# 官网

http://ruoyi.vip

# 官方文档

http://doc.ruoyi.vip/ruoyi-cloud/

# 优点

1、快速开发：Ruoyi 框架提供了一系列常用的功能模块，并且内置了代码生成器和表单设计器等工具，可以快速地开发出各种管理系统。

2、易于集成：Ruoyi 框架采用了 Spring Boot 作为基础架构，无需配置繁琐的 XML 文件，同时还提供了 API 接口和插件机制，方便二次开发和扩展。

3、前后端分离：Ruoyi 框架采用前后端分离的设计思想，前端采用 Vue.js 技术栈，可以更好地实现前后端解耦，提高整个系统的灵活性和可维护性。

4、安全性高：Ruoyi 框架内置了安全框架，包括登录认证、权限控制等功能，有效保障了系统的安全性。

# 缺点

1、学习成本较高：Ruoyi 框架涉及到的技术比较广泛，需要对 Spring Boot、MyBatis、Vue.js 等技术有一定的掌握，对初学者来说学习成本较高。

2、不够灵活：虽然 Ruoyi 框架内置了各种常用的功能模块，但是在一些特殊的场景下，可能需要进行定制化开发或者二次开发，这时候就需要了解框架的底层原理和代码结构。

3、依赖版本升级困难：Ruoyi 框架包含了很多技术组件，并且实现了很多集成功能，因此在依赖版本升级时可能会有一定的困难。

# 演示地址

https://vue.ruoyi.vip/index

用户名：admin

密码：admin123

# 项目地址

## 微服务版本

https://gitee.com/y\_project/RuoYi-Cloud.git

# 常用工具

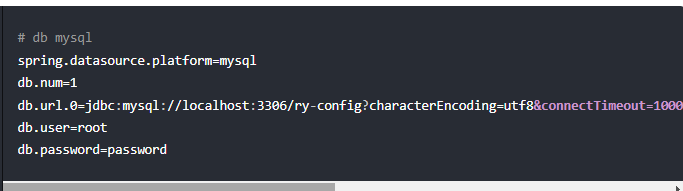
https://gitee.com/lpf\_project/common-tools

# 环境部署

## 运行系统

### 后端运行

1、前往Gitee下载页面([https://gitee.com/y\_project/RuoYi-Cloud (opens new window)](https://gitee.com/y_project/RuoYi-Cloud))下载解压到工作目录。  
2、导入到Eclipse，菜单 File -> Import，然后选择 Maven -> Existing Maven Projects，点击 Next> 按钮，选择工作目录，然后点击 Finish 按钮，即可成功导入。  
Eclipse会自动加载Maven依赖包，初次加载会比较慢（根据自身网络情况而定）  
3、创建数据库ry-cloud并导入数据脚本ry\_2021xxxx.sql（必须），quartz.sql（可选）  
4、创建数据库ry-config并导入数据脚本ry\_config\_2021xxxx.sql（必须）  
5、配置nacos持久化，修改conf/application.properties文件，增加支持mysql数据源配置。



提示

配置文件application.properties是在下载的nacos-server包conf目录下。  
最新RuoYi-Cloud版本>=3.0.0需要下载的nacos-server必须>=2.x.x版本。  
默认配置单机模式，nacos集群/多集群部署模式参考官网。

6、打开运行基础模块（启动没有先后顺序）

RuoYiGatewayApplication （网关模块 必须）

RuoYiAuthApplication （认证模块 必须）

RuoYiSystemApplication （系统模块 必须）

RuoYiMonitorApplication （监控中心 可选）

RuoYiGenApplication （代码生成 可选）

RuoYiJobApplication （定时任务 可选）

RuoYFileApplication （文件服务 可选）

7、集成seata分布式事务（可选配置，默认不启用）

创建数据库ry-seata并导入数据脚本ry\_seata\_2021xxxx.sql

提示

运行前需要先启动nacos，运行成功可以通过([http://localhost:8080 (opens new window)](http://localhost:8080/))访问，但是不会出现静态页面，可以继续参考下面步骤部署ruoyi-ui前端，然后通过前端地址来访问。

### 前端运行

#本地开发，启动项目

npm run dev

## 部署系统

### 说明

因为本项目是前后端分离的，所以需要前后端都部署好才能进行访问。

### 后端部署

1、打包工程文件

在ruoyi项目的bin目录下执行package.bat打包Web工程，生成war/jar包文件。然后会在项目下生成target文件夹包含war或jar。

提示：不同模块版本会生成在ruoyi/ruoyi-xxxx模块下target文件夹。

2、部署工程文件

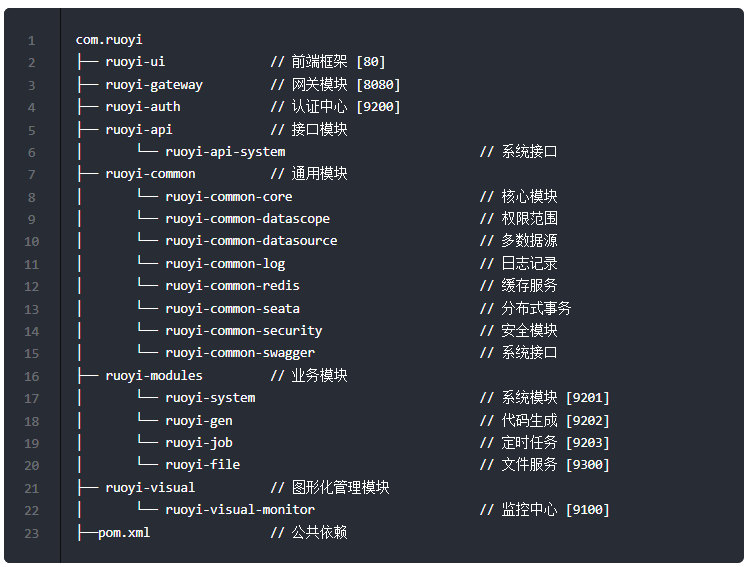
jar部署方式

使用命令行执行：java –jar ruoyi-xxxx.jar 或者执行脚本：ruoyi/bin/run-xxxx.bat

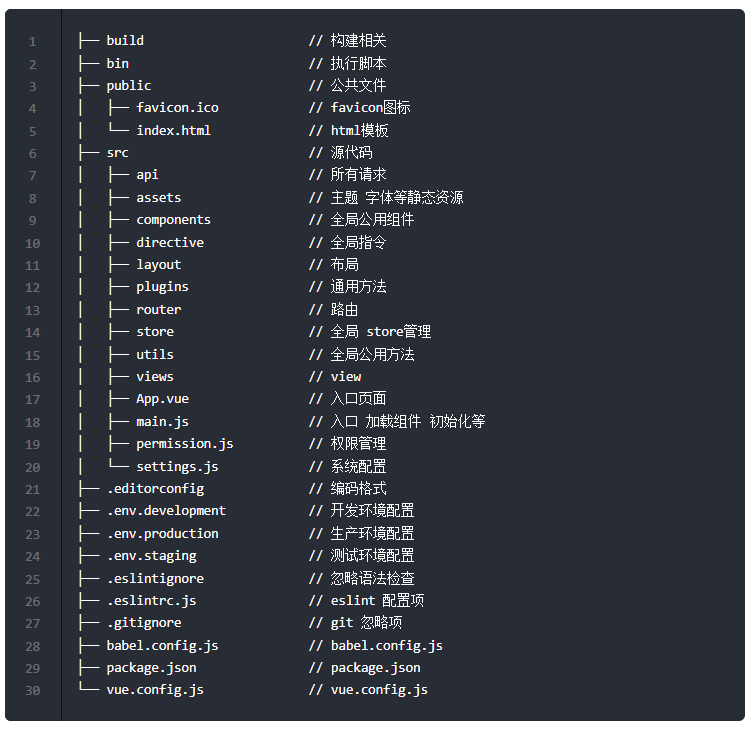
### 前端部署

# 项目介绍

## 后端结构

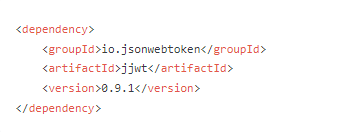


## 前端结构



# jjwt使用教程

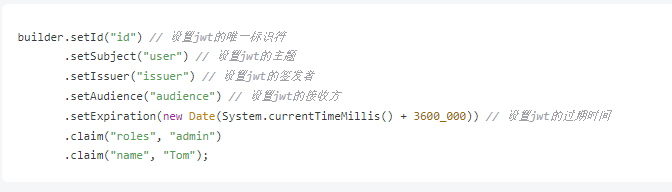
## 1、我们需要在pom.xml文件中引入jjwt依赖



## 2、需要先实例化一个JwtBuilder对象

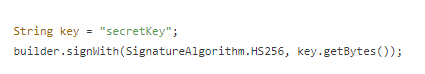


## 3、设置jwt的各种参数



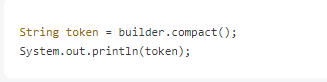
我们使用setId()、setSubject()、setIssuer()、setAudience()、setExpiration()方法设置了jwt的基本信息和过期时间；同时，使用claim()方法添加自定义的claim信息。其中，过期时间应该是一个Date类型的时间戳，表示从当前时间开始多少毫秒之后jwt过期。

## 4、使用signWith()方法设置jwt的签名密钥

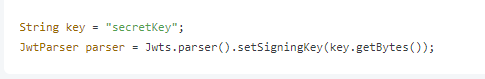


我们使用signWith()方法设置了jwt的签名算法为HS256，并指定了密钥key。

## 5、使用compact()方法生成jwt字符串



## 6、验证jwt，我们可以使用Jwts.parser()方法获取JwtParser对象，并使用setSigningKey()方法设置签名密钥



## 7、使用parseClaimsJws()方法解析jwt字符串，并获取其载荷信息



我们使用parseClaimsJws()方法解析jwt字符串，并通过getBody()方法获取其载荷信息。其中，getId()、getSubject()、getIssuer()、getAudience()、getExpiration()方法用于获取jwt的基本信息；而get()方法用于获取自定义的claim信息。

# 微服务

## 服务网关

## 认证中心

### 基本概念

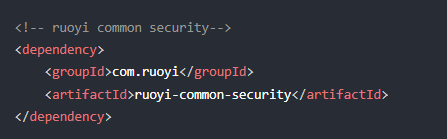
身份认证，就是判断一个用户是否为合法用户的处理过程。最常用的简单身份认证方式是系统通过核对用户输入的用户名和口令，看其是否与系统中存储的该用户的用户名和口令一致，来判断用户身份是否正确。

### 为什么要使用认证中心

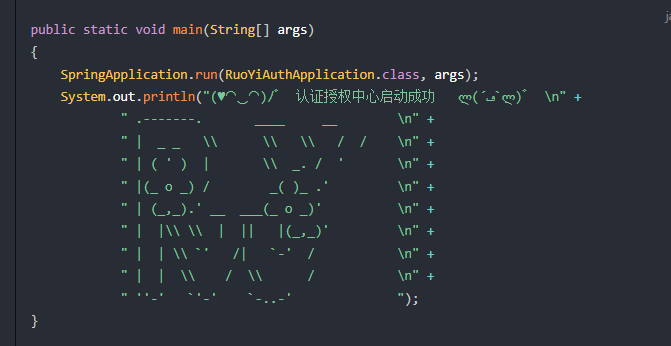
登录请求后台接口，为了安全认证，所有请求都携带token信息进行安全认证，比如使用vue、react后者h5开发的app，用于控制可访问系统的资源

### 使用方法

#### 1、添加依赖



#### 2、创建认证启动类



说明：目前已经存在ruoyi-auth认证授权中心，用于登录认证，系统退出，刷新令牌。

### 登录认证

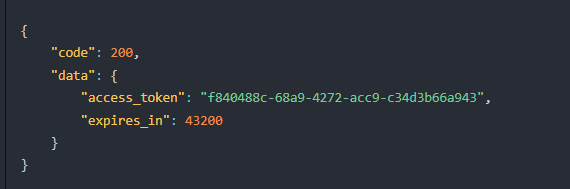
TokenController控制器login方法会进行用户验证，如果验证通过会保存登录日志并返回token，同时缓存中会存入login\_tokens:xxxxxx（包含用户、权限信息）

用户登录接口地址 http://localhost:9200/login

请求方法：post



返回结果



通过用户验证登录后获取access\_token，通过网关访问其他应用数据时必须携带此参数值。

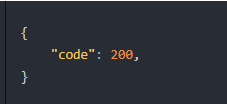
### 刷新令牌

就是对系统操作用户进行缓存刷新，防止过期，TokenController控制器refresh方法会在用户调用时更新令牌有效期。

刷新令牌接口地址 http://localhost:9200/refresh

请求头Authorization - f840488c-68a9-4272-acc9-c34d3b66a943，请求方式Post

返回结果：



### 系统退出

就是对系统登录用户的退出过程。TokenController控制器logout方法会在用户退出时删除缓存信息同时保存用户退出日志。

系统退出接口地址 http://localhost:9200/logout

请求头Authorization - f840488c-68a9-4272-acc9-c34d3b66a943，请求方式Delete



返回结果：



## 注册中心

## 配置中心

## 服务调用

## 服务监控

## 系统接口

## 链路追踪

# 后台手册（重点）

## 导入导出

## 上传下载

## 权限注解

### 基本说明

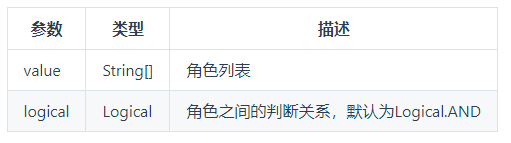
@RequiresLogin登录认证：只有登录之后才能进入该方法。

@RequiresPermissions权限认证：必须具有指定权限才能进入该方法。

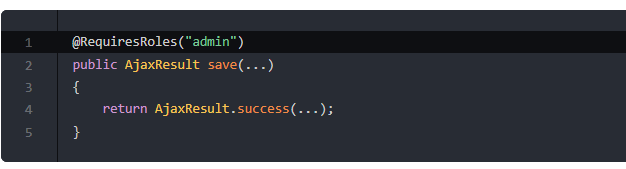
@RequiresRoles角色认证：必须具有指定角色标识才能进入该方法。

### @RequiresRoles

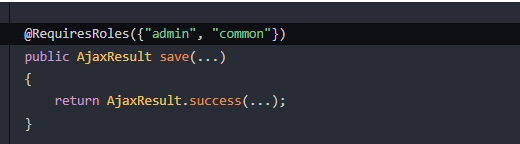
@RequiresRoles注解用于配置接口要求用户拥有某（些）角色才可访问，它拥有两个参数。



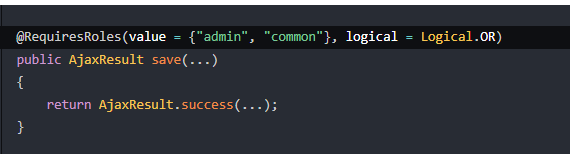
#### 1、以下代码表示必须拥有admin角色才可访问



#### 2、以下代码表示必须拥有admin和common角色才可访问



#### 3、以下代码表示需要拥有admin或common角色才可访问

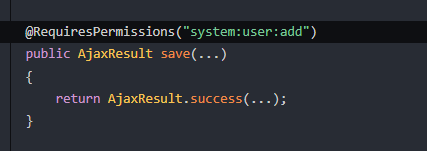


### @RequiresPermissions

@RequiresPermissions注解用于配置接口要求用户拥有某（些）权限才可访问，它拥有两个参数。



#### 1、以下代码表示必须拥有system:user:add权限才可访问



说明

认证注解处理是有内定的处理顺序的，如果有个多个注解的话，前面的通过了会继续检查后面的，若不通过则直接返回，处理顺序依次为（与实际声明顺序无关） RequiresLogin、RequiresRoles、RequiresPermissions。例如：你同时声明了RequiresRoles和RequiresPermissions，那就要求拥有此角色的同时还得拥有相应的权限。

## 事务管理

## 异常处理

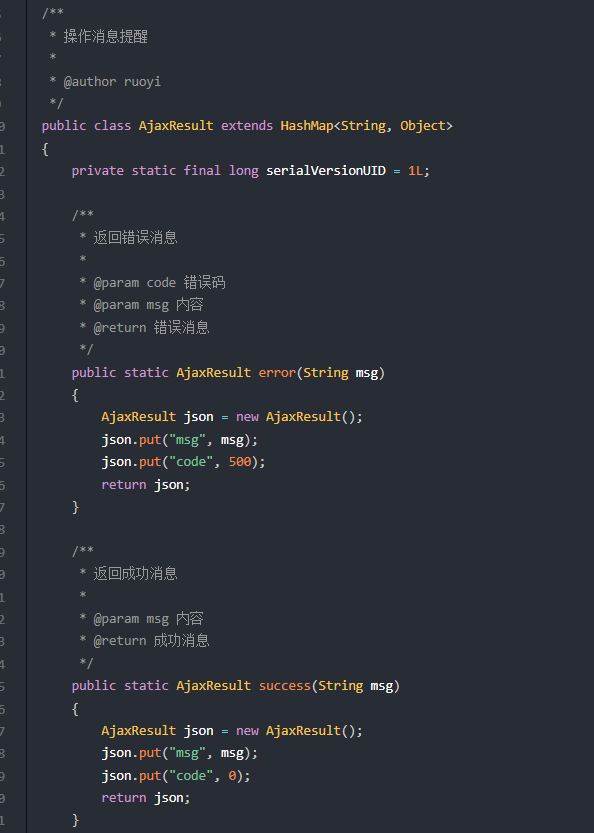
### 基本说明

通常一个web框架中，有大量需要处理的异常。比如业务异常，权限不足等等。前端通过弹出提示信息的方式告诉用户出了什么错误。 通常情况下我们用try.....catch....对异常进行捕捉处理，但是在实际项目中对业务模块进行异常捕捉，会造成代码重复和繁杂， 我们希望代码中只有业务相关的操作，所有的异常我们单独设立一个类来处理它。全局异常就是对框架所有异常进行统一管理。 我们在可能发生异常的方法里throw抛给控制器。然后由全局异常处理器对异常进行统一处理。 如此，我们的Controller中的方法就可以很简洁了。

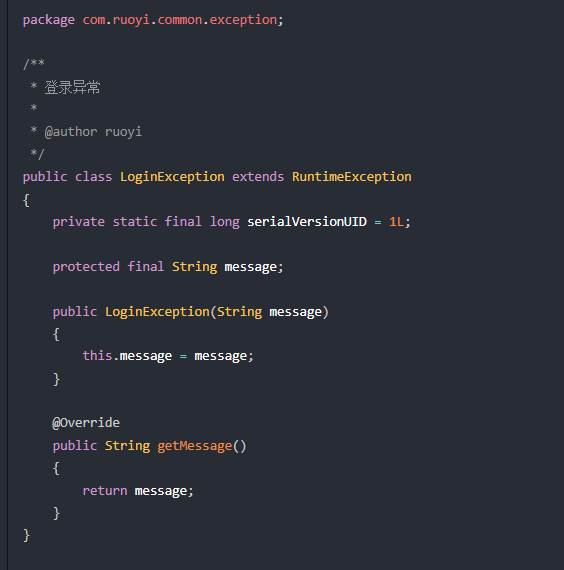
所谓全局异常处理器就是使用@ControllerAdvice注解。

### 示例

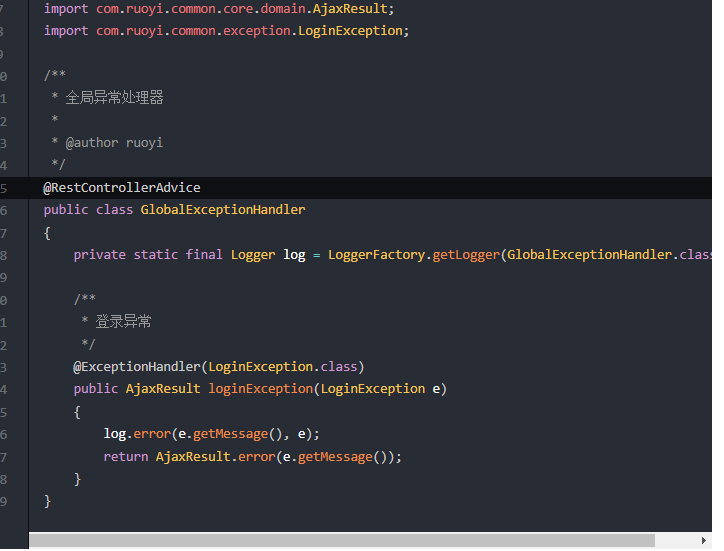
#### 1、统一返回实体定义



#### 2、定义登录异常定义



#### 3、基于@ControllerAdvice注解的Controller层的全局异常统一处理



#### 4、测试访问请求



根据上面代码含义，当我们未登录访问/index时就会发生LoginException业务逻辑异常，按照我们之前的全局异常配置以及统一返回实体实例化，访问后会出现AjaxResult格式JSON数据， 下面我们运行项目访问查看效果。  
界面输出内容如下所示：



对于一些特殊情况，如接口需要返回json，页面请求返回html可以使用如下方法：



### 若依系统的全局异常处理器

#### GlobalExceptionHandler

注意：如果全部异常处理返回json，那么可以使用@RestControllerAdvice代替@ControllerAdvice，这样在方法上就可以不需要添加@ResponseBody。

无法捕获异常？

如果您的异常无法捕获，您可以从以下几个方面着手检查，异常是否已被处理，即抛出异常后被catch，打印了日志或抛出了其它异常，异常是否非Controller抛出，即在拦截器或过滤器中出现的异常。

## 参数验证

### 基本说明

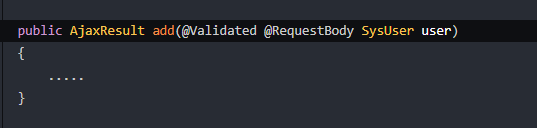
spring boot中可以用@Validated来校验数据，如果数据异常则会统一抛出异常，方便异常中心统一处理。

### 注解参数说明

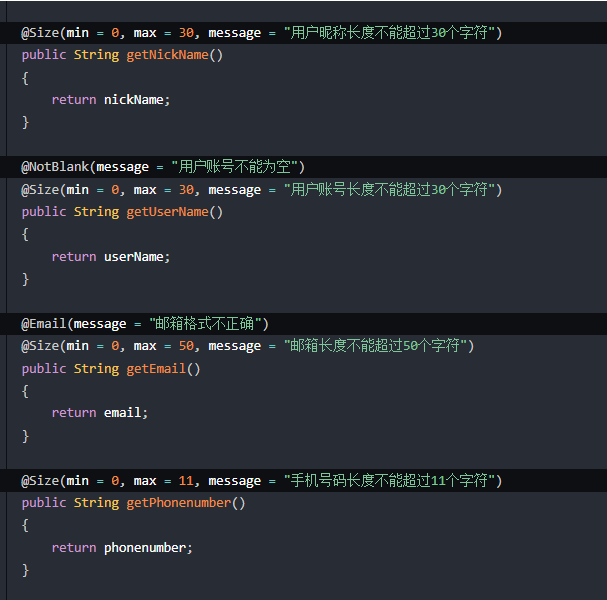


### 数据校验使用

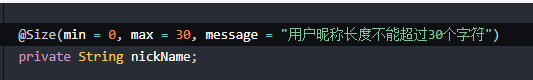
1、因为spring boot已经引入了基础包，所以直接使用就可以了。首先在controller上声明@Validated需要对数据进行校验。



2、然后在对应字段Get方法加上参数校验注解，如果不符合验证要求，则会以message的信息为准，返回给前端。



也可以直接放在字段上面声明。

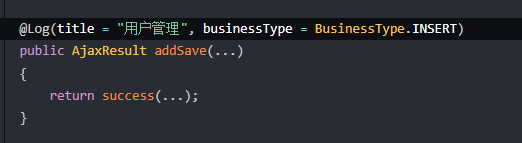


## 系统日志

### 基本说明

在实际开发中，对于某些关键业务，我们通常需要记录该操作的内容，一个操作调一次记录方法，每次还得去收集参数等等，会造成大量代码重复。 我们希望代码中只有业务相关的操作，在项目中使用注解来完成此项功能。

在需要被记录日志的controller方法上添加@Log注解，使用方法如下：



### @Log注解参数说明



### 自定义操作功能

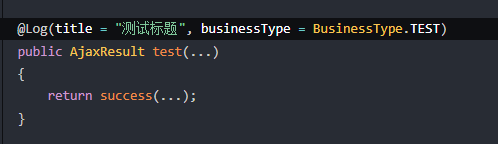
1、在BusinessType中新增业务操作类型如:



2、在sys\_dict\_data字典数据表中初始化操作业务类型。

insert into sys\_dict\_data values(25, 10, '测试', '10', 'sys\_oper\_type', '', 'primary', 'N', '0', 'admin', '2018-03-16 11-33-00', 'ry', '2018-03-16 11-33-00', '测试操作');

3、在Controller中使用注解。



操作日志记录逻辑实现代码[LogAspect.java](https://gitee.com/y_project/RuoYi/blob/master/ruoyi-framework/src/main/java/com/ruoyi/framework/aspectj/LogAspect.java)，登录系统（系统管理-操作日志）可以查询操作日志列表和详细信息。

## 数据权限

### 基本说明

在实际开发中，需要设置用户只能查看哪些部门的数据，这种情况一般称为数据权限。例如对于销售，财务的数据，它们是非常敏感的，因此要求对数据权限进行控制， 对于基于集团性的应用系统而言，就更多需要控制好各自公司的数据了。如设置只能看本公司、或者本部门的数据，对于特殊的领导，可能需要跨部门的数据， 因此程序不能硬编码那个领导该访问哪些数据，需要进行后台的权限和数据权限的控制。

提示：默认系统管理员admin拥有所有数据权限（userId=1），默认角色拥有所有数据权限（如不需要数据权限不用设置数据权限操作）

### @DataScope注解参数说明



### 数据权限使用

1、在（系统管理-角色管理）设置需要数据权限的角色 目前支持以下几种权限

全部数据权限

## 多数据源

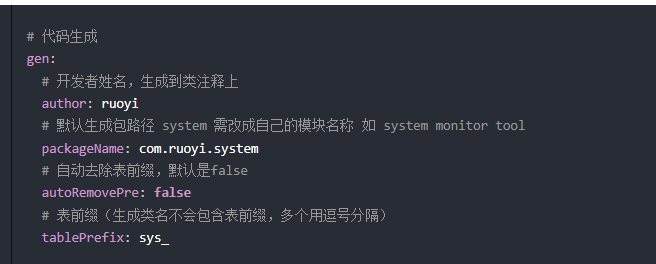
## 代码生成

### 基本说明

大部分项目里其实有很多代码都是重复的，几乎每个基础模块的代码都有增删改查的功能，而这些功能都是大同小异， 如果这些功能都要自己去写，将会大大浪费我们的精力降低效率。所以这种重复性的代码可以使用代码生成。

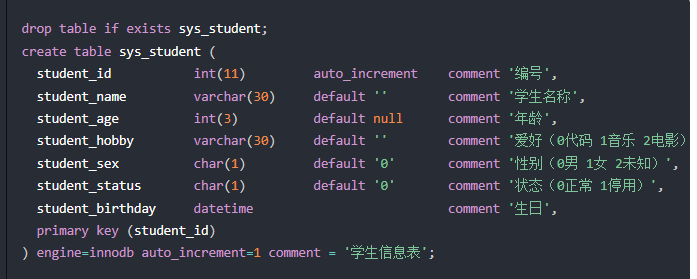
### 默认配置

单应用在resources目录下的application.yml，多模块ruoyi-generator中的resources目录下的generator.yml，可以自己根据实际情况调整默认配置。



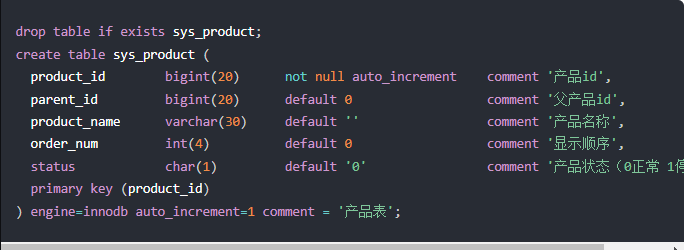
### 单表结构

#### ****新建数据库表结构（单表）****



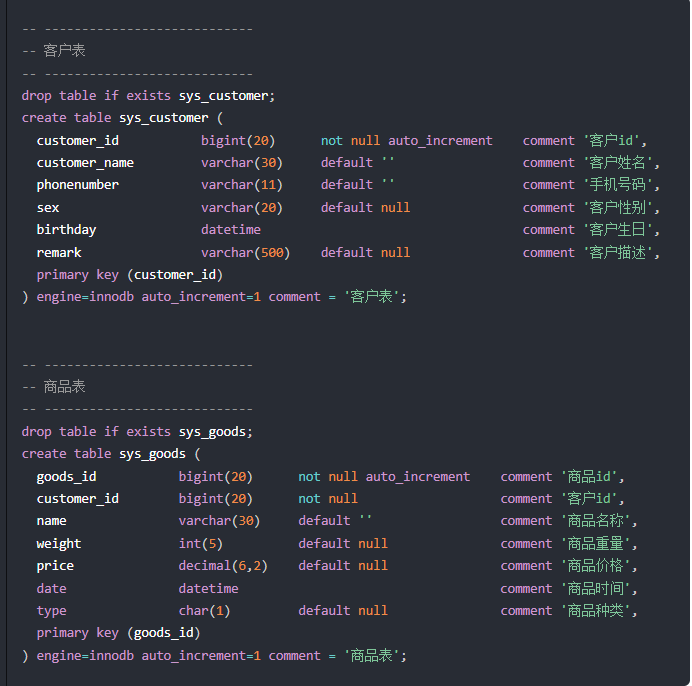
### 树表结构

#### ****新建数据库表结构（树表）****



### 主子表结构

#### ****新建数据库表结构（主子表）****



### 代码生成使用

1、登录系统（系统工具 -> 代码生成 -> 导入对应表）

2、代码生成列表中找到需要表（可预览、编辑、同步、删除生成配置）

3、点击生成代码会得到一个ruoyi.zip，解压文件，执行sql文件，按照包内目录结构复制到自己的项目中即可。

代码生成支持编辑、预览、同步

预览：对生成的代码提前预览，防止出现一些不符合预期的情况。

同步：对原表的字段进行同步，包括新增、删除、修改的字段处理。

修改：对生成的代码基本信息、字段信息、生成信息做一系列的调整。

另外多模块所有代码生成的相关业务逻辑代码在ruoyi-generator模块，不需要可以自行删除模块。

## 定时任务

### 基本说明

在实际项目开发中Web应用有一类不可缺少的，那就是定时任务。 定时任务的场景可以说非常广泛，比如某些视频网站，购买会员后，每天会给会员送成长值，每月会给会员送一些电影券； 比如在保证最终一致性的场景中，往往利用定时任务调度进行一些比对工作；比如一些定时需要生成的报表、邮件；比如一些需要定时清理数据的任务等。 所以我们提供方便友好的web界面，实现动态管理任务，可以达到动态控制定时任务启动、暂停、重启、删除、添加、修改等操作，极大地方便了开发过程。

### 示例

#### 1、后台添加定时任务处理类（支持Bean调用、Class类调用）

##### Bean调用示例

需要添加对应Bean注解@Component或@Service。调用目标字符串：ryTask.ryParams('ry')  
Class类调用示例

添加类和方法指定包即可。调用目标字符串：com.ruoyi.quartz.task.RyTask.ryParams('ry')



#### 2、前端新建定时任务信息（系统监控 -> 定时任务）

##### 任务名称

自定义，如：定时查询任务状态

##### 任务分组

根据字典sys\_job\_group配置

##### 调用目标字符串

设置后台任务方法名称参数

##### 执行表达式

cron表达式

##### 执行策略

定时任务自定义执行策略

##### 并发执行

是否需要多个任务间同时执行

##### 状态

是否启动定时任务

##### 备注

定时任务描述信息

#### 3、点击执行一次，测试定时任务是否正常及调度日志是否正确记录，如正常执行表示任务配置成功。

##### **执行策略详解**

###### 立即执行

（所有misfire的任务会马上执行）打个比方，如果9点misfire了，在10：15系统恢复之后，9点，10点的misfire会马上执行

###### **执行一次**

（会合并部分的misfire，正常执行下一个周期的任务）假设9，10的任务都misfire了，系统在10：15分起来了。只会执行一次misfire，下次正点执行。

###### **放弃执行**

(所有的misfire不管，执行下一个周期的任务)

##### 方法参数详解

字符串（需要单引号''标识，如：ryTask.ryParams(’ry’)）  
布尔类型（需要true false标识 如：ryTask.ryParams(true)）  
长整型（需要L标识 如：ryTask.ryParams(2000L)）  
浮点型（需要D标识 如：ryTask.ryParams(316.50D)）  
整型（纯数字即可）

说明：多模块所有定时任务的相关业务逻辑代码在ruoyi-quartz模块，可以自行调整或剔除。

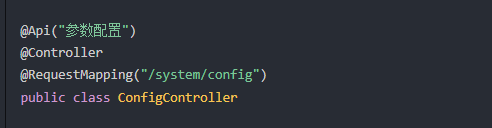
注意：不同数据源定时任务都有对应脚本，Oracle、Mysql已经有了，其他的可自行下载执行

## 系统接口

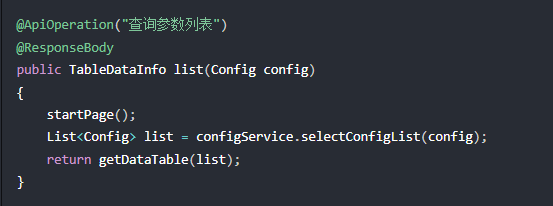
### 基本说明

通过Swagger， 只要你在写代码的时候，稍加几个注解，文档自动生成。

1、在控制层Controller中添加注解来描述接口信息如:



2、在方法中配置接口的标题信息



3、在系统工具-系统接口测试相关接口

注意：SwaggerConfig可以指定根据注解或者包名扫描具体的API

### API详细说明



### @Api

用在类上，说明该类的作用。可以标记一个Controller类做为Swagger文档资源，使用方式：



与Controller注解并列使用。

#### 属性



### @ApiOperation

用在方法上，说明方法的作用，每一个url资源的定义,使用方式：



与Controller中的方法并列使用。

#### 属性



## 国际化支持

## 新建子模块

# 注解

## @Log

# 接口

## 后台登陆接口

# 问题

## 1、Access denied for user 'root'@'localhost' (using password: YES)

### 解决方法

## 2、如何不登录直接访问

登录nacos在配置管理中配置列表，修改ruoyi-gateway-dev.yml，在ignore中设置whites，表示允许匿名访问。



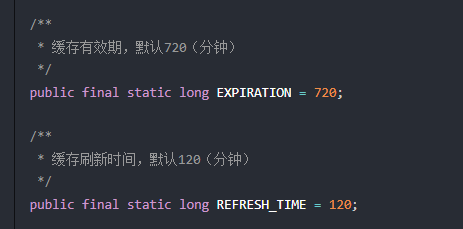
## 3、如何获取用户登录信息

### ****后端获取当前用户信息****



## 4、如何设置令牌有效期

可以在ruoyi-common-core模块中的com.ruoyi.common.core.constant.CacheConstants类中设置。



5、