

## 目录

目录	1
首页	7
平台简介(公测)	7
软件架构图	8
贡献代码	8
其他	8
捐献作者	9
业务功能	9
更新日志	10
更新日志	10
v1.2.0 - 2022-08-09	10
重大更新	10
依赖升级	10
功能更新	10
新功能	10
问题修复	10
v1.1.0 - 2022-07-18	11
重大更新	11
依赖升级	11
功能更新	11
新功能	11
问题修复	11
v1.0.0 - 2022-06-20	11
新增/优化 工程模块	11
代码依赖改动	12
后续/进行中计划	12
系统演示	13
微服务版本 与 分布式集群版本 功能基本一致	13
专栏与视频	14
粉丝整理 欢迎投稿	14
快速开始	15
项目初始化	15
项目分支说明	15
项目必备环境	15
需勾选 maven 对应环境	15
默认 JDK1.8 如有变动 需更改以下配置	15
sql导入	16
配置 Nacos	16
更改 Nacos 自定义配置	18
idea环境配置	20
配置项目编码	20
配置运行看板	20
配置spring与docker看板	20
配置服务器SSH连接	21
配置服务器FTP连接	22
配置Docker连接	23
可操作远程docker与构建上传docker镜像(代替原来maven docker插件)	23
应用部署	25
版本 >= 1.3.0	25

请优先阅读 idea环境配置	25
手动部署	25
docker 后端部署	25
请优先阅读 idea环境配置	25
将配置使用FTP上传到根目录	25
给docker分配文件夹权限	25
构建应用镜像	26
创建基础服务	27
创建中心服务(需要先构建服务镜像)	27
创建业务服务(需要先构建服务镜像)	27
docker其他操作(idea的docker插件 推荐使用)	27
前端部署	28
更改后端代理路径或者后端ip地址	28
(旧)应用部署	29
手动部署	29
docker 后端部署	29
将源码内 docker 文件夹上传到服务器(注意: 不要放到根目录)	29
开放外网防火墙端口(内网服务无需开启)	29
放置挂载文件(切勿多次执行)	29
分配文件夹权限	29
启动基础服务	29
启动可视化服务	29
启动与停止业务服务(需要先构建服务镜像)	29
停止所有服务	29
删除所有容器	30
删除所有空版本镜像	30
构建服务镜像	30
前端部署	30
Seata服务搭建	32
版本 >= 0.12.0 源码集成 Seata 1.5.X 服务端	32
版本 < 0.12.0	32
参考博客	32
如下配置修改	32
框架功能	34
项目结构	34
目录结构	34
创建新服务	36
最简单的方式	36
多团队开发	38
功能介绍	38
重点说明	38
分页功能	39
重点说明	39
代码用法	39
事务相关	41
多数据源	42
关于多数据源	42
用法参考demo模块	42
用法	42
更多用法	42

OSS功能	43
关于OSS模块使用	43
重点注意事项	43
代码使用	43
功能配置	43
配置OSS	43
切换OSS	47
扩展分类	48
上传图片或文件	48
列表展示	50
删除功能	53
下载功能	54
数据权限	56
关于数据权限	56
使用教程	56
数据权限功能:	56
数据权限相关代码	56
使用方式 参考demo模块	56
创建角色 test1 为 本部门及以下	56
创建角色 test2 为 仅本人	56
将其分配给用户 test	57
编写列表查询(注意: 数据权限注解只能在 Mapper 层使用)	57
编写数据权限模板	58
测试代码	58
自定义SQL模板	60
mybatis-plus 原生方法 增加数据权限过滤	62
支持类标注	62
防重幂等	64
功能介绍	64
美团GTIS系统流程图	64
使用方法	64
数据脱敏	66
功能说明	66
使用教程	66
脱敏逻辑修改	67
国际化	68
国际化方案	68
获取 code 对应国际化内容	68
使用 Validator 框架校验 controller 参数返回国际化	69
使用 Validator 框架校验 Bean 返回国际化	71
短信模块	74
配置功能	74
功能使用	74
重点须知	75
接口放行	76
使用方式	76
代码生成	77
功能介绍	77
数据源配置	77
导入数据表	77

编辑表生成结构	79
生成条件影响	80
树表配置	81
主子表说明	81
预览功能	82
代码结构同步	82
接口文档	83
版本 >= 1.2.0	83
说明	83
文档工具使用	83
Swagger升级SpringDoc指南	83
建议使用 Apifox	83
接入框架	83
(旧)接口文档	88
版本 <= 1.1.0	88
访问方式	88
前端访问 系统工具 -> 系统接口	88
文档配置	88
ruoyi-doc 文档服务配置说明	89
内网鉴权	91
功能介绍	91
开启/关闭内网鉴权	91
放行内网鉴权	91
修改包名	92
关于修改包名	92
主键使用说明	93
关于如何使用分布式id或雪花id	93
重点说明	93
修改应用路径	94
修改访问后端接口路径	94
修改前端页面访问路径	94
关于多表查询	96
建议单表查询	96
前端相关文档	97
扩展功能	98
ELK搭建	98
环境搭建	98
运行命令	98
参考文章	98
项目内配置	98
ES搜索引擎	99
环境搭建(如果已经搭建了ELK则跳过)	99
运行命令	99
Easy-ES 文档	99
用法	99
RabbitMQ搭建	100
环境搭建	100
用法参考	100
RocketMQ搭建	101
环境搭建	101

用法参考	101
Kafka搭建	102
环境搭建	102
用法参考	102
常见问题	103
Lombok注解爆红	103
如何使用Tomcat	104
关于如何使用Tomcat	104
更改 ruoyi-common-web 模块依赖 使用 springboot 默认依赖 即为 tomcat	104
更改 application.yml 文件, 将 undertow 配置改为 tomcat 配置	104
导入excel实体类为空	106
如何同步项目更新	107
ParseException SQL解析异常	108
异常内容	108
异常由来	109
解决方案	109
swagger相关问题	111
空指针问题	111
参数不显示问题	111
实体bean为空问题	112
问题排查	112
原因	112
Redis 报错 Permission denied	113
此报错为无权限	113
关于RDB报错 /etc 无权限问题	113
关于HTTPS配置	114
后端 HTTPS 改造	114
监控中心 与 任务调度中心 改造	114
Minio https 改造	114
放行接口提示认证失败	115
可能的原因	115
解决方案	115
打包jar运行报错问题	116
打包jar运行报错问题	116
解决方案	116
如何指定dubbo注册ip	117
重点说明	117
在nacos指定协议IP地址(全局生效)	117
docker指定dubbo环境变量(单服务生效)	117
Sentinel页面404问题	118
原因	118
解决方案	118
无法读取nacos配置	119
检查 group 与 namespace 是否一致	119
检查是否手动改过 nacos 数据库数据	119
不支持ST请求	120
问题原因	120
多方用户提供修复意见	120
Only one connection receive subscriber allowed	121
问题原因	121

## 目录

多方用户提供修复意见	121
扩展项目	122
基于 RuoYi-Cloud-Plus 的扩展项目列表	122
精品PR 欢迎投稿	122
欢迎投稿 项目介绍+项目地址	122
加群与捐献	123
贡献代码	123
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)	123
捐献作者	123
关注作者	123
使用者登记	124
使用本开源项目的公司或者组织	124

# 首页

## 平台简介(公测)

830 Stars 29 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)

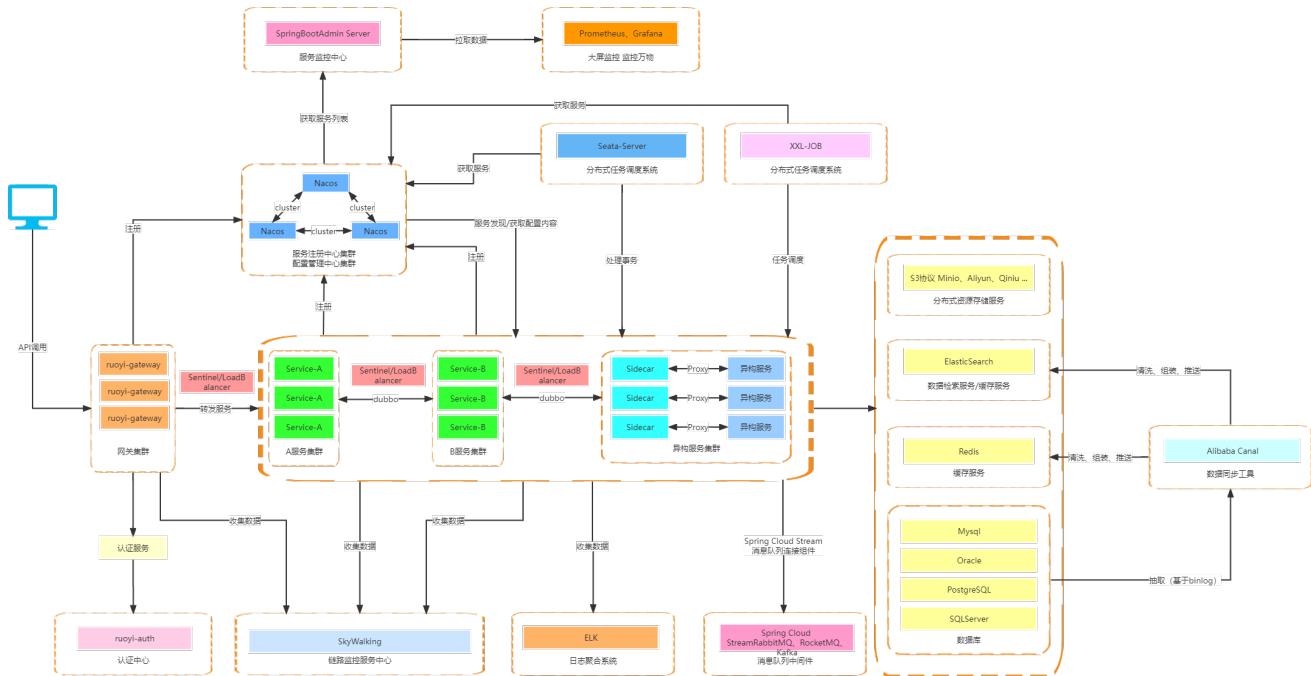
项目代码、文档 均开源免费可商用 遵循开源协议在项目中保留开源协议文件即可 活到老写到老 为兴趣而开源 为学习而开源 为让大家真正可以学到技术而开源

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(未完成)	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(未完成)	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

- 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

v1.2.0 - 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 密等组件 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
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

# 快速开始

## 项目初始化

### 项目分支说明

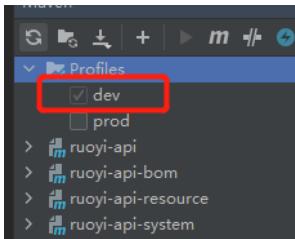
master 主分支 稳定发布分支 dev 开发分支 代码随时更新 不推荐使用 经测试后会发布到主分支 future/\* 新功能预览分支

### 项目必备环境

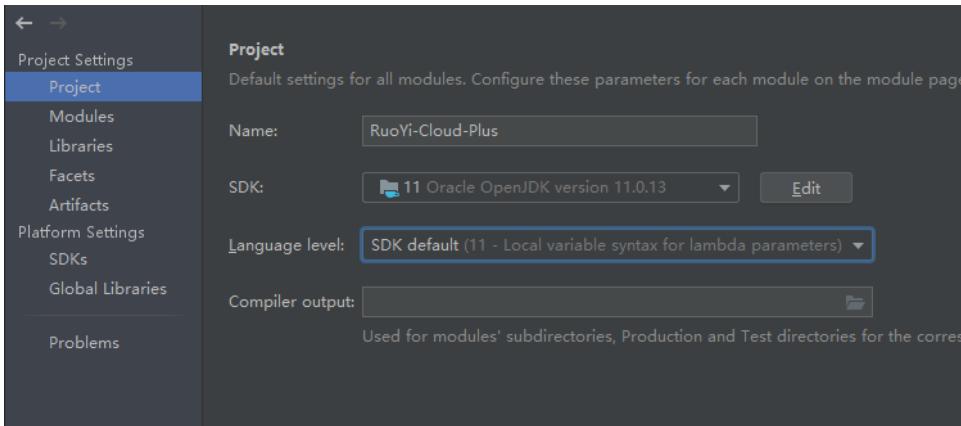
推荐使用 docker 安装 项目内置 docker 编排文件

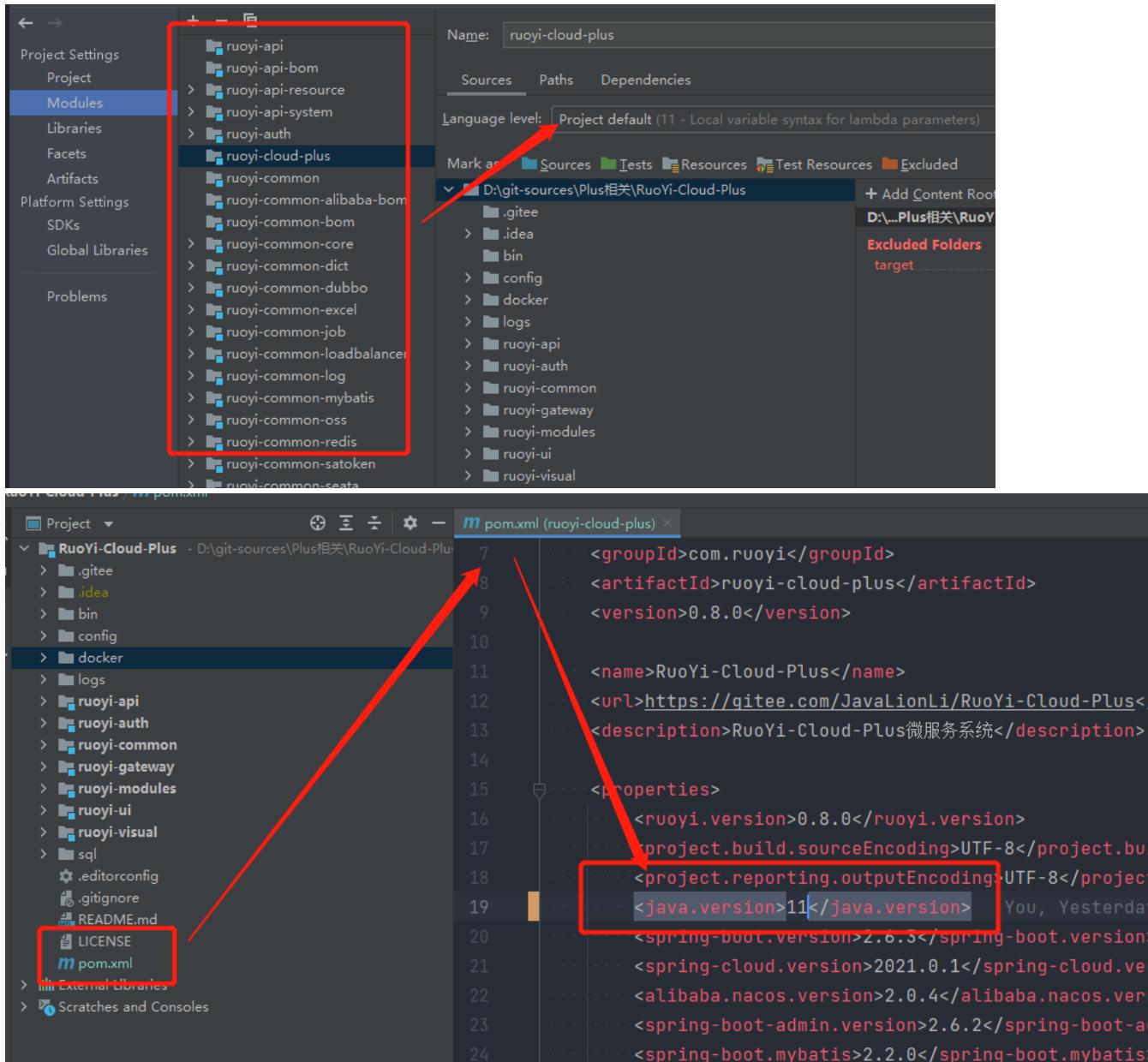
- oracle jdk 8.11 (暂时不支持 17 不支持大于 jdk8\_202 因为202是最后一个免费版本)
- mysql 5.7.8.0 (5.6未适配可能会有问题)
- redis 5.X 6.X 由于框架大量使用了redis特性 版本必须 >= 5.X ([win redis 下载地址](#))
- minio 本地文件存储 或 阿里云 腾讯云 七牛云等一切支持S3协议的云存储
- maven 3.6.3 3.8.X
- nodejs >= 12
- npm 6.X 8.X (7.X确认有问题)
- nacos >= 2.0.X
- sentinel 框架内置
- seata 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

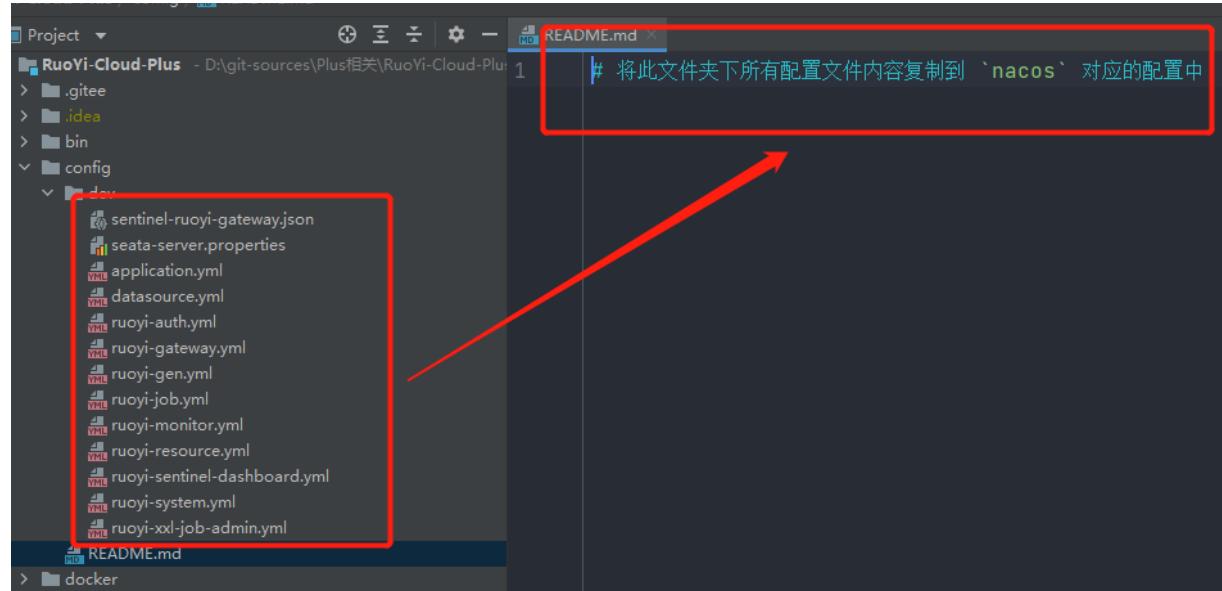


将sql导入到与sql文件名对应的数据库(不要放到一个库下)

## 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)



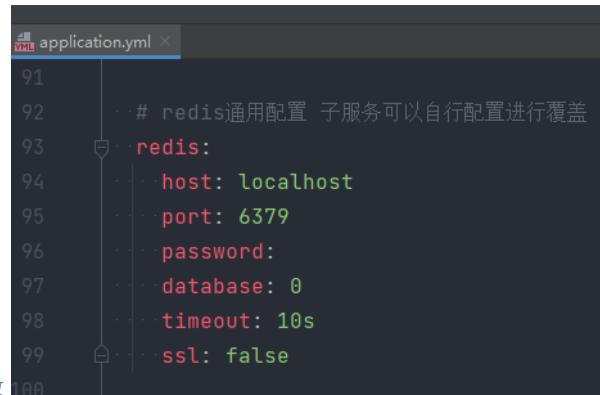
Data ID	Group	Group
application.yml	DEFAULT_GROUP	
datasource.yml	DEFAULT_GROUP	
ruoyi-gateway.yml	DEFAULT_GROUP	
ruoyi-auth.yml	DEFAULT_GROUP	
ruoyi-monitor.yml	DEFAULT_GROUP	
ruoyi-system.yml	DEFAULT_GROUP	
ruoyi-gen.yml	DEFAULT_GROUP	
ruoyi-job.yml	DEFAULT_GROUP	
ruoyi-resource.yml	DEFAULT_GROUP	
ruoyi-sentinel-dashboard.yml	DEFAULT_GROUP	
ruoyi-xxl-job-admin.yml	DEFAULT_GROUP	
sentinel-ruoyi-gateway.json		
seata-server.properties		

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>
```

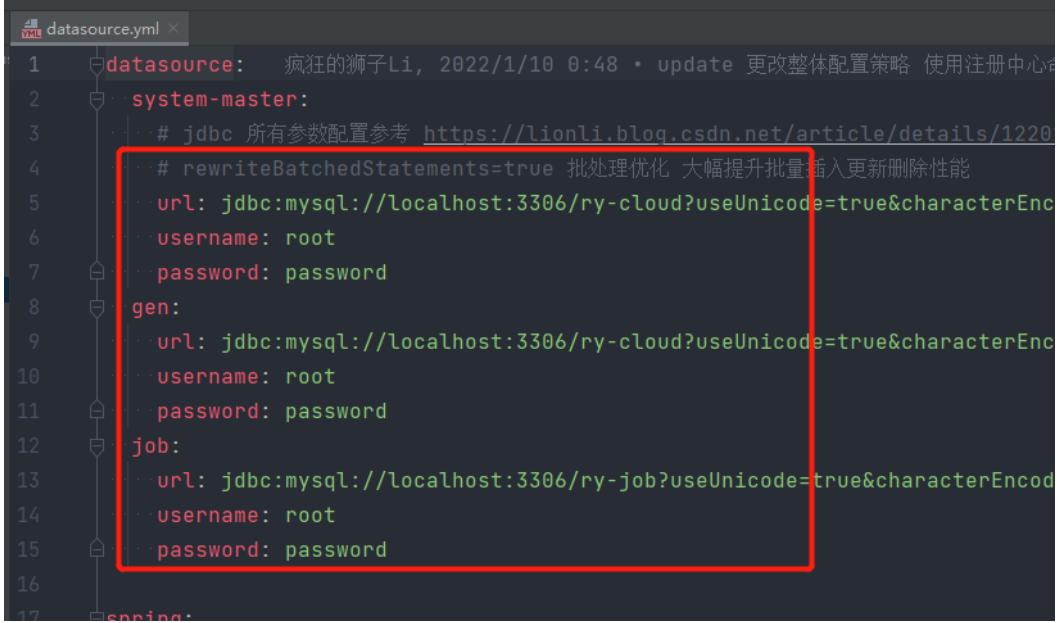
## 更改 Nacos 自定义配置

忠告: 微服务配置相当复杂 请勿在不懂原理的情况下乱改



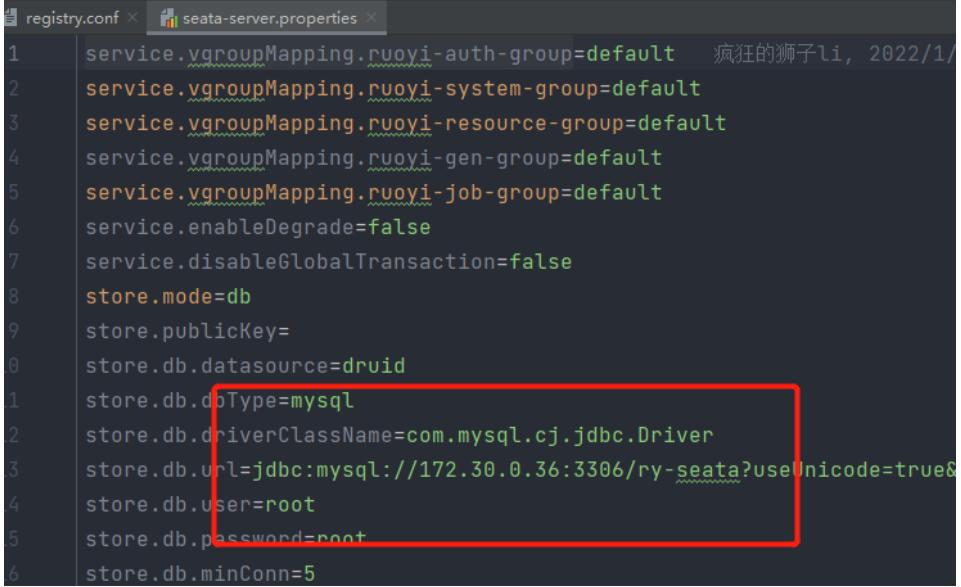
The screenshot shows a code editor window with the file 'application.yml' open. The content of the file is as follows:

```
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



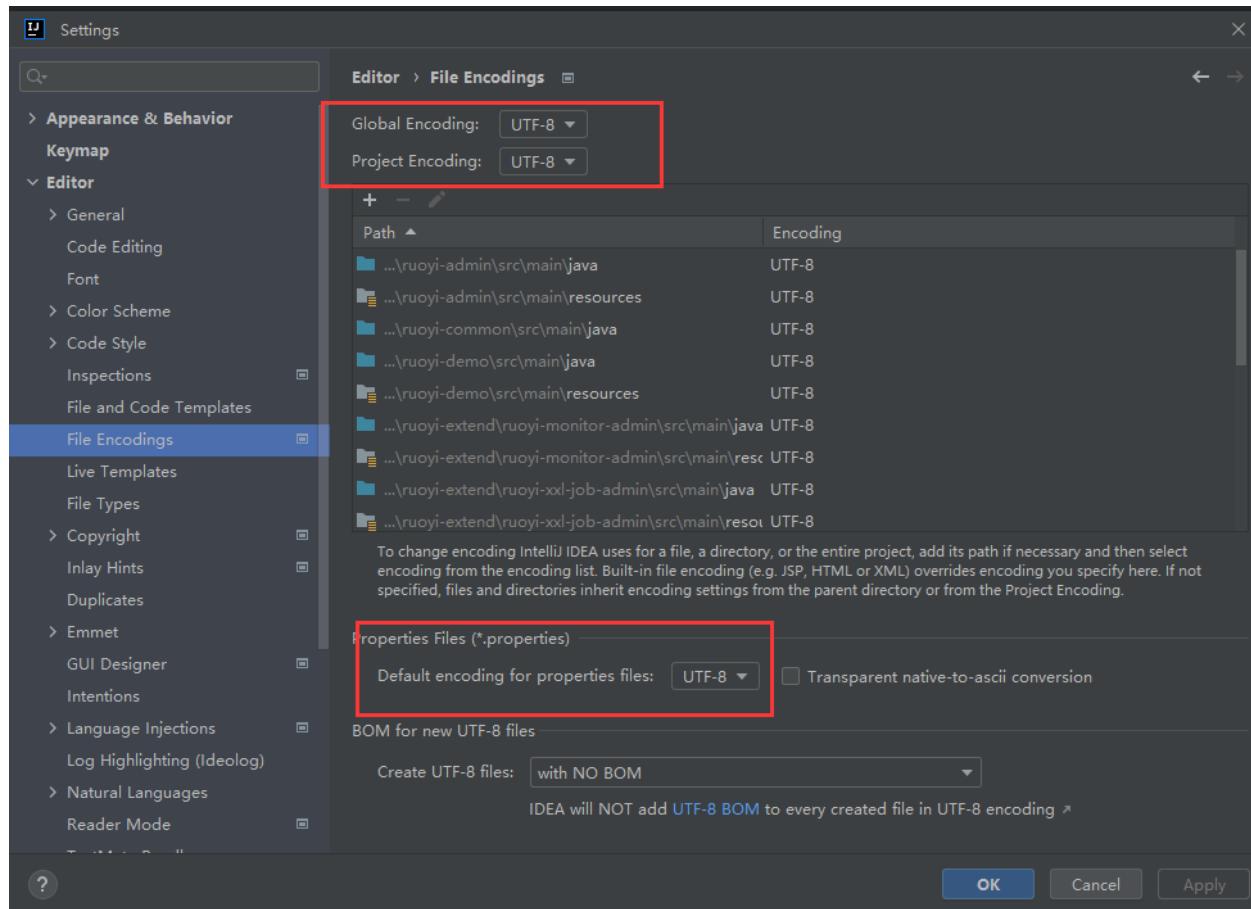
```
service.vgroupMapping.ruoyi-auth-group=default 疯狂的狮子li, 2022/1/
service.vgroupMapping.ruoyi-system-group=default
service.vgroupMapping.ruoyi-resource-group=default
service.vgroupMapping.ruoyi-gen-group=default
service.vgroupMapping.ruoyi-job-group=default
service.enableDegrade=false
service.disableGlobalTransaction=false
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&
store.db.user=root
store.db.password=root
store.db.minConn=5
```

seata-server.properties 更改

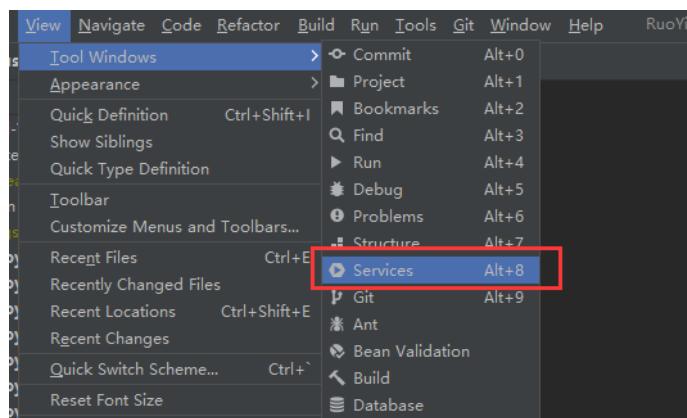
配置项目编码

## idea环境配置

### 配置项目编码

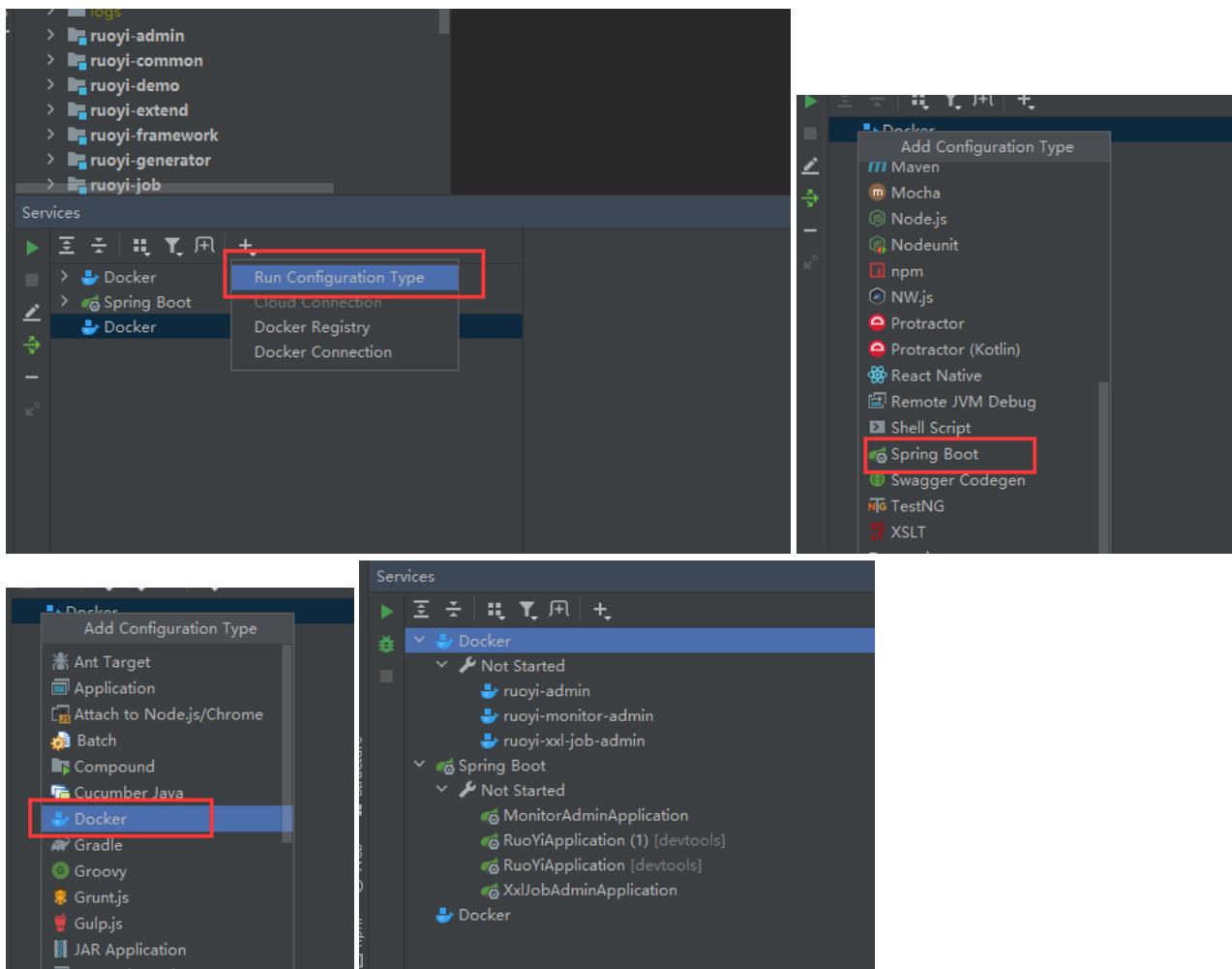


### 配置运行看板



### 配置spring与docker看板

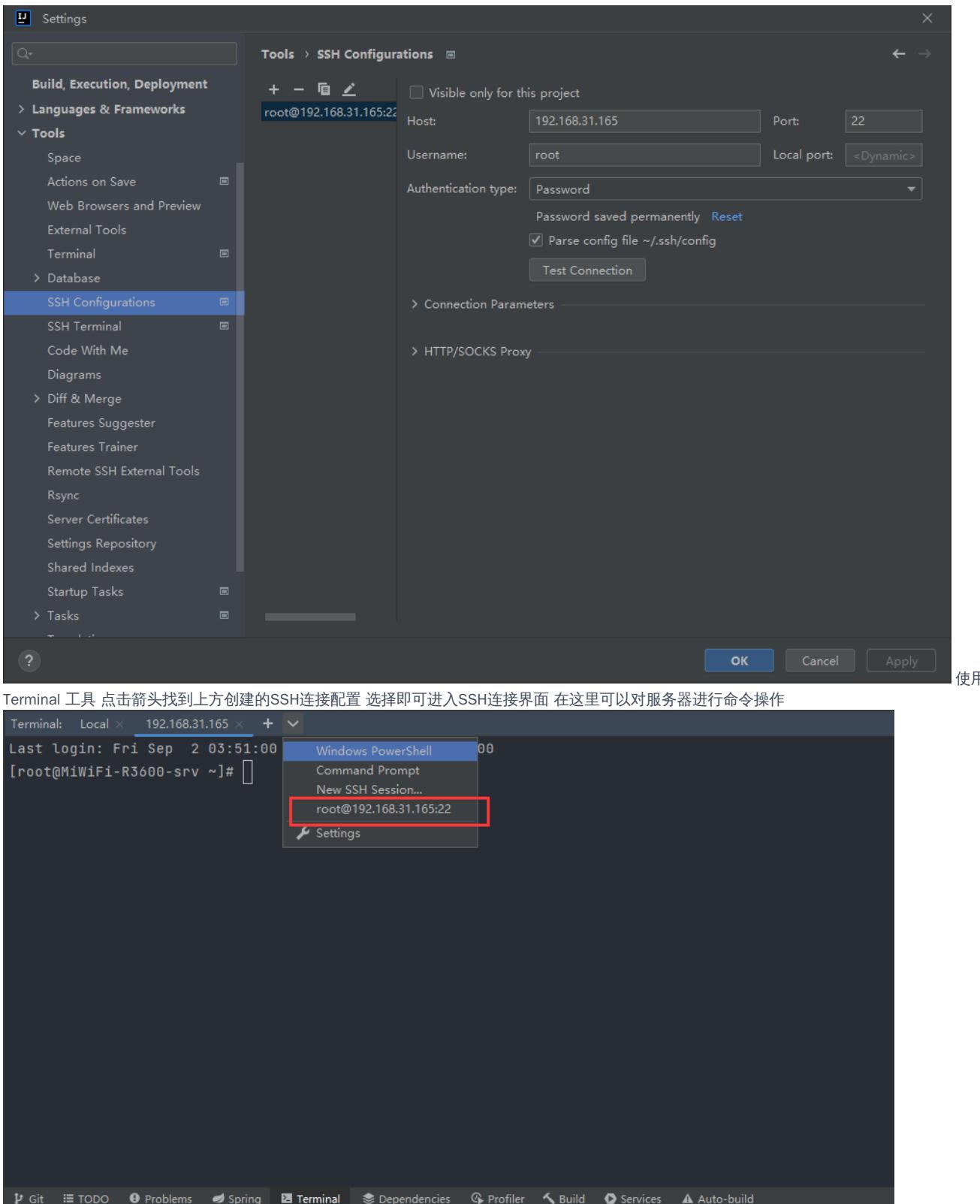
配置服务器SSH连接



## 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

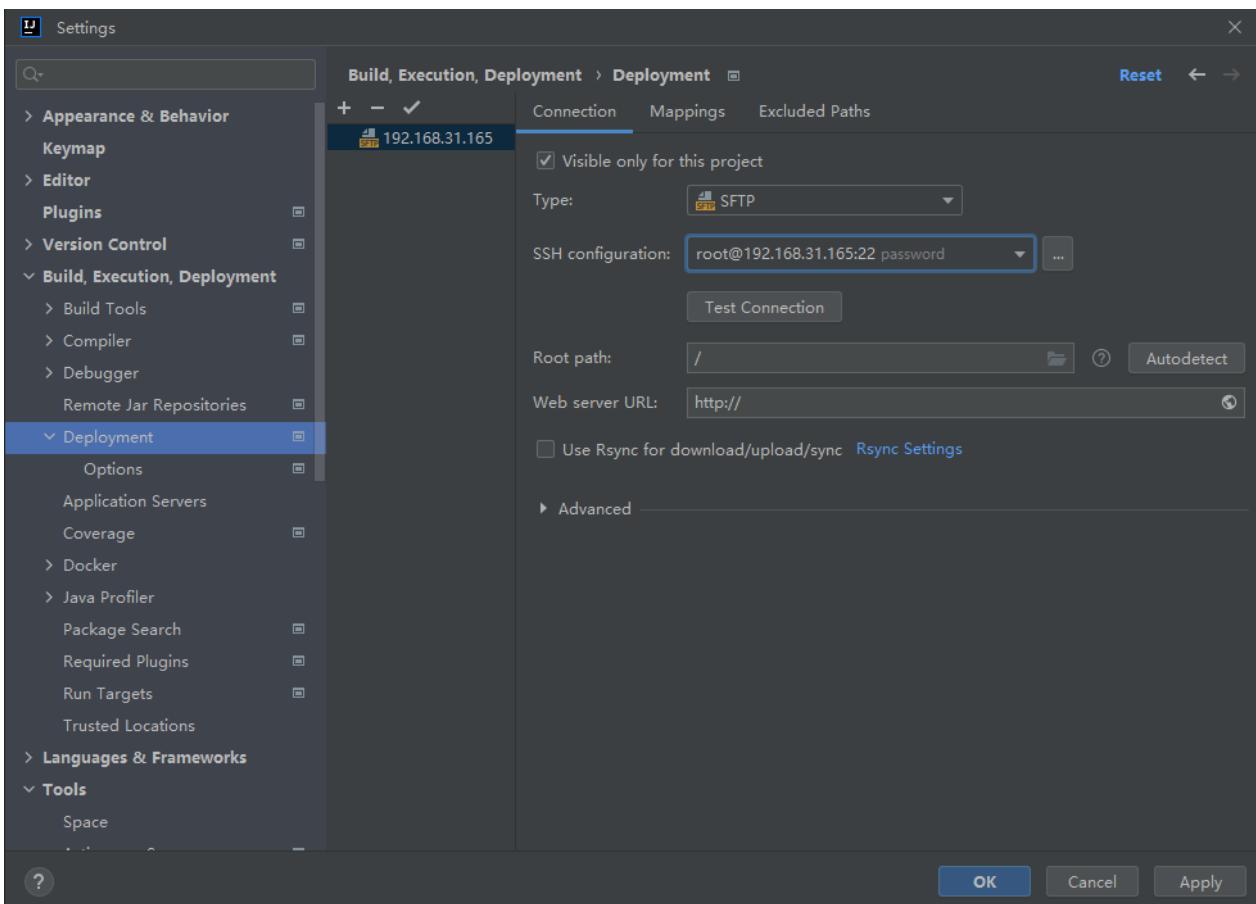
## 配置服务器FTP连接



## 配置服务器FTP连接

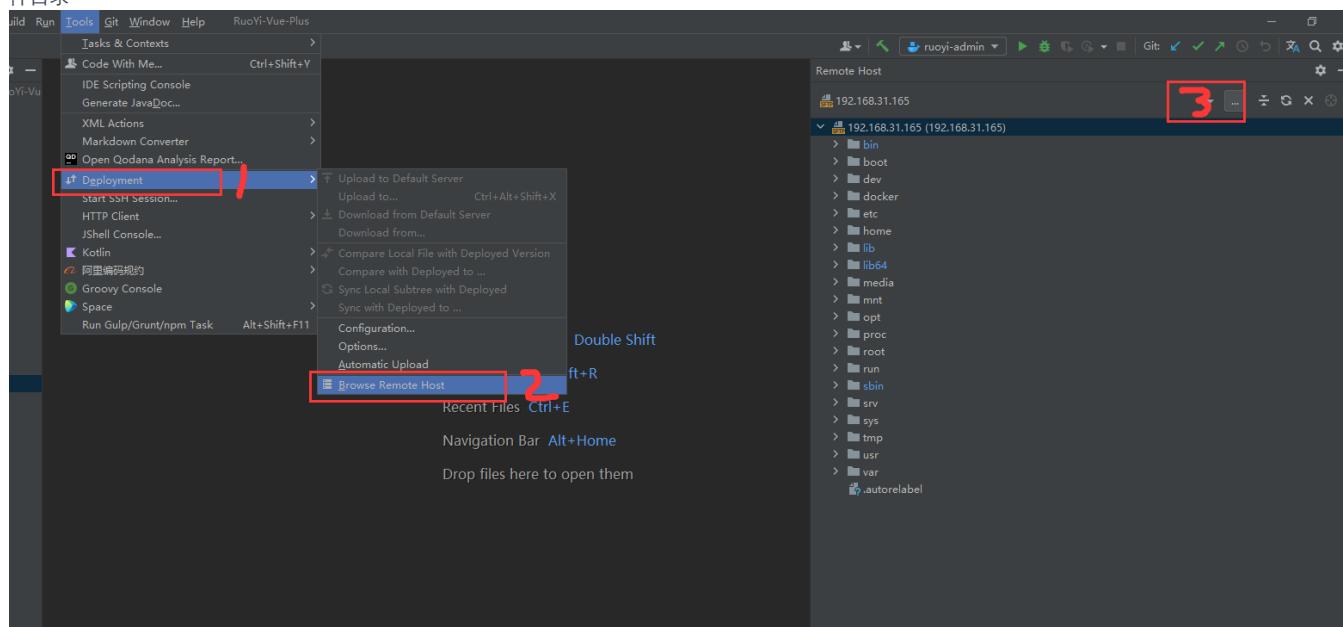
进入 `Settings -> Build-> Deployment` 点击加号 选择SFTP 创建 FTP 连接配置 选择之前创建好的SSH配置 点击 `Test Connection` 测试连接

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

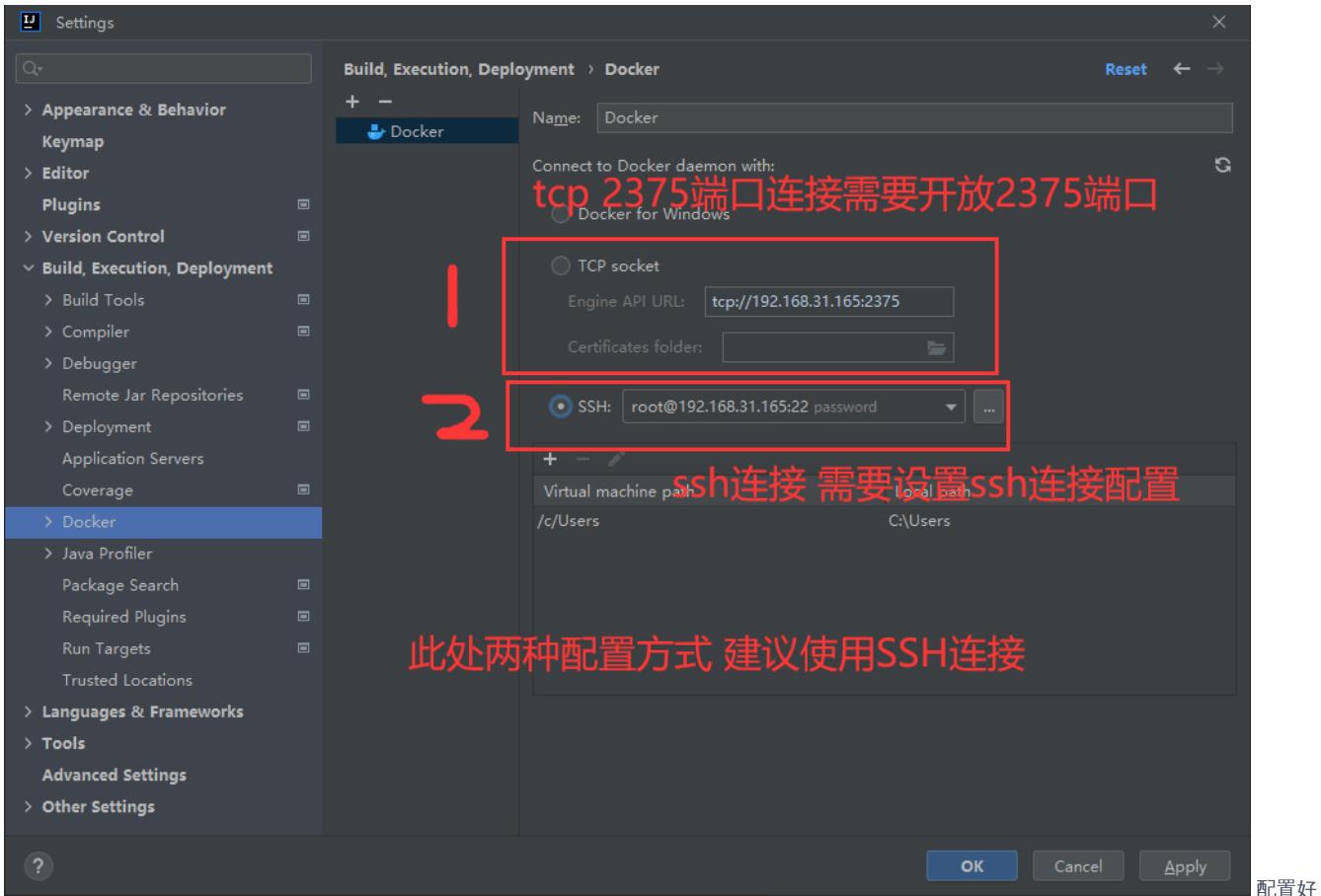


## 配置Docker连接

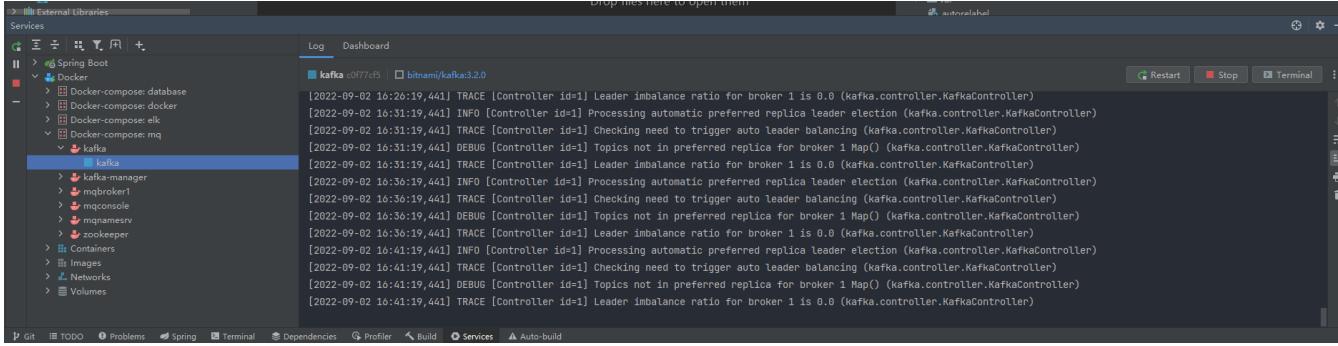
可操作远程docker与构建上传docker镜像(代替原来maven docker插件)

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

idea拖拽文件到远程目录即可上传



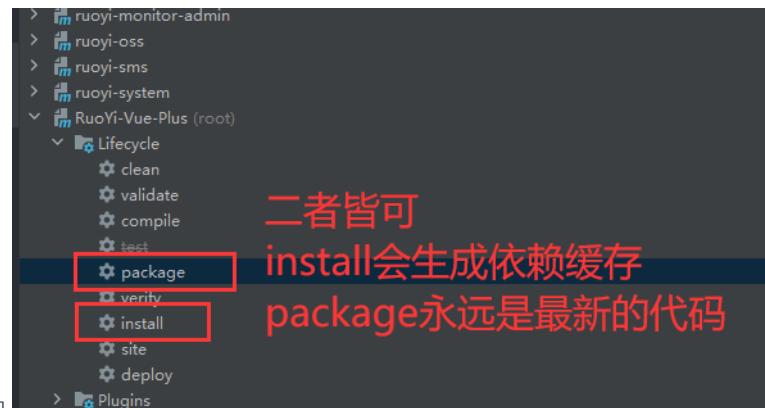
给docker分配文件夹权限

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

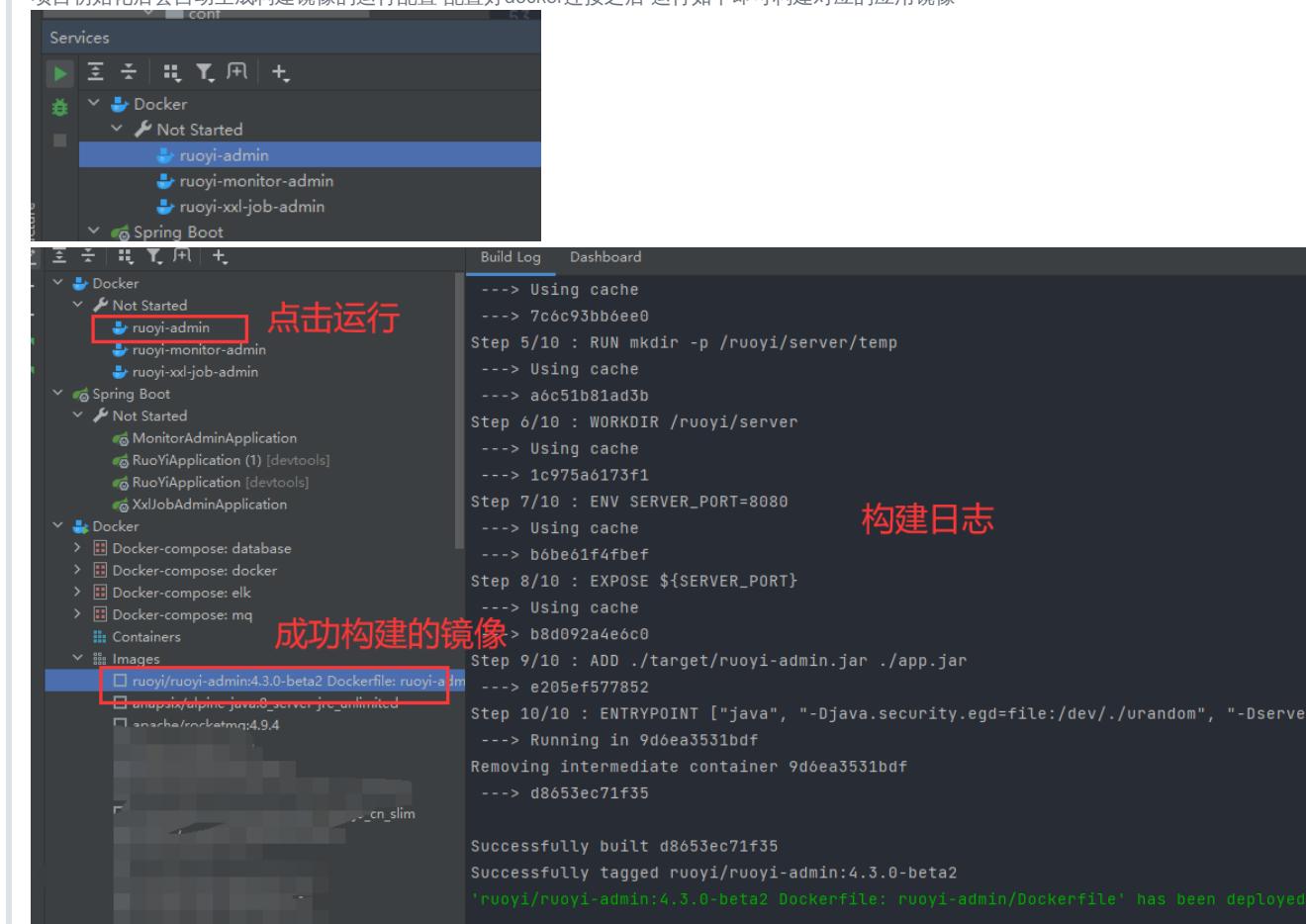
## 构建应用镜像

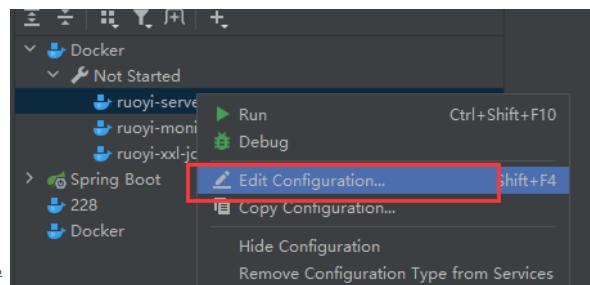


1. 需要先使用maven打包成jar包

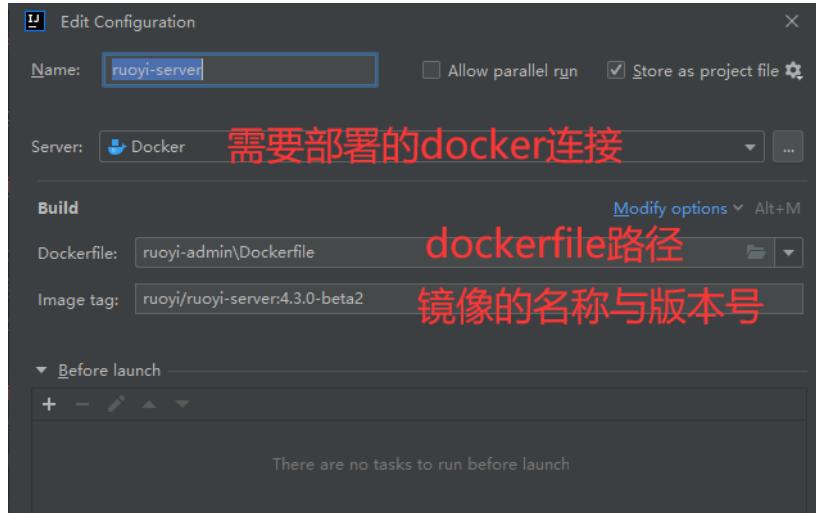
2. 执行构建

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。





3.结构讲解 右键编辑 即可看到内部配置



创建基础服务

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

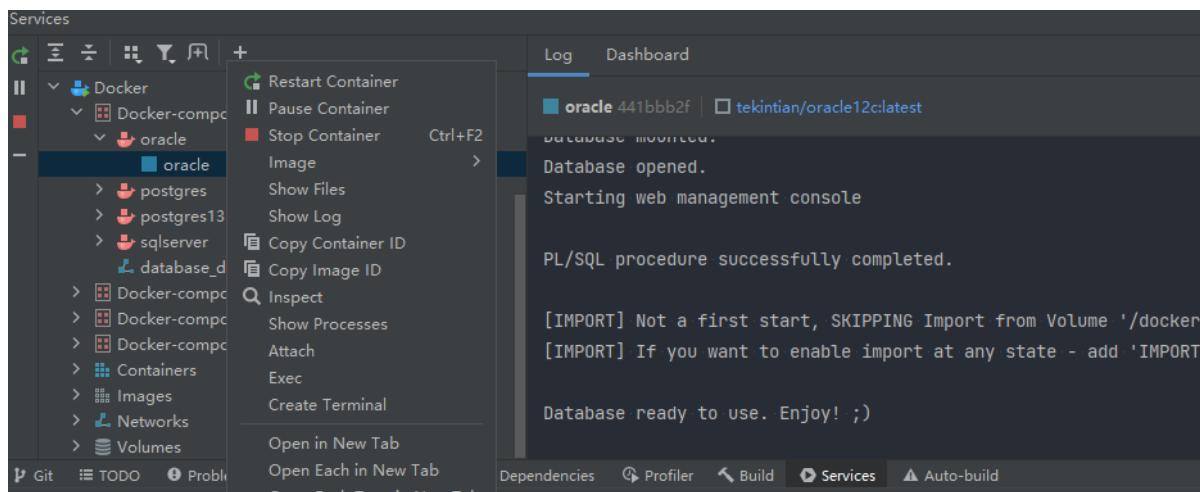
创建中心服务(需要先构建服务镜像)

```
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/;
...     }
```

路径对应前端环境文件

```
12
13 # 监控地址
14 VUE_APP_XXL_JOB_ADMIN = '/xxl-job-admin'
15
16 # 若依管理系统/生产环境
17 VUE_APP_BASE_API = '/prod-api'
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

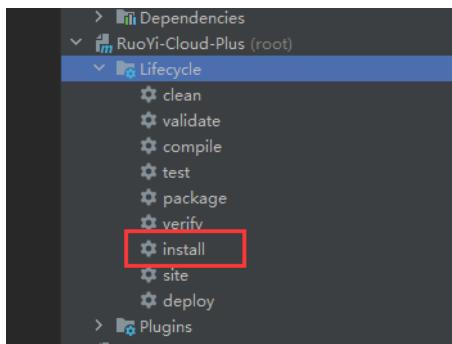
```
sh deploy.sh rm
```

### 删除所有空版本镜像

```
sh deploy.sh rmiNoneTag
```

### 构建服务镜像

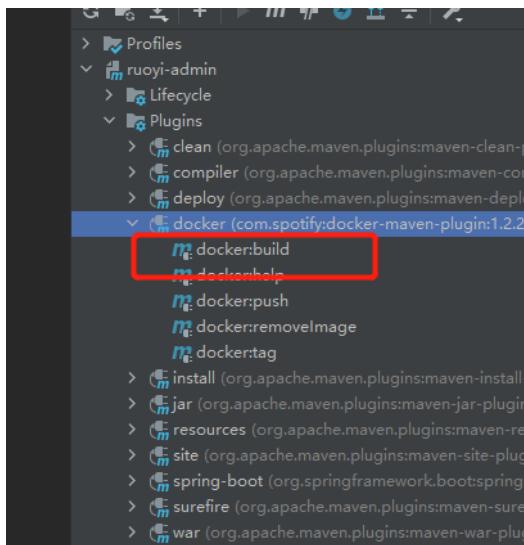
首先 使用 maven 总工程打包 所有模块jar包(注意: 此处需使用 install 因有几个依赖工程需要装载到本地)



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

版本 < 0.12.0

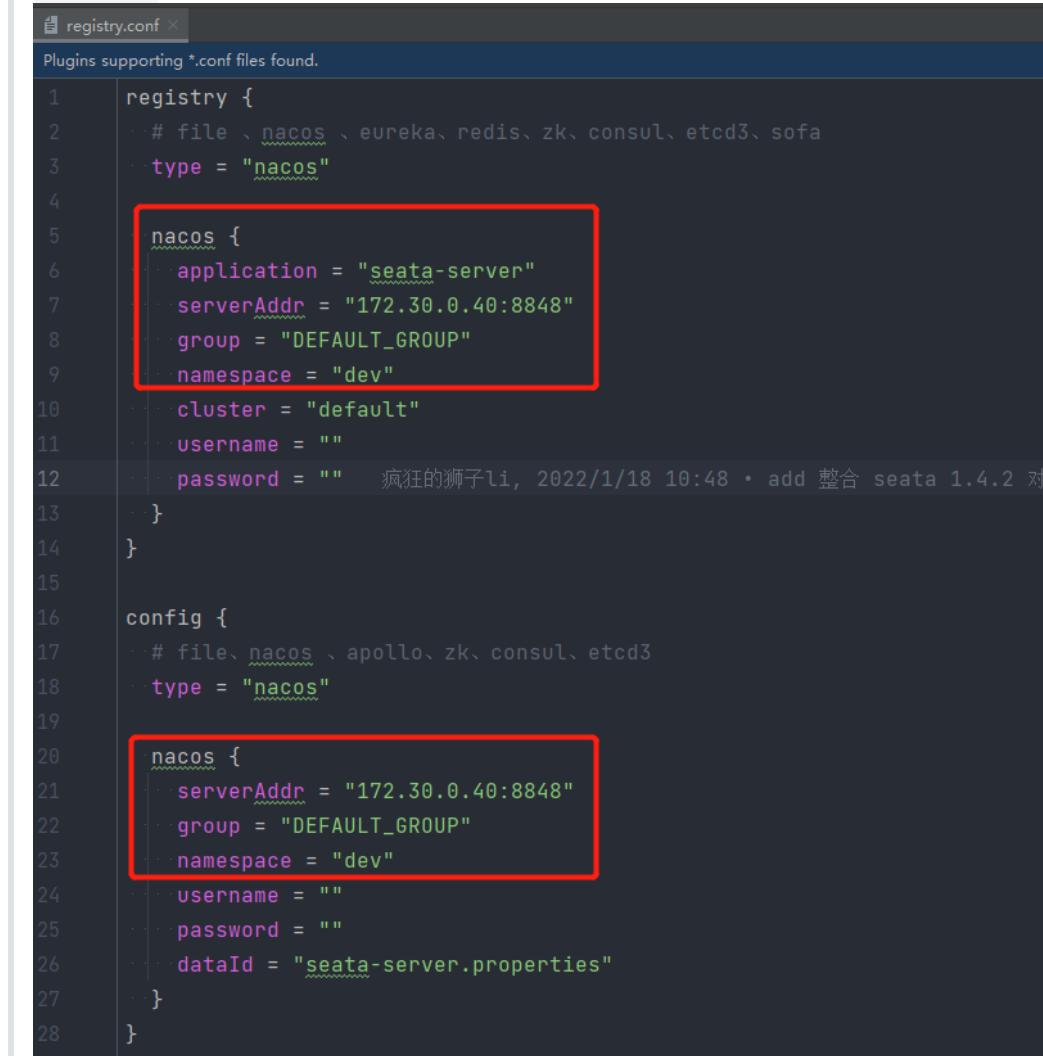
参考博客

[SpringCloud Alibaba 2021 整合 Seata 1.4.X 对接 Nacos 注册配置中心 Mysql 存储 避坑指南](#)

如下配置修改

项目已内置(`registry.conf` 与 `seata-server.properties` 文件)

`registry.conf` 修改 Nacos Ip地址 其余与图内保持一致



```
registry {
    # file、nacos、eureka、redis、zk、consul、etcd3、sofa
    type = "nacos"

    nacos {
        application = "seata-server"
        serverAddr = "172.30.0.40:8848"
        group = "DEFAULT_GROUP"
        namespace = "dev"
        cluster = "default"
        username = ""
        password = ""    疯狂的狮子li, 2022/1/18 10:48 · add 整合 seata 1.4.2 对
    }
}

config {
    # file、nacos、apollo、zk、consul、etcd3
    type = "nacos"

    nacos {
        serverAddr = "172.30.0.40:8848"
        group = "DEFAULT_GROUP"
        namespace = "dev"
        username = ""
        password = ""
        dataId = "seata-server.properties"
    }
}
```

`seata-server.properties` 在 Nacos 上修改 对应的数据库地址 与 服务关联组

```
registry.conf x seata-server.properties x
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=UTF-8
store.db.user=root
store.db.password=root
store.db.minConn=5
store.db.maxConn=30
store.db.globalTable=global_table
store.db.branchTable=branch_table
store.db.distributedLockTable=distributed_lock
store.db.queryLimit=100
```

# 框架功能

## 项目结构

### 目录结构

v1.0.0

```

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 [9200]
└─ 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-dubbo     // dubbo集成模块
    └─ ruoyi-common-excel     // excel集成模块
    └─ ruoyi-common-idempotent // 繁等功能模块
    └─ 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-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-sms         // 短信集成模块
    └─ ruoyi-common-swagger     // 接口文档集成模块
    └─ ruoyi-common-web         // web服务集成模块
└─ ruoyi-example   // 例子模块
└─ ruoyi-demo      // 演示模块 [9401]
└─ ruoyi-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]
    └─ ruoyi-resource  // 资源模块 [9300]
    └─ ruoyi-system    // 系统模块 [9201]
└─ ruoyi-visual    // 可视化模块
    └─ ruoyi-doc      // 接口文档模块 [18000]
    └─ ruoyi-monitor   // 服务监控模块 [9100]
    └─ ruoyi-seata-server // seata服务模块 [7091,8091]
    └─ ruoyi-sentinel-dashboard // sentinel控制台模块 [8718]
    └─ ruoyi-xxl-job-admin // xxljob 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);
        System.out.println("(*_*) 系统模块启动成功");
    }
}

```

服务启动类 名称改掉

```

<configuration scan="true" scanPeriod="60 seconds" debug="false">
    <!-- 日志存放路径 -->
    <property name="log.path" value="logs/ruoyi-system" />
    <!-- 日志输出格式 -->
    <property name="console.log.pattern"
        value="%red(%d{yyyy-MM-dd HH:mm:ss}) %green([%thread]) %highlighted%" />
    <property name="log.pattern" value="%d{yyyy-MM-dd HH:mm:ss} [%thread] %highlighted%" />

```

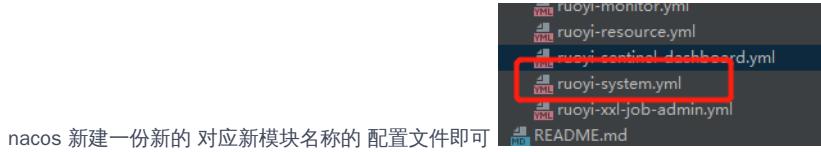
logback.xml 日志存储名 改掉

```

# 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@

```

bootstrap.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 配置文件即可

## 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) 访问站点为 域名 + 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

```
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 (highlighted with a red arrow), 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 (highlighted with a red arrow). The main content area shows the OSS configuration page with tabs for 配置管理 (highlighted with a red arrow) and 文件展示. It includes search and filter fields for 文件名, 原名, 文件后缀, 上传人, 服务商, and search/refresh buttons. A message indicates "暂无数据". Below this is a table listing OSS configurations:

	配置key	访问站点	自定义域名	桶名称	前缀	域	状态	操作
<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>

At the bottom, pagination controls show "共 5 条" and "10条/页" with a current page of 1.

## 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 修改 删除

<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

对象存储

存储区域: 华北 存储量: 0 对象数: 0 访问控制: 公开 空间类型: 自有空间

空间概览 文件管理 域名管理 图片样式 转码样式 空间设置

跨区域同步

统计分析

空间管理

CDN 加速域名 [自定义域名](#) CDN 测试域名

基础配置

访问控制 [点击进入](#) 生命周期 [点击进入](#) 空间授权 [点击进入](#) 事件通知 [点击进入](#) 镜像回源 [点击进入](#) S3 域名 [点击查看](#)

S3 域名

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]

阿里云 访问站点

## 关于OSS模块使用

The screenshot shows the Aliyun OSS console for a bucket named 'ruoyi' in the '华北2 (北京)' region. The left sidebar lists various management options like用量查询 (Usage Query), 文件管理 (File Management), 权限管理 (Permission Management), etc. The main area is divided into sections: 基础数据 (Basic Data) and 访问域名 (Access Domains). In the 基础数据 section, there's a note about average delay of 1-2 hours for statistical data. It displays storage usage (0 Byte), traffic (127.5 KB), and requests (170). The 访问域名 section shows the Endpoint (地域节点) as 'oss-cn-beijing.aliyuncs.com', which is highlighted with a red box. Other listed domains include ECS的经典网络访问 (内网) and ECS的VPC网络访问 (内网), both pointing to 'oss-cn-beijing-internal.aliyuncs.com'. A note at the bottom says '传输加速域名 (全地域上传下载加速)' is not enabled.

## 腾讯云 访问站点

The screenshot shows the Tencent Cloud OSS console for a bucket named 'ruoyi'. The left sidebar includes options like 概览 (Overview), 文件列表 (File List), 基础配置 (Basic Configuration), 安全管理 (Security Management), etc. The main area has two tabs: 用量概览 (Usage Overview) and 基本信息 (Basic Information). The 用量概览 tab shows storage (0 B), traffic (286.94 KB), and requests (279 times). The 基本信息 tab shows the bucket name 'ruoyi', location '北京 (中国) (ap-beijing)', creation time '2022-03-30 15:29:09', and access rights '公有读私有写'. The 域名信息 (Domain Information) section shows the '访问域名' (Access Domain) as 'https://ruoyi-100711111.cos.ap-beijing.myqcloud.com', which is also highlighted with a red box. A note at the bottom right says '注意: 此处只需要黄线内的部分' (Note: Only the part inside the yellow line is required).

## 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	localhost:9000	ruoyi	image
<input type="checkbox"/> image			

指定需要使用的配置

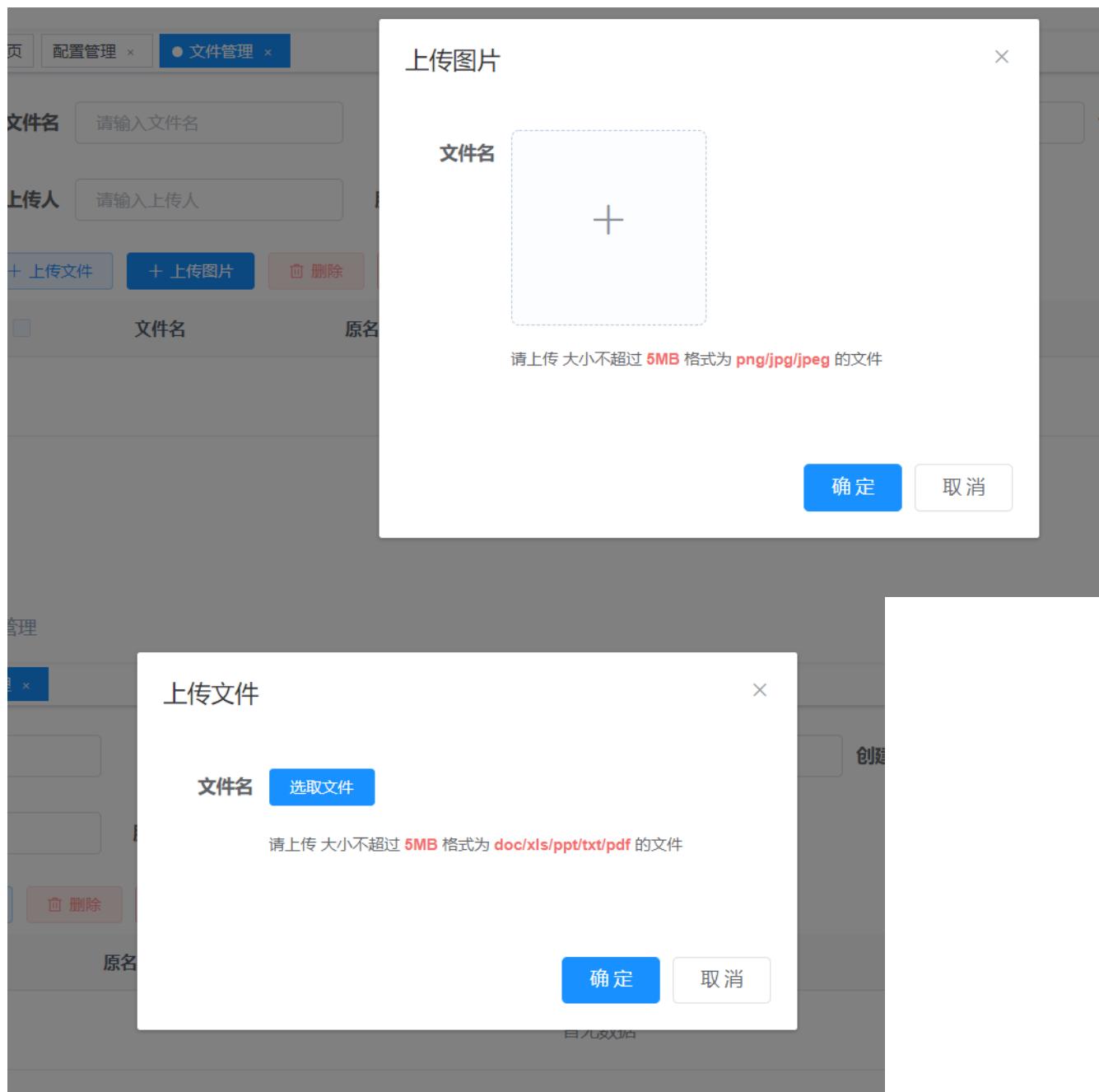
```
OssClient storage = OssFactory.instance(configKey: "image");
```

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



文件管理

上传图片

文件名



请上传 大小不超过 5MB 格式为 png/jpg/jpeg 的文件

确定 取消

上传文件

文件名 选取文件

请上传 大小不超过 5MB 格式为 doc/xls/ppt/txt/pdf 的文件

8341a0ee84ec4480aa267a5fe9dff264.txt 删除

确定 取消

### 列表展示

默认展示图片(可预览) 文件会展示路径

## 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  开始日期 结束日期

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

The screenshot shows a file list interface for the OSS module. At the top, there are search and configuration management buttons. A red box highlights the '预览开关: 禁用' (Preview Switch: Disabled) button. Below it, the file list includes columns for file name, original name, file suffix, preview, and creation time. The first item is a file named 'test.jpg' which has a preview thumbnail displayed.

文件名	原名	文件后缀	文件展示	创建时间
20210817/38a0fff1c5dc4 4e686fca57b1ba02428.j pg	test.jpg	.jpg		2021-08-17
20210817/613380f4496 8462cb9ba76d994b9365 8.txt	test.txt	.txt	<a href="http://qwoqr5oqq.hb-bkt.clouddn.com/20210817/613380f44968462cb9ba76d994b93658.txt">http://qwoqr5oqq.hb-bkt.clouddn.com/20210817/613380f44968462cb9ba76d994b93658.txt</a>	2021-08-17
20210817/8341a0ee84e c4480aa267a5fe9dff264. txt	test.txt	.txt	<a href="http://qwoqr5oqq.hb-bkt.clouddn.com/20210817/8341a0ee84ec4480aa267a5fe9dff264.txt">http://qwoqr5oqq.hb-bkt.clouddn.com/20210817/8341a0ee84ec4480aa267a5fe9dff264.txt</a>	2021-08-17

点击禁用后 图片会变成路径展示

The screenshot shows the same file list interface, but the '预览开关: 启用' (Preview Switch: Enabled) button is highlighted with a red box. The preview thumbnails for the files have disappeared, and instead, the full URL paths are displayed in the '文件展示' (File Preview) column.

文件名	原名	文件后缀	文件展示	创建时间
20210817/38a0fff1c5dc4 4e686fca57b1ba02428.j pg	test.jpg	.jpg	<a href="http://qwoqr5oqq.hb-bkt.clouddn.com/20210817/38a0fff1c5dc44e686fca57b1ba02428.jpg">http://qwoqr5oqq.hb-bkt.clouddn.com/20210817/38a0fff1c5dc44e686fca57b1ba02428.jpg</a>	202
20210817/613380f4496 8462cb9ba76d994b9365 8.txt	test.txt	.txt	<a href="http://qwoqr5oqq.hb-bkt.clouddn.com/20210817/613380f44968462cb9ba76d994b93658.txt">http://qwoqr5oqq.hb-bkt.clouddn.com/20210817/613380f44968462cb9ba76d994b93658.txt</a>	202
20210817/8341a0ee84e c4480aa267a5fe9dff264. txt	test.txt	.txt	<a href="http://qwoqr5oqq.hb-bkt.clouddn.com/20210817/8341a0ee84ec4480aa267a5fe9dff264.txt">http://qwoqr5oqq.hb-bkt.clouddn.com/20210817/8341a0ee84ec4480aa267a5fe9dff264.txt</a>	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

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

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> <a href="#">删除</a>
20210817/613380f44968462cb9ba76d994b93658.txt	test.txt	.txt	<a href="http://qwoqr5oqq.hbbkt.clouddn.com/20210817/613380f44968462cb9ba76d994b93658.txt">http://qwoqr5oqq.hbbkt.clouddn.com/20210817/613380f44968462cb9ba76d994b93658.txt</a>	2021-08-17	admin	qiniu	<a href="#">下载</a> <a href="#">删除</a>
20210817/47151c4e4bfe45ed88e482270d2e187d.txt	test.txt	.txt	<a href="http://ruoyi-vue-plus.oss-cn-beijing.aliyuncs.com/20210817/47151c4e4bfe45ed88e482270d2e187d.txt">http://ruoyi-vue-plus.oss-cn-beijing.aliyuncs.com/20210817/47151c4e4bfe45ed88e482270d2e187d.txt</a>	2021-08-17	admin	aliyun	<a href="#">下载</a> <a href="#">删除</a>

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

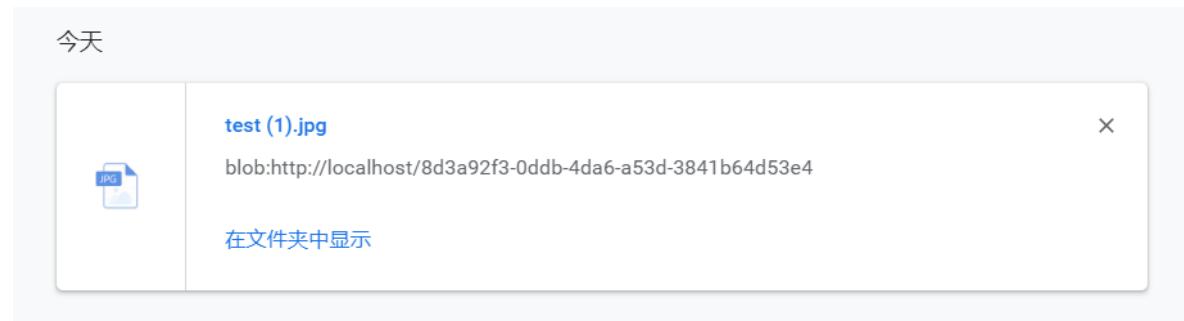
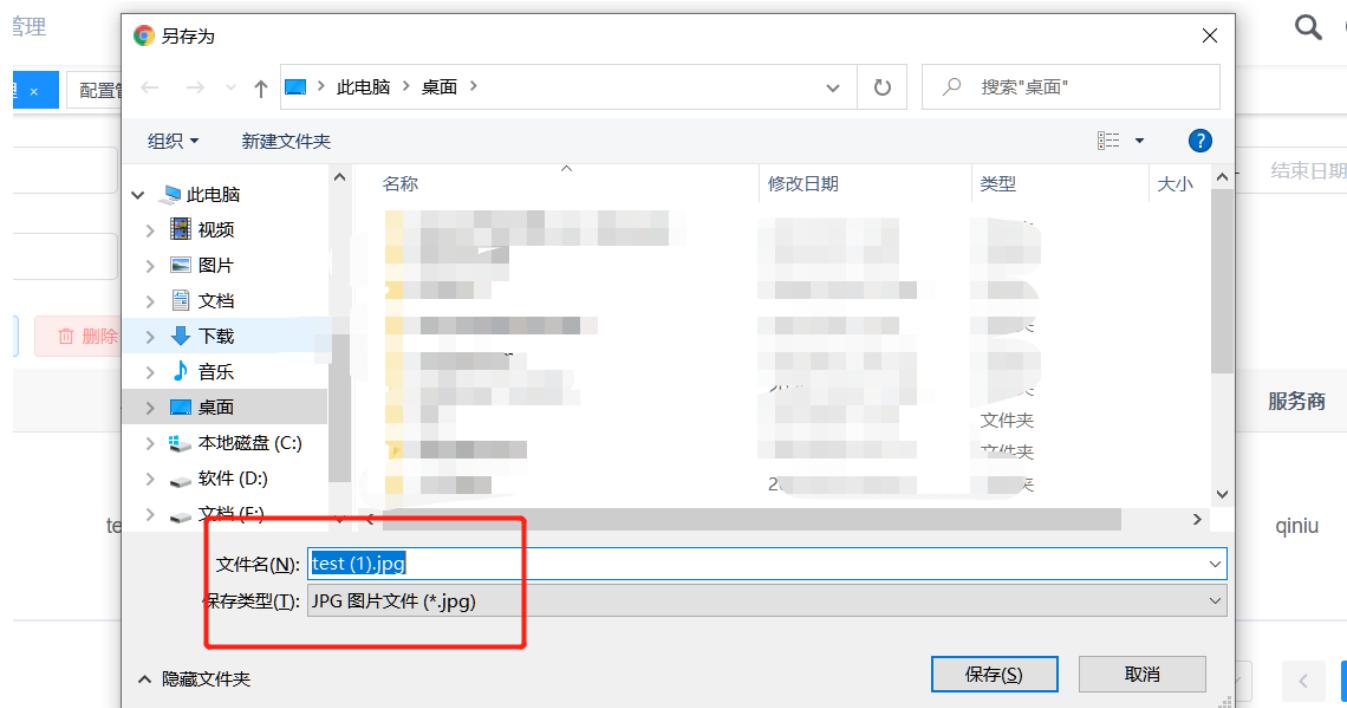
文件名	原名	文件后缀	文件展示	创建时间	上传人
20210817/38a0fff1c5 dc44e686fca57b1ba0 2428.jpg	test.jpg	.jpg		2021-08-17	admin

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

## 关于OSS模块使用

The screenshot shows a web application interface for managing data. At the top, there are tabs for '岗位管理' and '测试单表'. A prominent red banner at the top right says '操作失败' (Operation Failed). Below the banner, there are search and filter fields: '键' (Key) with placeholder '请输入key键', '值' (Value) with placeholder '请输入值', and '创建时间' (Created Time) with dropdowns for '开始日期' (Start Date) and '结束日期' (End Date). There are also two blue '搜索' (Search) buttons. Below these are several buttons: '新增' (Add), '修改' (Modify), '删除' (Delete), and '导出' (Export). The main content area is a table with columns: '部门id' (Dept ID), '用户id' (User ID), '排序号' (Sort Order), 'key键' (Key Key), '值' (Value), and '创建时间' (Created Time). The table contains six rows of data. The fourth row, which has a red border around its entire row, is highlighted. It has values: 108, 4, 4, '测试数据', 'demo', and '2021-06-01'. The log output below the table shows SQL queries and their execution times:

```
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(query);  
}
```

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
```

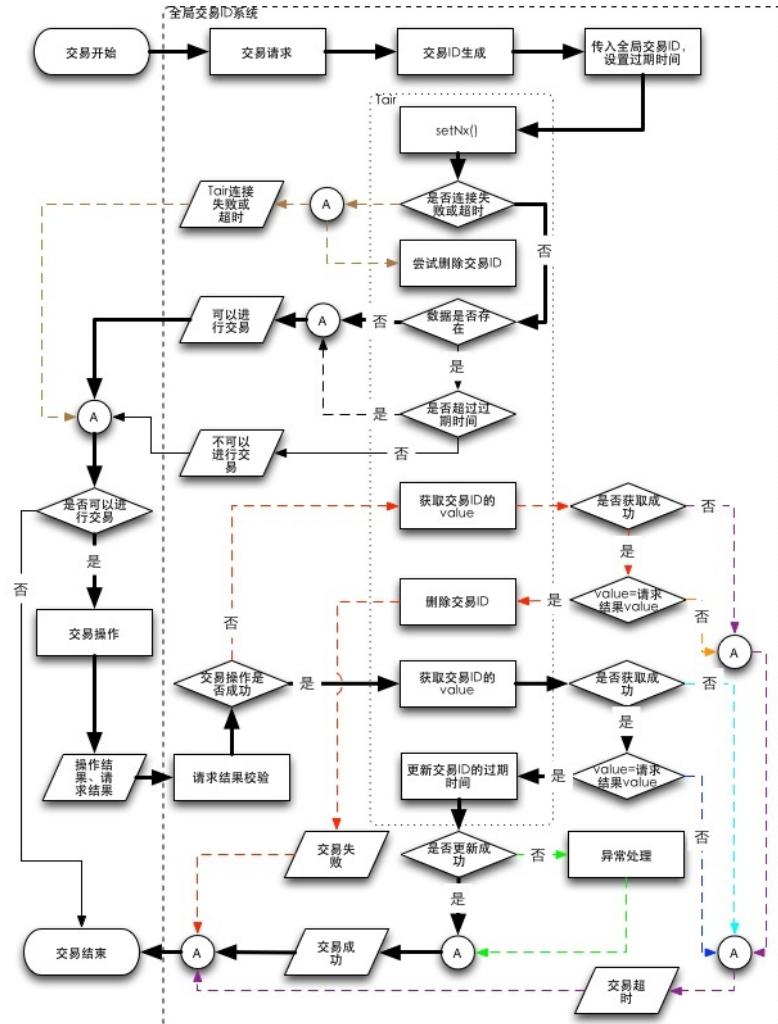
防重幂等

# 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();  
}
```

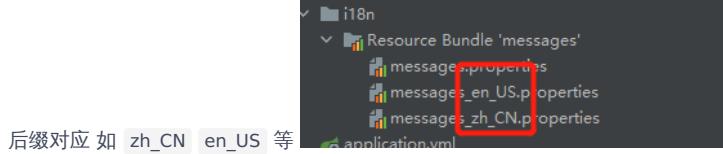
系统默认处理逻辑为 非管理员脱敏 可自行修改默认实现

```
/*  
 * 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: }
    
```

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 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:** Contains a title bar "标题: RuoYi-Vue-Plus后台管理系统\_接口文档" and a sub-navigation bar with tabs: 主页, Validator 校验国际化, and 通过code获取国际化内容.
- API Testing Section:** Shows a request configuration for a GET request to "/dev-api/demo/i18n". The "请求参数" tab is selected, showing a parameter named "code" with value "user.register.success". The "响应内容" tab shows the JSON response: 

```

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

```
- Second API Testing Section:** Shows a request configuration for a GET request to "/dev-api/demo/i18n". The "请求头部" tab is selected, showing headers: Authorization (Bearer eyJhbGciOiJIUzUxMiJ9.eyJsb2dpbiI91c2VyX...) and content-language (en\_US). The "响应内容" tab shows the JSON response: 

```

1 | {
2 |   "code": 200,
3 |   "msg": "Register successful",
4 |   "data": null
5 |

```

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, there is a sidebar with various system modules like System Management, System Monitoring, System Tools, Form Construction, Code Generation, and System Interfaces. The 'System Interfaces' module is currently selected. In the main content area, there is a navigation bar with tabs for '首页' (Home), '系统接口' (System Interface), and a dropdown menu for '演示案例' (Demo Examples). Below this, a search bar is present. The main content area displays the title '标题: RuoYi-Vue-Plus后台管理系统\_接口文档' (Title: RuoYi-Vue-Plus Backend Management System\_Interface Documentation). Under this title, there is a section for 'Validator 校验国际化' (Validator Validation Internationalization). It shows a 'GET /dev-api/demo/i18n/test1' endpoint. The 'Headers' tab is selected, showing a 'content-language' header set to 'en\_US'. The 'Raw' tab shows the response content:

```

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 API documentation interface. On the left, there is a navigation sidebar with various system modules like '系统管理', '系统监控', '系统工具', '表单构建', '代码生成', and '系统接口'. Under '系统接口', there are several sub-options including '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', '测试国际化管理', and 'Bean 校验国际化'. The 'Bean 校验国际化' option is highlighted.

The main content area displays the API documentation for the 'Bean 校验国际化' endpoint. It includes the title '标题: RuoYi-Vue-Plus后台管理系统\_接口文档', the URL '/dev-api/demo/i18n/test2', and a 'GET' method section. The '请求头' (Request Headers) section is expanded, showing two headers: 'Authorization' (with value 'Bearer eyJhbGciOiJIUzUxMiJ9eyJsb2d...' and checked) and 'content-language' (with value 'zh\_CN' and checked). A red box highlights these two headers. Below the headers, the '响应内容' (Response Content) section shows 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

### 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. The 'Bean 校验国际化' endpoint is selected. The request details are as follows:

**GET /dev-api/demo/i18n/test2**

**请求头部**

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

**响应内容**

```

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

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

```

## 接口放行

### 使用方式

```
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
```

nacos 中 ruoyi-gateway.yml 白名单放行

## 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/100000000
  # 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

<input type="checkbox"/>	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

数据源 master 表名称 请输入表名称 表描述 请输入表描述

Q 搜索 C 重置

生成 导入 修改 删除

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

数据源 master 表名称 请输入表名称 表描述 请输入表描述 创建时间 开始日期 - 结束日期

Q 搜索 C 重置

生成 导入 修改 删除

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="radio"/> 编辑 <input type="radio"/> 删除 <input type="radio"/> 同步 <input type="radio"/> 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="radio"/> 编辑 <input type="radio"/> 删除 <input type="radio"/> 同步 <input type="radio"/> 生成代码

共 2 条 10条/页 1 前

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)**

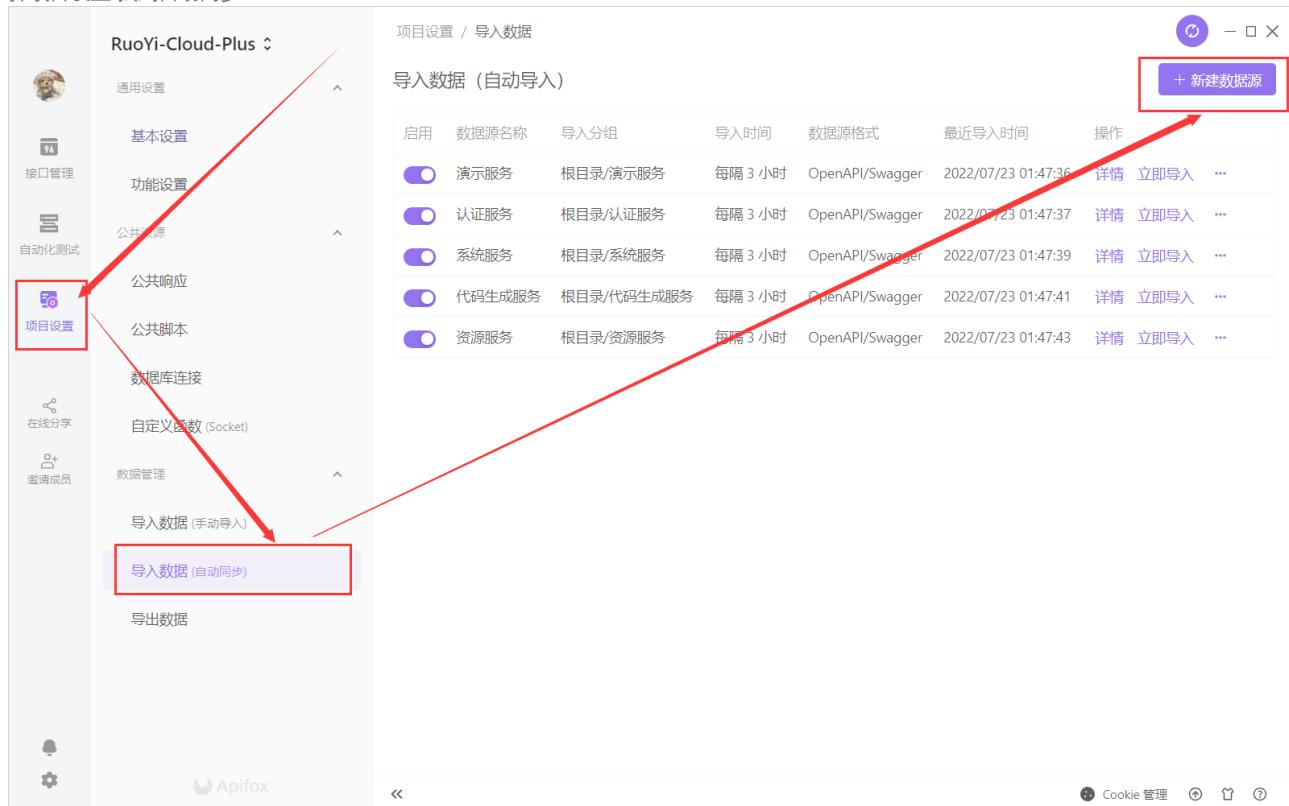
* 项目名称	RuoYi-Cloud-Plus	项目图标
成员权限		
成员	权限	
疯狂的狮子Li (@疯狂的狮子Li)	管理员	▼

\* 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**

**导入数据 (手动导入)**

**导出数据**

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)

**全局授权设置**

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

**Auth 配置**

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

**配置文件截图**

```

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, 系统管理, 系统监控, 系统工具 (which is expanded), 表单构建, 代码生成, 系统接口 (which is highlighted with a red arrow), PLUS官网, 测试菜单, and 测试菜单 (under 测试菜单). The main content area has a title bar with '首页 / 系统工具 / 系统接口'. Below this, there is a sub-menu for '1. 演示案例' containing items like '主页', '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', '测试国际化管理', and '测试批量方法'. To the right, there is a detailed configuration panel for the '1. 演示案例' group. It includes fields for '标题' (Title), '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
```

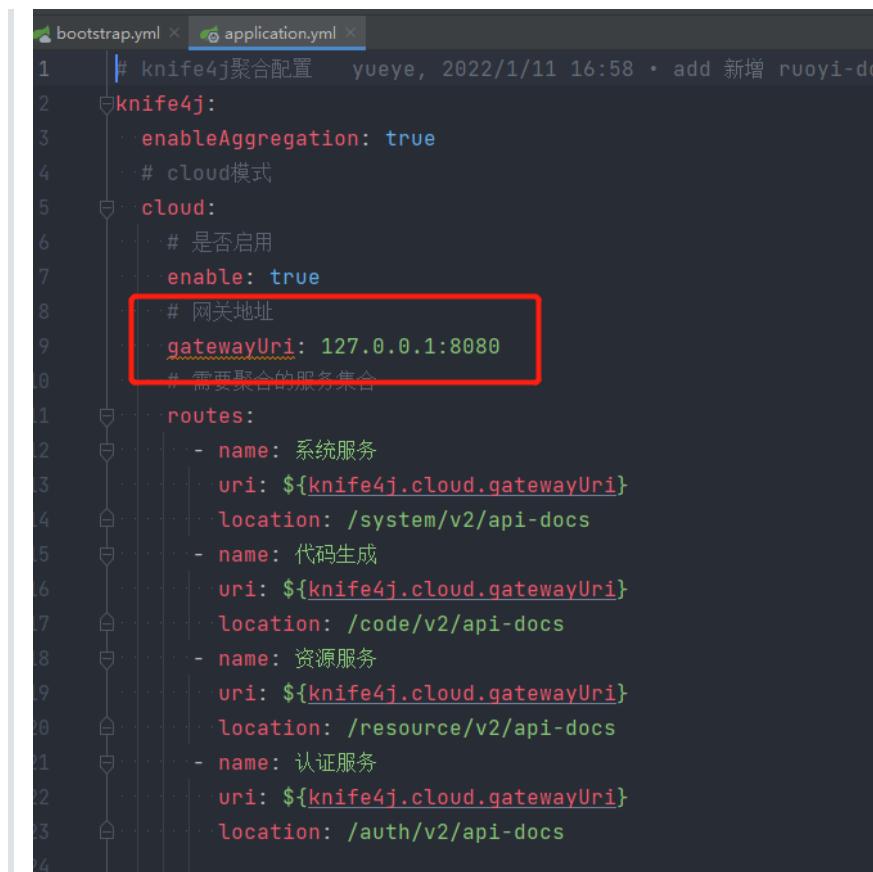
swagger 前缀配置为基础配置 knife4j 前缀配置为增强配置 增强配置可以查看 [knife4j文档](#) 框架文档

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
  # 为false时如果同时登录失败则会清空所有token
  allow-concurrent-login: false
```

## 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



The screenshot shows the project structure of the `ruoyi-common-security` module. A red box highlights the `com.ruoyi.common.security` package, which contains the `SecurityConfiguration` class. In the code editor, a red arrow points to the `.addExclude(...paths: "/actuator/**")` method call, indicating where to add the exclude path for the actuator endpoint.

```
28 }
29 /**
30  * 校验是否从网关转发
31 */
32 /**
33 */
34 @Bean
35 public SaServletFilter getSaServletFilter() {
36     return new SaServletFilter()
37         .addInclude(...paths: "/**")
38         .addExclude(...paths: "/actuator/**")
39         .setAuth(obj -> SaIdUtil.checkCurrentRequestToken())
40         .setError(e -> SaResult.error( msg: "认证失败, 无法访问系统资源").setCode(401))
41 }
```

## 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

## 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。

## 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

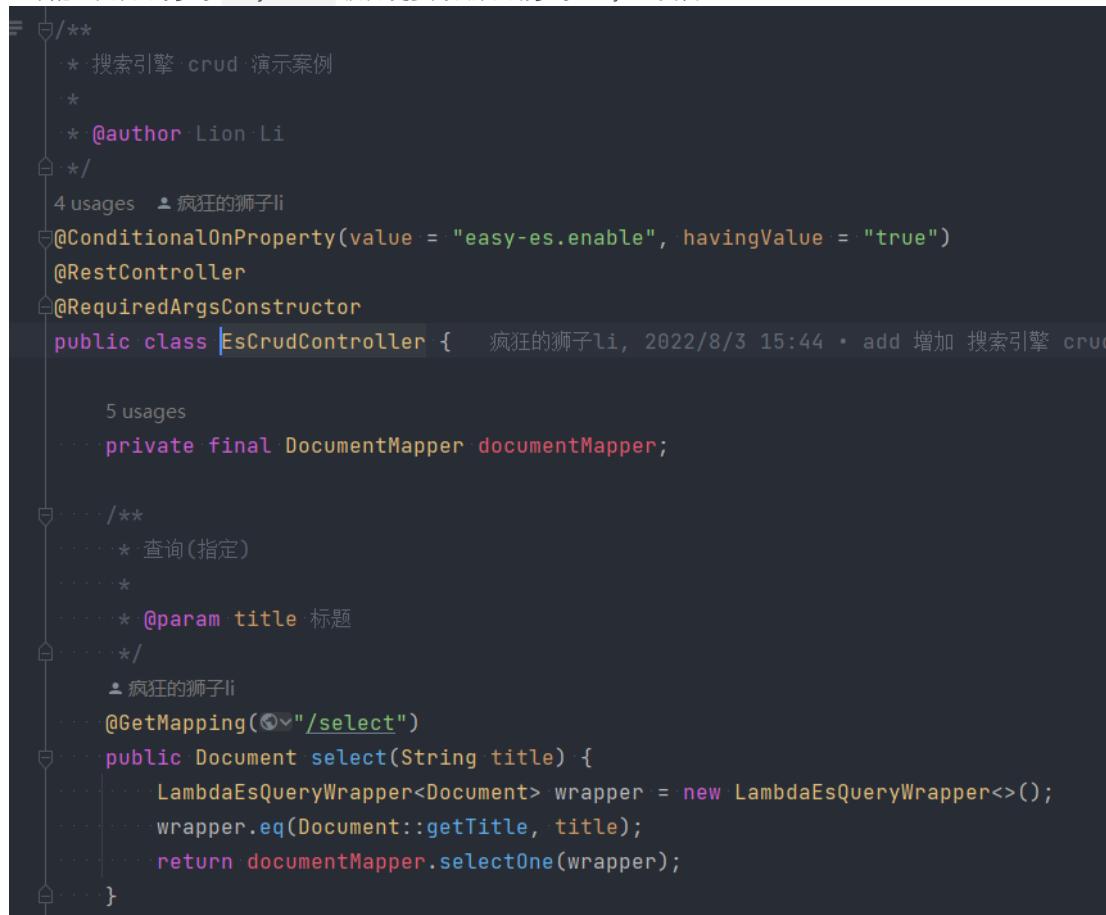
```
docker-compose up -d elasticsearch
```

## 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 for an Elasticsearch CRUD controller. The code includes annotations like `@ConditionalOnProperty`, `@RestController`, and `@RequiredArgsConstructor`. It defines a class `EsCrudController` with methods for crud operations. 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);  
    }  
}
```

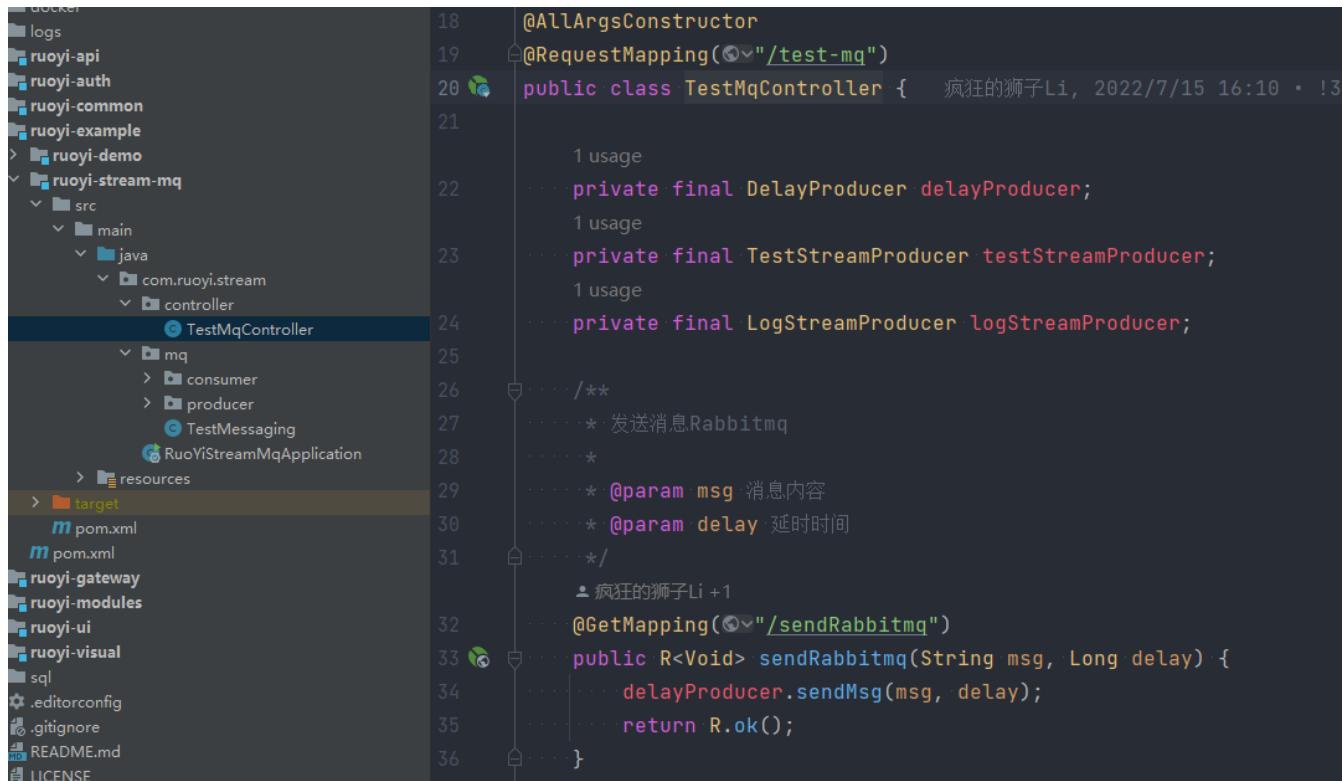
## RabbitMQ搭建

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

```

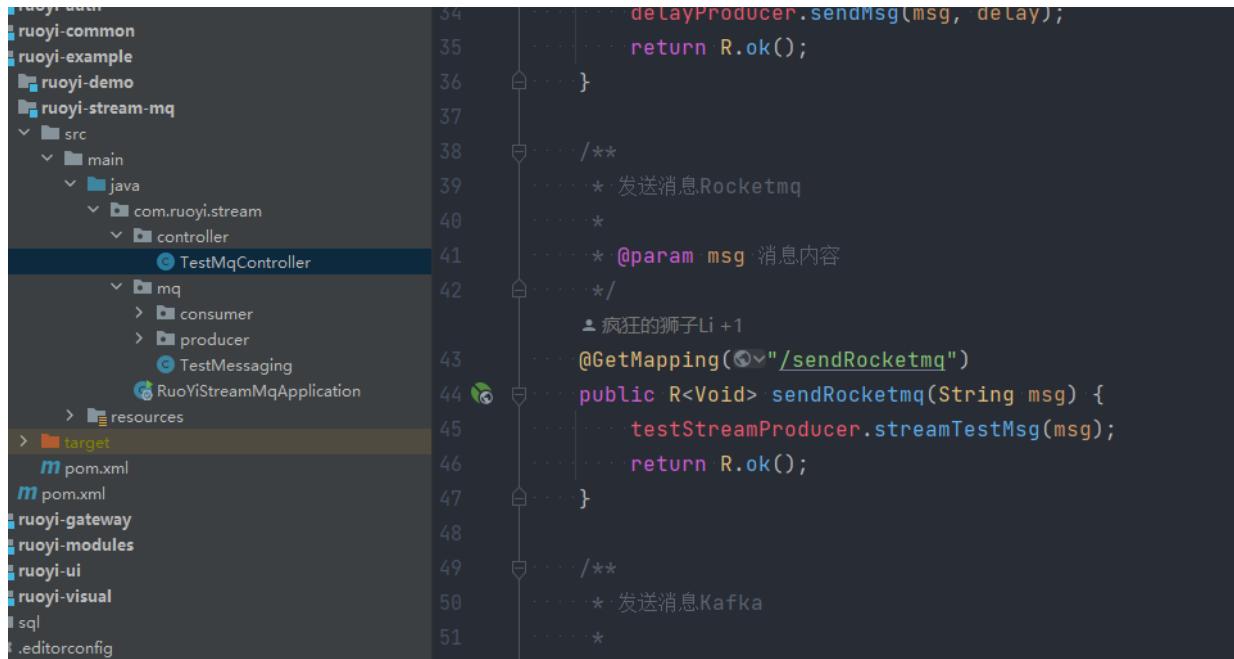
## RocketMQ搭建

## 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



```
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

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 @GetMapping("/sendRocketmq")
44 public R<Void> sendRocketmq(String msg) {
45   testStreamProducer.streamTestMsg(msg);
46   return R.ok();
47 }
48 /**
49  * 发送消息Kafka
50  */
51 *
```

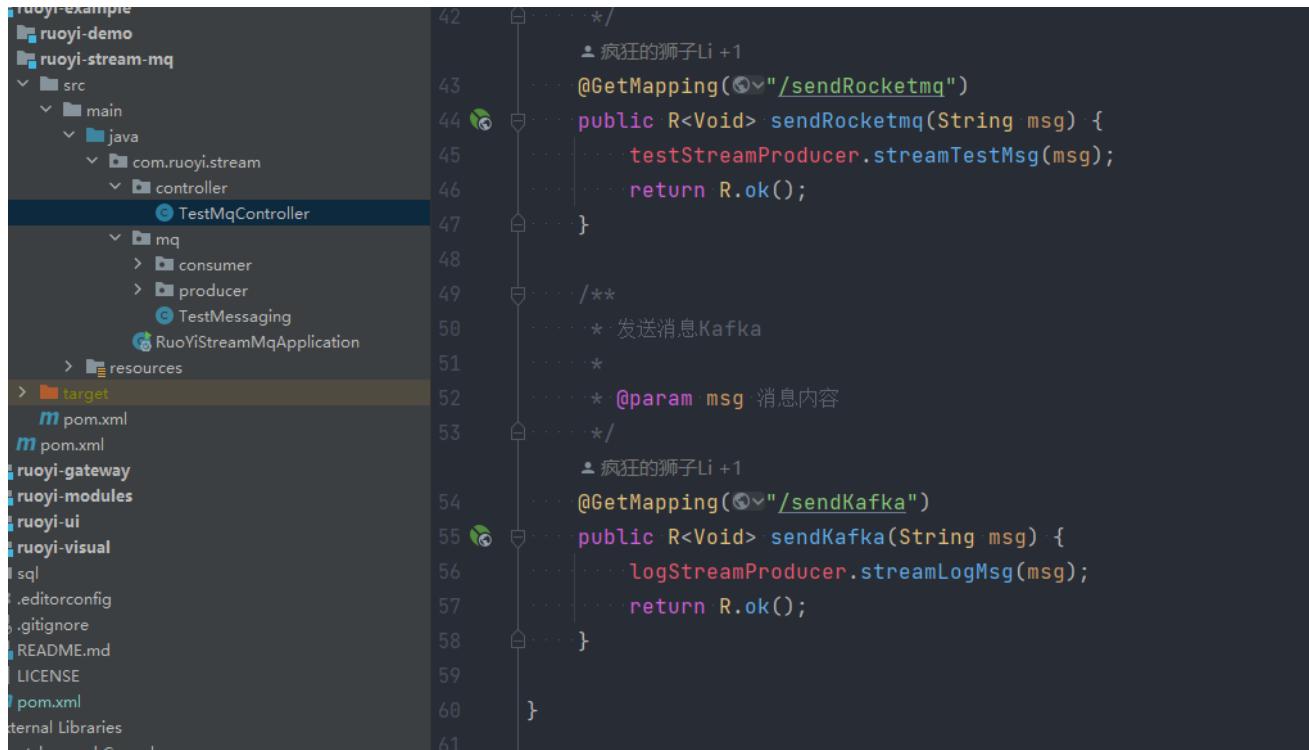
## Kafka搭建

## 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.

**File Tree:**

- 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

**Code Snippet (TestMqController.java):**

```
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 }
```

## 常见问题

### Lombok注解爆红

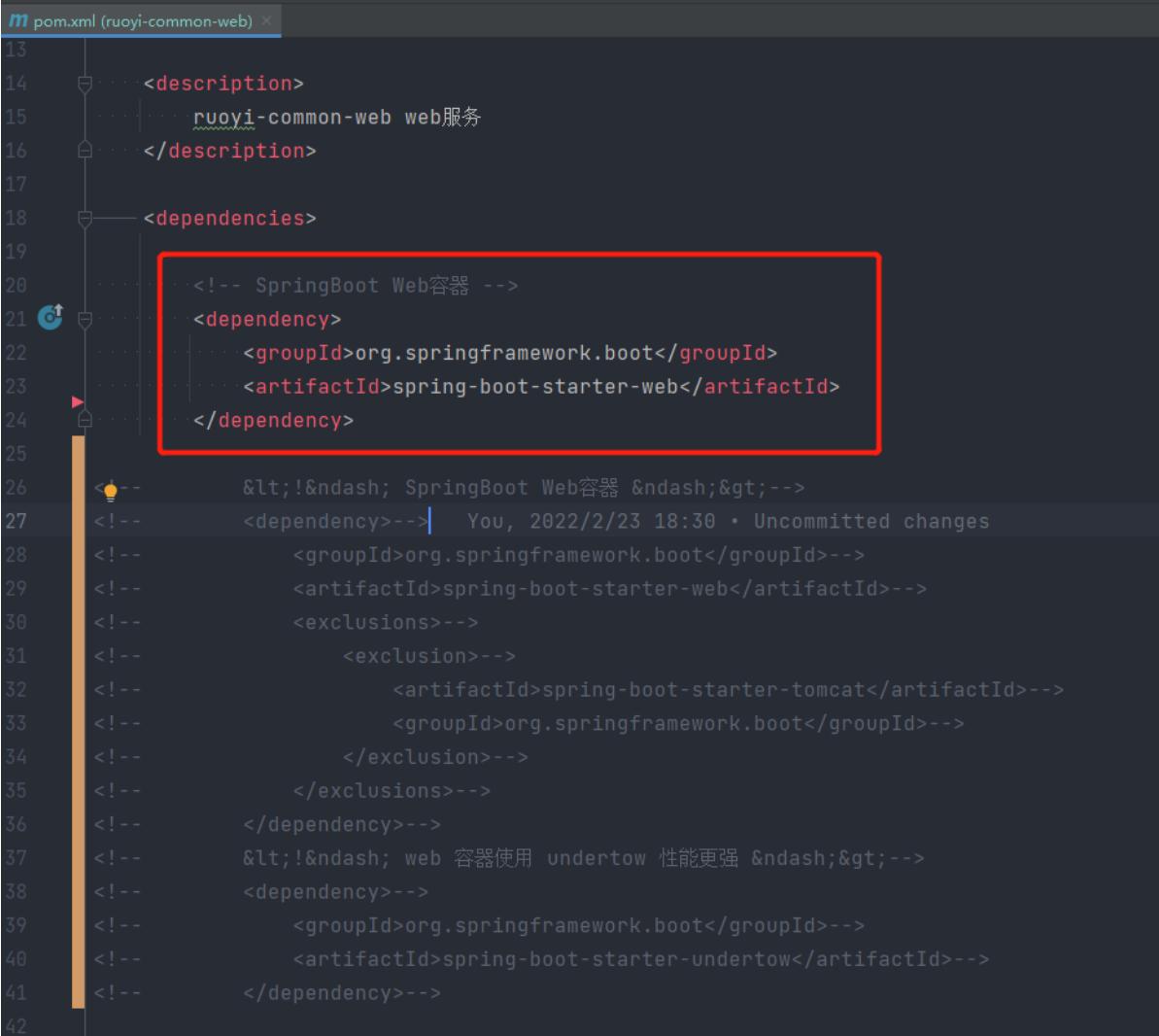
关于lombok注解爆红

- 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

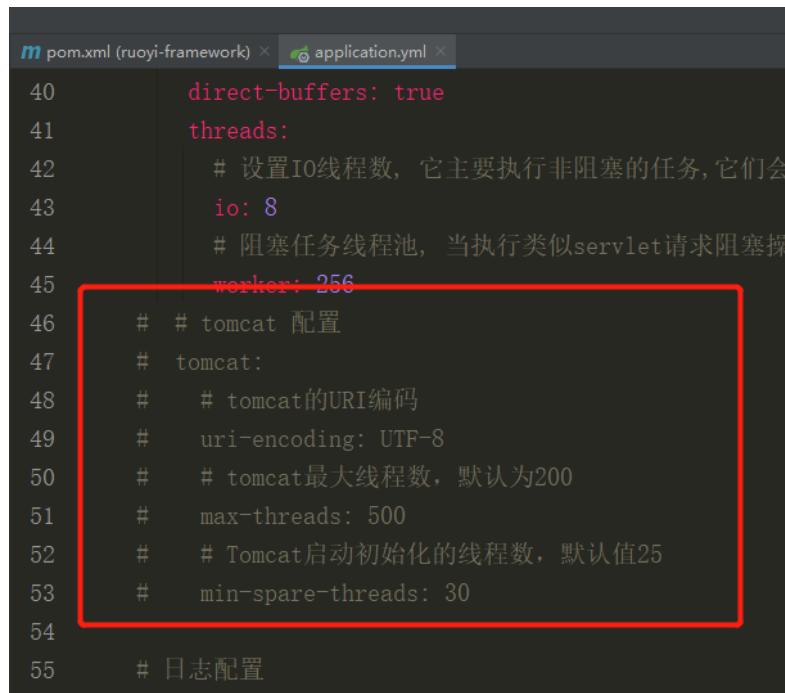
### 关于如何使用Tomcat

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         <!--
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         <!--
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
```

## 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

## 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, 2
        m.menu_name, 3
        m.path, 4
        m.component, 5
        m.query, 6
        m.visible,
        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);
}
```

## swagger相关问题

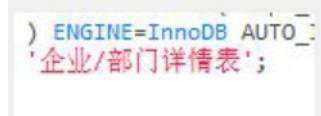
### 空指针问题

```
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

## 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

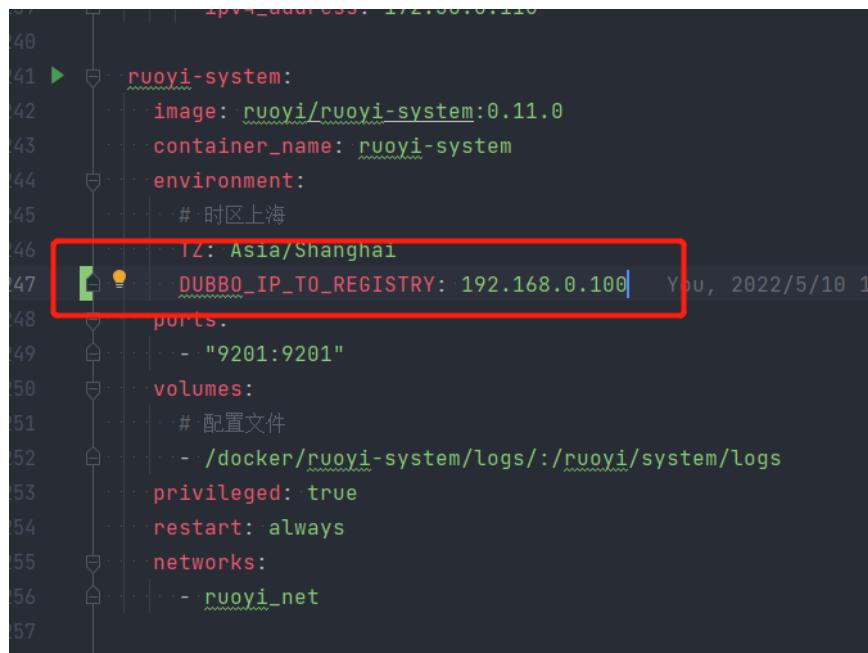
### 重点说明

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
```

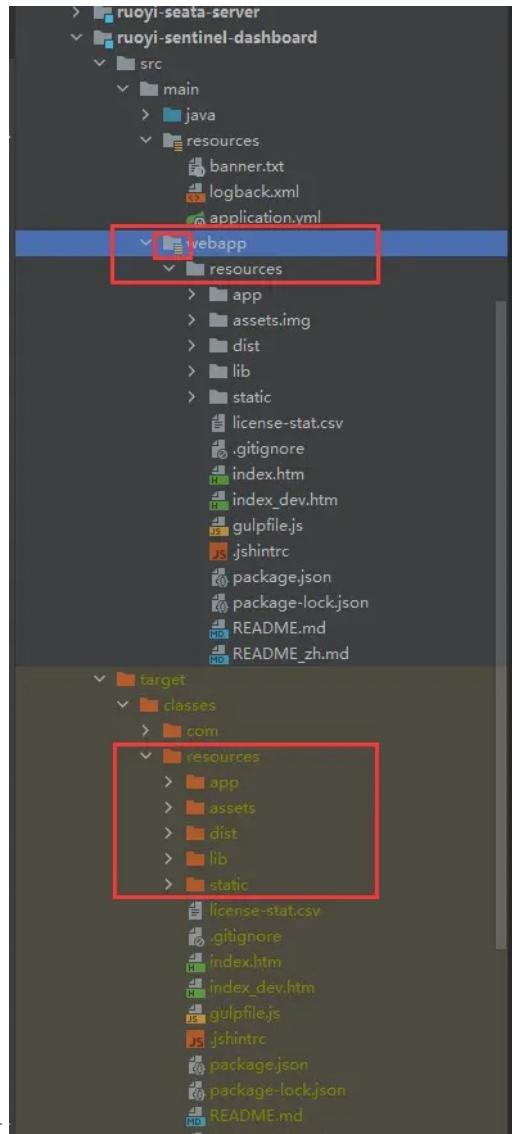
原因

## 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

多方用户提供修复意见

使用原生网络请求会降低问题出现频率

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

## 多方用户提供修复意见

使用原生网络请求会降低问题出现频率

## 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>

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>

## 加群与捐献

### 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

| 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

群号 : <637757165>

### 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



### 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>	

