展
关系数据库	MySQL	<a href="#">MySQL官网</a>	适配 8.X 最低 5.7
关系数据库(未完成)	Oracle	<a href="#">Oracle官网</a>	适配 12c
关系数据库(未完成)	PostgreSQL	<a href="#">PostgreSQL官网</a>	适配 14
关系数据库(未完成)	SQLServer	<a href="#">SQLServer官网</a>	适配 2019
缓存数据库	Redis	<a href="#">Redis官网</a>	适配 6.X 最低 5.X
分布式注册中心	Alibaba Nacos	<a href="#">Alibaba Nacos文档</a>	采用2.X 基于GRPC通信高性能
分布式配置中心	Alibaba Nacos	<a href="#">Alibaba Nacos文档</a>	采用2.X 基于GRPC通信高性能
服务网关	SpringCloud Gateway	<a href="#">SpringCloud Gateway文档</a>	响应式高性能网关
负载均衡	SpringCloud Loadbalancer	<a href="#">SpringCloud Loadbalancer文档</a>	负载均衡处理
RPC远程调用	Apache Dubbo	<a href="#">Apache Dubbo官网</a>	原生态使用体验、高性能
分布式限流熔断	Alibaba Sentinel	<a href="#">Alibaba Sentinel文档</a>	无侵入、高扩展
分布式事务	Alibaba Seata	<a href="#">Alibaba Seata文档</a>	无侵入、高扩展 支持 四种模式
分布式消息队列	SpringCloud Stream	<a href="#">SpringCloud Stream文档</a>	门面框架兼容各种MQ集成
分布式消息队列	Apache Kafka	<a href="#">Apache Kafka文档</a>	高性能高速度
分布式消息队列	Apache RocketMQ	<a href="#">Apache RocketMQ文档</a>	高可用功能多样
分布式消息队列	RabbitMQ	<a href="#">RabbitMQ文档</a>	支持各种扩展插件功能多样性
分布式搜索引擎	ElasticSearch、Easy-Es	<a href="#">Easy-Es官网</a>	以 Mybatis-Plus 方式操作 ElasticSearch
分布式数据同步(未完成)	Alibaba Canal	<a href="#">Alibaba Canal官网</a>	采集数据同步各种数据库 ES Redis Mysql
分布式链路追踪	Apache SkyWalking	<a href="#">Apache SkyWalking文档</a>	链路追踪、网格分析、度量聚合、可视化
分布式日志中心	ELK	<a href="#">ElasticSearch官网</a>	ELK业界成熟解决方案
分布式锁	Lock4j	<a href="#">Lock4j官网</a>	注解锁、工具锁 多种多样
分布式幂等	Redisson	<a href="#">Lock4j文档</a>	拦截重复提交

分布式任务调度	Xxl-Job	Xxl-Job官网	高性能 高可靠 易扩展
分布式文件存储	Minio	Minio文档	本地存储
分布式云存储	七牛、阿里、腾讯	OSS使用文档	云存储
短信模块	阿里、腾讯	短信使用文档	短信发送
分布式监控	Prometheus、Grafana	Prometheus文档	全方位性能监控
服务监控	SpringBoot-Admin	SpringBoot-Admin文档	全方位服务监控
数据库框架	Mybatis-Plus	Mybatis-Plus文档	快速 CRUD 增加开发效率
数据库框架	P6spy	p6spy官网	更强劲的 SQL 分析
多数据源框架	Dynamic-Datasource	dynamic-ds文档	支持主从与多种类数据库异构
序列化框架	Jackson	Jackson官网	统一使用 jackson 高效可靠
Redis客户端	Redisson	Redisson文档	支持单机、集群配置
校验框架	Validation	Validation文档	增强接口安全性、严谨性 支持国际化
Excel框架	Alibaba EasyExcel	EasyExcel文档	性能优异 扩展性强
文档框架	SpringDoc、javadoc	接口文档	无注解零入侵基于java注释
工具类框架	Hutool、Lombok	Hutool文档	减少代码冗余 增加安全性
代码生成器	适配MP、Knife4j规范化代码	Hutool文档	一键生成前后端代码
部署方式	Docker	Docker文档	容器编排 一键部署业务集群
国际化	SpringMessage	SpringMVC文档	Spring标准国际化方案

## 软件架构图



## 贡献代码

欢迎小伙伴 PR 代码 请提交到 dev 开发分支 统一测试发版

## 其他

## 项目简介

- 同步升级 RuoYi-Cloud
- github 地址 [RuoYi-Cloud-Plus-github](#)
- 分离版分支 [RuoYi-Vue-Plus](#)
- 单模块 fast 分支 [RuoYi-Vue-Plus-fast](#)

## 捐献作者

作者为兼职做开源,平时还需要工作,如果帮到了您可以请作者吃个盒饭



## 业务功能

功能	介绍
用户管理	用户是系统操作者，该功能主要完成系统用户配置。
部门管理	配置系统组织机构（公司、部门、小组），树结构展现支持数据权限。
岗位管理	配置系统用户所属担任职务。
菜单管理	配置系统菜单，操作权限，按钮权限标识等。
角色管理	角色菜单权限分配、设置角色按机构进行数据范围权限划分。
字典管理	对系统中经常使用的一些较为固定的数据进行维护。
参数管理	对系统动态配置常用参数。
通知公告	系统通知公告信息发布维护。
操作日志	系统正常操作日志记录和查询；系统异常信息日志记录和查询。
登录日志	系统登录日志记录查询包含登录异常。
文件管理	系统文件上传、下载等管理。
定时任务	在线（添加、修改、删除）任务调度包含执行结果日志。
代码生成	前后端代码的生成（java、html、xml、sql）支持CRUD下载。
系统接口	根据业务代码自动生成相关的api接口文档。
服务监控	监视集群系统CPU、内存、磁盘、堆栈、在线日志、Spring相关配置等。
缓存监控	对系统的缓存信息查询，命令统计等。
在线构建器	拖动表单元素生成相应的HTML代码。
连接池监视	监视当前系统数据库连接池状态，可进行分析SQL找出系统性能瓶颈。
使用案例	系统的一些功能案例

## 更新日志

### 更新日志

v1.4.0 - 2022-12-01

#### 重大更新

- [重大更新] 新增 对接 skywalking 全功能(详细看下方新功能列表)
- [重大更新] 重构 ruoyi-nacos 使用官方依赖整合 解决一些问题 并升级 2.1.2 版本
- [重大更新] 新增 oss 私有库功能(数据库结构改动 需执行升级sql)
- [重大更新] 优化 数据源连接池从 druid 切换到 hikari(原因看文档)
- [重大更新] 新增 对接 prometheus + grafana 全功能(详细看下方新功能列表)

#### 依赖升级

- update springcloud 2021.0.4 => 2021.0.5
- update springboot 2.7.4 => 2.7.6
- update springboot-admin 2.7.5 => 2.7.7
- update springdoc 1.6.11 => 1.6.13
- update poi 5.2.2 => 5.2.3
- update hutool 5.8.6 => 5.8.10
- update aliyun-sms 2.0.18 => 2.0.22
- update tencent-sms 3.1.591 => 3.1.611
- update sa-token 1.30.0 => 1.33.0
- update redisson 3.17.6 => 3.18.0
- update easy-es 1.0.2 => 1.1.0
- update easyexcel 3.1.1 => 3.1.3
- update lock4j 2.2.2 => 2.2.3
- update s3-adk 1.12.300 => 1.12.349
- update sentinel 1.8.5 => 1.8.6
- update nacos 2.1.1 => 2.1.2
- update ELK 7.17.2 => 7.17.6 升级镜像版本
- update nginx 1.21.6 => 1.22.1 修复漏洞
- update mysql-docker 8.0.29 => 8.0.31

#### 功能更新

- update 优化 分页对象 PageQuery 支持多排序 适配 文件管理 页面支持多排序
- update 优化 获取用户信息getInfo接口 使用缓存数据获取
- update 优化 rpc文件上传 增加 ossId 数据返回
- update 优化 nacos 集群模式搭建 关于 nacos.home 注释说明
- update 优化 修改头像在小屏幕上页面布局错位的问题
- update 优化 oss 云厂商增加 华为obs关键字
- update 优化 重置时取消部门选中
- update 优化 新增返回警告消息提示
- update 优化 抽取 logback 通用配置 logback-common.xml 简化其他服务日志文件书写
- update 更改 nacos 配置文件目录 从dev文件夹迁移到nacos文件夹与其他配置区分
- update 优化 gateway 只缓存body
- update 优化 Dockerfile 创建目录命令简化操作
- update 优化 gateway filter顺序 与 代码工具封装
- update 优化 将空 catch 块形参重命名为 ignored
- update 优化 satoken 依赖传递
- update 优化 重写字典查询 使用本地缓存优化网络开销 提升到上级实现减少rpc调用频率 使用流处理减少字符串操作
- update 优化 减小腾讯短信引入jar包的体积

## 更新日志

- update 优化 简化一些方法的写法
- update 优化 消除Vue3控制台出现的警告信息
- update 优化 忽略不必要的属性数据返回
- update 优化 重构 mysql-jdbc 依赖到 mybatis 包内 替换为最新坐标

## 新功能

- add 新增 所有服务 docker 部署对接 skywalking
- add 新增 三大 mq 整合 skywalking
- add 新增 seata 整合 skywalking 手动编译 seata 插件包
- add 新增 ruoyi-common-skylog 整合 skywalking 日志推送
- add 增加 skywalking docker编排
- add 增加 ruoyi-seata-server redis模式配置
- add 新增 ruoyi-common-prometheus 模块 用于对接 prometheus 监控
- add 新增 docker prometheus + grafana 容器编排
- add 新增 ruoyi-monitor 监控服务 提供 prometheus http\_sd 服务发现功能
- add 新增 所有服务整合 ruoyi-common-prometheus 模块
- add 新增 grafana 监控大屏配置文件(框架定制)
- add 新增 使用 mica-metrics 为 undertow 提供健康检查
- add 新增 字典数据映射翻译注解
- add 增加 RedisUtils 获取缓存Map的key列表

## 问题修复

- fix 修复 开启账号同端互斥登录 被顶掉后登出报null异常问题
- fix 修复 设置NameMapper导致队列功能异常问题
- fix 修复 EnvironmentPostProcessor 不生效问题
- fix 修复 文件上传组件格式验证问题
- fix 修复 ruoyi-xxl-job-admin 服务健康检查配置缺失问题
- fix 修复 Excel导出字典值转换方法由于内部调用缓存不生效bug
- fix 修复 SysOss 方法内部调用导致缓存不生效 bug
- fix 修复 主题颜色在Drawer组件不会加载问题
- fix 修复 修改用户信息 校验用户名未排除当前用户问题
- fix 修复 升级 nginx 修复漏洞 <https://www.oschina.net/news/214309>
- fix 修复 用户编辑时角色和部门存在无法修改情况
- fix 修复 RemoteDictServiceImpl 代理对象获取异常bug
- fix 修复 菜单激活无法填充颜色 去除某些svg图标的fill属性
- fix 修复 使用透明底png图片时, 自动填充黑色背景
- fix 修复 table中更多按钮切换主题色未生效修复问题
- fix 修复 dubbo 使用 tri 协议 header 请求头变为小写导致无法获取参数问题
- fix 修复 DubboRequestFilter 优先级过高导致的 skywalking tid 取不到问题
- fix 修复 前端脚本乱码问题
- fix 修复 WebFluxUtils 读取空 body 报 null 问题
- fix 修复 Log注解GET请求记录不到参数问题
- fix 修复 某些特性的环境生成代码变乱码TXT文件问题
- fix 修复 开启TopNav没有子菜单隐藏侧边栏
- fix 修复 回显数据字典数组异常问题
- fix 修复 升级 satoken 导致白名单热更不生效问题
- fix 修复 swagger 版本与 springdoc 版本不一致导致找不到class问题
- fix 修复 grafana 监控模板绑定数据源ID 导致无法正常读取数据问题

v1.3.0 - 2022-09-29

## 重大更新

- [重大更新] 新增 ruoyi-nacos 源码集成 nacos 服务端控制台 支持单机/集群模式

## 更新日志

- [重大更新] 重写 spring-cache 实现更人性化的操作 支持注解指定ttl等一些参数
- [重大更新] 新增 Ruoyi-Cloud-Plus-UI 项目 Vue3 前端分支
- [重大更新] 移除maven docker插件 过于老旧功能缺陷大 使用idea自带的docker插件替代
- [重大更新] 优化 ruoyi-common-job 支持通过调度中心服务名注册 xxl-job-admin
- [重大更新] 新增 ruoyi-common-sentinel 模块 支持使用服务名注册 sentinel 控制台

## 依赖升级

- update spring-cloud 2021.0.3 => 2021.0.4
- update springboot 2.7.2 => 2.7.4
- update springboot-admin 2.7.3 => 2.7.5
- update sentinel 1.8.4 => 1.8.5 集成新 dubbo3 插件
- update springdoc 1.6.9 => 1.6.11
- update easy-es 0.9.80 => 1.0.2
- update dubbo 3.0.10 => 3.1.1
- update redisson 3.17.5 => 3.17.6
- update druid 1.2.11 => 1.2.12
- update hutool 5.8.5 => 5.8.6
- update dynamic-ds 3.5.1 => 3.5.2
- update aws-java-sdk-s3 1.12.264 => 1.12.300
- update aliyun-sms 2.0.16 => 2.0.18
- update tencent-sms 3.1.555 => 3.1.591
- update snakeyaml 1.30 => 1.32

## 功能更新

- update 优化 getLoginId 增加必要参数空校验
- update 优化 将 elasticsearch 解压后放入 避免造成用户误解
- update 优化 修改资料头像与部门被覆盖的问题
- update 优化 字典管理操作类型新增其他
- update 优化 使用 spring-cache 注解优化缓存
- update 优化 easy-es.enable=false 关闭 actuator 健康检查
- update 优化 优化多角色数据权限匹配规则
- update dubbo 升级 3.1.0 删除自行处理的源码修复 采用官方修复后的代码
- update 优化 页面内嵌iframe切换tab不刷新数据
- update 优化 调整 oss表key 与 ossconfig的服务 字段长度不匹配
- update 优化 操作日志密码脱敏
- update 优化 补全缺失的接口 更改更新日志链接
- update 优化 插入 SysOperLog 时，限制 operUrl 属性的长度
- update 优化 satoken 鉴权拦截器 优化多次校验

## 新功能

- add 增加 项目中使用到的请求头放行跨域
- add 新增 获取oss对象元数据方法
- add 新增 字典管理操作类型 其他

## 问题修复

- fix 修复 个人中心卡死或鼠标点击和键盘输入无效
- fix 修复 BaseMapperPlus 方法命令不一致问题
- fix 修复 图片预览组件src属性为null值控制台报错问题
- fix 修复 短信功能是否启用判断不生效
- fix 修复 web模块 不引入nacos依赖报错问题
- fix 修复 sentinel 构建无法读取webapp目录问题
- fix 修复 菜单管理遗漏的prop属性
- fix 修复 minio配置https遇到的问题

- fix 修复 点击删除后点击取消控制台报错问题
- fix 修复 文件/图片上传组件 第一次上传报错导致后续上传无限loading问题
- fix 修复 ruoyi-auth 服务与 elasticsearch 端口号冲突问题
- fix 修复 ruoyi-resource 服务与 elasticsearch 端口号冲突问题
- fix 修复 角色部门状态字典错误 与 菜单注释错误
- fix 修复 hutool 存在多版本问题
- fix 修复 openapi结构体 因springdoc缓存导致多次拼接接口路径问题
- fix 修复 oss配置删除内部数据id匹配类型问题
- fix 修复 没有权限的用户编辑部门缺少数据
- fix 修复 用户导入存在则更新不生效
- fix 修复 日志转换非json数据导致报错
- fix 修复 p6spy输出sql语句时间格式化不正确问题
- fix 修复 不同网段因reset请求头导致下载导出跨域问题
- fix 修复 在线用户设置永不过期 超时时间-1推送redis无效问题
- fix 修复 snakeyaml 1.31 依旧存在漏洞 升级 1.32

v1.2.0 - 2022-08-09

## 重大更新

- [重大更新] 新增 ruoyi-common-elasticsearch 模块 集成 easy-es 傻瓜式操作搜索引擎
- [重大更新] 新增 ruoyi-common-doc 整合 springdoc 基于 javadoc 实现无注解零入侵生成接口文档
- [不兼容更新] 移除 swagger 所属 ruoyi-doc ruoyi-common-swagger 两个模块 建议使用 ruoyi-common-doc 模块

## 依赖升级

- update springboot 2.6.9 => 2.7.2 重构使用最新自动配置方式
- update springboot-admin 2.6.7 => 2.7.3
- update dubbo 3.0.9 => 3.0.10
- update redisson 3.17.4 => 3.17.5
- update hutool 5.8.3 => 5.8.5
- update okhttp 4.9.1 => 4.10.0
- update aws-java-sdk-s3 1.12.248 => 1.12.264 修复依赖安全漏洞
- update aliyun.sms 2.0.9 => 2.0.16
- update tencent.sms 3.1.537 => 3.1.555
- update guava 30.0-jre => 31.1-jre

## 功能更新

- update 修改 资源服务 不提供默认短信 sdk 依赖
- update 优化表格上右侧工具条 (搜索按钮显隐&右侧样式凸出)
- update 优化 前后端多环境部署保持一致 删除无用环境文件
- update 优化 错误登录锁定与新增解锁功能
- update 优化字典数据使用store存取
- update 优化布局设置使用el-drawer抽屉显示
- update 更新框架文档 专栏与视频 链接地址
- update 优化 对象上传 主动设置文件公共读 解决天翼云OSS文件私有问题
- update 优化 网关验证码过滤器 路径匹配改为严格匹配
- update 优化 数据导致权限生成 SQL 重复问题

## 新功能

- add 增加 全局跨域过滤器 处理跨域请求 适配移动端访问
- add 增加 搜索引擎 crud 演示案例

## 问题修复

- fix 防止date-picker组件报错，降级element-ui版本

- fix 修复 RedisUtils 并发 set ttl 错误问题
- fix 防止vue3主键字段名与row或ids一致导致报错的问题
- fix 修复 幂等组件 逻辑问题导致线程变量未清除
- fix 修复 图片回显查询 路径错误问题
- fix 修复 脱敏没有实现类导致返回数据异常问题
- fix 修复 xxljob 错误导入配置文件引发的问题
- fix 修复 gateway模块 dockerfile 端口编写错误
- fix 修复用户导出字典使用错误
- fix 修复 demo 模块 远程调用失败问题
- fix 修复 sentinel 控制台未适配 springboot 2.6 新路由策略导致无法登录问题

v1.1.0 - 2022-07-18

## 重大更新

- [重大更新] 新增 ELK 分布式日志中心整合
- [重大更新] 新增 ruoyi-stream-mq 演示模块 完成 RabbitMQ RocketMQ Kafka 整合
- [重大更新] 优化 docker 部署方式 使用 host 模式简化部署流程 降低使用成本
- [重大更新] 调整 dubbo 服务注册命名空间与 cloud 服务保持一致 通过注册组区分访问服务
- [安全性] 优化 nginx 限制外网访问内网 actuator 相关路径 建议升级

## 依赖升级

- update springboot 2.6.8 => 2.6.9
- update easyexcel 3.1.0 => 3.1.1
- update hutool 5.8.2 => 5.8.3
- update redisson 3.17.2 => 3.17.4
- update aws-java-sdk-s3 1.12.215 => 1.12.248
- update tencentcloud-sdk-java 3.1.500 => 3.1.537
- update dubbo 3.0.8 => 3.0.9
- update seata 1.5.1 => 1.5.2

## 功能更新

- update 增加 redisson key 前缀配置
- update 优化 DateColumn 支持单模板多key场景
- update 优化部署脚本 增加 elk kafka rabbitmq rocketmq 等配置
- update 修改 oss 客户端自定义域名 统一使用https开关控制协议头
- update 优化 使用 StreamUtils 简化业务流操纵
- update 优化 ruoyi-demo 模块 去除用不上的 seata 依赖
- update 优化 接口文档 接口地址与服务地址不匹配问题
- update 优化字典数据回显样式下拉框显示值
- update 默认不启用压缩文件缓存防止node\_modules过大
- update 优化登出方法

## 新功能

- add 增加 rocketmq docker编排
- add 新增 rabbitmq docker编排 包含延迟插件
- add 新增 kafka docker编排
- add 增加 es ik 分词器插件集成
- add 增加 StreamUtils 流工具 简化 stream 流操纵

## 问题修复

- fix 修复 获取 SensitiveService 空问题 增加空兼容
- fix 修复 演示页面导出路径错误
- fix 修复 minio 上传自定义域名回显路径错误问题

- fix 修复 hutool 工具返回不可操纵类型 导致报错问题
- fix 修复 远程调用短信功能返回实体 SysSms 序列化报错问题
- fix 修复 复制过程错误 导致演示excel文件损坏问题
- fix 修复 dubbo 注册组不生效问题 通过覆盖源码方式
- fix 修复代码生成首字母大写问题

v1.0.0 - 2022-06-20

### 新增/优化 工程模块

- add 新增 ruoyi-common-alibaba-bom 工程管理 alibaba 相关依赖
- add 新增 ruoyi-common-bom 工程管理 ruoyi-common 相关依赖
- add 新增 ruoyi-api-bom 工程管理 ruoyi-api 依赖项
- add 新增 ruoyi-api-resource 模块 规范用法 移除 ruoyi-file 模块
- add 新增 ruoyi-common-web 模块 使用 undertow 替换 tomcat
- add 新增 ruoyi-common-dubbo 整合 dubbo 3.X 实现高性能 rpc 远程调用 替换 feign
- add 新增 ruoyi-common-dict 实现字典多服务调用
- add 新增 ruoyi-common-loadbalancer 自定义负载均衡模块 用于多团队开发
- add 新增 ruoyi-common-excel 模块 集成 Alibaba EasyExcel 替换 自带excel实现
- add 新增 ruoyi-common-oss 模块 支持 AWS S3 协议 分布式文件存储
- add 新增 ruoyi-common-mail 邮件模块
- add 新增 ruoyi-common-sms 短信模块 整合 阿里云、腾讯云 短信功能
- add 新增 ruoyi-common-idempotent 分布式幂等模块
- add 新增 ruoyi-common-satoken 整合 sa-token 重写所有权限
- add 新增 ruoyi-xxl-job-admin 整合 xxljob 替换 quartz 支持分布式任务调度
- add 新增 ruoyi-job 模块 统一远程处理任务 规范用法
- add 新增 ruoyi-doc 模块 集成 Knife4j 替换 swagger
- add 新增 ruoyi-seata-server 源码集成 Seata 1.5.X 服务端
- add 新增 ruoyi-sentinel-dashboard 模块 源码集成 sentinel 控制台
- update 抽取所有公用配置到 maven profile 管理

### 代码依赖改动

- update SpringCloud 2021.0.3
- update 适配 SpringCloudAlibaba 2021.0.1.0 全新配置方式
- update poi 4.1.2 => 5.2.2 性能大幅提升
- update 重构 整合 jackson 替换 fastjson
- update 重构 整合 redisson 客户端
- update 重构 整合 mybatis-plus
- update 重写 数据权限实现 基于 mybatis-plus
- add 增加 lombok 优化原生代码
- add 整合 hutool 优化相关代码
- add 新增 国际化 功能
- add 新增 lock4j 分布式锁
- add 增加监控中心 在线日志监控 优化日志文件格式
- add 适配 docker 部署方式

### 后续/进行中计划

- 增加 Vue3 前端工程
- 应用模块 适配 Oracle、PostgreSQL、SQLServer
- 增加 SpringCloud Stream 支持
- 适配 Apache Kafka、Apache RocketMQ、RabbitMQ
- 适配 ElasticSearch 分布式搜索引擎
- 适配 Alibaba Canal 分布式数据同步中心
- 适配 Apache SkyWalking 分布式链路追踪监控中心

- 适配 ELK 分布式日志中心
- 适配 Prometheus、Grafana 分布式全方位数据大屏监控

## 系统演示

微服务版本 与 分布式集群版本 功能基本一致

分布式集群版本 演示环境 [传送门](#)

# 专栏与视频

粉丝整理 欢迎投稿

作者	文档地址	说明
程序猿一枚_	<a href="https://www.yuque.com/xiaoqiang-qgsmv/rfgglm">https://www.yuque.com/xiaoqiang-qgsmv/rfgglm</a>	玩转RuoYi-Cloud-Plus
程序猿一枚_	<a href="https://www.bilibili.com/medialist/play/400188320?business=space_series&amp;business_id=2667665">https://www.bilibili.com/medialist/play/400188320?business=space_series&amp;business_id=2667665</a>	环境搭建以及进阶开发
抓蛙师	<a href="https://www.bilibili.com/video/BV1TG41157Ef/">https://www.bilibili.com/video/BV1TG41157Ef/</a>	学会问问题(小白必看)
MichelleChung	<a href="https://blog.csdn.net/michelle_zhong/category_12058476.html">https://blog.csdn.net/michelle_zhong/category_12058476.html</a>	Cloud源码解析专栏
MichelleChung	<a href="https://blog.csdn.net/michelle_zhong/category_11109741.html">https://blog.csdn.net/michelle_zhong/category_11109741.html</a>	源码解析专栏
抓蛙师	<a href="https://www.bilibili.com/video/BV1mr4y1j75M">https://www.bilibili.com/video/BV1mr4y1j75M</a>	框架基础视频专栏(新人必看)
抓蛙师	<a href="https://www.bilibili.com/video/BV1Na411u7eC">https://www.bilibili.com/video/BV1Na411u7eC</a>	框架改造视频专栏(新人必看)
抓蛙师	<a href="https://www.bilibili.com/video/BV1te4y1D7hi">https://www.bilibili.com/video/BV1te4y1D7hi</a>	小程序鉴权与uniapp联动
抓蛙师	<a href="https://www.bilibili.com/video/BV1zt4y137UP">https://www.bilibili.com/video/BV1zt4y137UP</a>	公众号集成
mayuanfei	<a href="https://note.youdao.com/s/XpvKnxAb">https://note.youdao.com/s/XpvKnxAb</a>	入门专栏(新人必看)

# 快速开始

## 项目初始化

### 项目分支说明

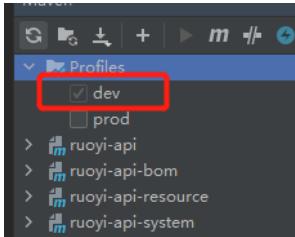
master 主分支 稳定发布分支 dev 开发分支 代码随时更新 不推荐使用 经测试后会发布到主分支 future/\* 新功能预览分支

### 项目必备环境

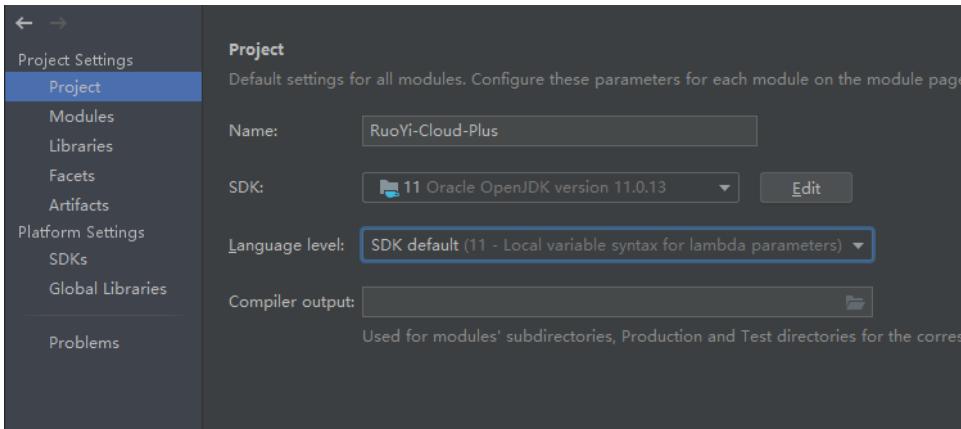
推荐使用 docker 安装 项目内置 docker 编排文件

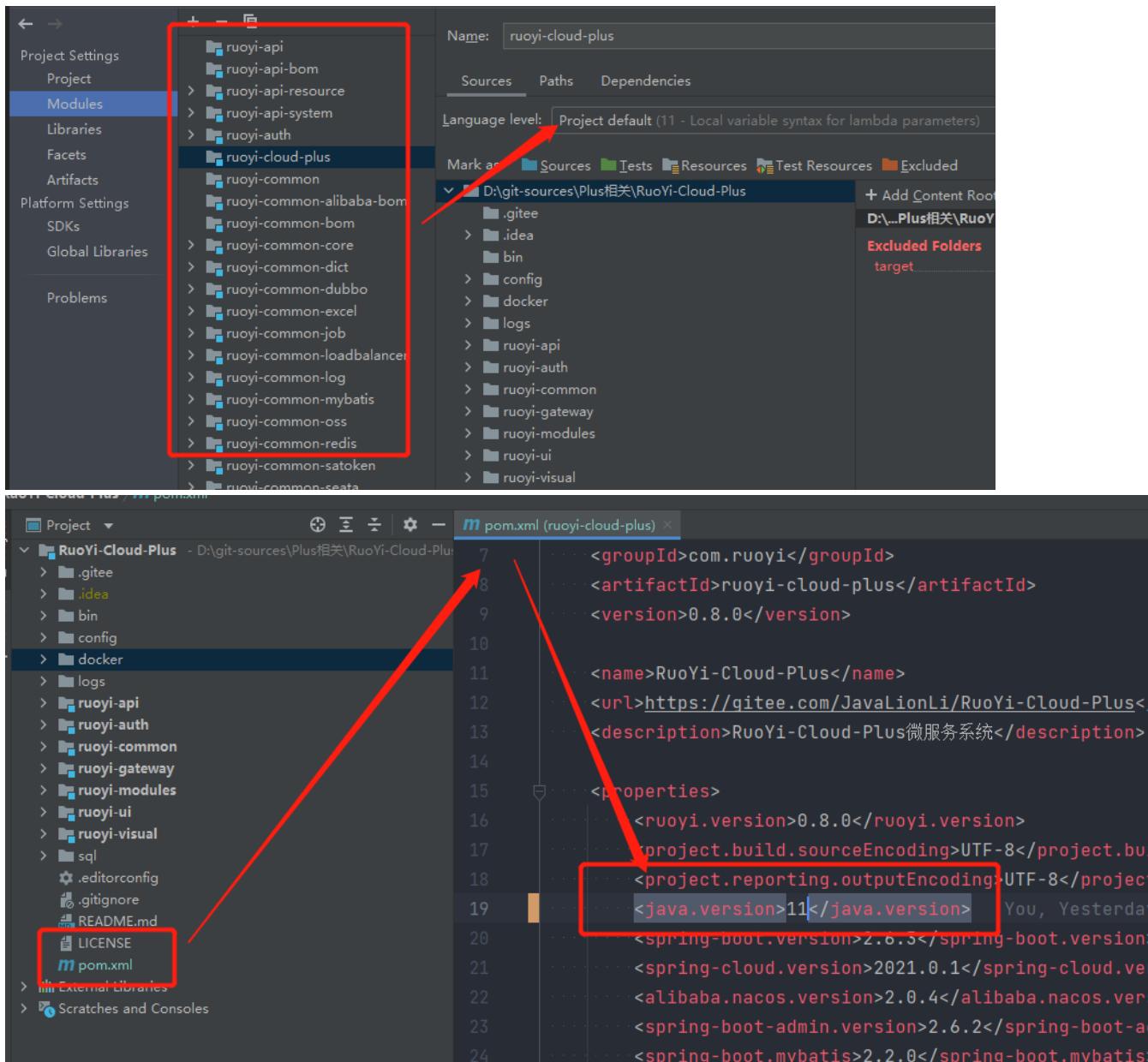
- oracle jdk 8.11 (暂时不支持 17 不支持大于 jdk8\_202 因为202是最后一个免费版本)
- mysql 5.7.8.0 (5.6未适配可能会有问题)
- redis 5.X 6.X 由于框架大量使用了redis特性 版本必须 >= 5.X ([win redis 下载地址](#))
- minio 本地文件存储 或 阿里云 腾讯云 七牛云等一切支持S3协议的云存储
- maven 3.6.3 3.8.X
- nodejs >= 12
- npm 6.X 8.X (7.X确认有问题)
- nacos >= 2.X(框架1.3.0内置nacos)
- sentinel 框架内置
- seata 框架内置

需勾选 maven 对应环境



默认 **JDK1.8** 如有变动 需更改以下配置





sql导入



将sql导入到与sql文件名对应的数据表(不要放到一个库下)

框架 >= 1.3.0 可直接使用内置 **ruoyi-naoCS** 服务

## 更改 ruoyi-nacos 数据库地址

```

30 # nacos.inetutils.ip-address=
31
32 spring.application.name=ruoyi-nacos
33 #***** Config Module Related Configurations *****
34 ##### If use MySQL as datasource:
35 spring.datasource.platform=mysql
36
37 ##### Count of DB:
38 db.num=1
39
40 ##### Connect URL of DB:
41 db.url.0=jdbc:mysql://127.0.0.1:3306/ry-config?characterEncoding=utf8&serverTimezone=UTC
42 db.user.0=root
43 db.password.0=root
44 db.pool.config.connectionTimeout=30000
45 db.pool.config.validationTimeout=10000
46 db.pool.config.maximumPoolSize=20
47 db.pool.config.minimumIdle=2
48

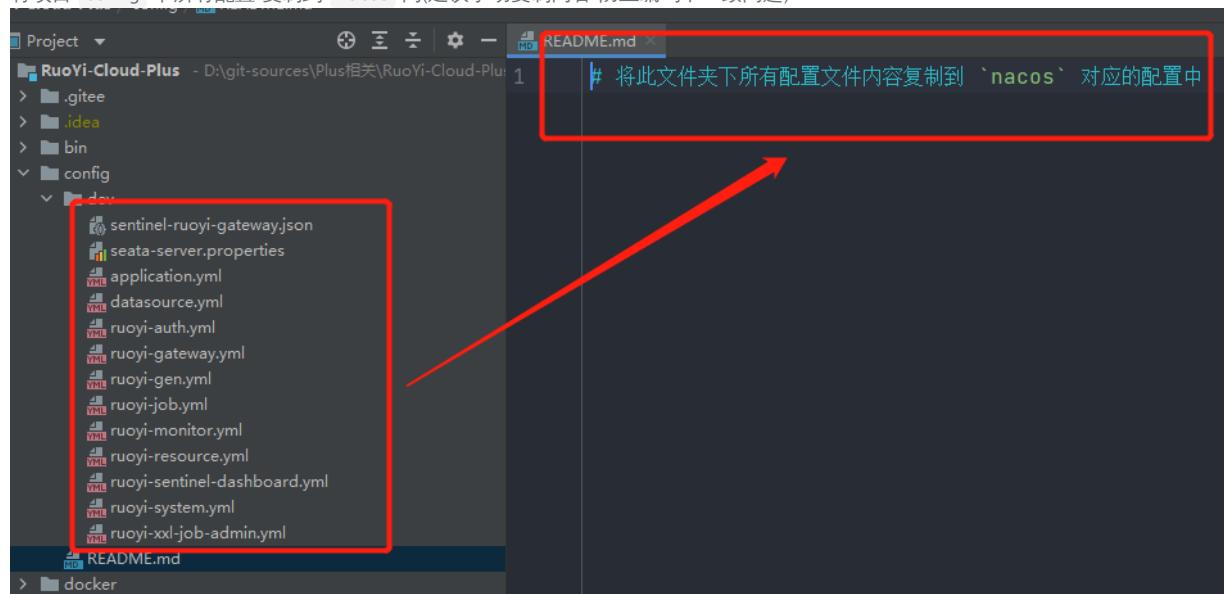
```

其余流程同下方步骤一致

### 自建 Nacos 或 框架 < 1.3.0 配置 Nacos

Nacos 数据库指向 ry-config 数据库(此处重点: 此数据库为定制数据 未使用此库会无法读取配置)

将项目 config 下所有配置 复制到 nacos 内(建议手动复制内容 防止编码不一致问题)



NACOS 2.0.3

public | dev | prod

配置管理 | dev dev 查询结果: 共查询到 13 条满足要求的配置。

配置列表 Data ID: 添加通配符'\*'进行模糊查询 Group: 添加通配符'\*'进行模糊查询

历史版本

	Data Id ↓	Group ↓
<input type="checkbox"/>	application.yml	DEFAULT_GROUP
<input type="checkbox"/>	datasource.yml	DEFAULT_GROUP
<input type="checkbox"/>	ruoyi-gateway.yml	DEFAULT_GROUP
<input type="checkbox"/>	ruoyi-auth.yml	DEFAULT_GROUP
<input type="checkbox"/>	ruoyi-monitor.yml	DEFAULT_GROUP
<input type="checkbox"/>	ruoyi-system.yml	DEFAULT_GROUP
<input type="checkbox"/>	ruoyi-gen.yml	DEFAULT_GROUP
<input type="checkbox"/>	ruoyi-job.yml	DEFAULT_GROUP

监听查询

任务管理

权限控制

命名空间

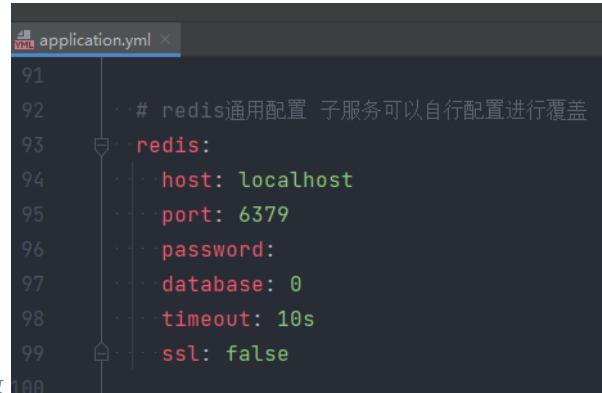
集群管理

更改 主pom文件 对应环境的 nacos 地址

```
<profiles>
    <profile>
        <id>dev</id>
        <properties>
            <!-- 环境标识, 需要与配置文件的名称相对应 -->
            <profiles.active>dev</profiles.active>
            <nacos.server>127.0.0.1:8848</nacos.server>
            <nacos.discovery.group>DEFAULT_GROUP</nacos.discovery.group>
            <nacos.config.group>DEFAULT_GROUP</nacos.config.group>
        </properties>
        <activation>
            <!-- 默认环境 -->
            <activeByDefault>true</activeByDefault>
        </activation>
    </profile>
    <profile>
        <id>prod</id>
        <properties>
            <profiles.active>prod</profiles.active>
            <nacos.server>127.0.0.1:8848</nacos.server>
            <nacos.discovery.group>DEFAULT_GROUP</nacos.discovery.group>
            <nacos.config.group>DEFAULT_GROUP</nacos.config.group>
        </properties>
    </profile>
</profiles>
```

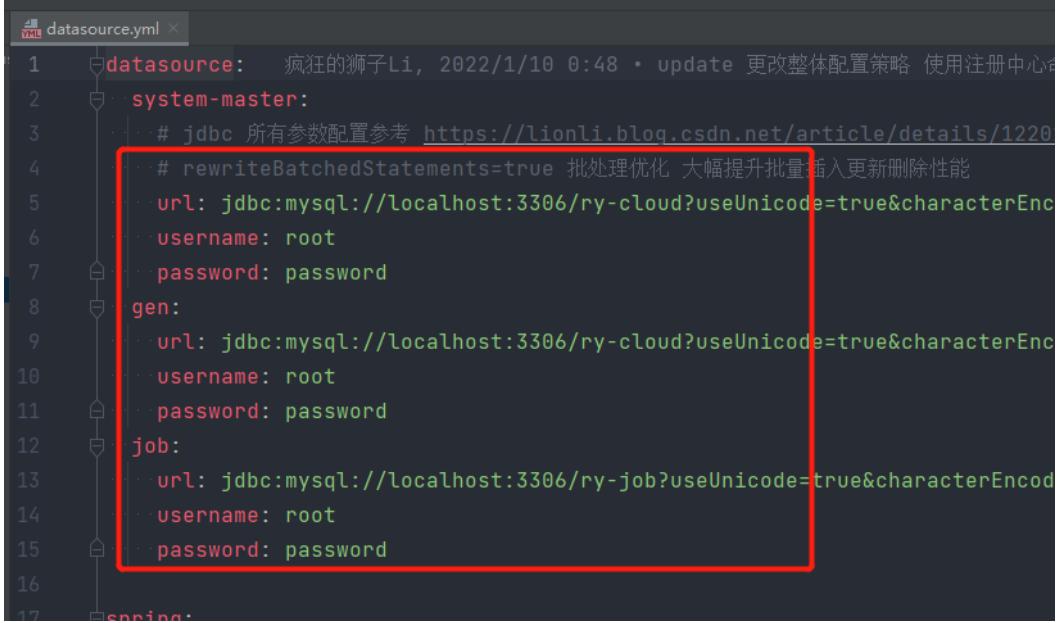
## 更改 Nacos 自定义配置

忠告: 微服务配置相当复杂 请勿在不懂原理的情况下乱改



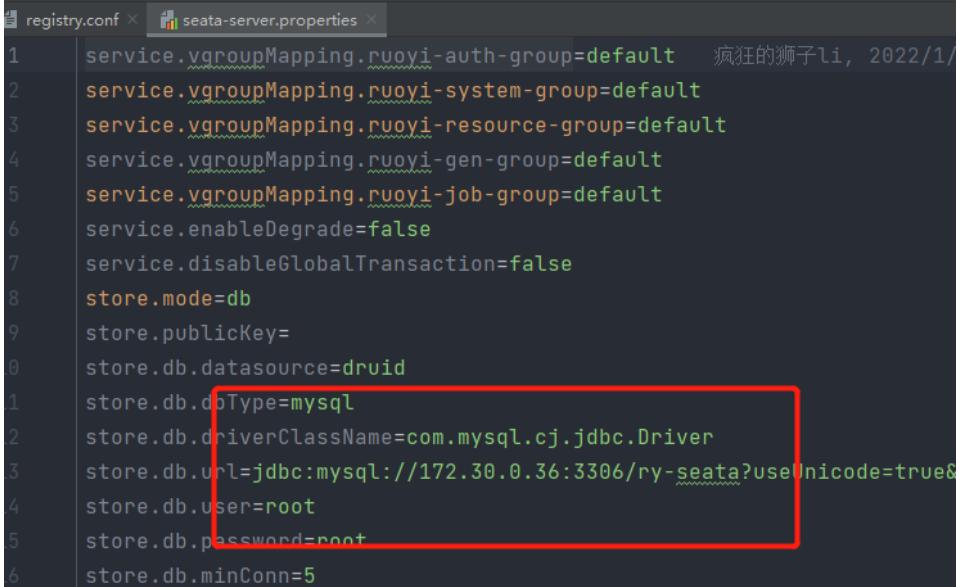
The screenshot shows a code editor window with the file 'application.yml' open. The content of the file is as follows:

```
91
92     # redis通用配置 子服务可以自行配置进行覆盖
93     redis:
94         host: localhost
95         port: 6379
96         password:
97         database: 0
98         timeout: 10s
99         ssl: false
```



```
datasource: 疯狂的狮子Li, 2022/1/10 0:48 · update 更改整体配置策略 使用注册中心
system-master:
    # jdbc 所有参数配置参考 https://lionli.blog.csdn.net/article/details/1220
    # rewriteBatchedStatements=true 批处理优化 大幅提升批量插入更新删除性能
    url: jdbc:mysql://localhost:3306/ry-cloud?useUnicode=true&characterEncoding=utf8
    username: root
    password: password
gen:
    url: jdbc:mysql://localhost:3306/ry-cloud?useUnicode=true&characterEncoding=utf8
    username: root
    password: password
job:
    url: jdbc:mysql://localhost:3306/ry-job?useUnicode=true&characterEncoding=utf8
    username: root
    password: password
```

datasource.yml 更改

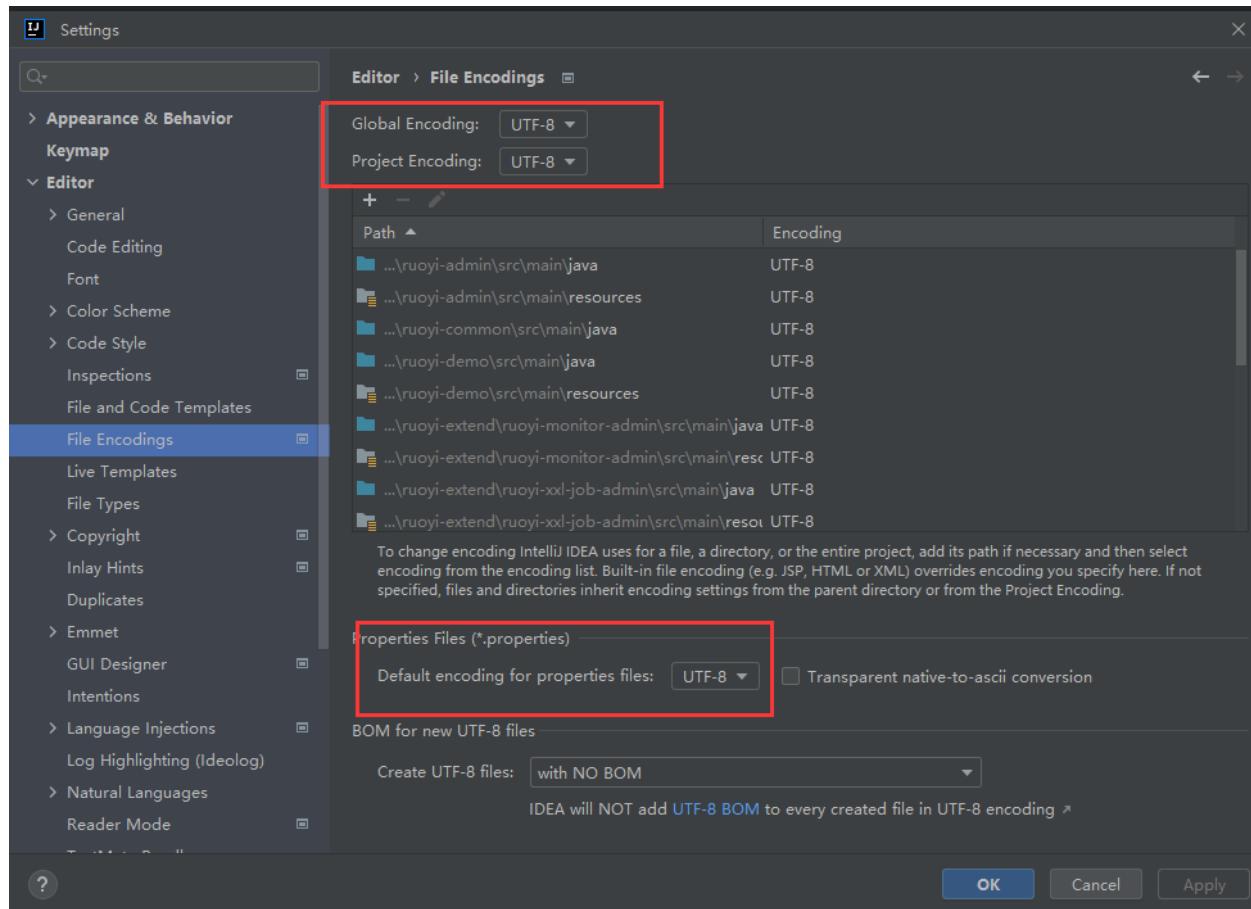


```
service.vgroupMapping.ruoyi-auth-group=default
service.vgroupMapping.ruoyi-system-group=default
service.vgroupMapping.ruoyi-resource-group=default
service.vgroupMapping.ruoyi-gen-group=default
service.vgroupMapping.ruoyi-job-group=default
service.enableDegrade=false
service.disableGlobalTransaction=false
store.mode=db
store.publicKey=
store.db.datasource=druid
store.db.dbType=mysql
store.db.driverClassName=com.mysql.cj.jdbc.Driver
store.db.url=jdbc:mysql://172.30.0.36:3306/ry-seata?useUnicode=true&characterEncoding=utf8
store.db.user=root
store.db.password=root
store.db.minConn=5
```

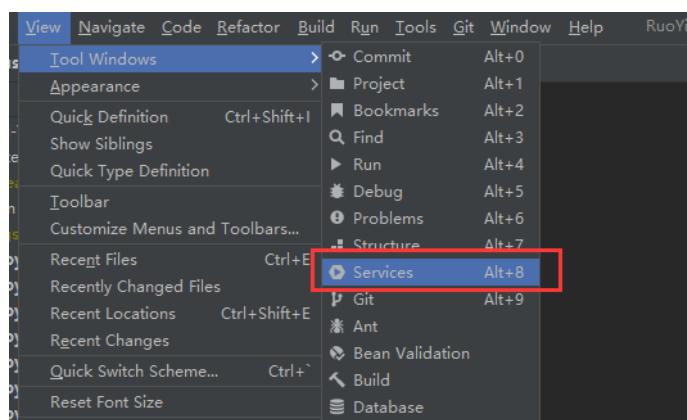
seata-server.properties 更改

## idea环境配置

### 配置项目编码

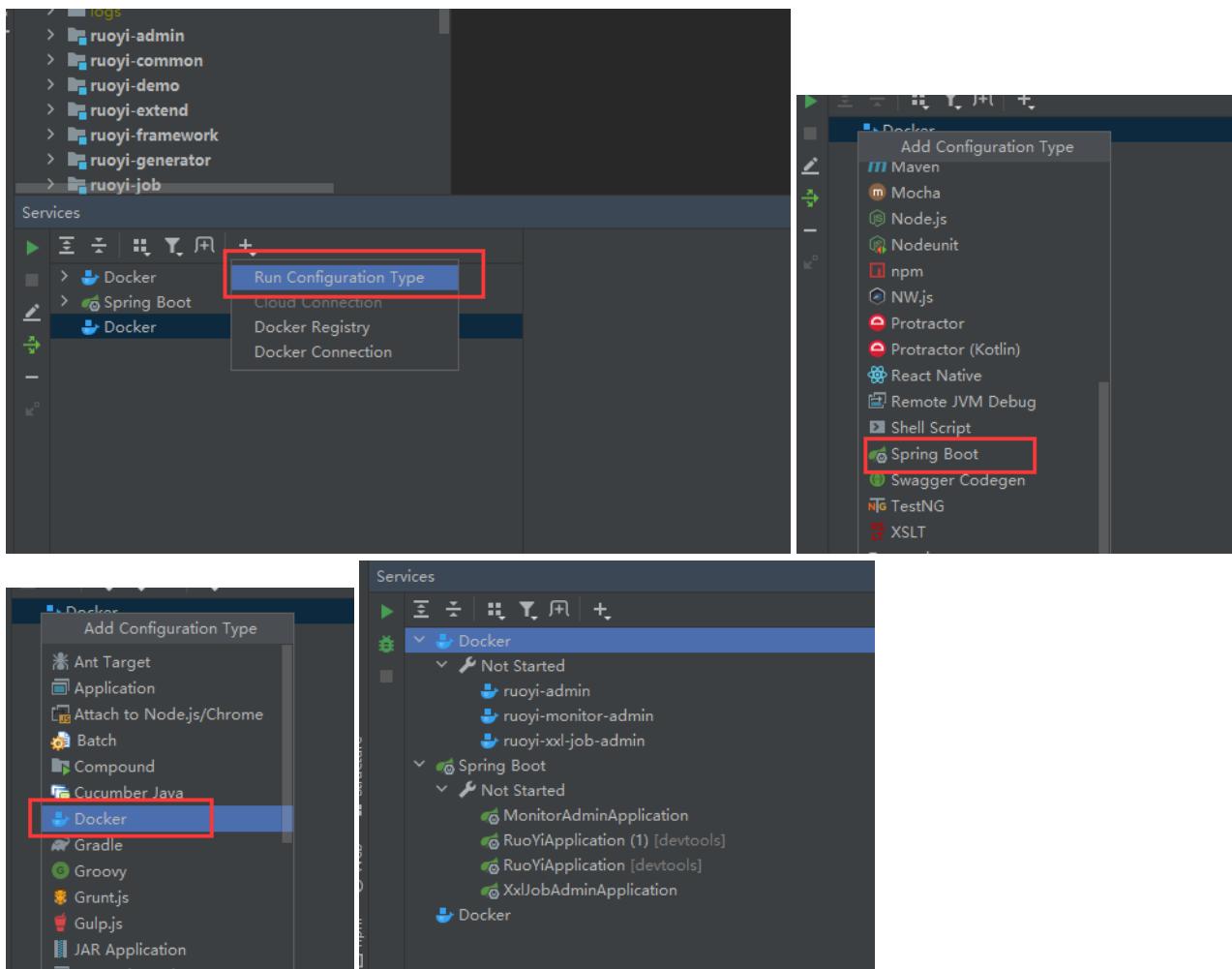


### 配置运行看板



### 配置spring与docker看板

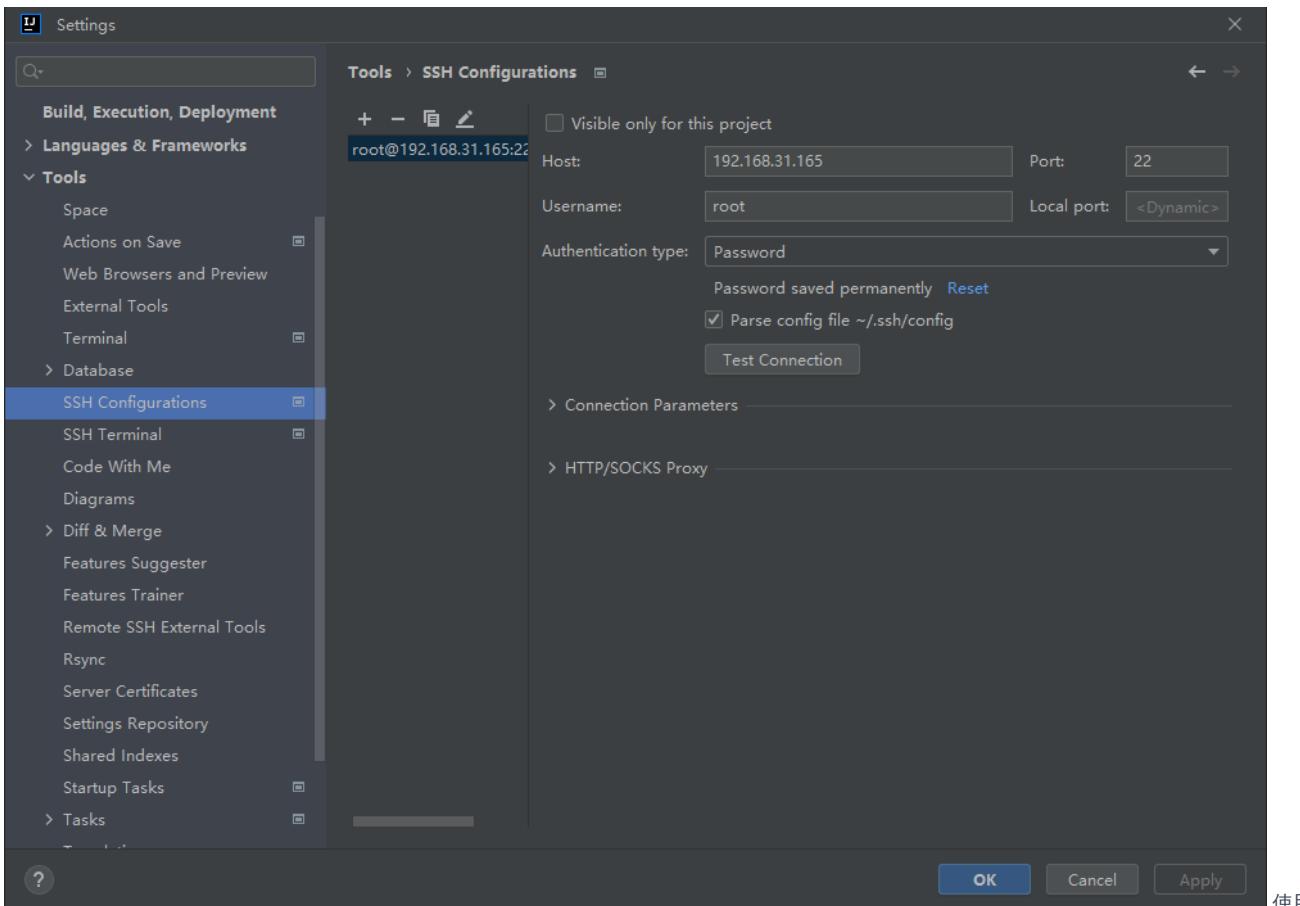
配置服务器SSH连接



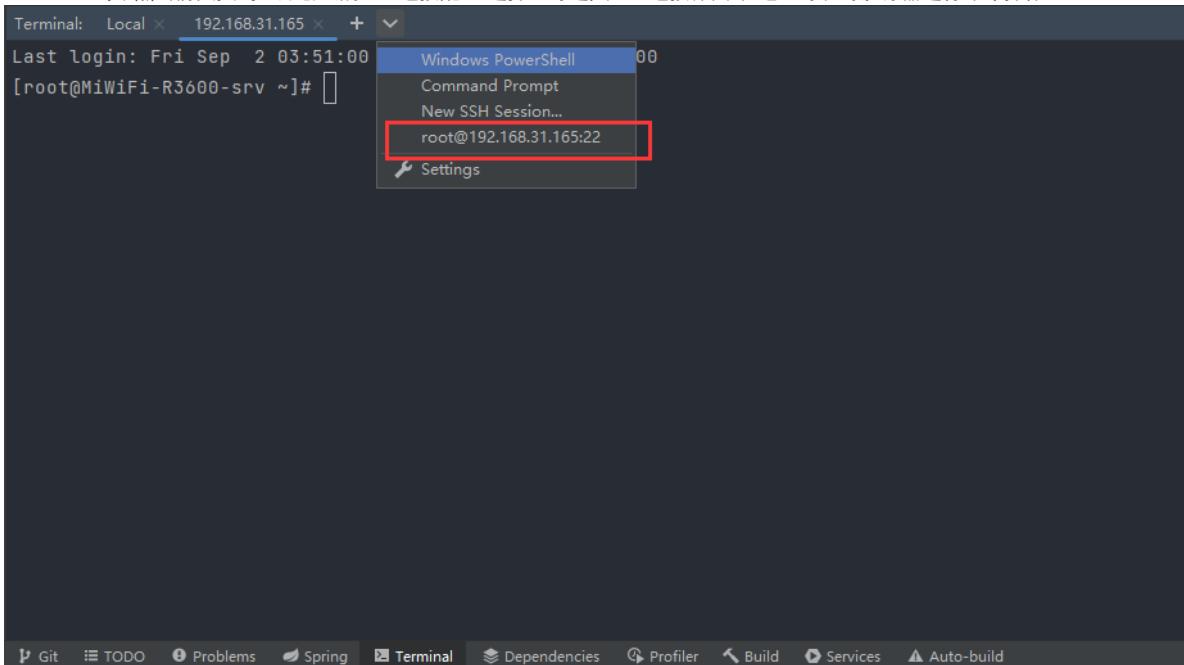
## 配置服务器SSH连接

进入 `Settings -> Tools -> SSH Configurations` 点击加号创建SSH连接配置 填写 服务器IP 用户名 密码 端口号 点击 `Test Connection` 测试连接

## 配置服务器FTP连接



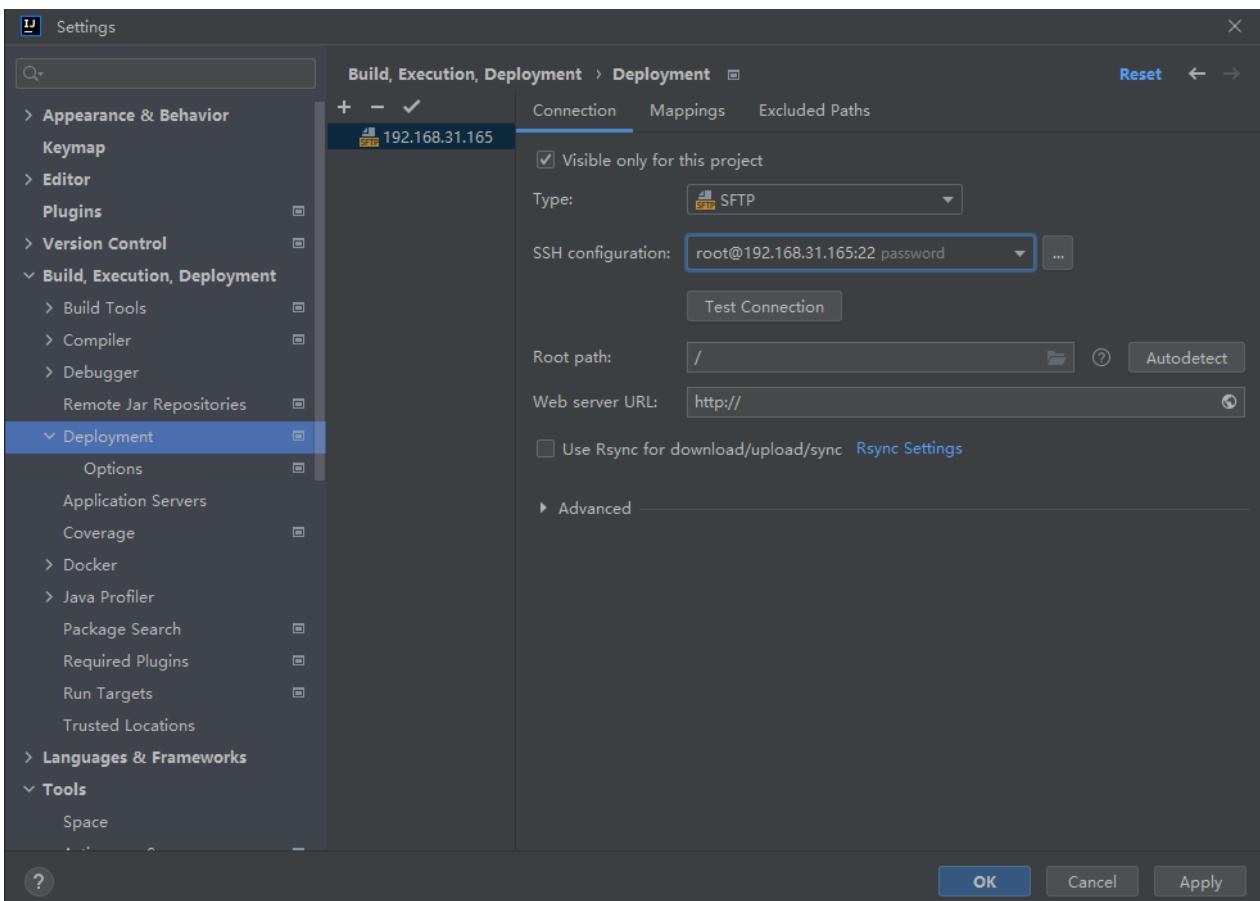
Terminal 工具 点击箭头找到上方创建的SSH连接配置 选择即可进入SSH连接界面 在这里可以对服务器进行命令操作



## 配置服务器FTP连接

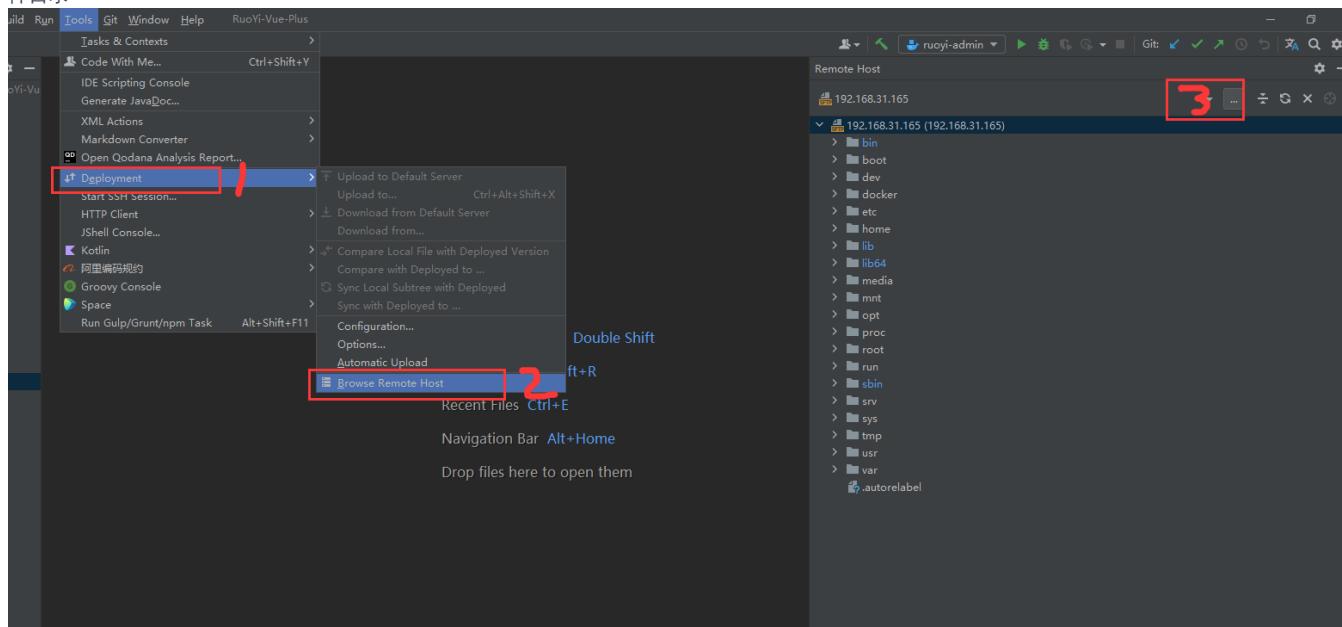
进入 `Settings -> Build-> Deployment` 点击加号 选择SFTP 创建 FTP 连接配置 选择之前创建好的SSH配置 点击 Test Connection 测试连接

配置Docker连接



在IDEA

上方工具栏找到 Tools -> Deployment -> Browse Remote Host 打开远程界面 点击箭头找到我们上方配置的SFTP连接配置 即可连接到服务器的文件目录

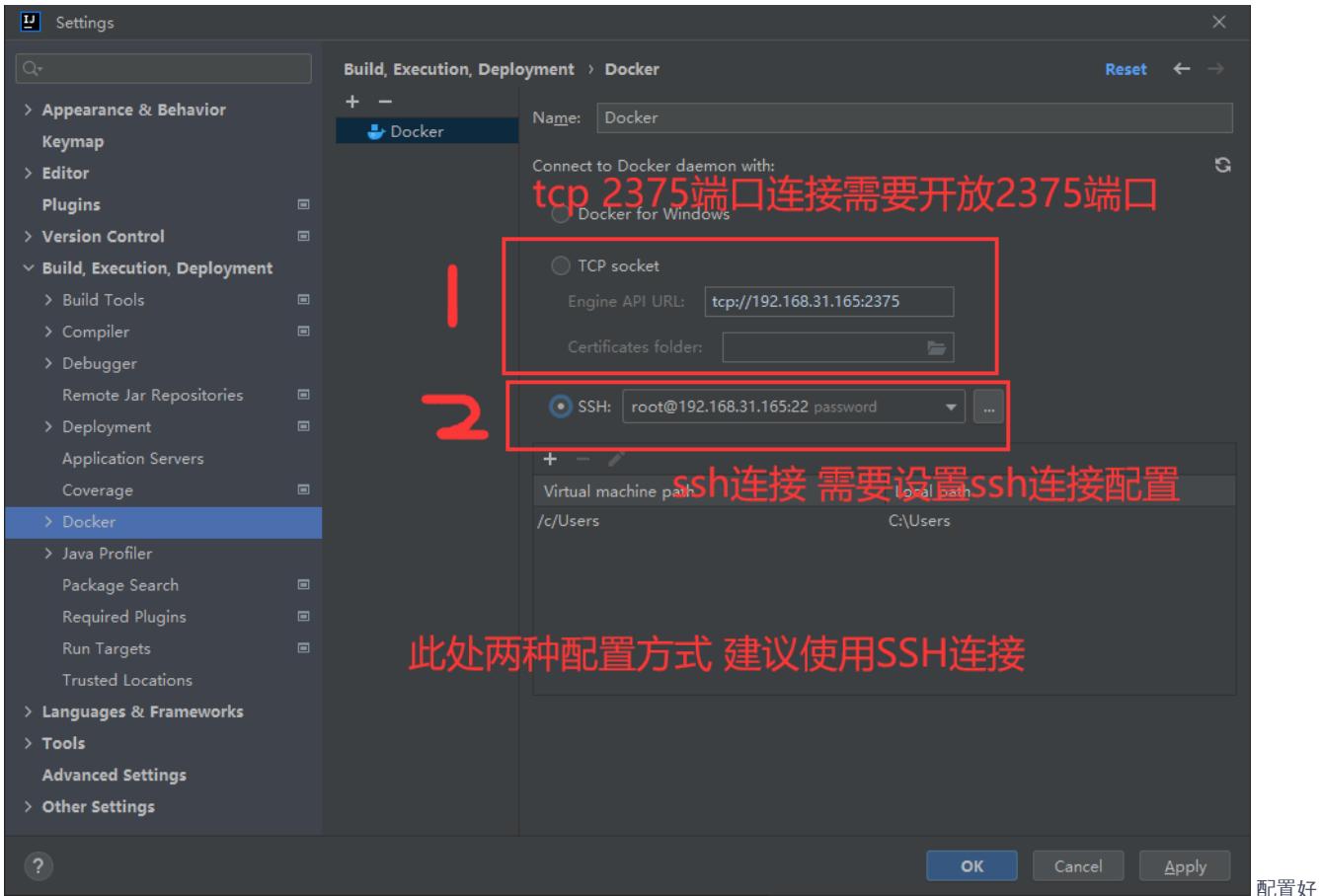


## 配置Docker连接

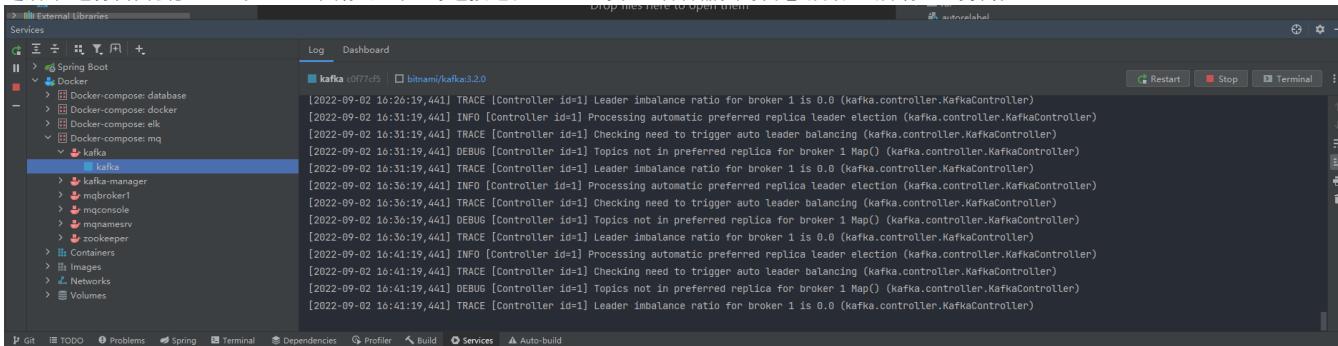
可操作远程docker与构建上传docker镜像(代替原来maven docker插件)

tcp连接需要开放服务器2375端口 ssh需要使用上方的SSH连接配置 建议使用SSH连接

## 配置Docker连接



之后 在运行窗口会多出一个Docker图标 双击即可连接远程docker 可以查看容器实时日志 启动 重启 停止 等操作



## 应用部署

### 版本 >= 1.3.0

请优先阅读 [idea环境配置](#)

## 手动部署

在服务器安装 nacos(对接mysql) mysql redis nginx minio 等其他组件

将项目内 docker/ 文件夹下的文件内容 放到对应的组件内 例如: 将项目内 docker/nginx/nginx.conf 配置文件 复制到 nginx 配置内 将项目内 docker/redis/redis.conf 配置文件 复制到 redis 配置内 将项目内 docker/nacos/custom.properties 配置文件 复制到 nacos 配置内

并修改相关参数如 前端页面存放位置 后端ip地址 等使其生效

jar包部署后端服务 打包命令如下

```
mvn clean install -D maven.test.skip=true -P prod
```

前端参考下方前端部署章节

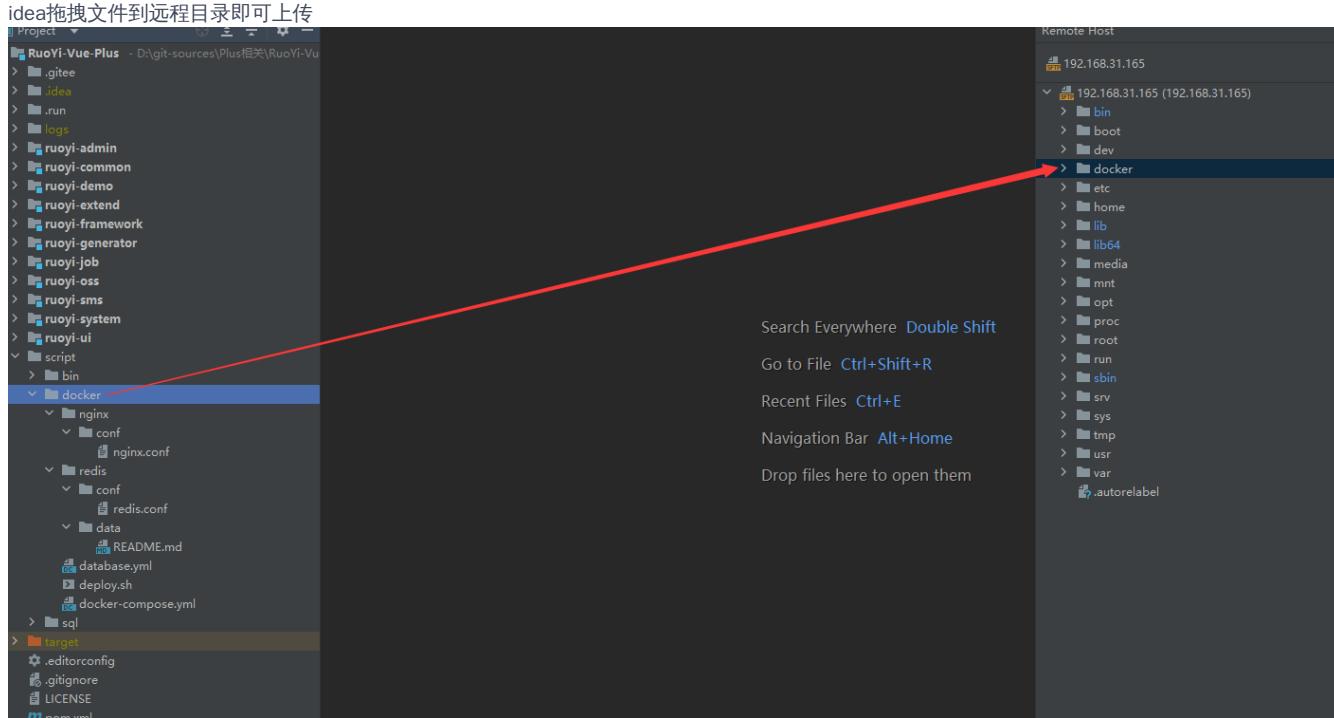
## docker 后端部署

请优先阅读 [idea环境配置](#)

重点: 一知半解的必看

[docker安装](#) [docker-compose安装](#) [docker网络模式讲解](#) [docker 开启端口 2375 供外部程序访问](#)

将配置使用FTP上传到根目录



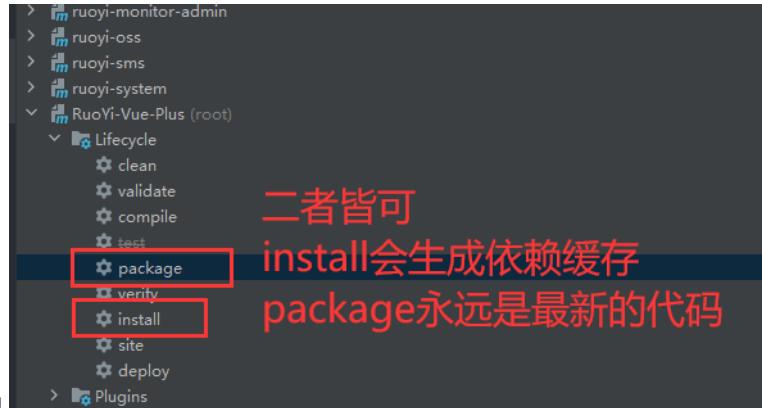
给docker分配文件夹权限

重点注意: 一定要确保目录 `/docker` 及其所有子目录 具有写权限 如果后续出现权限异常问题 重新执行一遍分配权限

```
Last login: Fri Sep 2 04:39:52 2022 from 192.168.31.100
[root@MiWiFi-R3600-srv ~]# chmod -R 777 /docker
[root@MiWiFi-R3600-srv ~]#
```

chmod -R 777 /docker

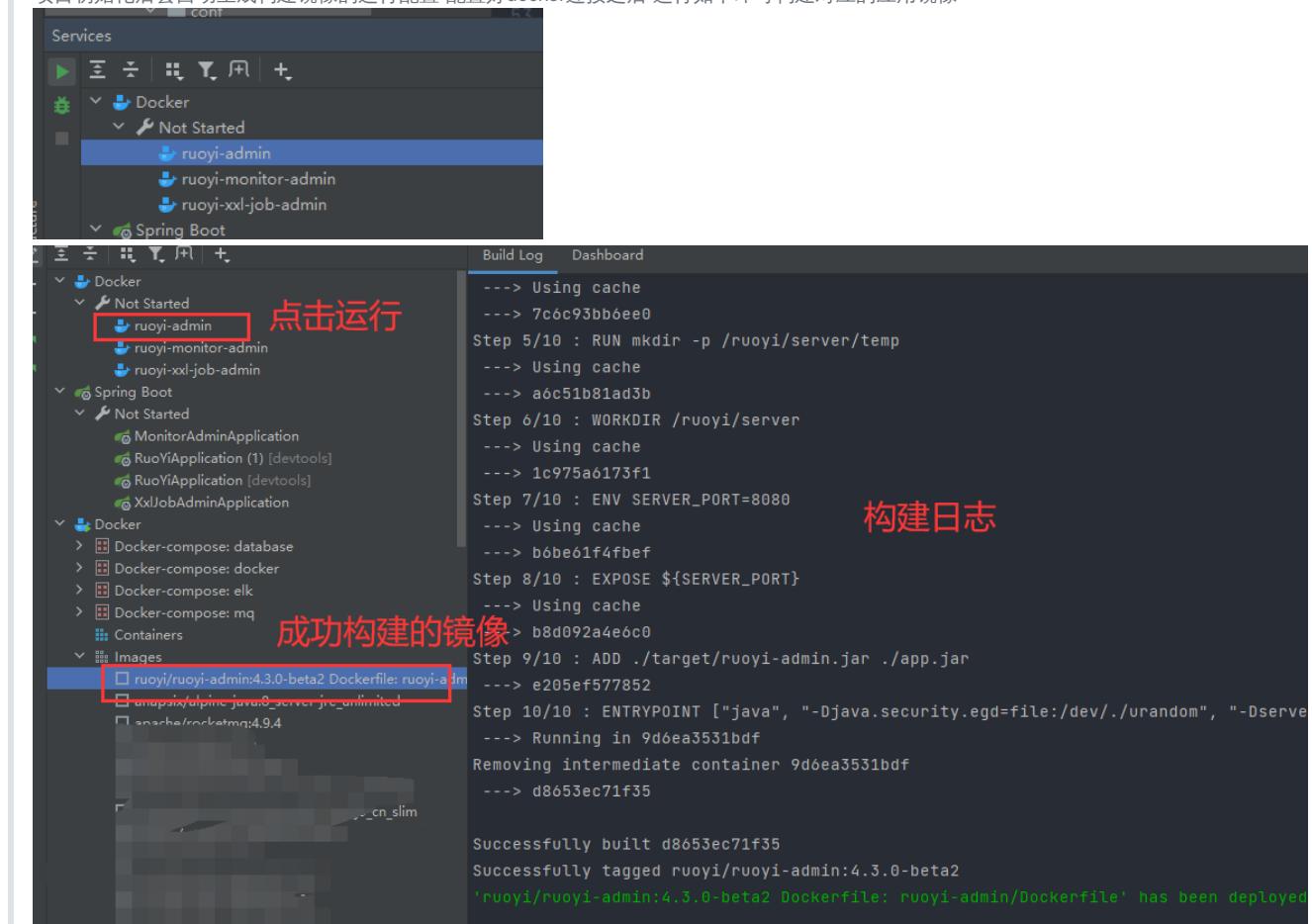
## 构建应用镜像

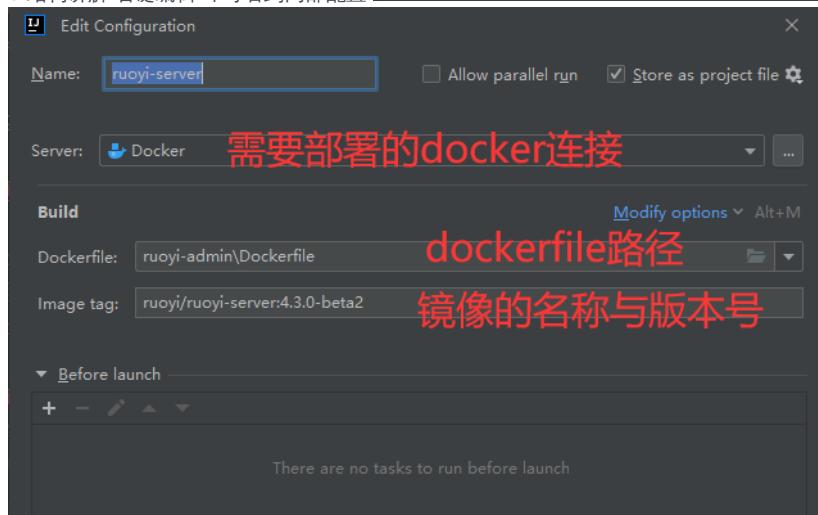
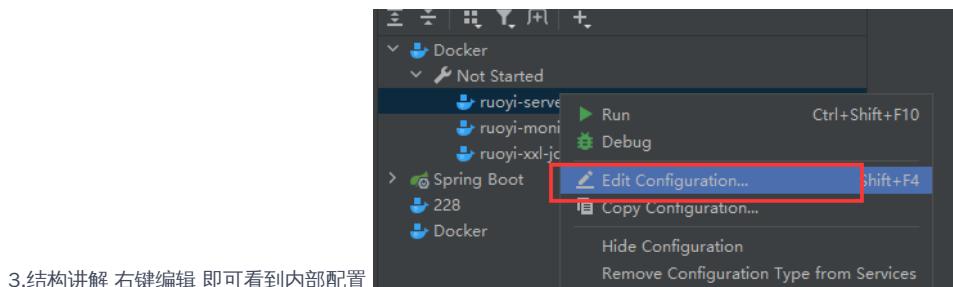


1. 需要先使用maven打包成jar包

2. 执行构建

项目初始化后会自动生成构建镜像的运行配置。配置好docker连接之后，运行如下即可构建对应的应用镜像。





### 创建基础服务

```
docker-compose up -d mysql nginx-web redis minio
```

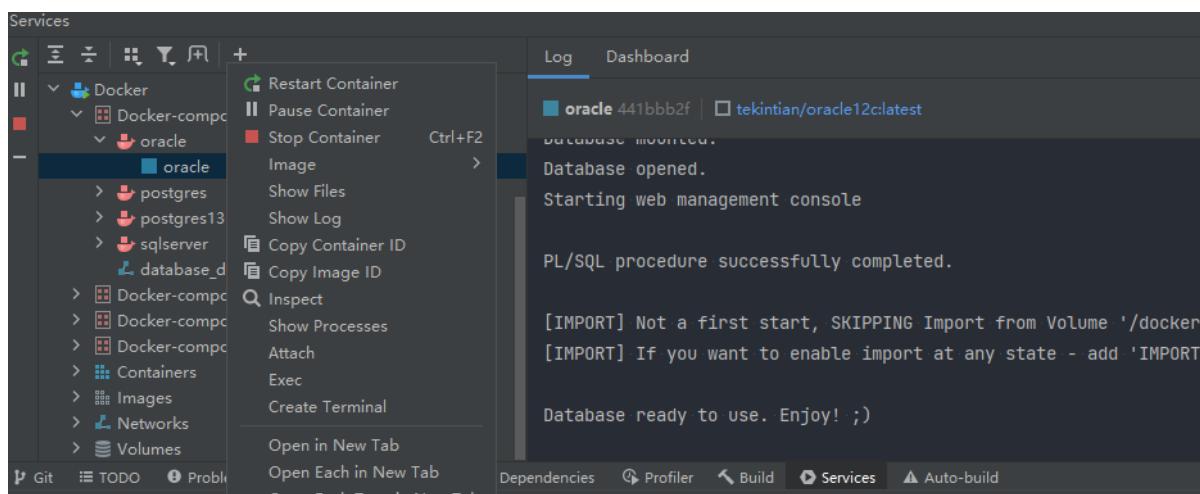
### 创建中心服务(需要先构建服务镜像)

```
docker-compose up -d nacos seata-server sentinel ruoyi-monitor ruoyi-xxl-job-admin
```

### 创建业务服务(需要先构建服务镜像)

```
docker-compose up -d ruoyi-gateway ruoyi-auth ruoyi-system ruoyi-resource
```

### docker其他操作(idea的docker插件 推荐使用)



# 前端部署

执行打包命令

```
# 打包正式环境
npm run build:prod
```

打包后生成打包文件在 `ruoyi-ui/dist` 目录 将 `dist` 目录下文件(不包含 `dist` 目录) 上传到部署服务器 `docker/nginx/html` 目录下(手动部署放入自己配置的路径即可)



重启 `nginx` 服务即可

更改后端代理路径或者后端ip地址

更改代理路径(注意: /开头/结尾)

```
... location /prod-api/ {
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header REMOTE-HOST $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass http://server/;
}

# 监控地址
VUE_APP_XXL_JOB_ADMIN = '/xxl-job-admin'
```

路径对应前端环境文件

```
12
13 # 若依管理系统/生产环境
14 VUE_APP_BASE_API = '/prod-api'
15
16
17
18
```

更改后端ip地址

```
upstream server {
    ip_hash;
    server 127.0.0.1:8080;
    server 127.0.0.1:8081;
}
```

## (旧)应用部署

### 手动部署

在服务器安装 nacos(对接mysql) mysql redis nginx minio 等其他组件

将项目内 docker/ 文件夹下的文件内容 放到对应的组件内 例如: 将项目内 docker/nginx/nginx.conf 配置文件 复制到 nginx 配置内 将项目内 docker/redis/redis.conf 配置文件 复制到 redis 配置内 将项目内 docker/nacos/custom.properties 配置文件 复制到 nacos 配置内

并修改相关参数如 前端页面存放位置 后端Ip地址 等使其生效

jar包部署后端服务 打包命令如下

```
mvn clean install -D maven.test.skip=true -P prod
```

### docker 后端部署

docker安装 docker-compose安装 docker网络模式讲解 docker 开启端口 2375 供外部程序访问

将源码内 docker 文件夹上传到服务器(注意: 不要放到根目录)

进入 docker 目录 给shell脚本分配执行权限

```
chmod 777 ~/docker/deploy.sh
```

开放外网防火墙端口(内网服务无需开启)

```
sh deploy.sh port
```

放置挂载文件(切勿多次执行)

```
sh deploy.sh mount
```

分配文件夹权限

重点注意: 一定要确保目录 /docker 及其所有子目录 具有写权限 如果后续出现权限异常问题 重新执行一遍分配权限

```
chmod -R 777 /docker
```

启动基础服务

```
sh deploy.sh base
```

启动可视化服务

```
sh deploy.sh monitor
```

启动与停止业务服务(需要先构建服务镜像)

```
sh deploy.sh start  
sh deploy.sh stop
```

停止所有服务

## 前端部署

```
sh deploy.sh stopall
```

### 删除所有容器

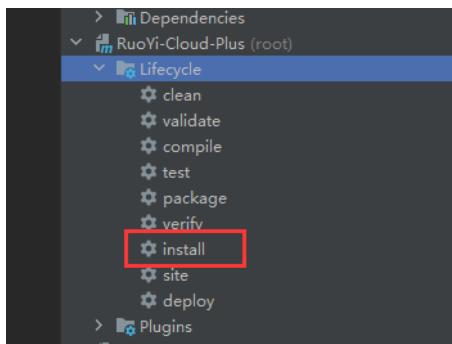
```
sh deploy.sh rm
```

### 删除所有空版本镜像

```
sh deploy.sh rmiNoneTag
```

### 构建服务镜像

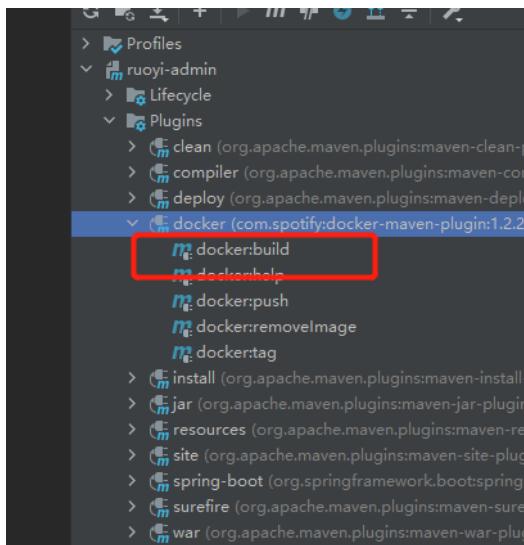
首先 使用 maven 总工程打包 所有模块jar包(注意: 此处需使用 install 因有几个依赖工程需要装载到本地)



然后 使用 docker-maven-plugin 插件上传构建 ruoyi-server 镜像

需修改 pom 文件对应 docker 服务器的 ip 地址与 docker 服务配置开启 2375 api端口 docker 开启端口 2375 供外部程序访问

未开启 2375 api端口将无法远程连接 docker 可选使用 ssh 上传 jar 包 在服务器执行 docker build 构建镜像 命令具体用法参考文章:  
docker build 方式部署jar包



对应maven命令(对应模块目录内执行)

```
cd ruoyi-admin  
mvn docker:build
```

## 前端部署

执行打包命令

```
# 打包正式环境  
npm run build:prod
```

打包后生成打包文件在 `ruoyi-ui/dist` 目录 将 `dist` 目录下文件(不包含 `dist` 目录) 上传到部署服务器 `docker/nginx/html` 目录下(手动部署放入自己配置的路径即可) 重启 `nginx` 服务即可

## Seata服务搭建

版本 >= 0.12.0 源码集成 Seata 1.5.X 服务端

执行 `ry-seata.sql` 文件 初始化服务端数据库 修改 `nacos` 内的 `seata-server.properties` 的数据库地址 启动 `ruoyi-seata-server` 服务即可

# 框架功能

## 项目结构

### 目录结构

v1.3.0

```

RuoYi-Cloud-Plus
├─ ruoyi-api      // api模块
│  └─ ruoyi-api-bom    // api模块依赖管理
│  └─ ruoyi-api-resource // 资源api模块
│  └─ ruoyi-api-system   // 系统api模块
└─ ruoyi-auth     // 通用模块 [9210]
└─ ruoyi-common   // 通用模块
    └─ ruoyi-common-alibaba-bom // alibaba 依赖管理
    └─ ruoyi-common-bom       // common 依赖管理
    └─ ruoyi-common-core     // 核心功能模块
    └─ ruoyi-common-dict      // 字典集成模块
    └─ ruoyi-common-doc       // 文档集成模块
    └─ ruoyi-common-dubbo     // dubbo集成模块
    └─ ruoyi-common-elasticsearch // ES集成模块
    └─ ruoyi-common-excel      // excel集成模块
    └─ ruoyi-common-idempotent // 幂等功能模块
    └─ ruoyi-common-job        // job定时任务集成模块
    └─ ruoyi-common-loadbalancer // 团队负载均衡集成模块
    └─ ruoyi-common-log         // 日志集成模块
    └─ ruoyi-common-logstash    // elk日志集成模块
    └─ ruoyi-common-mail        // 邮件集成模块
    └─ ruoyi-common-mybatis     // mybatis数据库相关集成模块
    └─ ruoyi-common-oss          // oss相关集成模块
    └─ ruoyi-common-redis        // redis集成模块
    └─ ruoyi-common-satoken      // satoken集成模块
    └─ ruoyi-common-seata        // seata分布式事务集成模块
    └─ ruoyi-common-security     // 框架权限鉴权集成模块
    └─ ruoyi-common-sentinel      // sentinel集成模块
    └─ ruoyi-common-sms          // 短信集成模块
    └─ ruoyi-common-web          // web服务集成模块
└─ ruoyi-example   // 例子模块
└─ ruoyi-demo     // 演示模块 [9401]
└─ ruoyi-stream-mq // mq演示模块 [9402]
└─ ruoyi-gateway  // 网关模块 [8080]
└─ ruoyi-modules  // 功能模块
    └─ ruoyi-gen        // 代码生成模块 [9202]
    └─ ruoyi-job        // 任务调度模块 [9203,9901]
    └─ ruoyi-resource    // 资源模块 [9204]
    └─ ruoyi-system      // 系统模块 [9201]
└─ ruoyi-visual    // 可视化模块
    └─ ruoyi-monitor    // 服务监控模块 [9100]
    └─ ruoyi-nacos      // nacos服务模块 [8848,9848,9849]
    └─ ruoyi-seata-server // seata服务模块 [7091,8091]
    └─ ruoyi-sentinel-dashboard // sentinel控制台模块 [8718]
    └─ ruoyi-xxl-job-admin // 任务调度控制台模块 [9900]
└─ ruoyi-ui        // 前端框架 [80]
└─ config/dev      // nacos配置文件(需复制到nacos配置中心使用)
    └─ sentinel-ruoyi-gateway.json // sentinel对接gateway限流配置文件
    └─ seata-server.properties // seata服务配置文件
    └─ application.yml      // 所有应用主共享配置文件
    └─ datasource.yml        // 所有应用共享数据源配置文件
    └─ ruoyi-auth.yml        // auth 模块配置文件
    └─ ruoyi-gateway.yml     // gateway 模块配置文件
    └─ ruoyi-gen.yml         // gen 模块配置文件
    └─ ruoyi-job.yml         // job 模块配置文件
    └─ ruoyi-monitor.yml     // monitor 模块配置文件
    └─ ruoyi-resource.yml    // resource 模块配置文件
    └─ ruoyi-system.yml      // system 模块配置文件
    └─ ruoyi-sentinel-dashboard.yml // sentinel 控制台 模块配置文件
    └─ ruoyi-xxl-job-admin.yml // xxljob 控制台 模块配置文件

```

```
|─ sql      // sql脚本
|  └─ ry-cloud.sql    // 主sql文件
|  └─ ry-config.sql   // 配置中心sql文件
|  └─ ry-job.sql      // 任务调度sql文件
|  └─ ry-seata.sql    // 分布式事务sql文件
└─ docker     // docker 配置脚本
  └─ nacos        // nacos配置文件
  └─ nginx        // nginx配置文件
  └─ redis        // redis配置文件
  └─ seata        // seata配置文件
  └─ deploy.sh    // 运行脚本
  └─ docker-compose.yml // docker编排文件
.EDITORCONFIG // 编辑器编码格式配置
LICENSE      // 开源协议
pom.xml      // 公共依赖
README.md    // 框架说明文件
```

## 创建新服务

最简单的方式

找个配置好的 例如 ruoyi-system 直接copy一份



```
</parent>
<modelVersion>4.0.0</modelVersion>

<artifactId>ruoyi-system</artifactId>

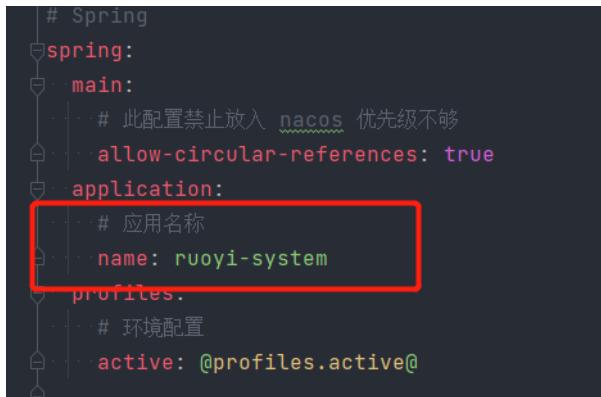
<description>
    ruoyi-system系统模块
</description>
```

将 pom 名称改掉



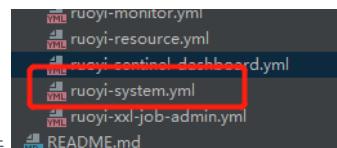
```
/*
 * 系统模块
 *
 * @author ruoyi
 */
@EnableDubbo
@SpringBootApplication
public class RuoYiSystemApplication {
    疯狂的狮子li, 2021/
    public static void main(String[] args) {
        SpringApplication.run(RuoYiSystemApplication.class, args);
        System.out.println("(*_*) 系统模块启动成功");
    }
}
```

服务启动类 名称改掉



```
# Spring
spring:
  main:
    # 此配置禁止放入 nacos 优先级不够
    allow-circular-references: true
  application:
    # 应用名称
    name: ruoyi-system
  profiles:
    # 环境配置
    active: @profiles.active@
```

application.yml 配置服务应用名 改掉



nacos 新建一份新的 对应新模块名称的 配置文件

更改 nacos 上的 ruoyi-gateway.yml 增加新服务路由 新服务访问路径 网关ip:端口/服务路径/controller路径/接口路径 例子:

public | **dev** | prod

配置管理 | dev dev 查询结果：共查询到 44 条满足要求的配置。

Data ID: 已开启默认模糊查询 Group: 已开启默认模糊查询

	Data Id ↓	Group ↓
<input type="checkbox"/>	application-common.yml	DEFAULT_GR
<input type="checkbox"/>	datasource.yml	DEFAULT_GR
<input type="checkbox"/>	ruoyi-gateway.yml	DEFAULT_GR
<input type="checkbox"/>	ruoyi-auth.yml	DEFAULT_GR

http://localhost:8080/system/user/list

```
0      # 系统模块
1      - id: ruoyi-system
2          uri: lb://ruoyi-system
3          predicates:
4              - Path=/system/**
5          filters:
6              - StripPrefix=1
```

## 多团队开发

### 功能介绍

多人员/团队开发往往会出现 调试程序 被负载均衡到别人那里 自己抓不到请求等问题 正确团队开发模式 测试机一台 公共服务都放到测试机上 本地开发人员 需启动 ruoyi-gateway 与 其他 调试的业务模块 将所有服务都统一指向同一个 Nacos 服务 前端连接本机 ruoyi-gateway 网关 调试程序

框架提供了 ruoyi-common-loadbalancer 多团队 负载均衡模块 可以将网关的请求锁定到与网关相同的IP服务

需要在 ruoyi-gateway ruoyi-auth ruoyi-modules 引入 ruoyi-common-loadbalancer 模块



```
pom.xml (ruoyi-modules) × pom.xml (ruoyi-gateway) × pom.xml (ruoyi-auth) ×
92     </dependency>
93
94     <!-- 自定义负载均衡(多团队开发使用) --&gt;
95     &lt;dependency&gt;
96         &lt;groupId&gt;com.ruoyi&lt;/groupId&gt;
97         &lt;artifactId&gt;ruoyi-common-loadbalancer&lt;/artifactId&gt;
98     &lt;/dependency&gt;
99</pre>
```

启动前端访问本机 ruoyi-gateway 网关在请求转发 和 dubbo 进行 RPC 调用时 会获取与本机IP地址相同的服务优先调用(如未找到 会随机返回)

## 重点说明

请检查本机是否有虚机网卡IP 如有多网卡获取IP地址会不准确

可使用如下代码检查本机IP是否正常

```
InetAddress.getLocalHost().getHostAddress()
```

## 分页功能

### 重点说明

项目使用 mybatis-plus 分页插件 实现分页功能 大致用法与 MP 一致 [MP分页文档](#) 项目已配置分页合理化 页数溢出 例如: 一共5页 查了第6页

```
/*
 * 分页插件，自动识别数据库类型
 */
public PaginationInnerInterceptor paginationInnerInterceptor() {
    PaginationInnerInterceptor paginationInnerInterceptor = new PaginationInnerInterceptor();
    // 设置最大单页限制数量，默认 500 条，-1 不受限制
    paginationInnerInterceptor.setMaxLimit(-1L);
    // 分页合理化
    paginationInnerInterceptor.setOverflow(true);
    return paginationInnerInterceptor;
}
```

默认返回第一页

## 代码用法

Controller 使用 PageQuery 接收分页参数 具体参数参考 PageQuery

```
/**
 * 查询测试单表列表
 */
@ApiOperation("查询测试单表列表")
@PreAuthorize("@ss.hasPermi('demo:demo:list')")
@GetMapping("/list")
public TableDataInfo<TestDemoVo> list(@Validated(QueryGroup.class) TestDemoBo bo, PageQuery pageQuery) {
    return iTestDemoService.queryPageList(bo, pageQuery);
}
```

构建 Mybatis-Plus 分页对象 使用 PageQuery#build() 方法 可快速(基于当前对象数据)构建 MP 分页对象

```
@Override
public TableDataInfo<TestDemoVo> queryPageList(TestDemoBo bo, PageQuery pageQuery) {
    LambdaQueryWrapper<TestDemo> lqw = buildQueryWrapper(bo);
    Page<TestDemoVo> result = pageVo(pageQuery, build().lqw);
    return TableDataInfo.build(result);
}
```

```
public <T> Page<T> build() {
    Integer pageNum = ObjectUtil.defaultIfNull(getPageNum());
    Integer pageSize = ObjectUtil.defaultIfNull(getPageSize());
    if (pageNum <= 0) {
        pageNum = DEFAULT_PAGE_NUM;
    }
    Page<T> page = new Page<T>(pageNum, pageSize);
    OrderItem orderItem = buildOrderItem();
    if (ObjectUtil.isNotEmpty(orderItem)) {
        page.addOrder(orderItem);
    }
    return page;
}
```

具体用法与 MP 一致

自定义 SQL 方法分页 只需在 Mapper 方法第一个参数和返回值 重点: 第一个参数 标注分页对象

```
Page<TestDemoVo> customPageList(@Param("page") Page<TestDemo> page, @Param("ew") Wrapper<TestDemo> wrapper)
    // resultmap
    <select id="customPageList" resultType="com.ruoyi.demo.domain.vo.TestDemoVo">
        SELECT * FROM test_demo ${ew.customSqlSegment}
    </select>
```

## 事务相关

若依文档对事务注解的描述 [关于事务](#) 以下对多数据源事务做补充:

- 同一个事务下是无法切换数据源的
- 禁止 父方法使用 `@Transactional` 创建事务 子方法使用 `@DS` 切换数据源

## 多数据源

### 关于多数据源

- 同一个事务下是无法切换数据源 具体参考关于事务篇章
- 切换数据源与二级缓存无法一起使用 会造成数据不一致(解决方案 移除Mapper接口缓存注解)

### 用法参考 **demo** 模块

加载顺序 方法 => 类 => 默认

### 用法

```
/*
 * 测试树表Service业务层处理
 *
 * @author Lion Li
 * @date 2021-07-26
 */
@Service
public class TestTreeServiceImpl extends ServicePlusImpl {

    @Override
    public TestTreeVo queryById(Long id) {
        return queryById(id);
    }

    /**
     * @DS("slave") // 切换从库查询
     */
    @Override
    public List<TestTreeVo> queryList(TestTreeBo bo) {
        LambdaQueryWrapper<TestTree> lqw = buildQueryWrapper(bo);
        return listVo(lqw);
    }
}
```

### 更多用法

参考多数据源框架官方文档: [dynamic-datasource](#) 文档

## OSS功能

# 关于OSS模块使用

## 重点注意事项

桶/存储区域 系统会根据配置自行创建分配权限 如手动配置需要设置 公有读-权限 否则文件无法访问( aliyun 还需开通跨域配置) 4.4.0 版本支持配置 公有/私有 权限( aliyun 还需开通跨域配置) 访问站点为 域名 + 端口 端口后严禁携带其他 url 例如: /, /ruoyi 等 阿里云与腾讯云SDK访问站点中不能包含桶名 系统会自动处理 minio 站点不允许使用 localhost 请使用 127.0.0.1 访问站点与自定义域名 都不要包含 **http https** 前缀 设置 **https** 请使用选项处理

## 代码使用

参考 `SysOssService.upload` 用法

```
    @Override
    public SysOss upload(MultipartFile file) {
        String originalfilename = file.getOriginalFilename();
        String suffix = StringUtils.substring(originalfilename, originalfilename.lastIndexOf("."));
        OssClient storage = OssFactory.instance(); // 疯狂的狮子Li, 4 minutes ago • !175 [重大改动] 基于S
        UploadResult uploadResult;
        try {
            uploadResult = storage.uploadSuffix(file.getBytes(), suffix, file.getContentType());
        } catch (IOException e) {
            throw new ServiceException(e.getMessage());
        }
        // 保存文件信息
        SysOss oss = new SysOss();
        oss.setUrl(uploadResult.getUrl());
    }
}
```

用 `OssFactory.instance()` 获取当前启用的 `OssClient` 实例 进行功能调用 获取返回值后 存储到对应的业务表

## 功能配置

### 配置OSS

进入 `系统管理 -> 文件管理 -> 配置管理` 填写对应的OSS服务相关配置

The screenshot displays the RuoYi-Vue-Plus system management interface. On the left, a dark sidebar lists various management modules: 首页, 系统管理 (selected), 用户管理, 角色管理, 菜单管理, 部门管理, 岗位管理, 字典管理, 参数设置, 通知公告, 日志管理, and 文件管理 (selected). Red arrows point from the '文件管理' link in both the sidebar and the breadcrumb navigation at the top to the configuration table on the right.

**Top Navigation:** 首页 / 系统管理 / 文件管理

**Search and Filter:**

- 文件名: 请输入文件名
- 原名: 请输入原名
- 文件后缀: 请输入文
- 上传人: 请输入上传人
- 服务商: 请输入服务商
- 搜索
- 重置

**Configuration Buttons:**

- + 上传文件
- + 上传图片
- 应 删除
- 预览开关: 禁用
- 配置管理

**Table Headers:**

	文件名	原名	文件后缀	文件展示
--	-----	----	------	------

**Data:** 暂无数据

**Bottom Table Headers:**

	配置key	访问站点	自定义域名	桶名称	前缀	域	状态	操作
--	-------	------	-------	-----	----	---	----	----

**Data Rows:**

<input type="checkbox"/>	minio	43.138.9.96:9000	[REDACTED]	ruoyi			<input checked="" type="checkbox"/>	<a href="#">修改</a> <a href="#">删除</a>
<input type="checkbox"/>	qiniu	s3-cn-north-1.qiniucs.com	[REDACTED]	ruoyi-vue-plus			<input type="checkbox"/>	<a href="#">修改</a> <a href="#">删除</a>
<input type="checkbox"/>	aliyun	oss-cn-beijing.aliyuncs.com	[REDACTED]	ruoyi-vue-plus			<input type="checkbox"/>	<a href="#">修改</a> <a href="#">删除</a>
<input type="checkbox"/>	qcloud	cos.ap-beijing.myqcloud.co m	[REDACTED]	ruoyi-1257110000		ap-beijing	<input type="checkbox"/>	<a href="#">修改</a> <a href="#">删除</a>
<input type="checkbox"/>	image	127.0.0.1:9000		ruoyi	image		<input type="checkbox"/>	<a href="#">修改</a> <a href="#">删除</a>

**Pagination:** 共 5 条 10条/页 < 1 > 前往 1 页

## 修改对象存储配置

&gt;

\* 配置key qiniu

\* 访问站点 s3-cn-north-1.qiniucs.com

自定义域名

\* accessKey

\* secretKey  ⚡

\* 桶名称 ruoyi-

前缀

是否HTTPS  是  否

域 

## 重点说明

云厂商只需修改 访问站点 对应的域 切勿乱改(云厂商强烈建议绑定自定义域名使用 七牛云必须绑定[官方规定])

+ 新增  

<input type="checkbox"/>	配置key	访问站点	自定义域名	桶名称
<input type="checkbox"/>	minio	40.122.222.222	mydomain-plus.top	ruoyi
<input type="checkbox"/>	qiniu	s3-cn-north-1.qiniucs.com	mydomain-plus.top	ruoyi-vue-plus
<input type="checkbox"/>	aliyun	oss-cn-beijing.aliyuncs.com	mydomain-plus.top	ruoyi-vue-plus
<input type="checkbox"/>	qcloud	cos.ap-beijing.myqcloud.com	mydomain-plus.top	ruoyi-1257412600
<input type="checkbox"/>	image	127.0.0.1:9000		ruoyi

七牛云 访问站点

## 关于OSS模块使用

The screenshot shows the Qiniu Cloud Storage (七牛云) OSS module interface. The main navigation bar includes '首页' (Home), '消息 (10)' (Messages), '文档' (Documentation), '工单' (Work Orders), '费用' (Costs), and '云商城' (Cloud Marketplace). The left sidebar has sections for '对象存储' (Object Storage), '概览' (Overview), '空间管理' (Space Management) (highlighted with a red arrow), '跨区域同步' (Cross-region Sync), and '统计分析' (Statistics Analysis). The top right shows storage information: 存储量: 0, 对象数: 0, 访问控制: 公开, 空间类型: 自有空间.

The '空间管理' tab is selected, showing tabs for '空间概览' (Space Overview), '文件管理' (File Management), '域名管理' (Domain Management), '图片样式' (Image Style), '转码样式' (Transcoding Style), and '空间设置' (Space Settings). The '空间概览' tab is active.

The 'CDN 加速域名' (CDN Acceleration Domains) section lists a single domain entry:

域名	协议	类型	状态	操作
七牛融合 CDN 测试域名	HTTP	普通域名	成功	[CNAME]

The '自定义域名' (Custom Domain) section contains a note about CDN test domains:

七牛融合 CDN 测试域名 (以 cloudn.com/qiniucdn.com/qnssl.com/qbox.me 结尾)，每个域名每日限回源总流量 10GB，每个测试域名自创建起 30 个自然日后系统会自动回收，仅供测试使用且不支持 HTTPS 访问，详情查看[七牛测试域名使用规范](#)。点击下列域名可查看每个域名剩余回收时间。

The '基础配置' (Basic Configuration) section includes links for '访问控制' (Access Control), '生命周期' (Lifecycle), '空间授权' (Space Authorization), '事件通知' (Event Notification), '镜像回源' (Mirror Origin), and 'S3 域名' (S3 Domain).

A modal window titled 'S3 域名' (S3 Domain) is open, showing the following details:

空间的 S3 域名仅限通过 AWS S3 兼容 api 对接时使用，且通过该域名访问空间时将产生外网流出流量。[了解 AWS S3 兼容详情](#).

Endpoint (区域节点) : s3-cn-north-1.qiniucs.com

空间域名②: [REDACTED]

The sidebar on the left shows '域名' (Domain), '协议' (Protocol), and 'HTTP'.

At the bottom left, there is a link '阿里云 访问站点' (Aliyun Access Site).

## 关于OSS模块使用

The screenshot shows the Aliyun OSS console for a bucket named 'ruoyi' in the Beijing region. The left sidebar includes links for Overview, Usage Query, File Management, Permission Management, Basic Settings, Redundancy and Reliability, Transfer Management, Log Management, Data Processing, and Data Security. The main area has two tabs: 'Overview' (selected) and 'Basic Data'. The 'Basic Data' tab displays storage usage (0 Byte), traffic (127.5 KB), and requests (170). It also shows a note about data being delayed by 1-2 hours. Below this are sections for 'Access Domains' and 'Transmission Acceleration Domains'. The 'Endpoint (Region Node)' field is highlighted with a red box and contains the value 'oss-cn-beijing.aliyuncs.com'. Other endpoints listed include 'oss-cn-beijing-internal.aliyuncs.com' for both ECS classic network and VPC network access.

## 腾讯云 访问站点

The screenshot shows the Tencent Cloud OSS console for a bucket named 'ruoyi'. The left sidebar lists various management categories. The main area shows basic statistics: object count (0), storage (0 B), traffic (286.94 KB), and requests (279). The 'Domain Information' section is highlighted with a red box and shows the 'Default CDN Acceleration Domain' as 'https://ruoyi-100711111.cos.ap-beijing.myqcloud.com'. A red annotation box with the text '注意: 此处只需要黄线内的部分' (Note: Only the part inside the yellow line is required) points to the URL field. The 'Basic Information' section shows the bucket name 'ruoyi', location 'Beijing (China) (ap-beijing)', creation time '2022-03-30 15:29:09', and access rights 'Public Read Private Write'.

## 切换OSS

再配置列表点击 状态 按钮开启即可(注意: 只能开启一个OSS默认配置) 手动使用 `OssFactory.instance("configKey")`

桶名称	前缀	域	状态	操作
ruoyi			<input type="checkbox"/>	<a href="#">修改</a> <a href="#">删除</a>
ruoyi [REDACTED]			<input checked="" type="checkbox"/>	<a href="#">修改</a> <a href="#">删除</a>
ruoyi [REDACTED]			<input type="checkbox"/>	<a href="#">修改</a> <a href="#">删除</a>
ruoyi [REDACTED]		ap-beijing	<input type="checkbox"/>	<a href="#">修改</a> <a href="#">删除</a>
ruoyi	image		<input type="checkbox"/>	<a href="#">修改</a> <a href="#">删除</a>

共 5 条 10条/页 < 1 > 前往 1 / 1

## 扩展分类

如有文件分类 建议创建多个 oss配置 进行切换存储

例如: 创建一个 图片存储的 oss配置 指定唯一的 configKey 与 前缀目录 或 直接使用独立的 桶 独立桶的特点 可以自定义访问权限 例如: 创建一个私有文件存储桶 不对外开放

m	image	localhost:9000	ruoyi	image

指定需要使用的配置

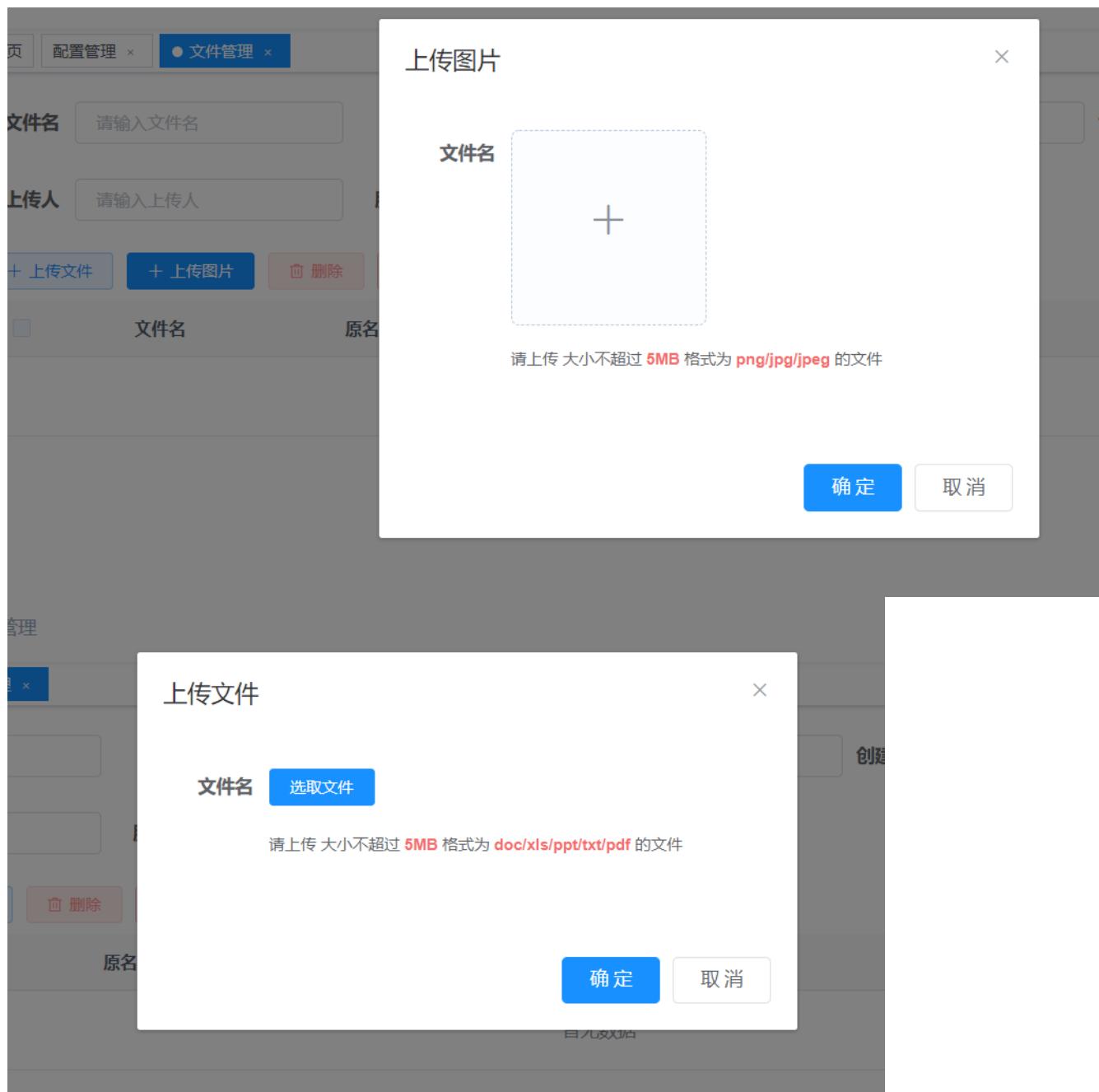
```
OssClient storage = OssFactory.instance(configKey: "image");
```

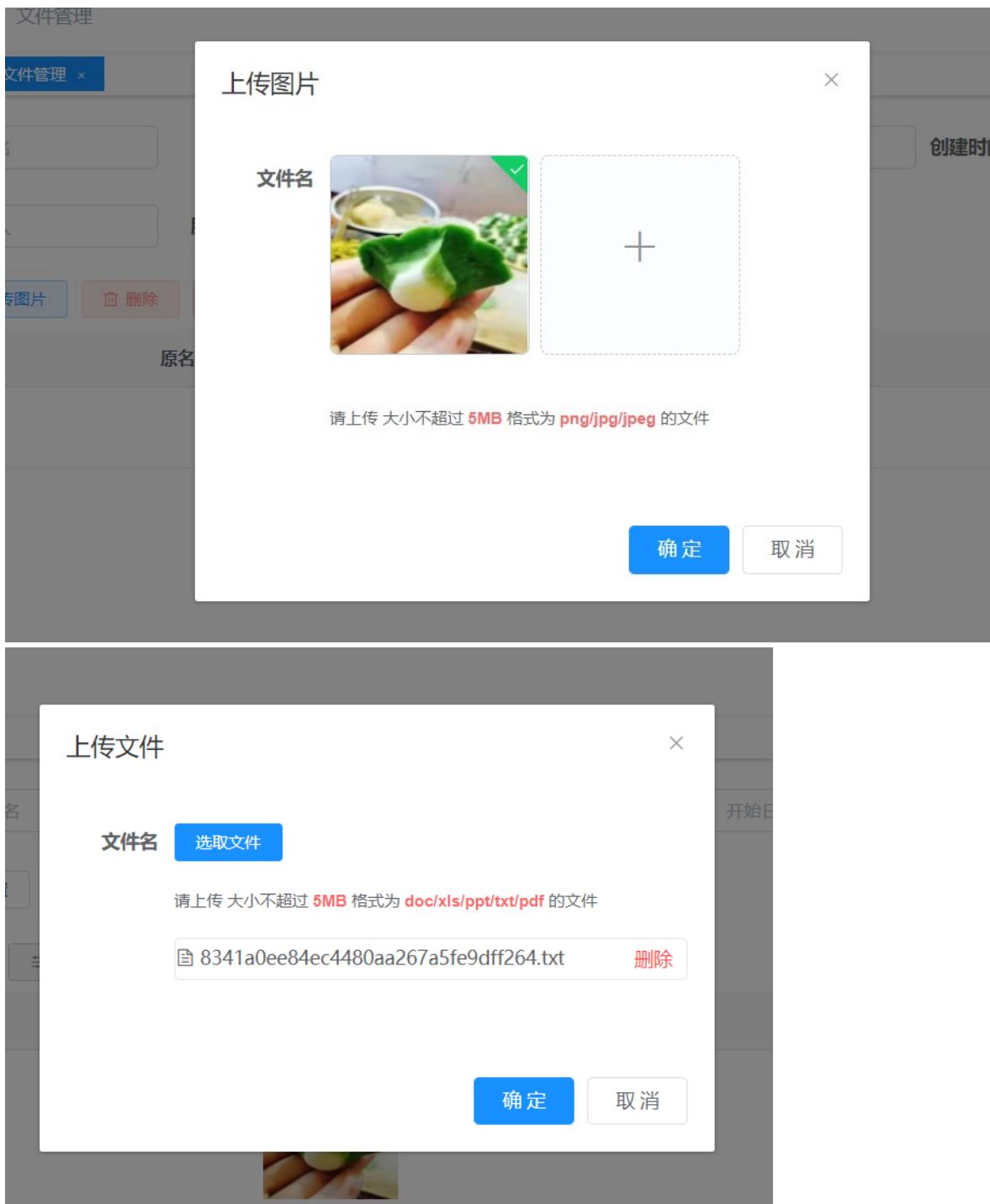
使用

OssFactory.instance("image") 获取的 OssClient 会加载上图的配置 从而达到上传不同的目录或桶

## 上传图片或文件

进入 系统管理 -> 文件管理 点击 上传文件 或 上传图片 根据选项选择即可 会对应上传到配置开启的OSS内





### 列表展示

Default display of images (with preview) - files will show their paths

## 关于OSS模块使用

<input type="checkbox"/>	文件名	原名	文件后缀	文件展示	创建时间	上传人
<input type="checkbox"/>	20210817/38a0fff1c5dc4 4e680fca57b1ba02428.j pg	test.jpg	.jpg		2021-08-17	admin
<input type="checkbox"/>	20210817/613380f4496 8462cb9ba76d994b9365 8.txt	test.txt	.txt	<p>http://qwoqr5oqq.hb-bkt. clouddn.com/20210817/ 613380f44968462cb9ba 76d994b93658.txt</p>	2021-08-17	admin

共 2 条 10:

原名

预览开关: 禁用 配置管理

原名	上传人
test.jpg	admin
test.txt	admin
test.txt	admin



Q Q [ ] C C

共 3 条 10

可以点击 预览禁用启用 按钮对是否展示进行更改

## 关于OSS模块使用

服务商

+ 上传文件 + 上传图片  预览开关: 禁用

<input type="checkbox"/>	文件名	原名	文件后缀	文件展示	创建时间
<input type="checkbox"/>	20210817/38a0fff1c5dc4 4e686fca57b1ba02428.j pg	test.jpg	.jpg		2021-08-17
<input type="checkbox"/>	20210817/613380f4496 8462cb9ba76d994b9365 8.txt	test.txt	.txt	http://qwoqr5oqq.hb-bkt. clouddn.com/20210817/ 613380f44968462cb9ba 76d994b93658.txt	2021-08-17
<input type="checkbox"/>	20210817/8341a0ee84e c4480aa267a5fe9dff264. txt	test.txt	.txt	http://qwoqr5oqq.hb-bkt. clouddn.com/20210817/ 8341a0ee84ec4480aa2 67a5fe9dff264.txt	2021-08-17

点击禁用后 图片会变成路径展示

服务商

+ 上传文件 + 上传图片  预览开关: 启用

<input type="checkbox"/>	文件名	原名	文件后缀	文件展示	创
<input type="checkbox"/>	20210817/38a0fff1c5dc4 4e686fca57b1ba02428.j pg	test.jpg	.jpg	http://qwoqr5oqq.hb-bkt. clouddn.com/20210817/ 38a0fff1c5dc44e686fca 57b1ba02428.jpg	202
<input type="checkbox"/>	20210817/613380f4496 8462cb9ba76d994b9365 8.txt	test.txt	.txt	http://qwoqr5oqq.hb-bkt. clouddn.com/20210817/ 613380f44968462cb9ba 76d994b93658.txt	202
<input type="checkbox"/>	20210817/8341a0ee84e c4480aa267a5fe9dff264. txt	test.txt	.txt	http://qwoqr5oqq.hb-bkt. clouddn.com/20210817/ 8341a0ee84ec4480aa2 67a5fe9dff264.txt	202

也可再 `参数设置` 更改预览状态 将 `OSS预览列表资源` 改为 `false` 即可关闭预览

首页

参数设置

参数名称: 请输入参数名称

参数键名: 请输入参数键名

系统内置

搜索 重置 新增 修改 删除 导出 刷新缓存

参数主键	参数名称	参数键名	参数键值	系统内
1	主框架页-默认皮...	sys.index.skinName	skin-blue	是
2	用户管理-账号初...	sys.user.initPassw...	123456	是
3	主框架页-侧边栏...	sys.index.sideThe...	theme-dark	是
4	账号自助-验证码...	sys.account.captc...	true	是
5	账号自助-是否开...	sys.account.regist...	false	是
11	OSS预览列表资源...	sys.oss.previewLi...	true	是

### 删除功能

点击列表上方或后方 **删除** 按钮 会根据OSS服务商类型 调用对应的删除(注意: 需确保对应的服务商配置正确) 可勾选多服务商类型的文件进行删除  
系统会自动判断

+ 上传文件 + 上传图片 **删除** 预览开关: 禁用 配置管理

文件名	原名	文件后缀	文件展示	创建时间	上传人	服务商	操作
20210817/38a0fff1c5dc44e686fca57b1ba02428.jpg	test.jpg	.jpg		2021-08-17	admin	qiniu	<a href="#">下载</a> <b>删除</b>
20210817/613380f44968462cb9ba76d994b93658.txt	test.txt	.txt	http://qwoqr5oqq.hbbkt.clouddn.com/20210817/613380f44968462cb9ba76d994b93658.txt	2021-08-17	admin	qiniu	<a href="#">下载</a> <b>删除</b>
20210817/47151c4e4bfe45ed88e482270d2e187d.txt	test.txt	.txt	http://ruoyi-vue-plus.oss-cn-beijing.aliyuncs.com/20210817/47151c4e4bfe45ed88e482270d2e187d.txt	2021-08-17	admin	aliyun	<a href="#">下载</a> <b>删除</b>

首页 / 系统管理 / 文件管理

参数设置 × ●文件管理 × 配置管理 ×

删除成功

件名  原名  文件后缀  创建时间

传人  服务商

上传文件   预览开关:禁用

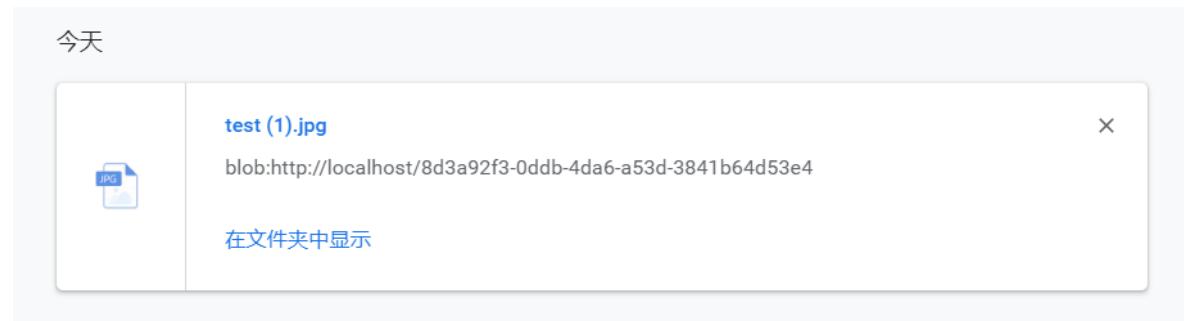
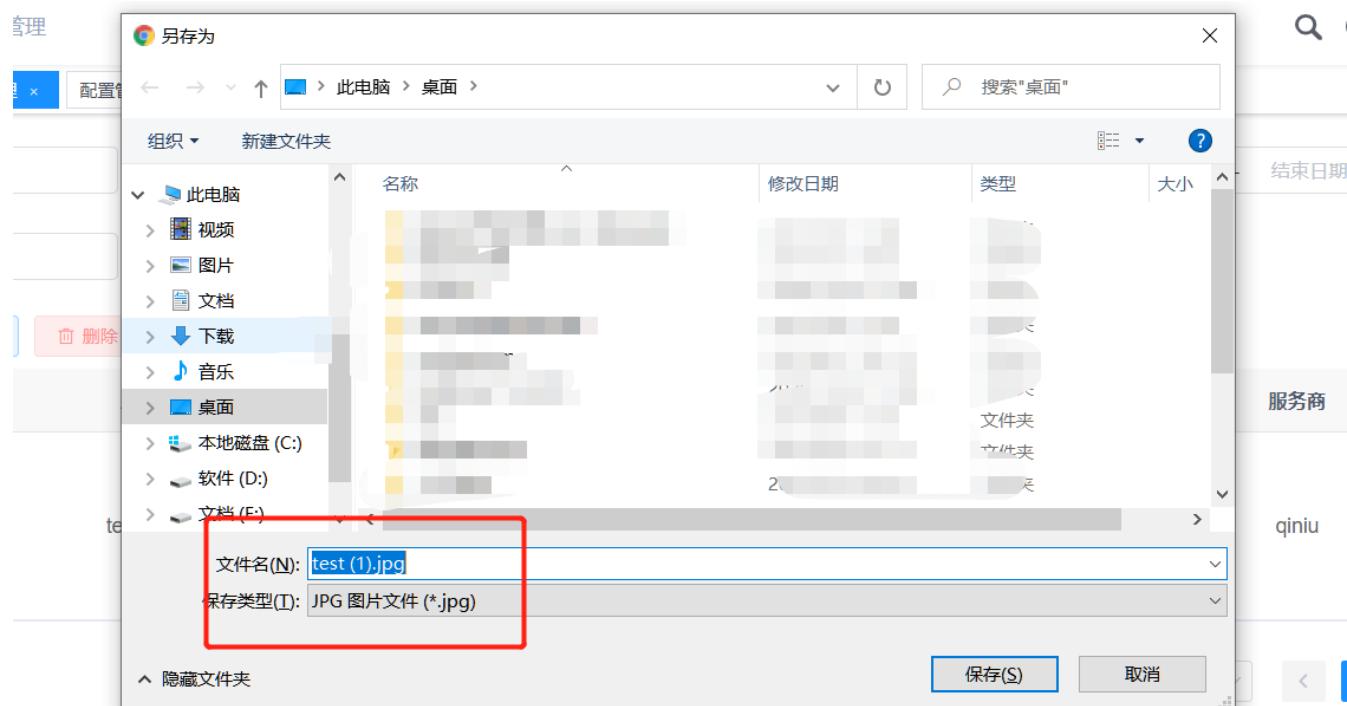
文件名	原名	文件后缀	文件展示	创建时间	上传人
20210817/38a0fff1c5 dc44e686fca57b1ba0 2428.jpg	test.jpg	.jpg		2021-08-17	admin

共 1 条

已删除

### 下载功能

点击列表后方对应资源的  按钮 根据需求填写文件名 点击确认即可完成下载



## 数据权限

### 关于数据权限

- 参考 demo 模块用法(需导入 test.sql 文件)

### 使用教程

数据权限功能:

1.支持自动注入 sql 数据过滤 2.查询、更新、删除 限制 3.支持自定义数据字段过滤 4.模板支持 spel 语法实现动态 Bean 处理

### 数据权限相关代码

类	说明	功能
DataScopeType	数据权限模板定义	用于定义数据权限模板
DataPermission	数据权限组注解	用于标注开启数据权限(默认过滤部门权限)
DataColumn	具体的数据权限字段标注	用于替换数据权限模板内的 key 变量
PlusDataPermissionInterceptor	数据权限 sql 拦截器	用于拦截所有 sql 检查是否标注了 DataPermission 注解
PlusDataPermissionHandler	数据权限处理器	用于处理被拦截到的 sql 为其添加数据权限过滤条件
DataPermissionHelper	数据权限助手	操作数据权限上下文变量
SysDataScopeService	自定义 Bean 处理数据权限	用于自定义扩展

使用方式 参考**demo**模块

数据权限体系 用户 -> 多角色 => 角色 -> 单数据权限

例子: 用户A 拥有两个角色 角色A 部门经理 可查看 本部门及以下部门的数据 角色B 兼职开发 可查看 仅自己的数据

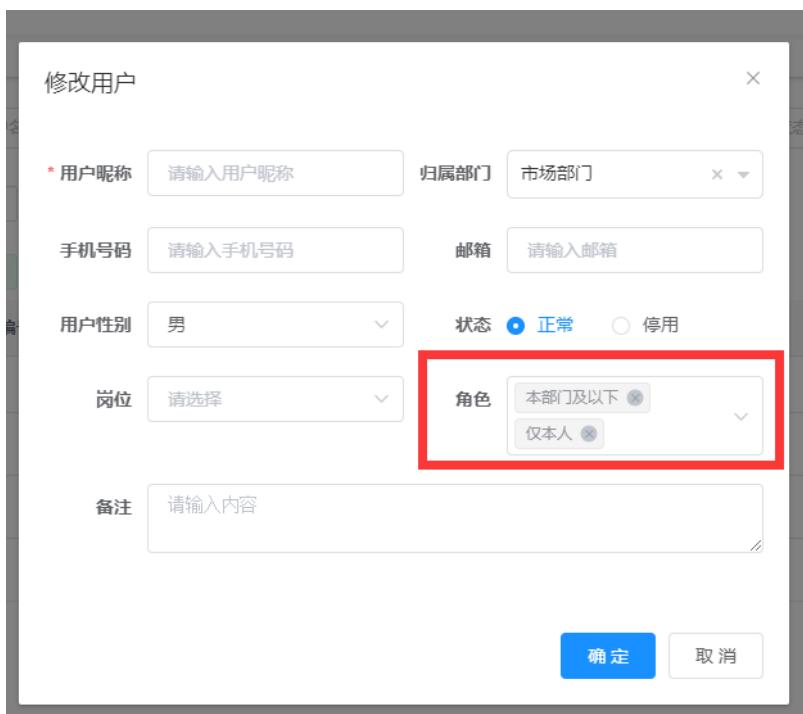
创建角色 test1 为 本部门及以下



创建角色 test2 为 仅本人



将其分配给用户 test



编写列表查询(注意: 数据权限注解只能在 Mapper 层使用)

标注数据权限注解 dept\_id 为过滤部门字段 user\_id 为过滤创建用户

```
/*
 * 测试单表Mapper接口
 *
 * @author Lion Li
 * @date 2021-07-26
 */
public interface TestDemoMapper extends BaseMapperPlus<TestDemo> {

    @DataPermission({
        @DataColumn(key = "deptName", value = "dept_id"),
        @DataColumn(key = "userName", value = "user_id")
    })
    Page<TestDemoVo> customPageList(@Param("page") Page<TestDemo> page, @Param("ew") Wrapper<TestDemo> wrapper);
}
```

重点注意: 如下情况不生效 有自定义实现方法 最终执行的mapper不是这个方法 所以无法生效



```
/*
 * @date 2021-07-26
 */
public interface TestDemoMapper extends BaseMapperPlus<TestDemoMapper, TestD

    2 usages  ↳ 疯狂的狮子Li *
    @DataPermission({
        @DataColumn(key = "deptName", value = "dept_id"),
        @DataColumn(key = "userName", value = "user_id")
    })
    default Page<TestDemoVo> customPageList(@Param("page") Page<TestDemo> pa
        return this.selectVoPage(page, wrapper);|  You, 2022/1/14 21:15 · Un
    }
}
```

## 编写数据权限模板

```
/*
 * 部门及以下数据权限
 */
DEPT_AND_CHILD(code: "4", sqlTemplate: "#{@deptName} IN ( #{@sdss.getDeptAndChild( #user.deptId )} )", elseSql:""),

/*
 * 仅本人数据权限
 */
SELF(code: "5", sqlTemplate: "#{@userName} = #{@user.userId} ", elseSql:"1 = 0");
```

1. code 为关联角色的数据权限 2. sqlTemplate 为 sql 模板 \${#deptName} 为模板变量 对应权限注解的 key \${@sdss} 为模板 Bean 调用 调用其 Bean 的处理方法 3. elseSql 为兜底 sql 处理当前角色与标注的注解 无对应的情况 例如 数据权限为仅本人 且 方法并未标注具体过滤注解 则 填充 1 = 0 使条件不满足 不允许查看 更详细用法可以参考 `DataScopeType` 注释

## 测试代码

使用 管理员 用户优先测试

```
2021-12-15 22:35:58 [XNIO-1 task-2] DEBUG c.r.f.i.PlusWebInvokeTimeInterceptor
- <0><9738843457186624> [PLUS]开始请求 => URL[GET /demo/demo/page], 参数类型[param], 参数:[{"pa
Consume Time: 16 ms 2021-12-15 22:35:58
Execute SQL: SELECT u.user_id, u.dept_id, u.user_name, u.nick_name, u.email, u.avatar, u.phon
Consume Time: 16 ms 2021-12-15 22:35:58
Execute SQL: SELECT COUNT(*) AS total FROM test_demo

Consume Time: 16 ms 2021-12-15 22:35:58
Execute SQL: SELECT * FROM test_demo LIMIT 10
```

```
2021-12-15 22:35:58 [XNIO-1 task-2] DEBUG c.r.f.i.PlusWebInvokeTimeInterceptor
- <0><9738843457186624> [PLUS]结束请求 => URL[GET /demo/demo/page], 耗时:[69]毫秒
```

使用 test 用户测试

```
2021-12-15 22:54:53 [XNIO-1 task-1] DEBUG c.r.f.i.PlusWebInvokeTimeInterceptor
- <0><9738917868202816> [PLUS]开始请求 => URL[GET /demo/demo/page], 参数类型[param], 参数:[{"pageSiz
Consume Time: 16 ms 2021-12-15 22:54:53
Execute SQL: SELECT u.user_id, u.dept_id, u.user_name, u.nick_name, u.email, u.avatar, u.phonenumber, u
自定义Bean执行
```

```
Consume Time: 16 ms 2021-12-15 22:54:53
Execute SQL: SELECT dept_id FROM sys_dept WHERE del_flag = '0' AND (dept_id = 108 OR find_in_set(108, a
```

```
Consume Time: 15 ms 2021-12-15 22:54:53
Execute SQL: SELECT COUNT(*) AS total FROM test_demo WHERE dept_id IN (108) OR user_id = 3
```

```
Consume Time: 15 ms 2021-12-15 22:54:53
Execute SQL: SELECT * FROM test_demo WHERE dept_id IN (108) OR user_id = 3 LIMIT 10
```

```
2021-12-15 22:54:53 [XNIO-1 task-1] DEBUG c.r.f.i.PlusWebInvokeTimeInterceptor
- <0><9738917868202816> [PLUS]结束请求 => URL[GET /demo/demo/page], 耗时:[88]毫秒
```

使用 test 删除一条不属于自己的数据

The screenshot shows a web application for managing department data. The table has columns: 部门id (dept\_id), 用户id (user\_id), 排序号 (sort\_order), key键 (key), 值 (value), and 创建时间 (create\_time). The row with dept\_id 108 and user\_id 4 is highlighted with a red box. Below the table, a terminal window displays the following SQL logs:

```

ime: 17 ms 2021-12-15 23:24:06
QL: SELECT u.user_id, u.dept_id, u.user_name, u.nick_name, u.email, u.avatar, u.phonenumber, u.password, u.se
ime: 15 ms 2021-12-15 23:24:06
QL: SELECT dept_id FROM sys_dept WHERE del_flag = '0' AND (dept_id = 108 OR find_in_set(108, ancestors))
ime: 32 ms 2021-12-15 23:24:06
QL: UPDATE test_demo SET del_flag = 2 WHERE id IN (4) AND del_flag = 0 AND dept_id IN (108) AND user_id = 3

```

sql执行行为不满足条件 不允许删除

使用 test 修改与删除同理 具体实现为 更新和删除方法 标注数据权限注解

```

@Override
@DataPermission({
    @DataColumn(key = "deptName", value = "dept_id"),
    @DataColumn(key = "userName", value = "user_id")
})
int updateById(@Param(Constants.ENTITY) TestDemo entity);

@Override
@DataPermission({
    @DataColumn(key = "deptName", value = "dept_id"),
    @DataColumn(key = "userName", value = "user_id")
})
int deleteBatchIds(@Param(Constants.COLLECTION) Collection<? extends Serializable> idList);
}

```

## 自定义SQL模板

1.首先在角色管理 数据权限下拉框 添加自定义模板 为什么不放置到系统字典问题: 因数据权限与模板绑定 不应随意改动 最好事先定义好



2.代码 DataScopeType 自定义一个SQL模板

```
/**  
 * 自定义模板 Demo  
 */  
Demo_(code: "6", sqlTemplate: " #{#demoKey} LIKE '%#{#demoValue}' ", elseSql: "");
```

3.标注权限注解

```
@PreAuthorize("@ss.hasPermi('demo:demo:list')")  
GetMapping("/page")  
public TableDataInfo<TestDemoVo> page(@Validated(QueryGroup.class) TestDemoQuery query,  
                                         DataPermissionHelper.setVariable("demoValue", "测试数据");  
                                         return iTestDemoService.customPageList(po, pageQuery);  
}
```

4.设置数据权限变量

```
- <0><1471849216900530176> [PLUS]开始请求 => URL[GET /demo/demo/page], 参数类型[param], 参数值[{"test_key": "%测试数据%"}]
Consume Time: 17 ms 2021-12-17 22:26:13
Execute SQL: select u.user_id, u.dept_id, u.user_name, u.nick_name, u.email, u.avatar, u.create_time, u.update_time from test_demo u where (test_key like '%测试数据%') limit 10
Consume Time: 22 ms 2021-12-17 22:26:13
Execute SQL: SELECT COUNT(*) AS total FROM test_demo WHERE (test_key LIKE '%测试数据%')
Consume Time: 16 ms 2021-12-17 22:26:13
Execute SQL: SELECT * FROM test_demo WHERE (test_key LIKE '%测试数据%') LIMIT 10
2021-12-17 22:26:13 [XNIO-1 task-1] DEBUG c.r.f.i.PlusWebInvokeTimeInterceptor
- <0><1471849216900530176> [PLUS]结束请求 => URL[GET /demo/demo/page], 耗时:[65]毫秒
5.测试
```

### mybatis-plus 原生方法 增加数据权限过滤

首先查看需要重写的方法源码 重点 方法源码 | 方法源码 | 方法源码 例如重写 selectPage 方法

```
1 /**
2  * 根据 entity 条件，查询全部记录（并翻页）
3  *
4  * @param page      分页查询条件（可以为 RowBounds.DEFAULT）
5  * @param queryWrapper 实体对象封装操作类（可以为 null）
6  */
7   <P extends IPage<T>> P selectPage(P page, @Param(Constants.WRAPPER) Wrapper<T> queryWrapper);
```

复制源码到自己的 Mapper 并增加数据权限注解 注意左边出现重写图标 即为重写成功

```
31
32     @Override
33     @DataPermission({
34         @DataColumn(key = "deptName", value = "dept_id"),
35         @DataColumn(key = "userName", value = "user_id")
36     })  疯狂的狮子Li, 2021/12/13 11:49 • update [重大更新] 重写数据权限实现
37     <P extends IPage<TestDemo>> P selectPage(P page, @Param(Constants.WRAPPER) Wrapper<TestDemo> queryWrapper);
38
39     @Override
```

### 支持类标注

获取规则 方法 > 类 注意: 类标注后 所有方法(包括父类方法) 都会进行数据权限过滤

```
/*
 * 测试树表Mapper接口
 *
 * @author Lion Li
 * @date 2021-07-26
 */
@DataPermission({})
    @DataColumn(key = "deptName", value = "dept_id"),
    @DataColumn(key = "userName", value = "user_id")
}
public interface TestTreeMapper extends BaseMapperPlus<Te
```

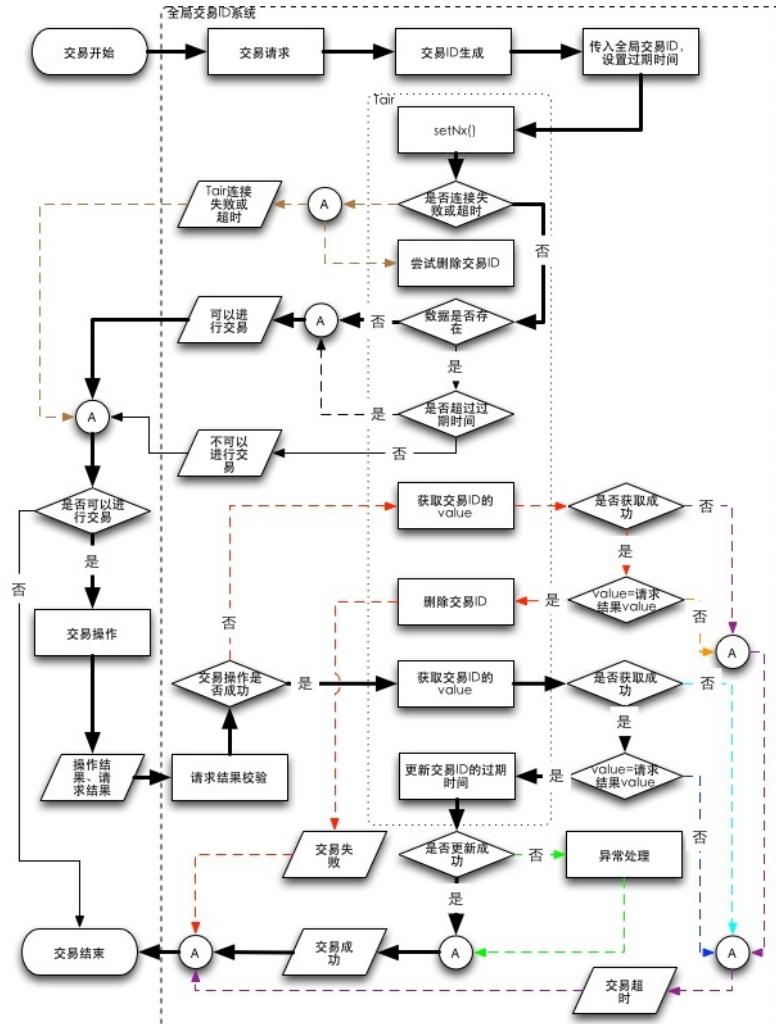
防重幂等

# 功能介绍

防重功能为防止两条相同的数据重复提交导致脏数据或业务错乱 注意: 重复提交属于小概率事件 请不要拿并发压测与之相提并论 框架防重功能参考 美团GTIS防重系统 使用 请求参数与用户Token或URL 生成全局业务ID 有效防止 同一个用户 在 限制时间 内对 同一个业务 提交 相同的数据

框架防重处理 支持业务失败或异常 快速释放限制 业务处理成功后 会在设置时间内 限制同一条数据的提交 注意: 只对同一个用户的同一个接口提交相同的数据有效

## 美团GTIS系统流程图



美团 分布式系统互斥性与幂等性问题的分析与解决

使用方法

在Controller标注 @RepeatSubmit 注解即可

```

/**
 * 自定义注解防止表单重复提交
 *
 * @author Lion.Li
 */
@Inherited
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface RepeatSubmit {

    /**
     * 间隔时间(ms), 小于此时间视为重复提交
     */
    int interval() default 5000;

    TimeUnit timeUnit() default TimeUnit.MILLISECONDS;

    /**
     * 提示消息
     */
    String message() default "{repeat.submit.message}"; You, A minute
}

/**
 * 新增测试单表
 */
@ApiOperation("新增测试单表")
@SaCheckPermission("demo:demo:add")
@Log(title = "测试单表", businessType = BusinessType.INSERT)
@RepeatSubmit(interval = 2, timeUnit = TimeUnit.SECONDS, message = "不允许重复提交")
@PostMapping()
public R<Void> add(@RequestBody TestDemoBo bo) {
    // 使用校验工具对标注 @Validated(AddGroup.class) 注解
    // 用于在非 Controller 的地方校验对象
    ValidatorUtils.validate(bo, AddGroup.class);
    return toAjax(iTestDemoService.insertByBo(bo) ? 1 : 0);
}
*/

```

## 数据脱敏

### 功能说明

系统使用 Jackson 序列化策略 对标注了 Sensitive 注解的属性进行脱敏处理

### 使用教程

使用注解标注需要脱敏的字段 选择对应的策略

```
    /**
     * 用户邮箱
     */
    @Sensitive(strategy = SensitiveStrategy.EMAIL)
    @ApiModelProperty(value = "用户邮箱")
    @Email(message = "邮箱格式不正确")
    @Size(min = 0, max = 50, message = "邮箱长度不能超过50个字符")
    private String email;

    /**
     * 手机号码
     */
    @Sensitive(strategy = SensitiveStrategy.PHONE)
    @ApiModelProperty(value = "手机号码")
    private String phonenumber;
```

可再 SensitiveStrategy 内自定义策略

```
    /**
     * 脱敏策略
     *
     * @author Yjoioooo
     * @version 3.6.0
     */
    13 usages  ↳ 疯狂的狮子li +1
    @AllArgsConstructor
    public enum SensitiveStrategy {

        /**
         * 身份证脱敏 疯狂的狮子li, 2021/12/28 12:03 · add 增加邮箱与银行卡脱敏策略
         */
        1 usage
        ID_CARD(s -> DesensitizedUtil.idCardNum(s, front: 3, end: 4)),

        /**
         * 手机号脱敏
         */
        2 usages
        PHONE(DesensitizedUtil::mobilePhone),

        /**
         * 地址脱敏
         */
    }
```

## 脱敏逻辑修改

系统使用通用接口处理是否需要脱敏 多个系统可以自定义不同的脱敏逻辑实现

```
/*  
 * 脱敏服务  
 * 默认管理员不过滤  
 * 需自行根据业务重写实现  
 *  
 * @author Lion.Li 疯狂的狮子li, 2021/12/28 11  
 * @version 3.6.0  
 */  
public interface SensitiveService {  
  
    /**  
     * 是否脱敏  
     */  
    boolean isSensitive();  
}
```

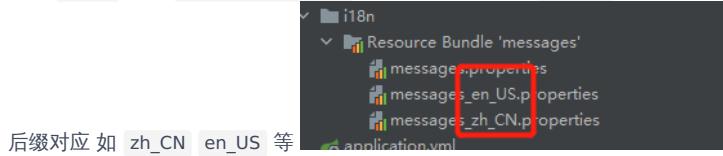
系统默认处理逻辑为 非管理员脱敏 可自行修改默认实现

```
/*  
 * 脱敏服务  
 * 默认管理员不过滤  
 * 需自行根据业务重写实现  
 *  
 * @author Lion.Li  
 * @version 3.6.0  
 */  
1 usage 疯狂的狮子li  
@Service  
public class SysSensitiveServiceImpl implements SensitiveService {  
  
    /**  
     * 是否脱敏  
     */  
    1 usage 疯狂的狮子li  
    @Override  
    public boolean isSensitive() { 疯狂的狮子li, 2021/12/28 11:51 · upd  
        return !LoginHelper.isAdmin();  
    }  
}
```

## 国际化

### 国际化方案

- 前端国际化参考 ruoyi前端国际化文档
- 参考 demo 模块 TestI18nController 国际化演示案例 在 Header 请求头 增加上下文语言参数 content-language 参数需与国际化配置文件



### 获取 code 对应国际化内容

请求头	内容
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9eyJsb2dpbl91c2VyX2tleSf6lBjZTE
content-language	zh_CN
请求头名称	请求头内容

```

1: {
2:   "code": 200,
3:   "msg": "注册成功",
4:   "data": null
5: }
    
```

响应内容 消息状态码  
消息内容

The screenshot displays two main sections of the RuoYi-Vue-Plus backend management system:

- Left Sidebar:** Shows the system navigation menu with categories like 首页, 系统管理, 系统监控, 系统工具, 表单构建, 代码生成, 系统接口, PLUS官网, and 测试菜单.
- Right Main Area:** Displays the "标题: RuoYi-Vue-Plus后台管理系统\_接口文档" (Title: RuoYi-Vue-Plus Backend Management System\_Interface Documentation).

In the main area, there are three tabs: 主页 (Home), Validator 校验国际化 (Validator Validation Internationalization), and 通过code获取国际化内容 (Get Internationalization Content via Code). The Validator tab is active.

A specific API endpoint is shown: **GET /dev-api/demo/i18n**. The request parameters section shows a parameter named "code" with the value "user.register.success".

The response content is displayed in a code editor-like pane:

```

1 | {
2 |   "code": 200,
3 |   "msg": "注册成功",
4 |   "data": null
5 |

```

Below the response content, there are tabs for 响应内容 (Response Content), Raw, Headers, and Curl. On the right side, there are buttons for 消息状态 (Message Status) and 消息内容 (Message Content).

使用 **Validator** 框架校验 **controller** 参数返回国际化

```

    /**
     * 测试国际化
     *
     * @author Lion Li
     */
    @Validated
    @ApiValue("测试国际化控制器", tags = {"疯狂的狮子"})
    @RestController
    @RequestMapping("/demo/i18n")
    public class TestI18nController {

```

controller 校验接口参数 需要在类增加 `@Validated` 注解  
用 `{code}` 形式标注使用国际化处理

```

    /**
     * Validator 校验国际化
     * 不传值 分别查看异常返回
     *
     * 测试使用 not.null
     */
    @ApiOperation("Validator 校验国际化")
    @GetMapping("/test1")
    public AjaxResult<Void> test1(@NotBlank(message = "not.null") String str) {
        return AjaxResult.success(str);
    }

```

The screenshot shows the RuoYi-Vue-Plus backend management system's API documentation interface. On the left, the navigation bar includes '首页', '系统管理', '系统监控', '系统工具' (selected), '表单构建', '代码生成', '系统接口' (selected), 'PLUS官网', and '测试菜单'. Under '系统工具', there are sections for 'Authorize', 'Swagger Models', '文档管理', 'Redis发布订阅', 'spring-cache 演示案例', '测试分布式锁的样例', '测试分布式限流样例', '测试单表管理', and '测试国际化管理'. The '测试国际化管理' section is currently selected. Below it, three API endpoints are listed: '通过code获取国际化内容' (GET), 'Validator 校验国际化' (selected, shown in blue), and 'Bean 校验国际化' (GET). The main content area displays the 'Validator 校验国际化' API details. It shows a '请求头' tab with 'Authorization' set to 'Bearer eyJhbGciOiJIUzUxMiJ9.eyJsb2dpbl91c2VyX2tleSI6IJBZTE2...' and 'content-language' set to 'en\_US'. The '响应内容' tab shows a JSON response: 

```

1 | {
2 |   "code": 500,
3 |   "msg": "* Required fill in",
4 |   "data": null
5 |

```

The screenshot shows the RuoYi-Vue-Plus backend management system's interface. On the left, there is a dark sidebar with various menu items such as 首页, 系统管理, 系统监控, 系统工具, 表单构建, 代码生成, 系统接口, PLUS官网, and 测试菜单. Under 测试菜单, there are several options like 通过code获取国际化内容, Validator 校验国际化 (which is highlighted in blue), Bean 校验国际化, and 测试批量方法.

The main content area has a title bar: 标题: RuoYi-Vue-Plus后台管理系统\_接口文档. Below it is a sub-header: Validator 校验国际化 X. There are tabs for 主页, 文档, 调试, and Open. The 调试 tab is selected, showing a GET request to /dev-api/demo/i18n/test1. The request header section shows 'content-language' set to 'zh\_CN'. The response content section shows a JSON object:

```

1 [
2   "code": 500,
3   "msg": "* 必须填写",
4   "data": null
5 ]

```

## 使用 Validator 框架校验 Bean 返回国际化

Bean 校验需要在接口校验 Bean 参数使用 @Validated 注解

```

    /**
     * Bean 校验国际化
     * 不传值 分别查看异常返回
     *
     * 测试使用 not.null
     */
    @ApiOperation("Bean 校验国际化")
    @GetMapping("/test2")
    public AjaxResult<TestI18nBo> test2(@Validated TestI18nBo bo) {
        return AjaxResult.success(bo);
    }

    @Data
    public static class TestI18nBo {

        @NotBlank(message = "{not.null}")
        private String name;

        @NotNull(message = "{not.null}")
        @Range(min = 0, max = 100, message = "{length.not.valid}")
        private Integer age;
    }
}

```

Bean 内属性校验注解 使用 {code} 形

式标注使用国际化处理

The screenshot shows the RuoYi-Vue-Plus backend management system's interface. On the left, there is a navigation sidebar with various modules like System Management, System Monitoring, System Tools, Form Construction, Code Generation, and System Interfaces. Under the 'System Interfaces' section, there are several examples: Redis publishing subscription, spring-cache demonstration example, testing distributed lock patterns, testing distributed flow patterns, testing table management, testing internationalization management, and the current example being tested.

The main content area displays the 'Swagger Models' documentation for the 'Validator 校验国际化' section. It includes a 'GET /dev-api/demo/i18n/test2' endpoint. The 'Headers' tab is selected, showing a table with two rows: 'Authorization' (containing 'Bearer eyJhbGciOiJIUzUxMiJ9.eyJsb2d...' and checked) and 'content-language' (containing 'zh\_CN' and checked). A red box highlights this table.

Below the headers, the '响应内容' (Response Content) tab is selected, showing a JSON response:

```

1 [
2   "code": 500,
3   "msg": "长度必须在0到100个字符之间",
4   "data": null
5 ]

```

The screenshot shows the RuoYi-Vue-Plus backend management system's sidebar menu. The 'System Interface' section is expanded, displaying various API endpoints:

- Validator 校验国际化
- Bean 校验国际化
- 测试批量方法
- GET 通过code获取国际化内容
- GET Validator 校验国际化
- GET Bean 校验国际化 (highlighted in blue)

## 三 标题: RuoYi-Vue-Plus后台管理系统\_接口文档

The screenshot shows the RuoYi-Vue-Plus API documentation interface. A GET request is being tested for the endpoint `/dev-api/demo/i18n/test2`. The '请求头部' (Request Headers) section is highlighted with a red box, showing the following headers:

请求头	内容
Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJj...
content-language	en_US
请求头名称	请求头内容

The '响应内容' (Response Content) section shows the JSON response:

```

1 | {
2 |   "code": 500,
3 |   "msg": "The length must be between 0 and 100 characters",
4 |   "data": null
5 | }

```

## 短信模块

### 配置功能

短信模块采用SPI加载 使用哪家的短信 引入哪家的依赖 即可动态加载 目前支持: 阿里云 腾讯云 欢迎扩展PR其他

```

5
6 <dependency>
7   <groupId>com.ruoyi</groupId>
8   <artifactId>ruoyi-sms</artifactId>
9 </dependency>

10
11 <!-- 短信 用哪个导入哪个依赖 -->
12 <dependency>-->
13 <groupId>com.aliyun</groupId>-->
14 <artifactId>dysmsapi20170525</artifactId>-->
15 </dependency>-->

16
17 <dependency>
18   <groupId>com.tencentcloudapi</groupId>
19   <artifactId>tencentcloud-sdk-java</artifactId>
20 </dependency>  疯狂的狮子li, Yesterday • add 增加 demo 短信演示案例
21

```

修改配置文件

```

# sms 短信
sms:
  enabled: false  疯狂的狮子li, Yesterday
  # 阿里云 dysmsapi.aliyuncs.com
  # 腾讯云 sms.tencentcloudapi.com
  endpoint: "dysmsapi.aliyuncs.com"
  accessKeyId: XXXXXXXX
  accessKeySecret: XXXXXX
  signName: 测试
  # 腾讯专用
  sdkAppId:

```

- `enabled` 为短信功能开关
- `endpoint` 为域名 各厂家域名固定 按照文档配置即可
- `accessKeyId` 密钥id
- `accessKeySecret` 密钥密匙
- `signName` 签名
- `sdkAppId` 应用id 腾讯专用

## 功能使用

参考 `demo` 模块 `SmsController` 短信演示案例 功能采用 模板模式 动态加载对应厂家的工具模板 引入 `SmsTemplate` 即可使用

```

@RestController
@RequestMapping("/demo/sms")
public class SmsController {

    2 usages
    private final SmsTemplate smsTemplate;

    ▲ 疯狂的狮子li
    @ApiOperation("发送短信Aliyun")
    @GetMapping("/sendAliyun")
    public R<Object> sendAliyun(@ApiParam("电话号") String phones,
                                  @ApiParam("模板ID") String templateId) {
        Map<String, String> map = new HashMap<>(initialCapacity: 1);
        map.put("code", "1234");
        Object send = smsTemplate.send(phones, templateId, map);
        return R.ok(send);
    }

    ▲ 疯狂的狮子li
    @ApiOperation("发送短信Tencent")
    @GetMapping("/sendTencent")
    public R<Object> sendTencent(@ApiParam("电话号") String phones,
                                 @ApiParam("模板ID") String templateId) {
        Map<String, String> map = new HashMap<>(initialCapacity: 1);
        // map.put("2", "测试测试");
        map.put("1", "1234");
        Object send = smsTemplate.send(phones, templateId, map);
        return R.ok(send);
    }
}

```

## 重点须知

由于各厂家参数解析不一致 请遵守以下规则

```

/**
 * 发送短信
 *
 * @param phones 电话号(多个逗号分割)
 * @param templateId 模板id
 * @param param 模板对应参数
 *          阿里 需使用 模板变量名称对应内容 例如: code=1234
 *          腾讯 需使用 模板变量顺序对应内容 例如: 1=1234, 1为模板内第一个参数
 */
2 usages 2 implementations ▲ 疯狂的狮子li
SmsResult send(String phones, String templateId, Map<String, String> param); 疯狂的狮子li

```

## 邮件功能

### 配置功能

修改配置文件

```
--- # mail 邮件发送
mail:
  enabled: false
  host: smtp.163.com
  port: 465
  # 是否需要用户名密码验证
  auth: true
  # 发送方, 遵循RFC-822标准
  from: xxx@163.com
  # 用户名 (注意: 如果使用foxmail邮箱, 此处user为qq号)
  user: xxx@163.com
  # 密码 (注意: 某些邮箱需要为SMTP服务单独设置密码, 详情查看相关帮助)
  pass: XXXXXXXXXXXX
  # 使用STARTTLS安全连接, STARTTLS是对纯文本通信协议的扩展。
  starttlsEnable: true
  # 使用SSL安全连接
  sslEnable: true
  # SMTP超时长, 单位毫秒, 缺省值不超时
  timeout: 0
  # Socket连接超时值, 单位毫秒, 缺省值不超时
  connectionTimeout: 0
```

- `enabled` 为邮件功能开关

## 功能使用

参考 `demo` 模块 `MailController` 邮件演示案例

```
evaluated
@RequiredArgsConstructor
@RestController
@RequestMapping("/demo/mail")
public class MailController { Zlyx, 2022/5/2 21:21 · [add]:
```

```
    /**
     * 发送邮件
     *
     * @param to 接收人
     * @param subject 标题
     * @param text 内容
     */
    @GetMapping("/sendSimpleMessage")
    public R<Void> sendSimpleMessage(String to, String subject, String text) {
        MailUtils.sendText(to, subject, text);
        return R.ok();
    }

    /**
     * 发送邮件（带附件）
     *
     * @param to 接收人
     * @param subject 标题
     * @param text 内容
     * @param filePath 附件路径
     */
    @GetMapping("/sendMessageWithAttachment")
    public R<Void> sendMessageWithAttachment(String to, String subject, String text, String filePath) {
        MailUtils.sendText(to, subject, text, new File(filePath));
    }
}
```

新增路由

## 网关路由与放行

### 新增路由

ruoyi-gateway.yml 配置文件 增加 routers 配置 注意: 路径格式为 /服务路径/controller路径/接口方法路径 \*代表任意一级 \*\*代表任意所有级 下图代表 resource/\*\* 将所有 resource开头的路径 都路由到 ruoyi-resource 服务 例如: /resource/sms/code resource路由到ruoyi-resource服务 sms路由

```
# 资源服务
- id: ruoyi-resource
  uri: lb://ruoyi-resource
  predicates:
    - Path=/resource/**
  filters:
    - StripPrefix=1
```

到对应的controller code 路由到对应的接口

```
/*
 * 疯狂的狮子li +1
 */
@Slf4j
@Validated
@RequiredArgsConstructor
@RestController
@RequestMapping("/sms")
public class SysSmsController extends BaseController {

    /**
     * 短信验证码
     *
     * @param phonenumber 用户手机号
     */
    @GetMapping("/code")
    public R<Void> smsCaptcha(@NotBlank(message = "...") String phone)
        if (!smsProperties.getEnabled()) {
            return R.fail(msg: "当前系统没有开启短信功能!");
        }
}
```

### 放行使用方式

nacos 中 ruoyi-gateway.yml 白名单放行 注意: 放行路径格式为 /服务路径/controller路径/接口方法路径 \*代表任意一级 \*\*代表任意所有级 示例:

```
15 # 防止xss攻击
16 xss:
17   | enabled: true
18   | excludeUrls:
19   |   - /system/notice
20   # 不校验白名单
21   ignore:
22     whites:
23       - /code
24       - /auth/logout
25       - /auth/login
26       - /auth/smsLogin
27       - /auth/xcxLogin
28       - /auth/register
29       - /resource/sms/code
30       - /*/v3/api-docs
31       - /csrf
```

/resource/sms/code 代表 ruoyi-resource服务 sms的controller code接口

## 代码生成

# 功能介绍

### 数据源配置

```
# 库数据源
master:
    driverClassName: com.mysql.cj.jdbc.Driver
    # jdbc 所有参数配置参考 https://lionli.blog.csdn.net/article/details/118330000
    # rewriteBatchedStatements=true 批处理优化 大幅提升批量插入更新删除性能
    url: jdbc:mysql://localhost:3306/ry-vue?useUnicode=true&characterSetResults=utf8
    username: root
    password: root
# 从库数据源
slave:
    lazy: true
    driverClassName: com.mysql.cj.jdbc.Driver
    url:
    username:
    password:
# oracle:
#     driverClassName: oracle.jdbc.OracleDriver
#     url: jdbc:oracle:thin:@//localhost:1521/XE
#     username: ROOT
#     password: root
# druid:
#     validationQuery: SELECT 1 FROM DUAL
# postgres:
#     driverClassName: org.postgresql.Driver
```

后续版本 项目适配多种类型数据库 可以在代码生成页面切换 填写对应的数据源名称 点击搜索按钮 即可切换到对应的数据源

数据源 master

表名称

表描述

Q 搜索
Q 重置

+ 生成
导入
修改
X 删除

序号	表名称	表描述	实体	创建时间
				暂无

### 导入数据表

The screenshot shows a user interface for managing database tables. At the top, there are input fields for '数据源' (DataSource) set to 'master' and '表名称' (Table Name) with a placeholder '请输入表名'. Below these are buttons for '搜索' (Search), '重置' (Reset), '生成' (Generate), and a large red-bordered button labeled '导入' (Import). A sub-menu below the main menu includes '序号' (Index), '表名称' (Table Name), '表描述' (Table Description), and '实体' (Entity). A tooltip message '点击导入按钮 会加载系统数据库所有的表' (Click the import button to load all tables in the system database) is displayed above the 'Import' button.

选择需要的表 点击确定即可

导入表

表名称 表描述

表名称	表描述	创建时间	更新时间
<input type="checkbox"/> sys_dept	部门表	2022-04-15 11:48:55	
<input checked="" type="checkbox"/> test_tree	测试树表	2022-03-24 09:55:56	2022-03-24 09:55:56
<input checked="" type="checkbox"/> test_demo	测试单表	2022-03-24 09:55:56	2022-03-24 09:58:11
<input type="checkbox"/> sys_dict_type	字典类型表	2022-03-24 09:55:45	2022-03-24 09:55:45
<input type="checkbox"/> sys_logininfor	系统访问记录	2022-03-24 09:55:45	2022-04-21 11:07:03

共 19 条 10条/页 < 1 2 > 前往 1 页

## 功能介绍

首页 / 系统工具 / 代码生成

操作成功

序号	表名称	表描述	实体	创建时间	更新时间
1	test_demo	测试单表	TestDemo	2022-03-24 09:55:56	2022-03-24 09:55:56
2	test_tree	测试树表	TestTree	2022-03-24 09:55:56	2022-03-24 09:55:56

## 编辑表生成结构

点击表对应的编辑按钮

序号	表名称	表描述	实体	创建时间	更新时间	操作
1	test_demo	测试单表	TestDemo	2022-03-24 09:55:56	2022-03-24 09:55:56	<input type="radio"/> 预览 <input checked="" type="button"/> 编辑 <input type="button"/> 删除 <input type="checkbox"/> 同步 <input type="button"/> 生成代码
2	test_tree	测试树表	TestTree	2022-03-24 09:55:56	2022-03-24 09:55:56	<input type="radio"/> 预览 <input checked="" type="button"/> 编辑 <input type="button"/> 删除 <input type="checkbox"/> 同步 <input type="button"/> 生成代码

更改要生成表的数据

## 功能介绍

首页 代码生成 × ● 修改[test\_demo]生成配置 ×

基本信息 字段信息 生成信息

\* 表名称: test\_demo

\* 表描述: 测试单表

\* 实体类名称: TestDemo

\* 作者: Lion Li

备注:

**提交** **返回**

基本信息 字段信息 生成信息

\* 生成模板: 单表 (增删改查)

\* 生成包路径: com.ruoyi.demo

\* 生成模块名: system

\* 生成业务名: demo

\* 生成功能名: 测试单表

上级菜单: 测试菜单

生成代码方式:  zip压缩包  自定义路径

**提交** **返回**

生成条件影响

## 功能介绍

首页 代码生成 × ● 修改[test\_demo]生成配置 ×

基本信息 字段信息 生成信息

序号	字段列名	字段描述	物理类型	Java类型	java属性	插入	编辑	列表	查询	查询方式	必填	显示类型	字典类型
1	id	主键	bigint	Long	id	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>	文本框	请选择
2	dept_id	部门id	bigint	Long	deptId	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	=	<input checked="" type="checkbox"/>	文本框	请选择
3	user_id	用户id	bigint	Long	userId	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	=	<input checked="" type="checkbox"/>	文本框	请选择
4	order_num	排序号	int	Long	orderNum	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	=	<input checked="" type="checkbox"/>	文本框	请选择
5	test_key	key键	varchar(255)	String	testKey	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	=	<input checked="" type="checkbox"/>	文本框	请选择
6	value	值	varchar(255)	String	value	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	=	<input checked="" type="checkbox"/>	文本框	请选择
7	version	版本	int	Long	version	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	=	<input type="checkbox"/>	文本框	请选择
8	create_time	创建时间	datetime	Date	createTime	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	=	<input type="checkbox"/>	日期控件	请选择

**提交** **返回**

- 插入 影响生成 BO 类是否有该字段
- 列表 影响生成 VO 类是否有该字段
- 查询 影响页面是否有该字段的搜索框与后端代码是否生成对应的查询条件
- 查询方式 影响生成查询条件的类型
- 必填 影响 BO 类 与 页面是否强制校验
- 显示类型 影响生成页面使用何种展示组件
- 字典类型 影响页面是否生成字段关联

## 树表配置

编辑表生成信息 生成模板为 树表 填写对应数据即可

首页 代码生成 × ● 修改[test\_tree]生成配置 ×

基本信息 字段信息 生成信息

\* 生成模板 树表 (增删改查) \* 生成包路径 com.ruoyi.demo

\* 生成模块名 system \* 生成业务名 tree

\* 生成功能名 测试树表 \* 上级菜单 测试菜单

生成代码方式  zip压缩包  自定义路径

其他信息

树编码字段 id: 主键 树父编码字段 parent\_id: 父id

树名称字段 tree\_name: 值

**提交** **返回**

## 主子表说明

框架不支持也不推荐使用主子表 原因一般业务场景 基本都是一对N表 多表关联场景 还有一些 主 => 子 <= 主 场景 需求很复杂 很少有单纯主子表场

## 功能介绍

景出现 另外主子表关联 很容易出现 笛卡尔积 或者数据错乱等问题 需要自行sql调优场景 所有建议大家都按照 单表生成 自行编写业务逻辑

## 预览功能



配置好生成信息后 可以点击预览按钮

共 2 条 10条/页

系统会根据已经配置好的数据 生成对应的代码预览

### 代码预览

domain.java vo.java bo.java mapper.java service.java serviceimpl.java controller.java mapper.xml sql api.js index.vue

```
package com.ruoyi.system.domain;

import com.baomidou.mybatisplus.annotation.*;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.io.Serializable;
import java.util.Date;
import java.math.BigDecimal;

import com.ruoyi.common.core.domain BaseEntity;

/**
 * 测试单对象 test_demo
 *
 * @author ruoyi
 * @date 2022-04-21
 */
@Data
@TableName("test_demo")
public class TestDemo extends BaseEntity {
```

可以再此处观察代码的生成结构和数据是否正确等

## 代码结构同步

实际开发中 难免会有表结构更改的需求 这时可以使用 同步功能 点击同步按钮 即可与实时数据库表进行字段同步



## 接口文档

版本 >= **1.2.0**

## 说明

由于 `springfox` 与 `knife4j` 均停止维护 bug众多 故从 1.2.0 开始 迁移到 `springdoc` 框架 基于 `javadoc` 无注解零入侵生成规范的 `openapi` 结构体 由于框架自带文档UI功能单一扩展性差 故移除自带UI 建议使用外置文档工具

## 文档工具使用

由于框架采用 `openapi` 行业规范 故市面上大部分的框架均支持 可自行选择 例如: `apifox` `postman` `torna` 等 根据对应工具的文档接入即可

## Swagger升级SpringDoc指南

常见功能如下 其他功能自行挖掘 注意: **javadoc** 只能替换基础功能 特殊功能还需要使用注解实现

swagger	springdoc	javadoc
<code>@Api(name = "xxx")</code>	<code>@Tag(name = "xxx")</code>	java类注释第一行
<code>@Api(description= "xxx")</code>	<code>@Tag(description= "xxx")</code>	java类注释
<code>@ApiOperation</code>	<code>@Operation</code>	java方法注释
<code>@ApiIgnore</code>	<code>@Hidden</code>	无
<code>@ApiParam</code>	<code>@Parameter</code>	java方法@param参数注释
<code>@ApiImplicitParam</code>	<code>@Parameter</code>	java方法@param参数注释
<code>@ApiImplicitParams</code>	<code>@Parameters</code>	多个@param参数注释
<code>@ApiModelProperty</code>	<code>@Schema</code>	java实体类注释
<code>@ApiModelProperty(hidden = true)</code>	<code>@Schema(accessMode = READ_ONLY)</code>	无
<code>@ApiResponse</code>	<code>@ApiResponse</code>	java方法@return值注释

## 建议使用 **Apifox**

官网连接: <https://www.apifox.cn/> 视频教程: `springdoc与apifox配合使用`

# Apifox

API 文档、API 调试、API Mock、API 自动化测试

Apifox = Postman + Swagger + Mock + JMeter

支持 文档编写 接口调试 Mock 接口压测 自动化测试 等一系列功能

接入框架

## 1. 下载或使用web在线版 创建一个自己的项目

**新建项目 (疯狂的狮子Li)**

* 项目名称	RuoYi-Cloud-Plus	项目图标
成员权限	成员	权限
	疯狂的狮子Li (@疯狂的狮子Li)	管理员

\* 是否公开  私有项目 (仅对项目成员可见)   
  公开项目 (公开后, 任何人都可以访问项目, 请谨慎考虑! )

**保存** **取消**

## 2. 进入项目 选

择项目设置 找到自动同步

**RuoYi-Cloud-Plus**

项目设置 / 导入数据

**导入数据 (自动导入)**

启用	数据源名称	导入分组	导入时间	数据源格式	最近导入时间	操作
<input checked="" type="checkbox"/>	演示服务	根目录/演示服务	每隔 3 小时	OpenAPI/Swagger	2022/07/23 01:47:36	<a href="#">详情</a> <a href="#">立即导入</a> <a href="#">...</a>
<input checked="" type="checkbox"/>	认证服务	根目录/认证服务	每隔 3 小时	OpenAPI/Swagger	2022/07/23 01:47:37	<a href="#">详情</a> <a href="#">立即导入</a> <a href="#">...</a>
<input checked="" type="checkbox"/>	系统服务	根目录/系统服务	每隔 3 小时	OpenAPI/Swagger	2022/07/23 01:47:39	<a href="#">详情</a> <a href="#">立即导入</a> <a href="#">...</a>
<input checked="" type="checkbox"/>	代码生成服务	根目录/代码生成服务	每隔 3 小时	OpenAPI/Swagger	2022/07/23 01:47:41	<a href="#">详情</a> <a href="#">立即导入</a> <a href="#">...</a>
<input checked="" type="checkbox"/>	资源服务	根目录/资源服务	每隔 3 小时	OpenAPI/Swagger	2022/07/23 01:47:43	<a href="#">详情</a> <a href="#">立即导入</a> <a href="#">...</a>

**+ 新建数据源**

**项目设置**

**导入数据 (手动导入)**

**导入数据 (自动同步)**

3. 根据项目内所有文档组完成所有数据源创建(拉取后端 openapi 结构体) 数据源URL格式 http://网关ip:端口/服务路径/v3/api-docs 项目内所需:  
 http://localhost:8080/demo/v3/api-docs 演示服务 http://localhost:8080/auth/v3/api-docs 认证服务 http://localhost:8080/resource/v3/api-docs 资源服务 http://localhost:8080/system/v3/api-docs 系统服务 http://localhost:8080/code/v3/api-docs 代码生成服务

## 编辑数据源

X

\* 导入频率  手动触发  每隔 3 小时  每隔 12 小时  每隔 24 小时

\* 数据源格式  OpenAPI (Swagger)  apiDoc  Apifox

\* 数据源名称 演示服务

\* 数据源 URL http://localhost:8080/demo/v3/api-docs

高级设置 展开 ^

接口 数据模型

导入到分组 演示服务

接口覆盖模式 同 URL 覆盖

接口路径加上 basePath

删除

立即导入

取消

确定

## 导入数据 (自动导入)

启用	数据源名称	导入分组	导入时间	数据源格式	操作
<input checked="" type="checkbox"/>	演示服务	根目录/演示服务	每隔 3 小时	OpenAPI/Swagger	<input type="button" value="编辑"/>
<input checked="" type="checkbox"/>	认证服务	根目录/认证服务	每隔 3 小时	OpenAPI/Swagger	<input type="button" value="编辑"/>
<input checked="" type="checkbox"/>	系统服务	根目录/系统服务	每隔 3 小时	OpenAPI/Swagger	<input type="button" value="编辑"/>
<input checked="" type="checkbox"/>	代码生成服务	根目录/代码生成服务	每隔 3 小时	OpenAPI/Swagger	<input type="button" value="编辑"/>
<input checked="" type="checkbox"/>	资源服务	根目录/资源服务	每隔 3 小时	OpenAPI/Swagger	<input type="button" value="编辑"/>

4. 选择 接口管理 项目概览 点击立即导入

并等待导入完成 后续会根据策略每3个小时自动导入一次 每次重新进入apifox也会自动同步一次 后端有改动也可以手动点击导入

Yi-Cloud-Plus Project Overview

**项目统计**

- 接口数: 159
- 接口用例数: 159
- 文档数: 0
- 数据模型数: 77

**在线分享**

① 将接口文档以 URL 形式分享出去，方便外部团队在线查看！

**自动导入**

① 自动将 OpenAPI (Swagger) 或 apiDoc 格式的在线 URL 接口数据定时导入到 Apifox。

**代码生成**

① 根据接口/模型定义 ObjectiveC、Kotlin (如 Model、Cont

**立即导入**

5. 设置鉴权 选择接口管理 项目概览 找到Auth 按照如下配置

RuoYi-Cloud-Plus Project Overview

**自动导入**

① 自动将 OpenAPI (Swagger) 或 apiDoc 格式的在线 URL 接口数据定时导入到 Apifox。

**代码生成**

① 根据接口/模型定义，自动生成各种语言/ObjectiveC、Kotlin、Dart、C++、C#、R (如 Model、Controller、单元测试代码)

**Auth**

① 全局 Authorization，项目内所有接口运行时都会执行此处设置的 Authorization。

类型: API Key
添加位置: Header
Key: Authorization
Value: Value

```

email: crazylionli@163.com
url: https://gitee.com/JavaLionLi/RuoYi-Vue-Plus
components:
  # 鉴权方式配置
  security-schemes:
    apiKey:
      type: APIKEY
      in: HEADER
      name: ${sa-token.token-name}
  
```

key对应项目配置 默认为 Authorization

```
# Sa-Token配置
sa-token:
  # token名称 (同时也是cookie名称)
  token-name: Authorization
  # token有效期 (设为一天 (必定过期) 单位: 秒)
  timeout: 86400
  # token临时有效期 (指定时间无操作就过期) 单位: 秒
  #auto-renew-timeout: 1800
```

## (旧)接口文档

版本 &lt;= 1.1.0

## 访问方式

前端访问 系统工具 -&gt; 系统接口

The screenshot shows the RuoYi-Vue-Plus system interface. On the left, there is a dark sidebar with various menu items: 首页 (Home), 系统管理 (System Management), 系统监控 (System Monitoring), 系统工具 (System Tools), 表单构建 (Form Building), 代码生成 (Code Generation), 系统接口 (System Interface), PLUS官网 (PLUS Official Website), 测试菜单 (Test Menu). The 'System Interface' item is highlighted with a red arrow. The main content area has a title '标题: RuoYi-Vue-Plus后台管理系统\_接口文档'. It contains sections for '简介' (Introduction), '作者' (Author), '版本' (Version), 'host' (Host), 'basePath' (Base Path), '服务Url' (Service URL), '分组名称' (Group Name), '分组Url' (Group URL), '分组location' (Group Location), and '接口统计信息' (Interface Statistics). The statistics table shows 20 GET requests and 7 POST requests.

## 文档配置

```

203 # swagger配置
204 swagger:
205   # 是否开启swagger
206   enabled: true
207   # 标题
208   title: '标题: RuoYi-Cloud-Plus微服务权限管理系统_接口文档'
209   # 描述
210   description: '描述: 微服务权限管理系统, 具体包括xxx,xxx模块...'
211   # 版本
212   version: '版本号: 系统版本...'
213   # 作者信息
214   contact:
215     name: Lion Li
216     email: crazylionli@163.com
217     url: https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus
218

```

```
knife4j:  
  # 是否开启Knife4j增强模式  
  enable: true  
  # 是否开启生产环境保护策略  
  production: false  
  # 登录认证  
  basic:  
    enable: true  
    username: ruoyi  
    password: 123456  
  # 前端Ui的个性化配置属性  
  setting:  
    # 默认语言  
    language: zh-CN  
    # 是否显示Footer  
    enableFooter: true  
    # 是否开启启动态参数调试功能  
    enableDynamicParameter: true  
    # 是否在每个Debug调试栏后显示刷新变量按钮  
    enableReloadCacheParameter: true
```

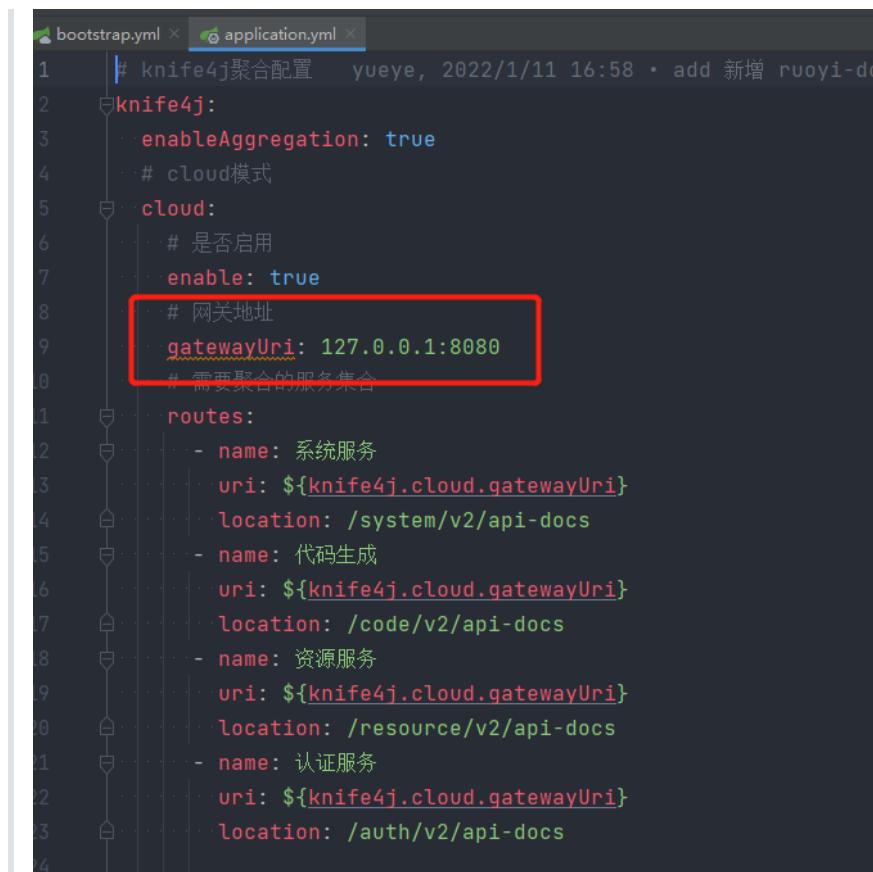
swagger 前缀配置为基础配置 knife4j 前缀配置为增强配置 增强配置可以查看 [knife4j文档](#) 框架文档

knife4j.production 生产保护 开启生产保护 prod 环境则接口文档无法访问

```
knife4j:  
  # 是否开启Knife4j增强模式  
  enable: true  
  # 是否开启生产环境保护策略  
  production: false  
  # 登录认证  
  basic:  
    enable: true  
    username: ruoyi  
    password: 123456
```

**ruoyi-doc** 文档服务配置说明

更改此处网关地址 指向本机的 `ruoyi-gateway` 服务



```
# knife4j聚合配置  yueye, 2022/1/11 16:58 • add 新增 ruoyi-dc
knife4j:
  enableAggregation: true
  # cloud模式
  cloud:
    # 是否启用
    enable: true
    # 网关地址
    gatewayUri: 127.0.0.1:8080
    # 需要聚合的服务集合
  routes:
    - name: 系统服务
      uri: ${knife4j.cloud.gatewayUri}
      location: /system/v2/api-docs
    - name: 代码生成
      uri: ${knife4j.cloud.gatewayUri}
      location: /code/v2/api-docs
    - name: 资源服务
      uri: ${knife4j.cloud.gatewayUri}
      location: /resource/v2/api-docs
    - name: 认证服务
      uri: ${knife4j.cloud.gatewayUri}
      location: /auth/v2/api-docs
```

内网鉴权

# 功能介绍

此功能用于防止外部请求访问内部服务应用 在请求经过 gateway网关 会生成一个 id-token 携带到后续服务进行校验 若未经过 gateway网关 调用内网服务会出现 id-token无效 异常 有效防止非法请求直接访问内网服务

## 开启/关闭内网鉴权

更改 application-common.yml 配置文件的 sa-token.check-id-token 配置即可

```
# Sa-Token配置
sa-token:
  # token名称 (同时也是cookie名称)
  token-name: Authorization
  # token有效期 设为一天 (必定过期) 单位: 秒
  timeout: 86400
  # token临时有效期 (指定时间无操作就过期) 单位: 秒
  activity-timeout: 1800
  # 开启内网服务调用鉴权
  check-id-token: true
  # Id-Token的有效期 (单位: 秒)
  id-token-timeout: 600
  # 是否允许同一账号并发登录 (为true时允许一起登录, 为false时不允许)
  allow-concurrent-login: true
```

## 放行内网鉴权

进入 ruoyi-common-security 模块找到 SecurityConfiguration 类增加排除路径即可



## 修改包名

### 关于修改包名

- 将文件夹全部修改为 com.xxx
- 使用IDEA全局替换 com.ruoyi 替换为 com.xxx
- 严禁手动修改

## 主键使用说明

# 关于如何使用分布式id或雪花id

参考 `MybatisPlusConfiguration` 如需自定义修改 `Bean` 实现即可

```
/*
 * 使用网卡信息绑定雪花生成器
 * 防止集群雪花ID重复
 */
@Bean
public IdentifierGenerator idGenerator() {
    疯狂的狮子li, 2021/12/17 13:47
    return new DefaultIdentifierGenerator(NetUtil.getLocalHost());
}
```

框架默认集成 雪花ID 只需全局更改 主键类型即可

```
logImpl: org.apache.ibatis.logging.noLogging.NoLoggingImpl
global-config:
# 是否打印 Logo banner
banner: true
# 是否初始化 SqlRunner
enableSqlRunner: false
dbConfig:
# 主键类型
# AUTO 自增 NONE 空 INPUT 用户输入 ASSIGN_ID 雪花 ASSIGN_UUID 唯一 UUID
idType: AUTO
# 表名是否使用驼峰转下划线命名, 只对表名生效
tableUnderline: true
# 大写命名, 对表名和字段名均生效
capitalMode: false
# 逻辑已删除值
1 -> 1, 0 -> 0
```

如单表使用 可单独配置注解

```
/*
 * 参数主键
 */
@ApiModelProperty(value = "参数主键")
@ExcelProperty(value = "参数主键")
@TableId(value = "config_id", type = IdType.ASSIGN_ID)
private Long configId;
```

## 重点说明

- 由于雪花id位数过长 `Long` 类型在前端会失真
- 框架已配置序列化方案 超越 JS 最大值自动转字符串 参考 `BigNumberSerializer` 类

修改访问后端接口路径

## 修改应用路径

### 修改访问后端接口路径

更改 前端环境配置文件 `VUE_APP_BASE_API` 代理路径

```
# 页面标题
VUE_APP_TITLE = RuoYi-Cloud-Plus后台管理系统

# 开发环境配置
ENV = 'development'

# 若依管理系统/开发环境
VUE_APP_BASE_API = '/dev-api/admin' You, Moments

# 应用访问路径 例如使用前缀 /admin/
VUE_APP_CONTEXT_PATH = '/'

# 路由懒加载
VUE_CLI_BABEL_TRANSPILE_MODULES = true
```

.env.development file content:

```
prod 生产环境需修改 nginx.conf 后端代理路径(上述配置文件也要改)
```

```
index index.html index.htm;

location /prod-api/admin/ { You, A minute ago • Uncommitted changes
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header REMOTE-HOST $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass http://server/;
}
```

## 修改前端页面访问路径

修改对应环境的 `.env` 环境 文件内的 `VUE_APP_CONTEXT_PATH` 应用访问路径即可

```
# 应用访问路径 例如使用前缀 /admin/
VUE_APP_CONTEXT_PATH = '/admin/' You, 2022/1/29 9

# 监控地址
VUE_APP_MONITRO_ADMIN = '/admin/login'

# 监控地址
VUE_APP_XXL_JOB_ADMIN = '/xxl-job-admin'

# 若依管理系统/生产环境
VUE_APP_BASE_API = '/prod-api'
```

.env.development file content:

修改访问后端接口路径

生产环境 nginx.conf 与之对应修改即可 注意: 文件真实目录为 /usr/share/nginx/html/admin/index.html 此功能一般为多项目部署需要 故会增

```
4
5     location /admin/ { ←
6         root /usr/share/nginx/html;
7         try_files $uri $uri/ /admin/index.html; You
8         index index.html index.htm; ←
9     }
```

加一层目录 如不需要可以自行修改

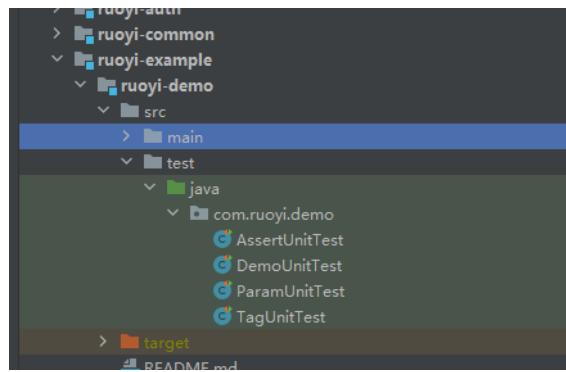
参考文章

## 单元测试

## 参考文章

[SpringBoot 2.X 整合 JUnit5 及全方位使用手册](#)

## 参考代码(1.4.0新增)



## 关于多表查询

# 建议单表查询

文章连接: [大连接查询分解好处](#)

## 分解大连接查询

### 1. 大连接查询分解好处

将一个大连接查询分解成对每一个表进行一次单表查询，然后将结果在应用程序中进行关联，这样做的好处有：

- **让缓存更高效。**对于连接查询，如果其中一个表发生变化，那么整个查询缓存就无法使用。而分解后的多个查询，即使其中一个表发生变化，对其它表的查询缓存依然可以使用。
- 分解成多个单表查询，这些单表查询的缓存结果更可能被其它查询使用到，从而减少冗余记录的查询。
- **减少锁竞争；**
- 在应用层进行连接，可以更容易对数据库进行拆分，从而更容易做到高性能和可伸缩。
- 查询本身效率也可能会有所提升。例如下面的例子中，使用 IN() 代替连接查询，可以让 MySQL 按照 ID 顺序进行查询，这可能比随机的连接要更高效。

### 一个复杂查询还是多个简单查询

设计查询的时候，一个需要考虑的重要问题是，是否需要将一个复杂的查询分成多个简单的查询。在传统实现中，总是强调需要数据库层完成尽可能多的工作，这样做的逻辑在于以前人们总是认为网络通信、查询解析和优化是一件代价很高的事情。

但是这样的想法对于MySQL并不适用，因为MySQL从设计上让连接和断开连接都很轻量，在返回一个小的查询结果方面很高效。现代的网络速度比以前要快很多，能在很大程度上降低延迟。在某些版本的MySQL中，即使在一台通用服务器上，也能够运行每秒超过10万次的简单查询，即使是一个千兆网卡也能轻松满足每秒超过2000次的查询。所以运行多个小查询现在已经不是大问题了。

在MySQL内部，每秒能够扫描内存中上百万行的数据，相比之下，MySQL响应数据给客户端就慢得多了。在其他条件都相同的时候，使用尽可能少的查询当然是更好的。但是有时候，将一个大查询分解为多个小查询是很有必要的。别害怕这样做，好好衡量一下这样做是不是会减少工作量。稍后我们将通过一个示例来展示这个技巧的优势。

## 分解联接查询

很多高性能的应用都会对联接查询进行分解。简单地说，可以对每一个表进行一次单表查询，然后将结果在应用程序中进行联接。例如，下面这个查询：

```
SELECT * FROM tag
JOIN tag_post ON tag_post.tag_id=tag.id
JOIN post ON tag_post.post_id=post.id
WHERE tag.tag='mysql';
```

可以分解成下面这些查询来代替：

```
SELECT * FROM tag WHERE tag='mysql';
SELECT * FROM tag_post WHERE tag_id=1234;
SELECT * FROM post WHERE post.id IN (123,456,567,9098,8904);
```

到底为什么要这样做？乍一看，这样做并没有什么好处，原本一条查询，这里却变成多条查询，返回的结果又是一模一样的。事实上，用分解联接查询的方式重构查询有如下优势：

- 让缓存的效率更高。许多应用程序可以方便地缓存单表查询对应的结果对象。例如，上面查询中

的tag mysql已经被缓存了，那么应用就可以跳过第一个查询。再例如，应用中已经缓存了ID为123、567、9098的内容，那么第三个查询的IN()中就可以少几个ID。

- 将查询分解后，执行单个查询可以减少锁的竞争。
- 在应用层做联接，可以更容易对数据库进行拆分，更容易做到高性能和可扩展。
- 查询本身的效果也可能会有所提升。在这个例子中，使用IN()代替联接查询，可以让MySQL按照ID顺序进行查询，这可能比随机的联接要更高效。
- 可以减少对冗余记录的访问。在应用层做联接查询，意味着对于某条记录应用只需要查询一次，而在数据库中做联接查询，则可能需要重复地访问一部分数据。从这点看，这样的重构还可能会减少网络和内存的消耗。

在有些场景下，在应用程序中执行联接操作会更加有效。比如，当可以缓存和重用之前查询结果中的数据时、当在多台服务器上分发数据时、当能够使用IN()列表替代联接查询大型表时、当一次联接查询中多次引用同一张表时。

(上图出自 <高性能MySQL>)

## 前端相关文档

因框架前端使用 `ruoyi-ui` 进行少部分功能改动 故与 `ruoyi-ui` 前端文档一致 文档地址: [ruoyi前端手册](#)

## 扩展功能

### ELK搭建

### 环境搭建

项目内置 ELK 的 docker-compose 编排 可查看 `/docker/docker-compose.yml` 文件下方扩展编排

注意: `/docker/elk/elasticsearch/` 目录下所有文件夹 均需要写权限

`chmod 777 /docker/elk/elasticsearch/data` `chmod 777 /docker/elk/elasticsearch/logs` `chmod 777 /docker/elk/elasticsearch/plugins` 注意: es插件  
需要解压后放入 `plugins` 目录

### 运行命令

```
docker-compose up -d elasticsearch kibana logstash
```

### 参考文章

[docker-compose 搭建 ELK 7.X 并整合 SpringBoot](#)

## 项目内配置

服务引入依赖项

```
<!-- ELK 日志收集 -->
<dependency>
    <groupId>com.ruoyi</groupId>
    <artifactId>ruoyi-common-logstash</artifactId>
</dependency>
```

```
... nacos.config.group>DEFAULT_GROUP</nacos.config.group>
... <logstash.address>127.0.0.1:4560</logstash.address>
</properties>
```

更改主 `pom` 文件 `logstash.address` 地址 `<activation>`

环境搭建(如果已经搭建了ELK则跳过)

## ES搜索引擎

### 环境搭建(如果已经搭建了ELK则跳过)

项目内置 ELK 的 docker-compose 编排 可查看 `/docker/docker-compose.yml` 文件下方扩展编排

注意: `/docker/elk/elasticsearch/` 目录下所有文件夹 均需要写权限

`chmod 777 /docker/elk/elasticsearch/data` `chmod 777 /docker/elk/elasticsearch/logs` `chmod 777 /docker/elk/elasticsearch/plugins` 注意: es插件  
需要解压后放入 `plugins` 目录

## 运行命令

```
docker-compose up -d elasticsearch
```

## Easy-ES 文档

[Easy-ES 文档](#)

## 用法

基本配置和用法可参考 `ruoyi-demo` 模块 更多高级用法请参考 Easy-ES 文档



The screenshot shows a code editor with Java code related to Elasticsearch. The code includes annotations like `@ConditionalOnProperty`, `@RestController`, and `@RequiredArgsConstructor`. It defines a class `EsCrudController` with methods for crud operations and search queries. The code is annotated with Javadoc comments and includes several usages of the `DocumentMapper` interface.

```
/** * 搜索引擎 crud 演示案例 */  
/* * @author Lion Li */  
/* */  
4 usages  ● 疯狂的狮子li  
@ConditionalOnProperty(value = "easy-es.enable", havingValue = "true")  
@RestController  
@RequiredArgsConstructor  
public class EsCrudController {    疯狂的狮子li, 2022/8/3 15:44 • add 增加 搜索引擎 crud  
  
    5 usages  
    private final DocumentMapper documentMapper;  
  
    /**  
     * 查询(指定)  
     *  
     * @param title 标题  
     */  
    ● 疯狂的狮子li  
    @GetMapping("/select")  
    public Document select(String title) {  
        LambdaEsQueryWrapper<Document> wrapper = new LambdaEsQueryWrapper<>();  
        wrapper.eq(Document::getTitle, title);  
        return documentMapper.selectOne(wrapper);  
    }  
}
```

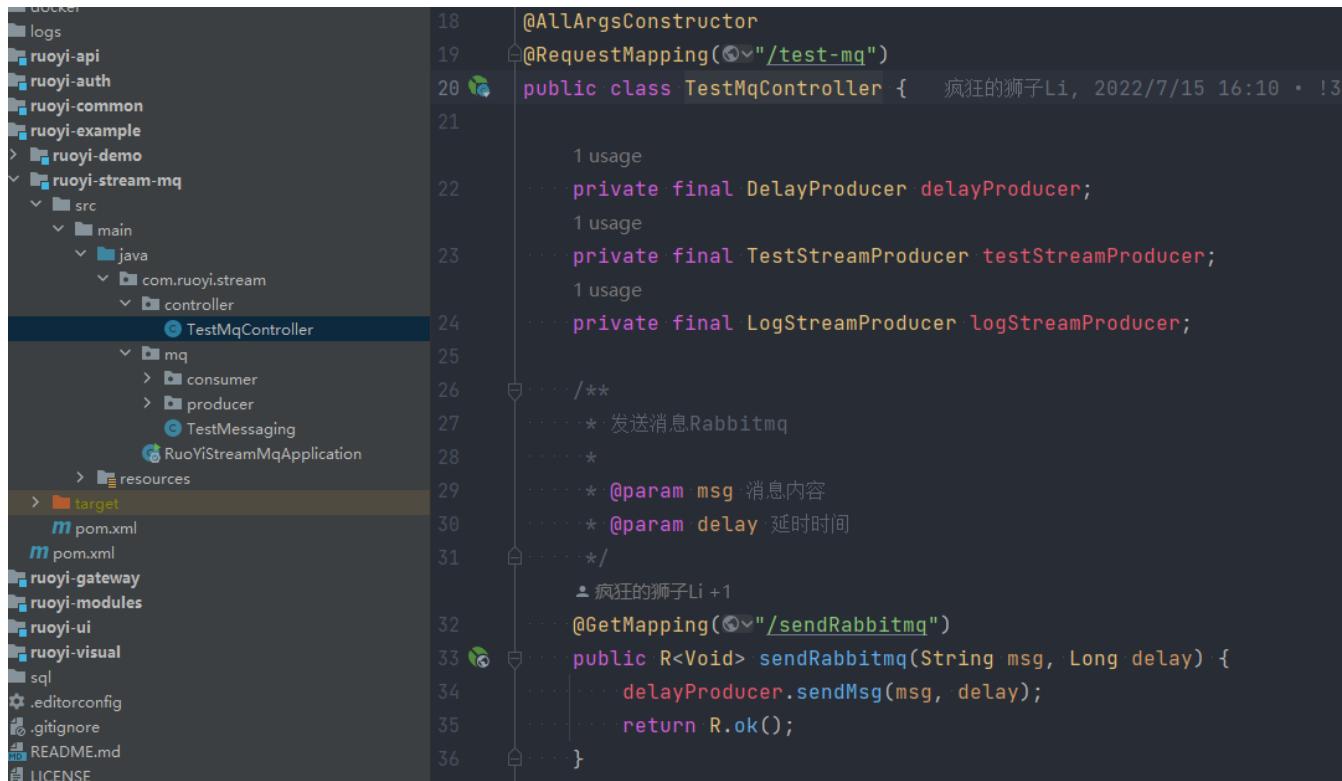
## RabbitMQ搭建

## 环境搭建

参考文章: [docker-compose 安装 RabbitMQ 3.X 附带延迟队列插件](#)

## 用法参考

参考 ruoyi-stream-mq 模块内的测试案例



```

18  @AllArgsConstructor
19  @RequestMapping("test-mq")
20  public class TestMqController {
21      ...
22      private final DelayProducer delayProducer;
23      private final TestStreamProducer testStreamProducer;
24      private final LogStreamProducer logStreamProducer;
25      ...
26      /**
27       * 发送消息Rabbitmq
28       */
29      /**
30       * @param msg 消息内容
31       * @param delay 延时时间
32       */
33      @GetMapping("/sendRabbitmq")
34      public R<Void> sendRabbitmq(String msg, Long delay) {
35          delayProducer.sendMsg(msg, delay);
36          return R.ok();
37      }

```

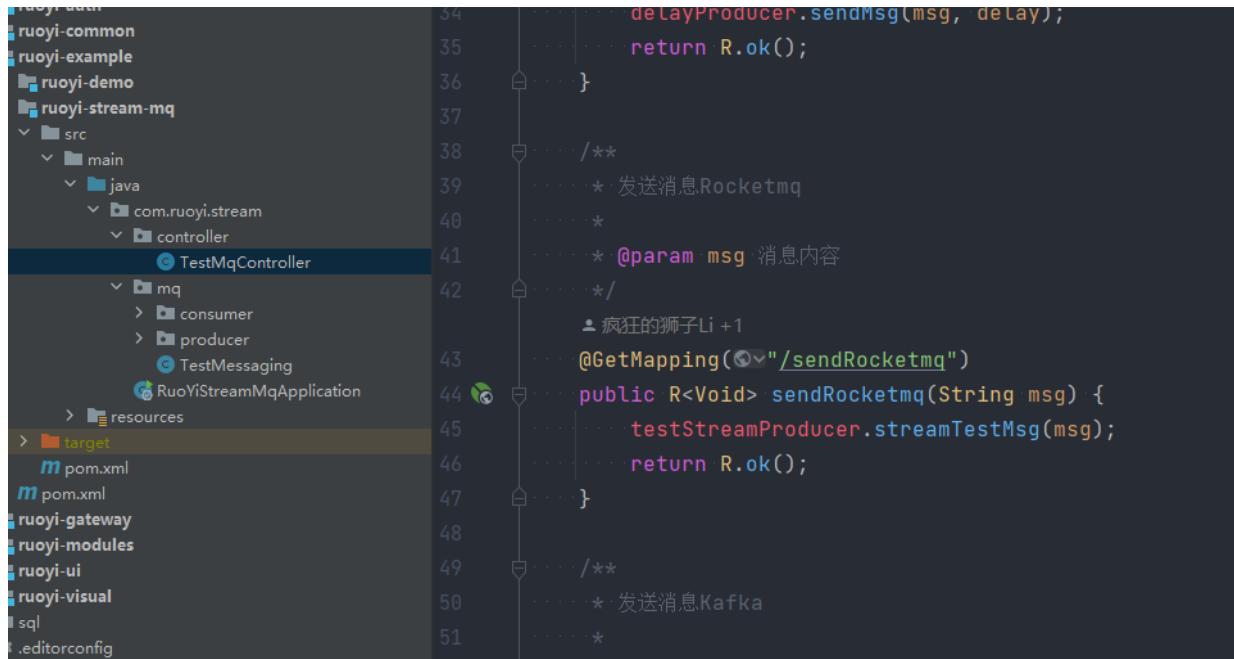
## RocketMQ搭建

## 环境搭建

参考文章: docker-compose 安装 RocketMQ 4.9.X (apache官方镜像) namesrv broker 与可视化控制台 console

## 用法参考

参考 ruoyi-stream-mq 模块内的测试案例



The screenshot shows a file browser interface with the following project structure:

```

ruoyi-data
ruoyi-common
ruoyi-example
ruoyi-demo
ruoyi-stream-mq
  src
    main
      java
        com.ruoyi.stream
          controller
            TestMqController
            mq
              consumer
              producer
              TestMessaging
            RuoYiStreamMqApplication
        resources
  target
  pom.xml
  pom.xml
ruoyi-gateway
ruoyi-modules
ruoyi-ui
ruoyi-visual
sql
.editorconfig

```

On the right, a code editor displays Java code for sending messages to RocketMQ. The code is annotated with comments and annotations:

```

34     delayProducer.sendMsg(msg, delay);
35   }
36 }
37 /**
38  * 发送消息Rocketmq
39  */
40 /**
41  * @param msg 消息内容
42  */
43 /**
44  * 疯狂的狮子Li +1
45  */
46 public R<Void> sendRocketmq(String msg) {
47   testStreamProducer.streamTestMsg(msg);
48   return R.ok();
49 }
50 /**
51  * 发送消息Kafka
52  */

```

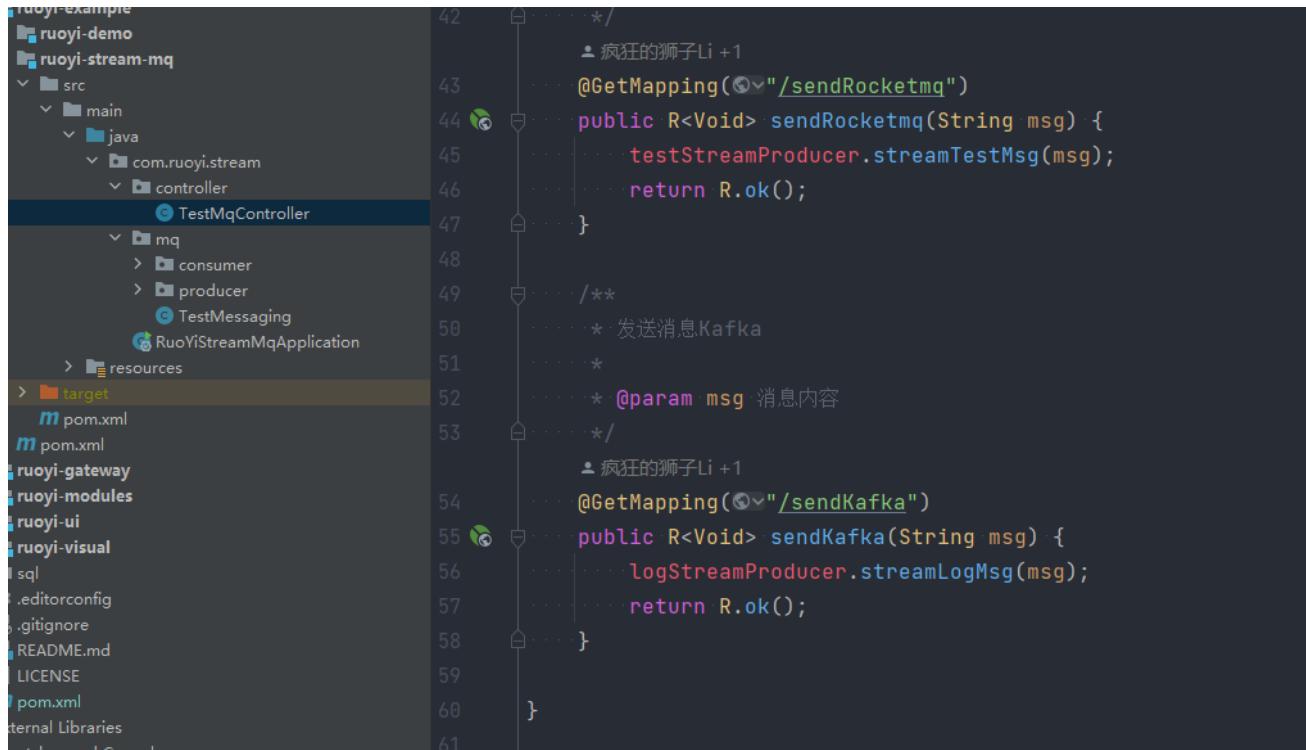
## Kafka搭建

## 环境搭建

参考文章: docker-compose 安装 Kafka 3.X 附带可视化界面

## 用法参考

参考 ruoyi-stream-mq 模块内的测试案例



The screenshot shows a Java code editor with a file tree on the left and code on the right. The file tree includes:

- ruoyi-example
- ruoyi-demo
- ruoyi-stream-mq
  - src
    - main
      - java
        - com.ruoyi.stream
      - controller
    - TestMqController
    - mq
      - consumer
      - producer
      - TestMessaging
    - RuoYiStreamMqApplication
  - resources
  - target
  - pom.xml
  - pom.xml
- ruoyi-gateway
- ruoyi-modules
- ruoyi-ui
- ruoyi-visual
- sql
- .editorconfig
- .gitignore
- README.md
- LICENSE
- pom.xml
- External Libraries

```
42     */
43     * 疯狂的狮子Li +1
44     @GetMapping("sendRocketmq")
45     public R<Void> sendRocketmq(String msg) {
46         testStreamProducer.streamTestMsg(msg);
47         return R.ok();
48     }
49     /**
50      * 发送消息Kafka
51      *
52      * @param msg 消息内容
53     */
54     @GetMapping("sendKafka")
55     public R<Void> sendKafka(String msg) {
56         logStreamProducer.streamLogMsg(msg);
57         return R.ok();
58     }
59 }
60 }
```

集群搭建两种方式

Nacos集群搭建

## 集群搭建两种方式

文件寻址集群

[【RuoYi-Cloud-Plus】学习笔记 02 - Nacos（二）寻址机制之文件寻址分析](#)

地址服务器寻址集群(推荐)

[【RuoYi-Cloud-Plus】学习笔记 03 - Nacos（三）使用 Nginx 实现地址服务器寻址及其原理分析](#)

## 集群路由代理设置

[【RuoYi-Cloud-Plus】学习笔记 04 - Nacos（四）使用 Nginx 简单实现 Nacos 集群负载均衡](#)

设置好代理之后 跟单机用法一致 后端nacos地址写代理 ip:端口 即可

## SkyWalking搭建与集成

### 服务搭建

参考文章: [SpringBoot 整合 SkyWalking 8.X \(包含 Logback 日志采集\)](#)

框架已经包含了 docker-compose 编排 执行如下命令启动容器即可

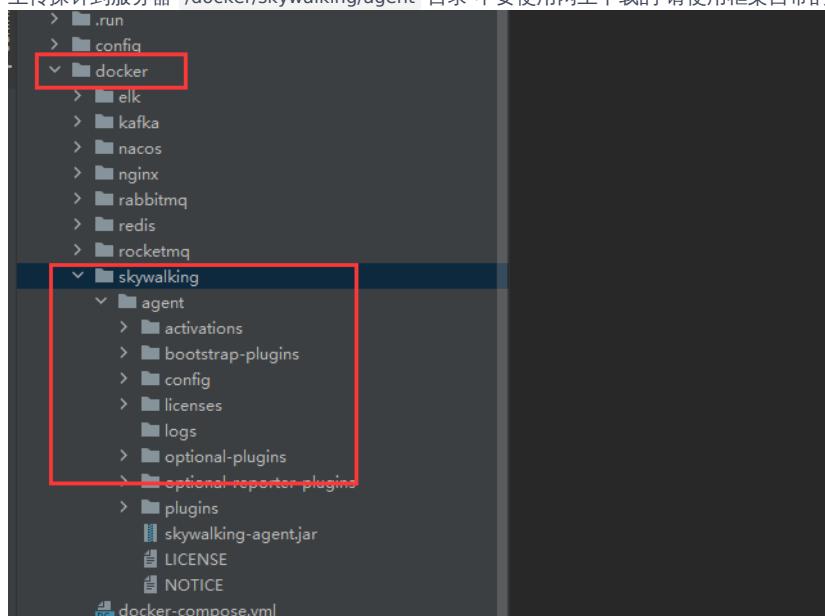
```
docker-compose up -d elasticsearch sky-oap sky-ui
```

#### 本地开发使用

参考上方文章

#### docker部署使用

上传探针到服务器 /docker/skywalking/agent 目录 不要使用网上下载的 请使用框架自带的 内含一些官网没有的插件



在对应服务的 dockerfile 内 打开 skywalking 相关参数注释

```
EXPOSE ${SERVER_PORT}

ADD ./target/ruoyi-auth.jar ./app.jar

ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/.urandom", "-Dserver.port=${SERVER_PORT}", # "-Dskywalking.agent.service_name=ruoyi-auth", # "-javaagent:/ruoyi/skywalking/agent/skywalking-agent.jar", "-jar", "app.jar"]
```

```

▶ ruoyi-auth:
  image: ruoyi/ruoyi-auth:1.3.0
  container_name: ruoyi-auth
  environment:
    # 时区上海
    TZ: Asia/Shanghai
  ports:
    - "9210:9210"
  volumes:
    # 配置文件
    - /docker/ruoyi-auth/logs/:/ruoyi/auth/logs
      # skywalking 探针
    - /docker/skywalking/agent/:/ruoyi/skywalking/agent
  privileged: true
  network_mode: "host"

```

服务编排增加探针路径映射

对接日志推送(不推荐 建议使用ELK收集日志)

框架已经封装好了对应的依赖和配置 在服务内添加如下依赖

```

<!-- skywalking 日志收集 -->
<dependency>
  <groupId>com.ruoyi</groupId>
  <artifactId>ruoyi-common-skylog</artifactId>
</dependency>

```

在 `logback.xml` 日志配置文件内引入 `skylog` 配置文件

```

</appender>
疯狂的狮子li, 2021/12/31 15:00 • add 增加监控中心 在线日志监控 优化用
<include resource="logback-common.xml" />

<include resource="logback-logstash.xml" />

<!-- 开启 skywalking 日志收集 -->
<include resource="logback-skylog.xml" />

<!--系统操作日志-->
<root level="info">
  <appender-ref ref="console"/>
</root>
</configuration>

```

## Prometheus+Grafana搭建

### 基础搭建

参考文章: <https://lionli.blog.csdn.net/article/details/127959009>

### 框架内扩展

框架已经包含了 docker-compose 编排 执行如下命令启动容器即可

```
docker-compose up -d prometheus grafana
```

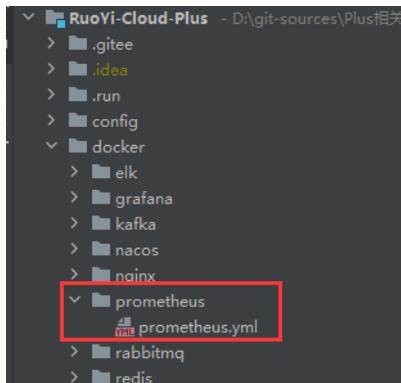
### 应用配置

各个服务引入 `ruoyi-common-prometheus` 模块



```
<!-- | prometheus 监控 --> 疯狂的狮子li, 2022/11/14 11:51 • ad
↑
<dependency>
    <groupId>com.ruoyi</groupId>
    <artifactId>ruoyi-common-prometheus</artifactId>
</dependency>
```

修改 `prometheus.yml` 配置采集数据源



修改 Nacos 地址 与 SpringBoot-Admin 监控地址 用于数据采集 如都为本地应用则无需更改

```
16 # -- "first_rules.yml"
17 # -- "second_rules.yml"
18
19 # A scrape configuration containing exactly one endpoint to scrape:
20 # Here it's Prometheus itself.
21 scrape_configs:
22   - job_name: 'Prometheus'
23     static_configs:
24       - targets: ['127.0.0.1:9090']
25
26   - job_name: 'Grafana'
27     static_configs:
28       - targets: ['127.0.0.1:3000']
29
30   - job_name: 'Nacos'
31     metrics_path: '/nacos/actuator/prometheus'
32     static_configs:
33       - targets: ['127.0.0.1:8848']
34
35   - job_name: RuoYi-Cloud-Plus
36     honor_timestamps: true    疯狂的狮子li, 2022/11/14 11:46 • add 新增 docker prometheus +
37     scrape_interval: 15s
38     scrape_timeout: 10s
39     metrics_path: /actuator/prometheus
40     scheme: http
41     http_sd_configs:
42       - url: 'http://127.0.0.1:9100/actuator/prometheus/sd'
```

## 导入框架特制模板

注意: 此处数据源名称必须与图片保持一致 不然会和模板对应不上导致无法读取数据

选择查看监控

The screenshot shows two main sections of the Grafana interface.

**Data Sources / Prometheus** (Top Section):

- Settings** tab is active.
- Name:** Prometheus (highlighted with a red box).
- Default:** Switch is off.
- Alerting supported:** Info message.

**HTTP** Configuration:

- URL:** http://127.0.0.1:9090
- Allowed cookies:** New tag (enter key to add)
- Timeout:** 5000

**Auth** section is present but empty.

**找到框架内的特制模板json文件 在grafana点击上传json文件 导入模板** (Text):

A file browser window is shown, displaying a directory structure:

- config
- grafana
- Nacos.json
- SLS JVM监控大盘.json
- Spring Boot 2.1 Statistics.json
- README.md

**Dashboards** (Bottom Section):

- Manage dashboards and folders**
- Browse**, **Playlists**, **Snapshots**, **Library panels** buttons.
- Upload JSON file** button (highlighted with a red box).
- Import via grafana.com**: Input field for URL or ID and **Load** button.
- Import via panel json**: Input field for JSON content.

选择查看监控

选择查看监控

点击右侧菜单浏览 选择想要查看的监控即可

The screenshot shows the Grafana interface for managing dashboards. At the top, there's a sidebar with 'Dashboards', 'Playlists', and 'Snapshots'. Below it, the main area has tabs for 'Browse', 'Playlists', 'Snapshots', and 'Library panels'. The 'Browse' tab is selected. A search bar and filter options ('Filter by tag', 'Starred') are present. The dashboard list includes:

Name	Type	Location	Tags
1 SLS JVM监控大盘	Dashboard	General	
Grafana metrics	Dashboard	General	grafana, prometheus
Nacos	Dashboard	General	
Prometheus 2.0 Overview	Dashboard	General	
Prometheus 2.0 Stats	Dashboard	General	prometheus
Spring Boot 2.1 Statistics	Dashboard	General	

Below this is a detailed view of the 'Spring Boot 2.1 Statistics' dashboard. It features a navigation bar with 'General / Spring Boot 2.1 Statistics' and various monitoring tabs like Application, Instance, HikariCP-Pool, Memory Pool (heap), etc. A red arrow points to the 'None' dropdown under 'Application' with the text '可切换监控的服务' (Services can be switched). The dashboard displays several cards and charts, including:

- Uptime:** 1.7 week
- Start time:** 2022-11-09 16:17:53
- Heap Used:** 12.5%
- Non-Heap Used:** 10.7%
- CPU Usage:** A line chart showing CPU usage over time.
- Load Average:** A line chart showing load average over time.

## 常见问题

### Lombok注解爆红

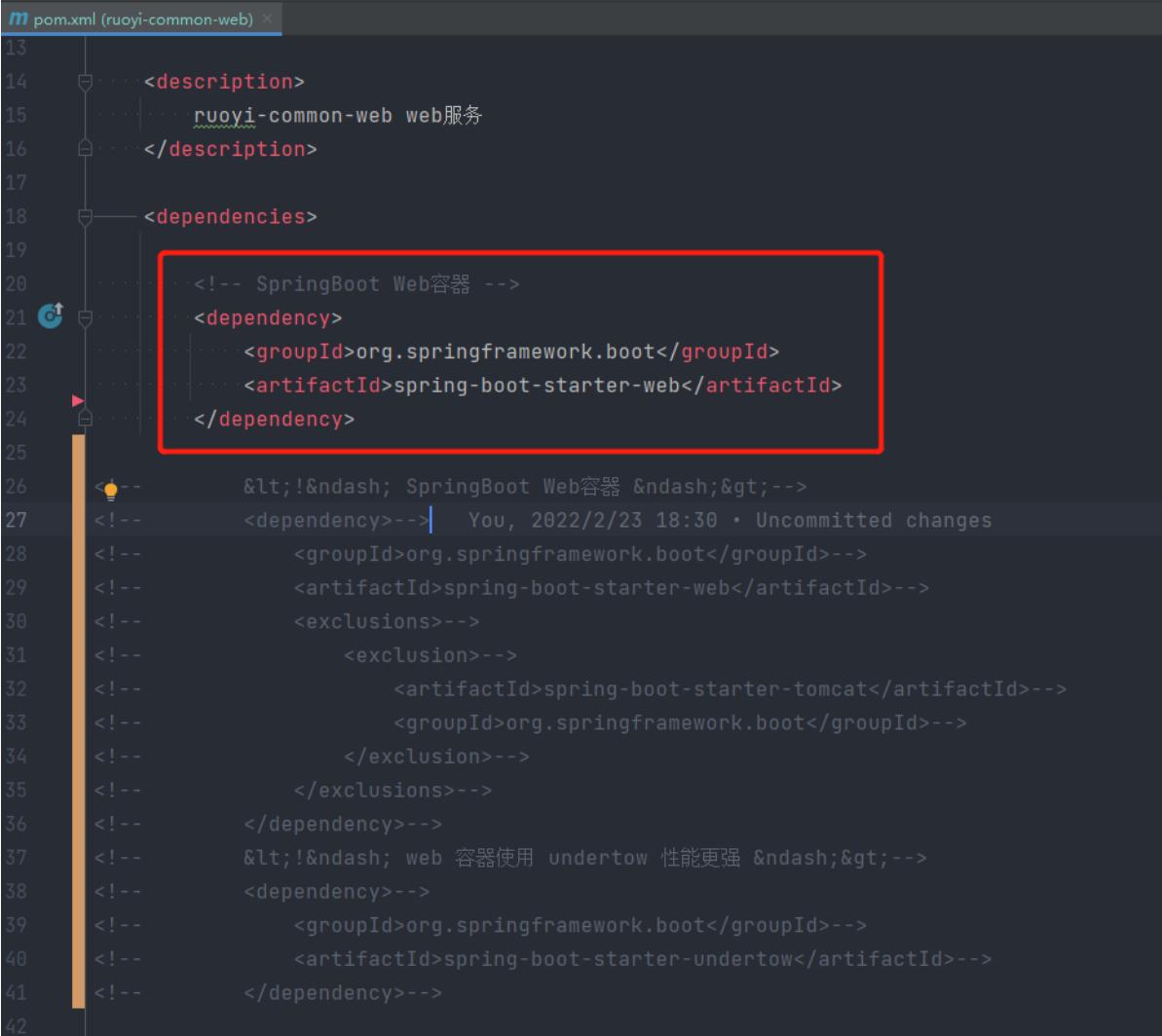
关于lombok注解爆红

- 已知 lombok 插件与 idea中文插件 存在兼容性问题
- 移除中文插件或手动关闭idea检查

## 如何使用Tomcat

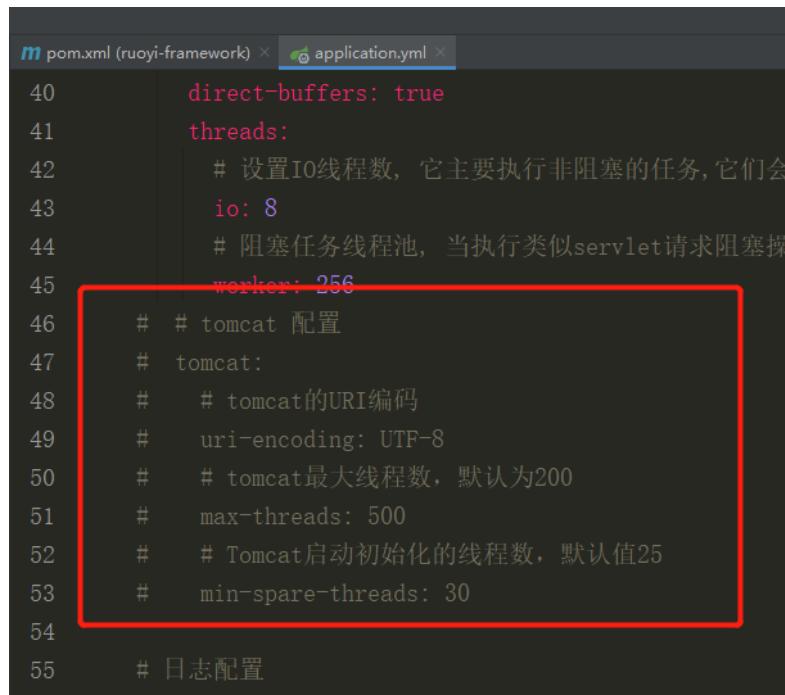
### 关于如何使用Tomcat

更改 `ruoyi-common-web` 模块依赖 使用 `springboot` 默认依赖 即为 `tomcat`



```
pom.xml (ruoyi-common-web) ×
13
14     <description>
15         ruoyi-common-web web服务
16     </description>
17
18     <dependencies>
19
20         <!-- SpringBoot Web容器 -->
21         <dependency>
22             <groupId>org.springframework.boot</groupId>
23             <artifactId>spring-boot-starter-web</artifactId>
24         </dependency>
25
26         <!-- &lt;!&ndash; SpringBoot Web容器 &ndash;&gt;-->
27         <!-- <dependency>--> You, 2022/2/23 18:30 • Uncommitted changes
28         <!-- <groupId>org.springframework.boot</groupId>-->
29         <!-- <artifactId>spring-boot-starter-web</artifactId>-->
30         <!-- <exclusions>-->
31         <!--     <exclusion>-->
32         <!--         <artifactId>spring-boot-starter-tomcat</artifactId>-->
33         <!--         <groupId>org.springframework.boot</groupId>-->
34         <!--     </exclusion>-->
35         <!--     </exclusions>-->
36         <!--     </dependency>-->
37         <!-- &lt;!&ndash; web 容器使用 undertow 性能更强 &ndash;&gt;-->
38         <!-- <dependency>-->
39         <!--     <groupId>org.springframework.boot</groupId>-->
40         <!--     <artifactId>spring-boot-starter-undertow</artifactId>-->
41         <!--     </dependency>-->
42 
```

更改 `application.yml` 文件, 将 `undertow` 配置改为 `tomcat` 配置



```
pom.xml (ruoyi-framework) × application.yml ×
40     direct-buffers: true
41     threads:
42         # 设置IO线程数, 它主要执行非阻塞的任务, 它们会
43         io: 8
44         # 阻塞任务线程池, 当执行类似servlet请求阻塞操
45         worker: 256
46     # # tomcat 配置
47     # tomcat:
48     #     # tomcat的URI编码
49     #     uri-encoding: UTF-8
50     #     # tomcat最大线程数, 默认为200
51     #     max-threads: 500
52     #     # Tomcat启动初始化的线程数, 默认值25
53     #     min-spare-threads: 30
54
55     # 日志配置
```

为何移除druid

## 如何使用druid连接池

## 为何移除druid

性能低下 bug频发 内含fastjson问题众多 监控不支持集群(鸡肋) 不支持一些高版本数据库 社区活跃度冰点

### 性能对比图

Benchmark	(pool)	Mode	Samples	Score	Units
c.z.h.b.ConnectionBench.cycleCnnection	hikari	thrpt	16	21206.330	ops/ms
c.z.h.b.ConnectionBench.cycleCnnection	bone	thrpt	16	10389.139	ops/ms
c.z.h.b.ConnectionBench.cycleCnnection	vibur	thrpt	16	6764.233	ops/ms
c.z.h.b.ConnectionBench.cycleCnnection	tomcat	thrpt	16	2117.792	ops/ms
c.z.h.b.ConnectionBench.cycleCnnection	c3p0	thrpt	16	128.447	ops/ms
c.z.h.b.ConnectionBench.cycleCnnection	druid	thrpt	16	110.370	ops/ms
c.z.h.b.StatementBench.cycleStatement	hikari	thrpt	16	130426.003	ops/ms
c.z.h.b.StatementBench.cycleStatement	tomcat	thrpt	16	62071.180	ops/ms
c.z.h.b.StatementBench.cycleStatement	druid	thrpt	16	54845.335	ops/ms
c.z.h.b.StatementBench.cycleStatement	vibur	thrpt	16	33414.774	ops/ms
c.z.h.b.StatementBench.cycleStatement	bone	thrpt	16	26551.919	ops/ms
c.z.h.b.StatementBench.cycleStatement	c3p0	thrpt	16	15956.412	ops/ms

### 包大小对比图

The screenshot shows two file browser interfaces. The top interface is for 'druid' version 1.2.12, and the bottom one is for 'HikariCP' version 4.0.3. Both show a list of files including '\_remote.repositories', 'druid-1.2.12.jar', 'druid-1.2.12.jar.sha1', and 'druid-1.2.12.pom'. The 'druid-1.2.12.jar' file is highlighted with a red box and labeled '3,676 KB'. The bottom interface shows similar files for HikariCP, with the 'HikariCP-4.0.3.jar' file highlighted with a red box and labeled '156 KB'.

名称	修改日期	类型	大小
_remote.repositories	2022/10/21 11:45	REPOSITORIES ...	1 KB
druid-1.2.12.jar	2022/9/26 17:51	Executable Jar File	3,676 KB
druid-1.2.12.jar.sha1	2022/9/26 17:51	SHA1 文件	1 KB
druid-1.2.12.pom	2022/9/26 17:50	POM 文件	17 KB

名称	修改日期	类型	大小
_remote.repositories	2022/10/25 10:07	REPOSITORIES ...	1 KB
HikariCP-4.0.3.jar	2022/9/26 17:51	Executable Jar File	156 KB
HikariCP-4.0.3.jar.sha1	2022/9/26 17:51	SHA1 文件	1 KB

## 为何使用hikari(中文: 光)

spring默认自带 代码量少结构简单 稳定可靠 性能突出(自行百度一堆测评)

## 参考提交记录反向操作即可

<https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus/commit/62d1304f9a77c3ecb66a6c4960bc517f29557f5a>

## 导入excel实体类为空

关于导入excel实体类为空

- 禁止在导入实体使用 `lombok` 链式调用注解 `@Accessors(chain = true)`
- 会导致找不到 `set` 方法无法注入内容

为何移除druid

## 如何同步项目更新

参考文章: [关于如何同步更新开源项目](#)

## ParseException SQL解析异常

### 异常内容

```
net.sf.jsqlparser.parser.ParseException: Encountered unexpected token:  
    at org.apache.ibatis.plugin.Plugin.invoke(Plugin.java:62) [internal call]  
    at org.apache.ibatis.session.defaults.DefaultSqlSession.selectList(DefaultSqlSession.java:  
    ... 153 common frames omitted  
Caused by: net.sf.jsqlparser.parser.ParseException: Encountered unexpected token: "."."  
    at line 6, column 8.  
  
Was expecting one of:  
  
"&"  
";;"  
";"  
"<<"  
">>>"  
"ACTION"  
"ACTIVE"  
"ALGORITHM"  
"ARCHIVE"
```

此异常为 SQL 解析异常，应检查 SQL 语句内是否包含 SQL 关键字

异常通常都会提供坐标

```
    at org.apache.ibatis.session.defaults.DefaultSqlSession.selectList(DefaultSqlSession.java:  
    ... 153 common frames omitted  
Caused by: net.sf.jsqlparser.parser.ParseException: Encountered unexpected token: "."."  
    at line 6, column 8.  
  
Was expecting one of:  
  
".;"
```

检查报错 SQL 相关坐标位置

```
at org.mybatis.spring.SqlSessionTemplate$SqlSessionInterceptor.invoke(SqlSession
... 146 common frames omitted
Caused by: com.baomidou.mybatisplus.core.exceptions.MybatisPlusException: Failed to
    m.parent_id, 1
    m.menu_name, 2
    m.path, 3
    m.component, 4
    m.query, 5
    m.visible, 6
    m.status,
    ifnull(m.perms, '') as perms,
    m.is_frame,
    m.is_cache,
    m.menu_type,
    m.icon,
    m.order_num,
    m.create_time
from sys_menu m
where m.menu_type in ('M', 'C')
and m.status = 0
```

## 异常由来

由 Mybatis-Plus 拦截器进行 SQL 解析导致 常见拦截器导致问题 TenantLineInnerInterceptor DataPermissionInterceptor

## 解决方案

将关键字增加标识符区别开

```
/*
 * 路由参数
 */
@ApiModelProperty(value = "路由参数")
@TableField("query")
private String query;
```

1. 实体类字段处理(以下仅限于mysql 其他数据库方法各不相同)

```
<select id="selectMenuTreeByUserId" parameterType=
    select distinct m.menu_id,
    m.parent_id,
    m.menu_name,
    m.path,
    m.component,
    m.`query` You, Moments ago
    m.visible,
    m.status
```

2. 自定义 SQL 或 XML 处理

3. Mapper排除

查看具体使用了哪些拦截器导致问题 使用忽略注解依次进行排除即可

```
/*
 * 业务字段 数据层
 *
 * @author Lion Li
 */
@InterceptorIgnore(dataPermission = "true", tenantLine = "true")
public interface GenTableColumnMapper extends BaseMapperPlus<GenTableColumn> {
    /**
     * 根据表名称查询列信息
     *
     * @param tableName 表名称
     * @return 列信息
     */
    @InterceptorIgnore(dataPermission = "true", tenantLine = "true")
    List<GenTableColumn> selectDbTableColumnsByName(String tableName);
}
```

为何移除druid

## swagger相关问题

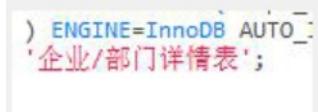
### 空指针问题

```
2022-01-20 17:32:22 [main] ERROR s.d.s.w.p.DocumentationPluginsBootstrapper
- Unable to scan documentation context 1.茶室模块
java.lang.NullPointerException: createBreakpoint() : null
    at springfox.documentation.schema.Example.equals(Example.java:131)
    at java.util.Objects.equals(Objects.java:59)
    at springfox.documentation.service.RequestParam.equals(RequestParameter.java:132)
    at java.util.HashMap.putVal(HashMap.java:634)
    at java.util.HashMap.put(HashMap.java:611)
    at java.util.HashSet.add(HashSet.java:219) <4 internal lines>
    at java.util.ArrayList$ArrayListSpliterator.forEachRemaining(ArrayList.java:1374) <4 internal lines>
    at java.util.stream.ReferencePipeline.collect(ReferencePipeline.java:499)
    at springfox.documentation.spring.web.readers.operation.OperationParameterReader.apply(OperationParameterReader.java:93)
    at springfox.documentation.spring.web.plugins.DocumentationPluginsManager.operation(DocumentationPluginsManager.java:144)
    at springfox.documentation.spring.web.readers.operation.ApiOperationReader.read(ApiOperationReader.java:72)
    at springfox.documentation.spring.web.scanners.CachingOperationReader.lambda$new$0(CachingOperationReader.java:43)
    at java.util.HashMap.computeIfAbsent(HashMap.java:1118)
    at springfox.documentation.spring.web.scanners.CachingOperationReader.read(CachingOperationReader.java:48)
    at springfox.documentation.spring.web.scanners.ApiDescriptionReader.read(ApiDescriptionReader.java:72)
    at springfox.documentation.spring.web.scanners.ApiListingScanner.scan(ApiListingScanner.java:169)
    at springfox.documentation.spring.web.scanners.ApiDocumentationScanner.scan(ApiDocumentationScanner.java:67)
    at springfox.documentation.spring.web.plugins.AbstractDocumentationPluginsBootstrapper.scanDocumentation(AbstractDocumentationPluginsBootstrapper.java:96)
    at springfox.documentation.spring.web.plugins.AbstractDocumentationPluginsBootstrapper.bootstrapDocumentationPlugins
(AbstractDocumentationPluginsBootstrapper.java:82)
```

一般为多对象参数字段重复问题 解决方案: 去掉或者改名 保证字段不重复即可

### 参数不显示问题

检查是否使用 jrebel 热更插件 swagger与之不兼容 使用idea自带方式启动即可解决 检查是否使用了 特殊符号 例如 / 注解内容不允许使用此类特殊



符号 请尽量使用 , 处理 错误示例

## 实体bean为空问题

### 问题排查

检查是否存在 链式调用 注解 `@Accessors(chain = true)` 删除即可

### 原因

java 规范 set 返回值为 `void` 链式调用 set 返回值为 `this` 故多数框架底层使用 jdk 工具导致找不到 set 方法 例如: `easyexcel` `cglib` `mybatis` 等

为何移除druid

## Redis 报错 Permission denied

此报错为无权限

需确保 根目录 /docker 存储文件夹具有写权限

```
chmod 777 /docker
```

没有写权限无法对数据进行存储

关于RDB报错 **/etc** 无权限问题

增加redis密码校验 无密码导致配置不安全

## 关于HTTPS配置

### 后端 HTTPS 改造

将申请的 https 证书放置到 nginx 对应目录 根据框架内 nginx https 示例 更改后端代理为 https

```
server {  
    listen      80;  
    server_name localhost;  
  
    # https配置参考 start  
    #listen      443 ssl;  
  
    # 证书直接存放 /docker/nginx/cert/ 目录下即可 更改证书名称即可 无需更改证书路径  
    #ssl on;  
    #ssl_certificate      /etc/nginx/cert/xxx.local.crt; # /etc/nginx/cert/ 为docker映射的路径  
    #ssl_certificate_key  /etc/nginx/cert/xxx.local.key; # /etc/nginx/cert/ 为docker映射的路径  
    #ssl_session_timeout 5m;  
    #ssl_ciphers  ECDHE-RSA-AES128-GCM-SHA256:ECDHE:ECDSA:RSA:AES:HIGH:!NULL:!aNULL:!MD5:  
    #ssl_protocols TLSv1 TLSv1.1 TLSv1.2;  
    #ssl_prefer_server_ciphers on;  
    # https配置参考 end
```

### 监控中心与任务调度中心改造

各种中心改造 属于系统管控服务 应在内网使用 不应该暴露到外网 也无需配置 https

### Minio https 改造

下方链接包含 minio+nginx 与 minio本身配置https 两种方案 [终极版minio配置https教程](#)

可能的原因

放行接口提示认证失败

可能的原因

接口放行后不需要token即可访问 但是没有token也就无法获取用户信息与鉴权

解决方案

删除接口上的鉴权注解 删除接口内获取用户信息功能 删除数据库实体类 自动注入 `createBy` `updateBy` 因为会获取用户数据

打包jar运行报错问题

打包jar运行报错问题

## 打包jar运行报错问题

常见于 windows 平台以命令方式启动

windows 平台默认编码为 GBK 所以读取到所有的配置都是乱码

## 解决方案

需要在命令增加 `-Dfile.encoding=utf-8` 指定文件编码

例如: `java -Dfile.encoding=utf-8 -jar ruoyi-xxx.jar`

## 如何指定dubbo注册ip

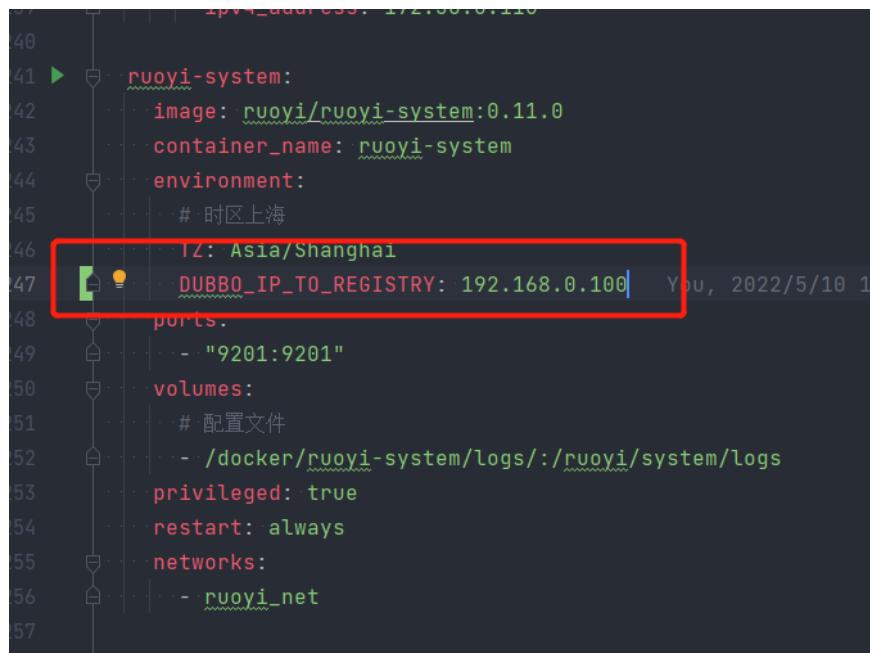
### 重点说明

以下方法指定IP必须是本地有网卡的自己可以访问的IP 不可以随意乱写 (云服务器公网IP是没有网卡的)

### 在 nacos 指定协议IP地址(全局生效)

```
dubbo:  
  protocol:  
    # 指定dubbo协议注册ip  
    host: 192.168.0.100
```

### docker指定dubbo环境变量(单服务生效)



```
40  
41 > ruoyi-system:  
42   image: ruoyi/ruoyi-system:0.11.0  
43   container_name: ruoyi-system  
44   environment:  
45     # 时区上海  
46     TZ: Asia/Shanghai  
47     DUBBO_IP_TO_REGISTRY: 192.168.0.100 | You, 2022/5/10 1  
48   ports:  
49     - "9201:9201"  
50   volumes:  
51     # 配置文件  
52     - /docker/ruoyi-system/logs/:/ruoyi/system/logs  
53   privileged: true  
54   restart: always  
55   networks:  
56     - ruoyi_net  
57
```

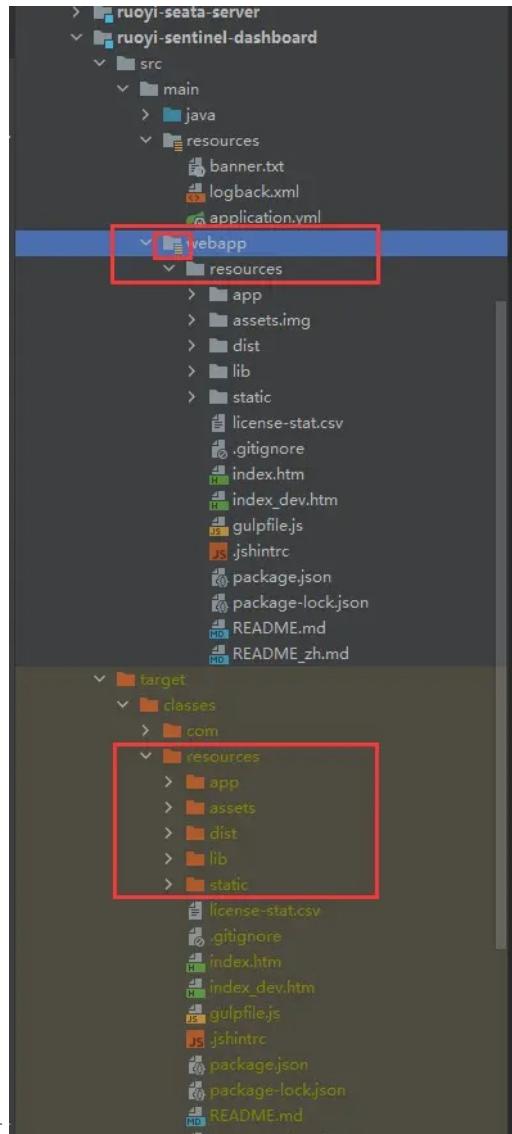
原因

## Sentinel页面404问题

### 原因

检查 webapp 目录是否为资源目录 低版本 idea 不会自动解析

### 解决方案



手动设置 webapp 为资源目录即可

原因

## 无法读取nacos配置

检查 **group** 与 **namespace** 是否一致

如果未使用框架自带 `ry-config.sql` 文件进行配置 会导致 `namespace` 不一致 无法查询配置

检查是否手动改过 **nacos** 数据库数据

`nacos` 数据表层层关联 不要自作聪明手动改数据库

已经改过的 需要重新导入 `ry-config.sql` 之后在页面进行改数据操作

问题原因

不支持ST请求

## 问题原因

经多人反馈 共同点为全都是做 小程序开发 使用的 **uniapp** 发送的网络请求而出现这种问题

**uniapp** 错误设置 `Content-Type` 将所有请求类型全都设置成了 `json` 导致不该读body的请求也读取了body 最终导致报错

## 解决方案

方案1: 升级 1.4.0 已经对这种不合规发的请求做了兼容处理(被迫) 方案2: **uniapp** 内的请求设置正确的 `Content-Type`

## 问题原因

Only one connection receive subscriber allowed

## 问题原因

经多人反馈 共同点为全都是做 小程序开发 使用的 **uniapp** 发送的网络请求而出现这种问题

**uniapp** 错误设置 `Content-Type` 将所有请求类型全都设置成了 `json` 导致不该读body的请求也读取了body 最终导致报错

## 解决方案

方案1: 升级 1.4.0 已经对这种不合规范的请求做了兼容处理(被迫) 方案2: **uniapp** 内的请求设置正确的 `Content-Type`

## 问题原因

nacos 报错 The Raft Group [naming\_instance\_metadata]

## Nacos 服务下线报错问题

问题描述：

Nacos 服务管理 > 服务列表 > 详情 > 下线 报错

报错详情：

```
caused: errCode: 500, errMsg: do metadata operation failed ;caused: com.alibaba.nacos.consistency.exception.ConsistencyException: The Raft G
```

解决方案：

删除 Nacos 根目录下 data 文件夹下的 protocol 文件夹

(推荐使用全局搜索软件查询，windows 环境根目录一般在 C:\Users\用户名\nacos)

问题原因：

Nacos 采用 raft 算法来计算 Leader，并且会记录上次启动的集群地址，所以当我们自己的服务器 IP 改变时(网络环境不稳定，如WIFI，IP 地址也经常变化)，导致 raft 记录的集群地址失效，导致选 Leader 出现问题。

参考目录：

[解决疑难问题之服务下线报：The Raft Group naming\\_instance\\_metadata\] did not find the Leader node; - 嘉美祥瑞 - 博客园 \(cnblogs.com\)](#)

问题原因

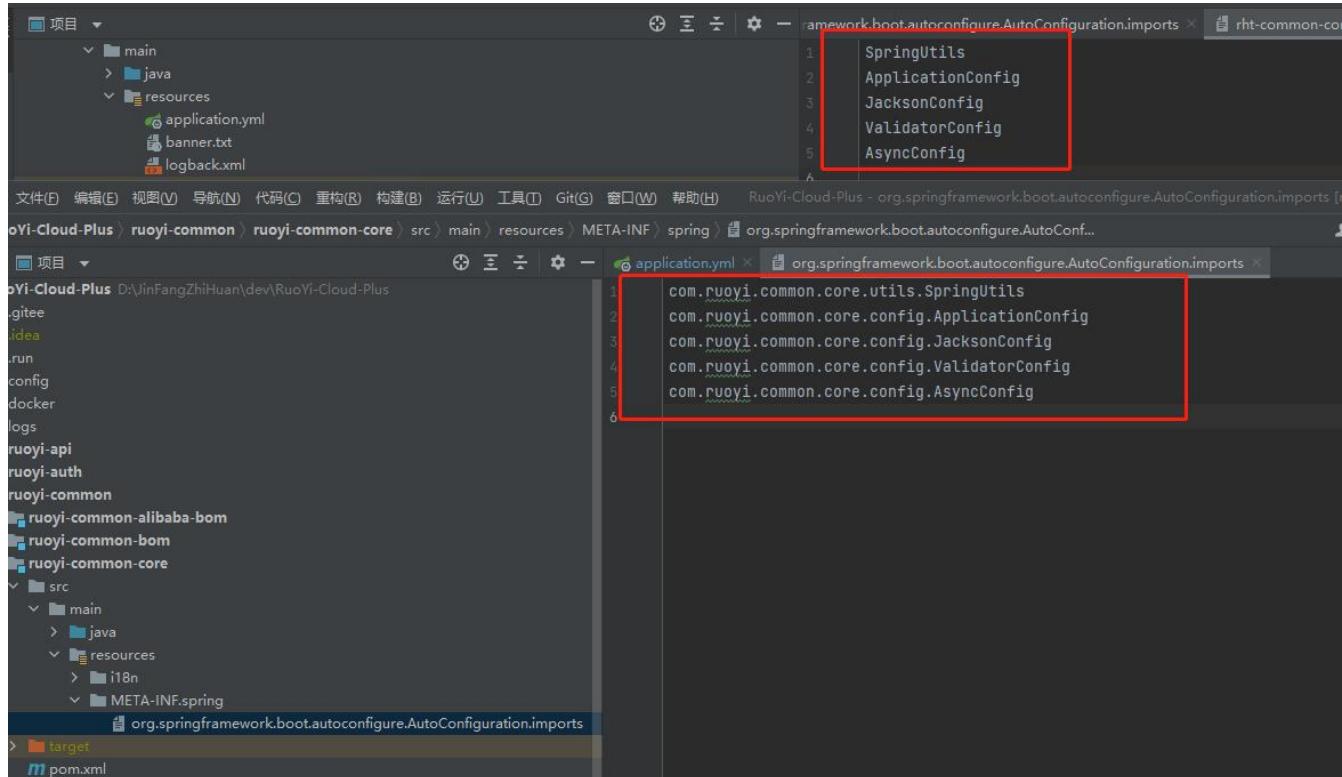
unable to read meta-data for class xxx

## 问题原因

此问题由改包名导致框架内组件 spring 的 spi 配置文件包名被改乱套

## 解决方案

更正组件包下的 spring spi 配置文件内的类包名



The screenshot shows the IntelliJ IDEA interface with two code editors open. The left editor displays the file `org.springframework.boot.autoconfigure.AutoConfiguration.imports`, which contains the following imports:

```
1 SpringUtils
2 ApplicationConfig
3 JacksonConfig
4 ValidatorConfig
5 AsyncConfig
```

The right editor displays the file `application.yml`, which contains the following configuration entries:

```
1 com.ruoyi.common.core.utils.SpringUtils
2 com.ruoyi.common.core.config.ApplicationConfig
3 com.ruoyi.common.core.config.JacksonConfig
4 com.ruoyi.common.core.config.ValidatorConfig
5 com.ruoyi.common.core.config.AsyncConfig
```

Both the imports in the Java file and the configuration entries in the YAML file are highlighted with red boxes.

## 扩展项目

### 基于 RuoYi-Cloud-Plus 的扩展项目列表

精品PR 欢迎投稿

功能介绍	PR地址
拖拽图片调整显示顺序	<a href="https://gitee.com/JavaLionLi/RuoYi-Vue-Plus/pulls/173">https://gitee.com/JavaLionLi/RuoYi-Vue-Plus/pulls/173</a>
增加Jasypt加密库对配置文件加密	<a href="https://gitee.com/JavaLionLi/RuoYi-Vue-Plus/pulls/177">https://gitee.com/JavaLionLi/RuoYi-Vue-Plus/pulls/177</a>
使用富文本wangeditor5替换Quill	<a href="https://gitee.com/JavaLionLi/RuoYi-Vue-Plus/pulls/213">https://gitee.com/JavaLionLi/RuoYi-Vue-Plus/pulls/213</a>
sentinel持久化nacos动态更改	<a href="https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus/pulls/37">https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus/pulls/37</a>
集成screw数据库文档功能模块	<a href="https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus/pulls/42">https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus/pulls/42</a>
Excel导入模板增加批注支持	<a href="https://gitee.com/JavaLionLi/RuoYi-Vue-Plus/pulls/222">https://gitee.com/JavaLionLi/RuoYi-Vue-Plus/pulls/222</a>
压缩包处理工具 支持本地文件/目录+oss文件/网络文件混合	<a href="https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus/pulls/44">https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus/pulls/44</a>
添加websocket模块 支持satoken鉴权	<a href="https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus/pulls/65">https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus/pulls/65</a>

欢迎投稿 项目介绍+项目地址

项目介绍	项目地址
分布式集群扩展	<a href="https://gitee.com/JavaLionLi/RuoYi-Vue-Plus">https://gitee.com/JavaLionLi/RuoYi-Vue-Plus</a>
单体fast扩展	<a href="https://gitee.com/JavaLionLi/RuoYi-Vue-Plus/tree/fast/">https://gitee.com/JavaLionLi/RuoYi-Vue-Plus/tree/fast/</a>
Plus学习笔记(常规功能)	<a href="https://gitee.com/michelle1028/ruoyi-vue-plus-notes">https://gitee.com/michelle1028/ruoyi-vue-plus-notes</a>
Plus学习笔记(微服务组件)	<a href="https://gitee.com/michelle1028/ruoyi-cloud-plus-notes">https://gitee.com/michelle1028/ruoyi-cloud-plus-notes</a>

## 加群方式

### 贡献代码

欢迎各路英雄豪杰 PR 代码 请提交到 dev 开发分支 统一测试发版

框架定位为 微服务通用权限管理系统 原则上不接受业务 PR

VIP群(提供 问题解答 技术支持 技术分享)(捐献100元提供订单号方可入群 开源不易 赚点饭钱)

首先感谢 RuoYi 提供分享开源 框架基于 RuoYi-Cloud 重写大部分功能实现 项目代码、文档 均开源免费可商用 遵循开源协议在项目中保留开源协议文件即可 VIP群是作者提供的私人服务 不代表着项目收费

问问题等于做习题 听作者解答问题等于习题讲解 一个人接触的问题有限 一群人接触的问题无限 早进群早接触更多的问题(每天99+) 承诺: 看见必回复 让你感受作者有多话痨 入群须知: 技术支持 不等于 基础教学

QQ群号 : <637757165>



加群扫码

关注作者



作者博客: <https://lionli.blog.csdn.net/?type=blog> 公众号: <狮子领域 程序圈>

## 使用者登记

使用本开源项目的公司或者组织

登记地址: <https://gitee.com/JavaLionLi/RuoYi-Cloud-Plus/issues/I4VJ7G>

公司名	公司官网	logo
武汉华智讯网络信息技术有限公司	<a href="http://www.xun188.com/">http://www.xun188.com/</a>	
易税信息技术有限公司	<a href="https://www.etax.top">https://www.etax.top</a>	

## 捐献作者

### 捐献作者

作者为兼职做开源,平时还需要工作,如果帮到了您可以请作者吃个盒饭



