**建行一世脚手架系统说明书**

目录

[建行一世脚手架系统说明书 1](#_Toc17988)

[一、 介绍 2](#_Toc2708)

[1.由来 2](#_Toc7567)

[2. 基本技术栈 3](#_Toc21896)

[（1）系统环境 3](#_Toc26830)

[（2）主框架 3](#_Toc5020)

[（3）持久层 3](#_Toc25279)

[（4）视图层 3](#_Toc20059)

[3. 功能介绍 3](#_Toc26983)

[4. 目录结构 5](#_Toc982)

[前端 5](#_Toc8774)

[后端 6](#_Toc11377)

[二、 下载与启动 6](#_Toc4459)

[1. 所需环境 6](#_Toc31804)

[2. 下载与准备 7](#_Toc16895)

[3. 启动 7](#_Toc9156)

[前端： 7](#_Toc21559)

[后端： 7](#_Toc13511)

[查验 8](#_Toc9170)

[三、 部署 8](#_Toc12591)

[前端打包 9](#_Toc5005)

[后端打包 9](#_Toc16192)

[前端部署 9](#_Toc11573)

[后端部署 10](#_Toc5086)

[几点解释 10](#_Toc826)

[Devops（基于Jenkins） 10](#_Toc27731)

[四、 定制化相关 23](#_Toc26610)

[1. 数据权限 23](#_Toc31490)

[（1） Mybatis-Plus和Mybatis-Plus-join 24](#_Toc28661)

[（2） Mybatis(xml方式) 26](#_Toc1765)

[（3） 相关解释 26](#_Toc8235)

[2. 通知模块 27](#_Toc15305)

[（1） 用户端三种类型通知的感受与反馈 27](#_Toc28043)

[（2） 关于创建三种通知 29](#_Toc22951)

[3. 多sheet页Excel导出 31](#_Toc9116)

[（1） 实现效果 32](#_Toc9101)

[（2） 工具介绍 32](#_Toc12345)

[（3） 整合数据，调用工具导出方法 34](#_Toc13035)

[4. 限制多端登录 34](#_Toc1569)

[5. 设置接口登录直接访问（不鉴权） 35](#_Toc14924)

[（1）后端 35](#_Toc5465)

[（3） 前端 37](#_Toc23223)

[6. 定时任务 37](#_Toc3960)

[（1） 介绍 37](#_Toc2392)

[（2） 使用流程 37](#_Toc8002)

[（3） 执行成功 40](#_Toc12056)

[7. 代码生成器 40](#_Toc13895)

[8. 数据及其组件库 41](#_Toc14927)

[ 在系统中，给出了几个相关组件的使用示例。 41](#_Toc26729)

[ 在系统中，给出了一些数据的示例。 41](#_Toc23225)

[五． 常用功能及注意点 41](#_Toc19553)

[1. 系统实体类 41](#_Toc8782)

[2. 代码生成器 42](#_Toc20940)

[3. 系统码表 42](#_Toc22964)

[4. 验证码使用 42](#_Toc16453)

[5. 登录相关 43](#_Toc3426)

[6. 流量限制控制 43](#_Toc28816)

[7. 前端相关 44](#_Toc16552)

# 介绍

## 1.由来

建行一世脚手架系统是一套快速开发平台，是根据开源框架[若依](https://github.com/yangzongzhuan/RuoYi-Vue)二次开发而来，支持加载动态权限菜单，多方式轻松权限控制，高效率开发，使用代码生成器可以一键生成前后端代码。

经过不同系统的使用总结以及其他技术大牛的改编参考，加入了许多适用于现在开发习惯的功能框架例如mybatis-plus、mybatis-plus-join、hutool等，并且同步接入了框架本身具有的数据权限控制以及代码生成器，包括mybatis-plus自带的伪删除与自动注入配置也可以使用。

还有一些包括开发习惯上的变动，例如在原本只有开发与正式环境的配置文件中加入了测试的配置文件、代码生成器中的特色代码、父级实体类中的属性等等。

目前，根据需求，建行一世脚手架系统另有一些格外加入的功能，包括：基于webSocket开发的三级通知功能、前端美化UI组件、logback日志、基于docker的部署等等。

除此之外，建行一世脚手架系统也一直跟随若依官方进行bug与漏洞的修补，现在已经同步迭代到了3.8.5版本。

## 基本技术栈

**（1）系统环境**

* Java EE 8
* Servlet 3.0
* Apache Maven 3

**（2）主框架**

* Spring Boot 2.2.14
* Spring Framework 5.2.x
* Spring Security 5.2.x

**（3）持久层**

* Apache MyBatis 3.5.x
* Hibernate Validation 6.0.x
* Alibaba Druid 1.2.x
* mybatis-plus 3.4.2
* Hutool 5.8.5

**（4）视图层**

* Vue 2.6.x
* Axios 0.21.x
* Element 2.15.x

## 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。

在线用户：当前系统中活跃用户状态监控。

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：监视当前系统CPU、内存、磁盘、堆栈等相关信息。

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。

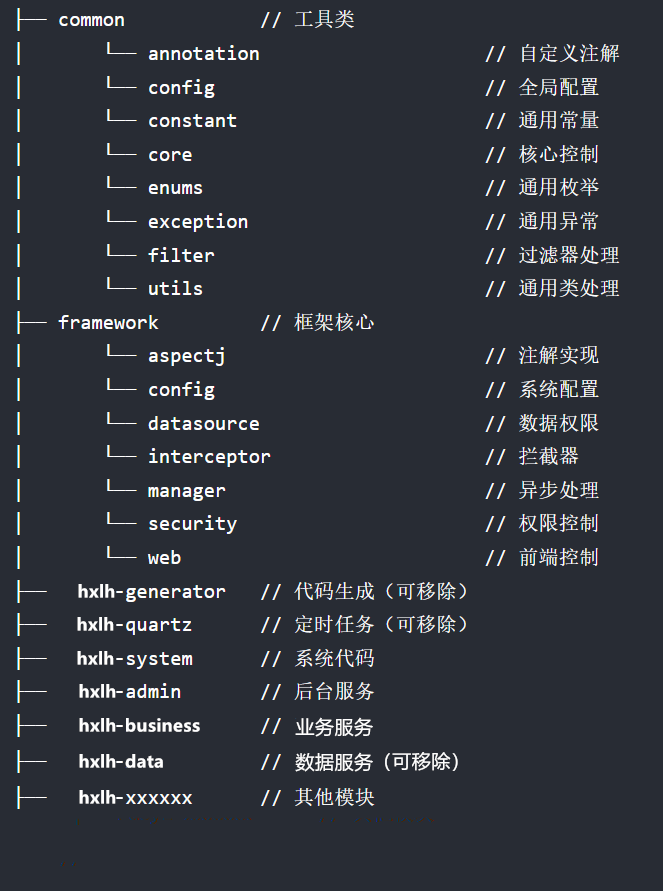
通知管理：后端封装有工具类，创建通知后将根据通知等级以不同程度实时反馈到用户前端

## 目录结构

### 前端



### 后端



PS： 以上图片中的【hxlh-】均看做【jxys-】即可！之前搞公司使用的架子时设计的，后期在重新做个图吧，现在还是先整重要的部分，求谅解哈❤

# 下载与启动

## 所需环境

* JDK1.8+
* MySQL 5.7
* Redis 4.0+
* Maven 3.3+
* Node.js 12+ 推荐v14.16.1

## 下载与准备

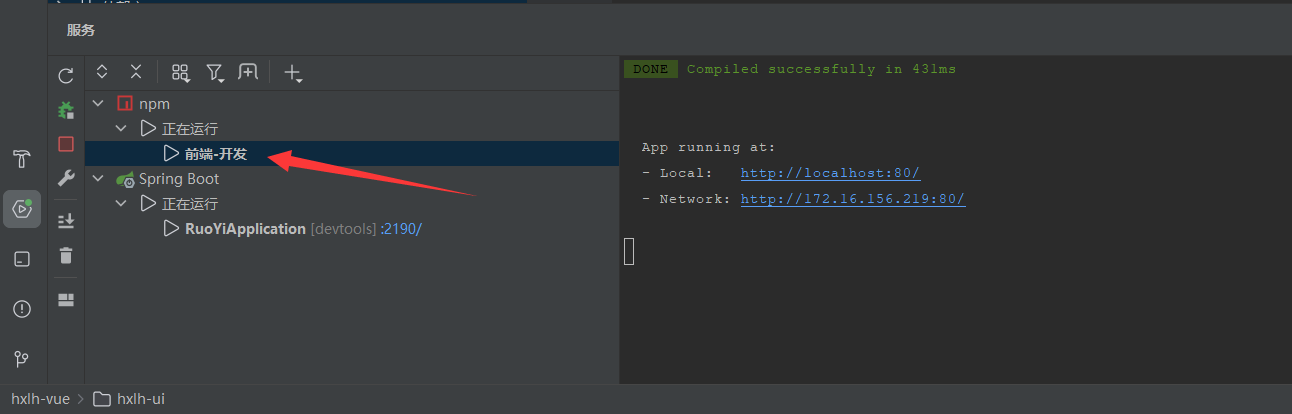
1. 拉取Git：<https://github.com/a241978181/jxys-vue.git>
2. 使用maven工具下载jar包
3. 将目录【jxys-vue/sql/】中的【jxys\_vue.sql】文件放入MySQL执行
4. 修改【jxys-vue/jxys-admin/src/main/resources】目录下【application-druid.yml】文件中的MySQL配置与Redis配置
5. 在命令行中进入到【jxys-vue/jxys-ui】目录下，执行如下指令：npm install或者yarn install。

## 启动

### **前端**：

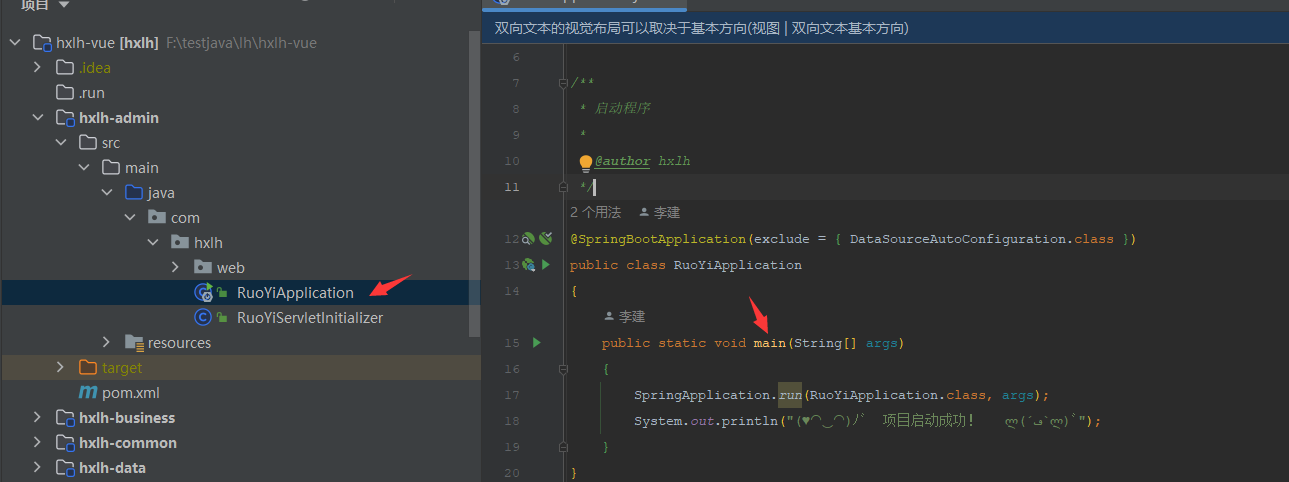
在命令行中进入到【jxys-vue/jxys-ui】目录下，执行 npm run dev或者yarn run dev。如果执行过慢可以使用npm install --registry=https://registry.npmmirror.com强烈建议不要用直接使用 cnpm 安装，会有各种诡异的 bug，可以通过重新指定 registry 来解决 npm 安装速度慢的问题。

如果在IDEA中则可以直接点击启动【服务】窗口中的【前端-开发】

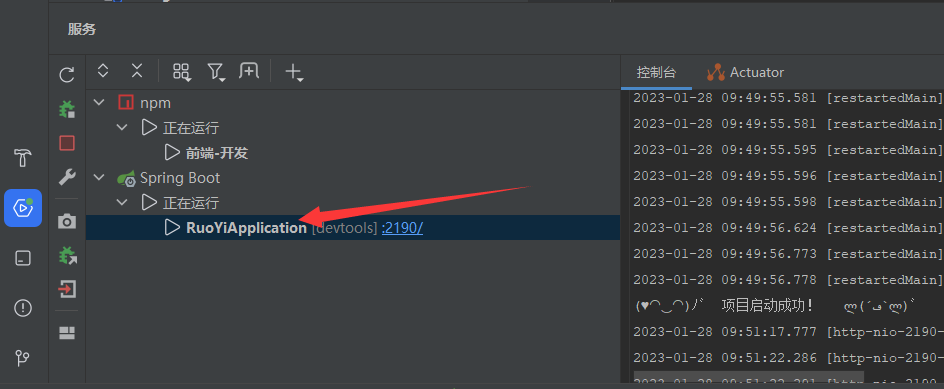


### **后端**：

在目录【jxys-vue/jxys-admin/src/main/java/com/Jxys】中找到【RuoYiApplication】文件。点击文件上的启动函数main即可启动后端项目。



如果在IDEA中则可以直接点击启动【服务】窗口中的【RuoYiApplication】



### 查验

打开浏览器，输入：(http://localhost:80)

默认账户：admin密码：admin123

若能正确展示登录页面，并能成功登录，菜单及页面展示正常，则表明环境搭建成功

# 部署

建行一世脚手架系统项目来说，部署的方式有很多种:

第1种方式:可以将前端打包整合到后端部署

第2种方式:可以使用tomcat部署(后端打war包+前端打dist包)

第3种方式:可以使用tomcat+nginx部署(后端打war包+前端打dist包+nginx配置)

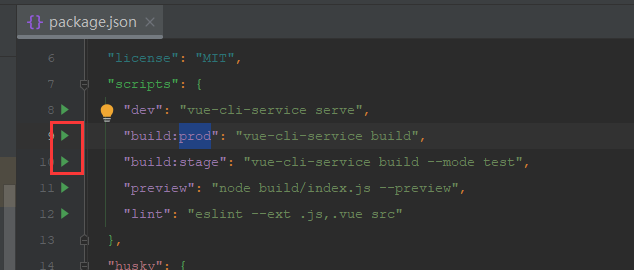
第4种方式:可以使用jar+nginx部署（后端打jar包+前端打dist包+nginx配置)

第5种方式:可以使用docker脚本一键部署

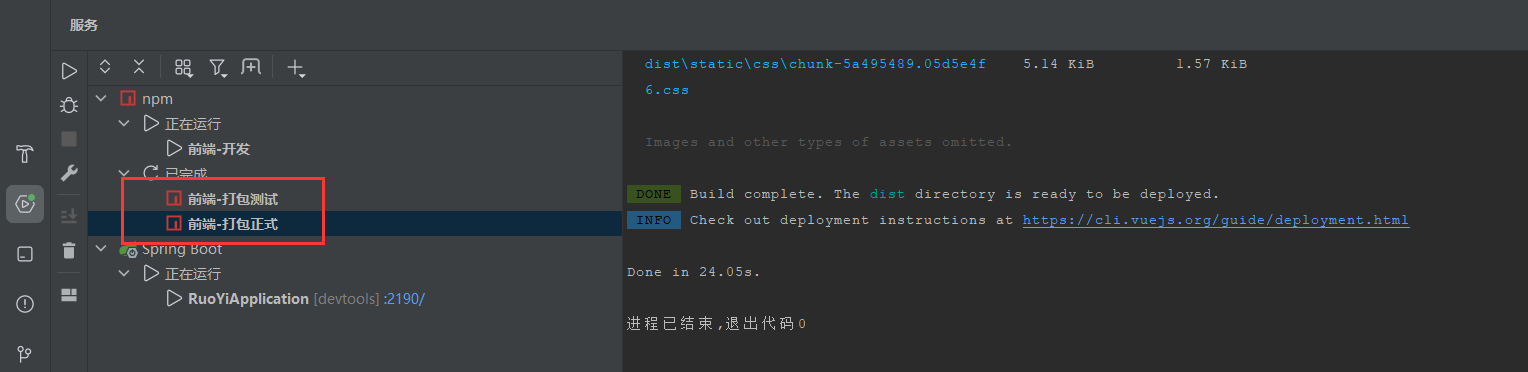
个人认为第4种方式，部署过程想比其它部署方式要简单一些，本章教程主要介绍一下这种部署方式的具体过程。（注意：这里会结合第5中方式，目的是为了后续对接Jenkins组成DevOps）

### 前端打包

**方式一**：在idea中，目录【jxys-ui】下找到【package.json】文件，直接点击build:stage（测试）或者build:prod（正式）前面的启动按钮即可生成dist文件夹



**方式二**：在idea中，可以直接点击启动【服务】窗口中的【前端-打包测试】或者【前端-打包正式】来生成包dist



方式三：在命令行中进入目录【jxys-vue/jxys-ui/】执行npm run build:stage或者yarn build:prod来生成dist文件

### 后端打包

在maven窗口中选择聚合项目的父项目（Jxys），在【生命周期】执行【package】，成功后可以在目录【jxys-vue/jxys-admin/target/】中生成jxys-admin.jar包

### 前端部署

[由于部署这块涉及很广，篇幅巨大，细节也比较多，固引用一篇提前写好的部署教程吧](https://blog.csdn.net/jxysgzs/article/details/105864442)

### 后端部署

后端部署同样需要Linux中已经安装并配置好了docker工具，将打包好的jar包放到一个目录上（/opt/Jxys\_jar/）。执行如下命令即可：

|  |
| --- |
| docker run -d --restart=always -e TZ="Asia/Shanghai" -p 2190:2190 -v /opt/Jxys\_jar/jxys-admin.jar:/usr/jxys-admin.jar -v /etc/localtime:/etc/localtime:ro --name Jxys\_admin java:8 java -jar /usr/jxys-admin.jar --spring.profiles.active=pro |

### 几点解释

1.部署启动之后，如果需要更新版本，只需要替换宿主机目录中的包，然后重启一下docker容器即可更新到新版本，十分方便

2.以上部署方式已经对容器内部的时间做了更改，无需担心。

3.如果想将容器直接挂载到宿主机上，只需要将执行的docker命令中加入【 --network=host】并删除【-p 2190:2190】即可。

4.需要安装Jenkins的话教程也是多又杂，以下举例做一个参考教程。

### Devops（基于Jenkins）

#### Jenkins部署

##### 1. 安装jdk

因为jenkins是用java编写的，所以本地服务器上需要安装jdk环境，推荐使用jdk1.8。本步骤能够自己完成的童鞋可以自行跳过。

yum install java-1.8.0-openjdk\* -y

默认安装到目录 /usr/lib/jvm下。

##### 2. 安装jenkins

由于使用weget太慢，我比较喜欢直接本地上传安装，所以给大家个连接，大家可以下载进行安装。

链接：https://pan.baidu.com/s/1bZqMcs15mu5fSJxcYh-W6g

提取码：p693

复制这段内容后打开百度网盘手机App，操作更方便哦--来自百度网盘超级会员V2的分享

将rpm文件上传到服务器后，执行以下命令

rpm -ivh jenkins-2.263.4-1.1.noarch.rpm

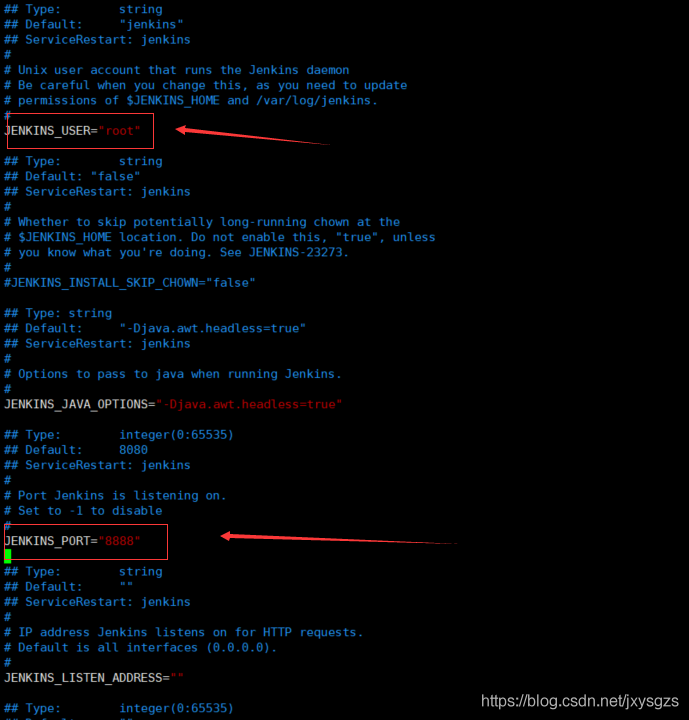
##### 3. 修改jenkins配置

vi /etc/syscofig/jenkins

修改内容：

JENKINS\_USER="root"

JENKINS\_PORT="8888"



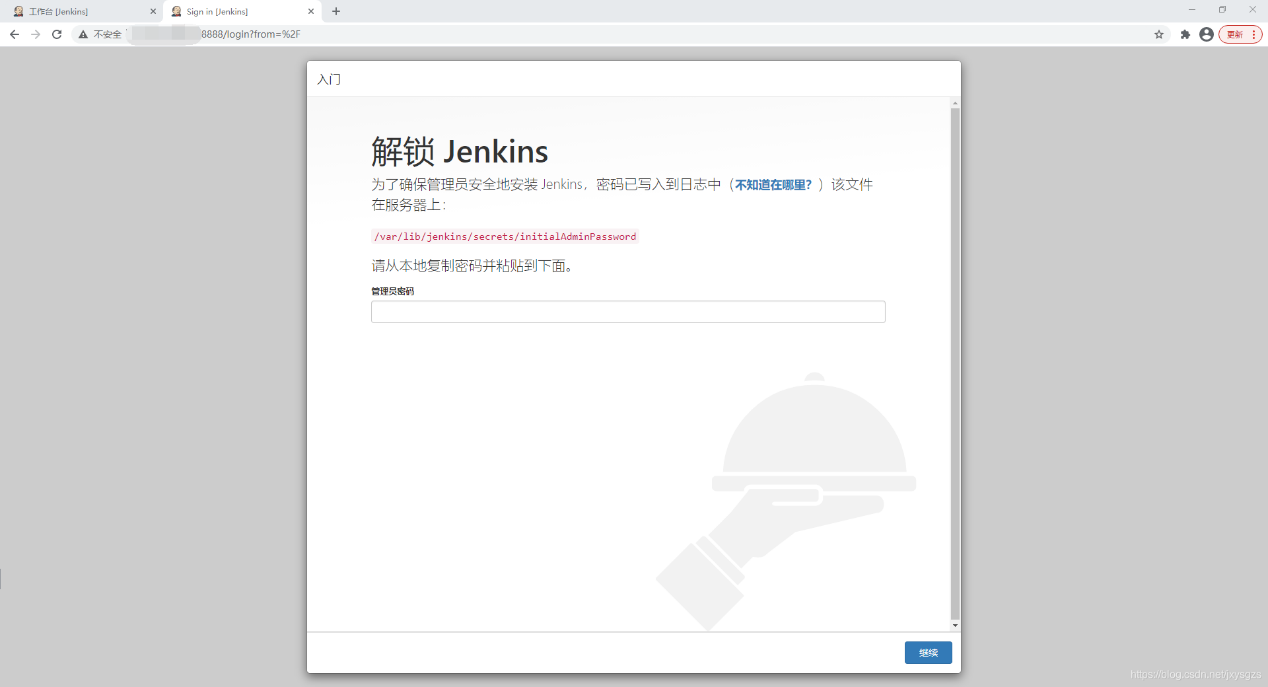
##### 4.启动jenkins

systemctl start jenkins

注意:启动之后，注意检查自己的防火墙与安全组是否开放8888，然后访问服务器8888端口，即可访问。

如果启动报错缺少policycoreutils-python，可以使用下面命令来解决

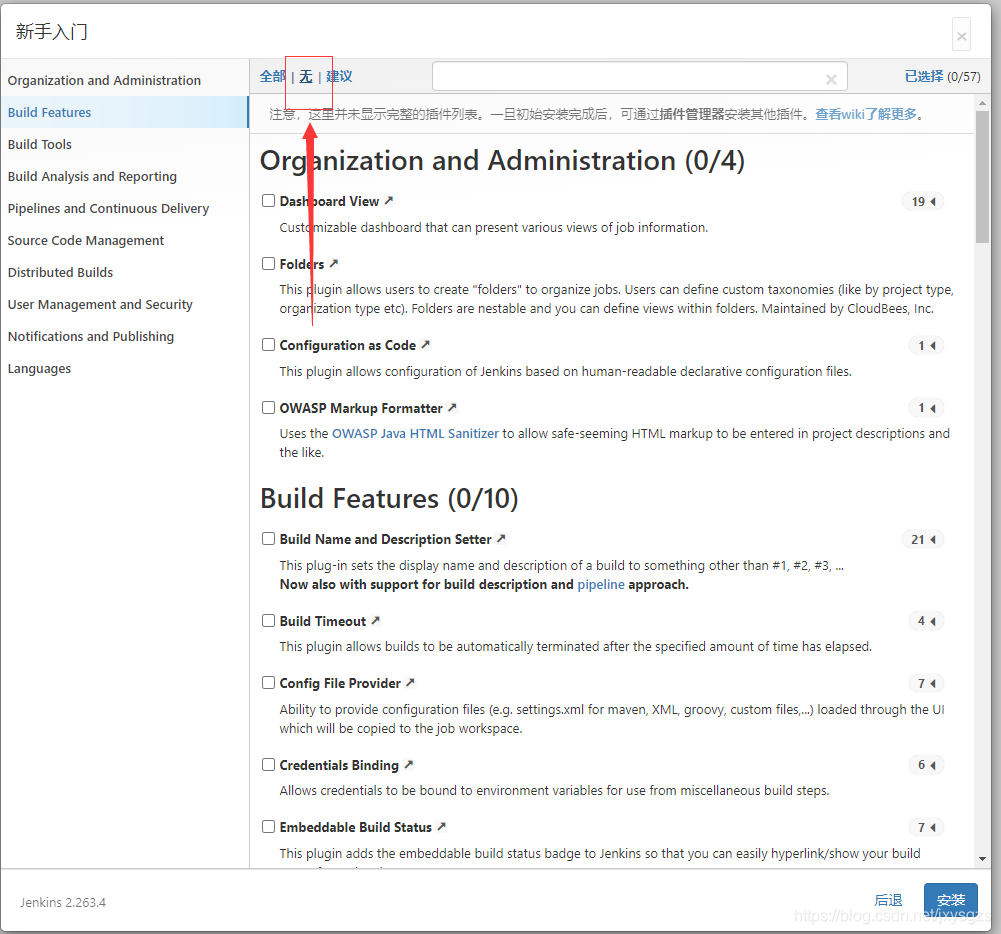
yum install -y curl policycoreutils-python openssh-server



##### 5.登录jenkins

这里登录需要使用到一个管理员密码，我们可以在服务器上使用如下命令获得

cat /var/lib/jenkins/secrets/initialAdminPassword

在这里插入图片描述复制密码，放到web的输入框内点击继续。  
按如下图片选择  
  


注意 后面页面忘了截屏了。。抱歉各位，好在后面就是一个设置管理员账号密码的页面，自己输入账号密码设置完成继续就没了。

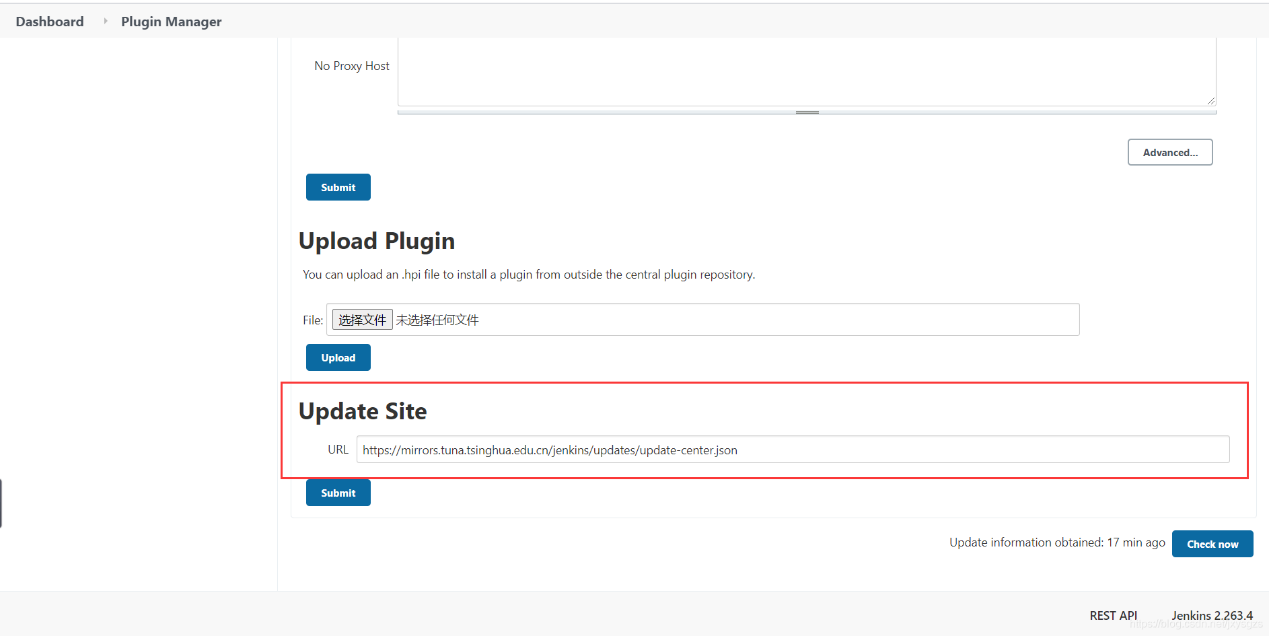
##### 6.配置jenkins下载插件国内清华大学链接

1. 服务器上使用如下命令：

sed -i 's/http:\/\/updates.jenkins-ci.org\/download/https:\/\/mirrors.tuna.tsinghua.edu.cn\/jenkins/g' default.json && sed -i's/http:\/\/www.google.com/https:\/\/www.baidu.com/g' default.json

1. 在页面中Manage Jenkins–>Manage Plugins点击Advanced后拉到最下面，将红色方框内的链接修改为

https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json

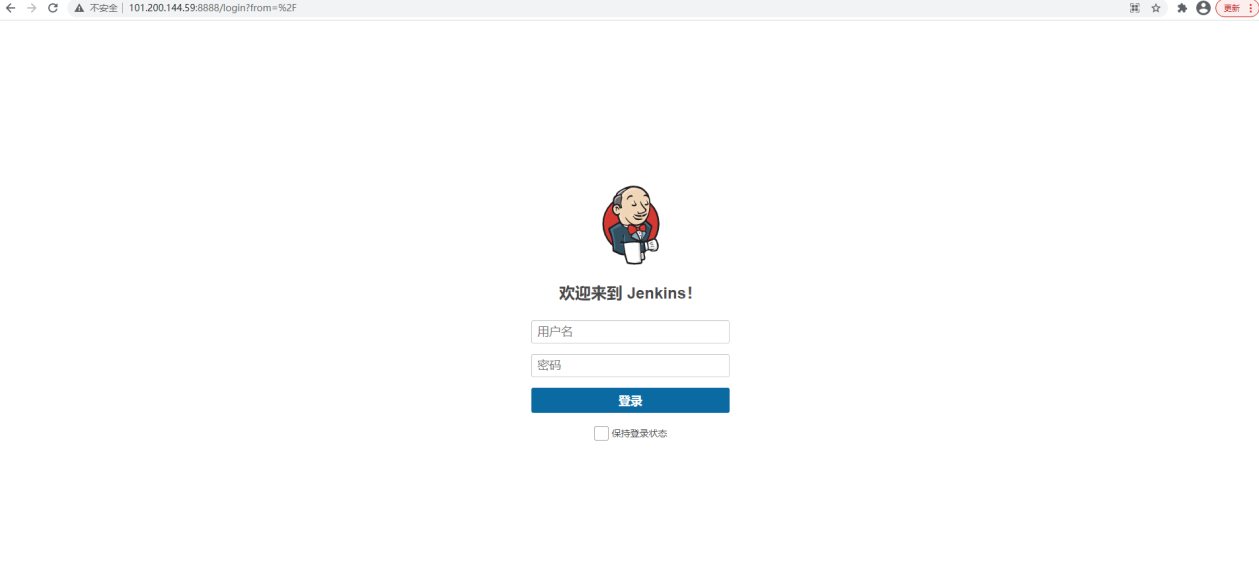


#### Jenkins使用

##### 地址

正式地址为：http://101.200.144.59:8888/login?from=%2F

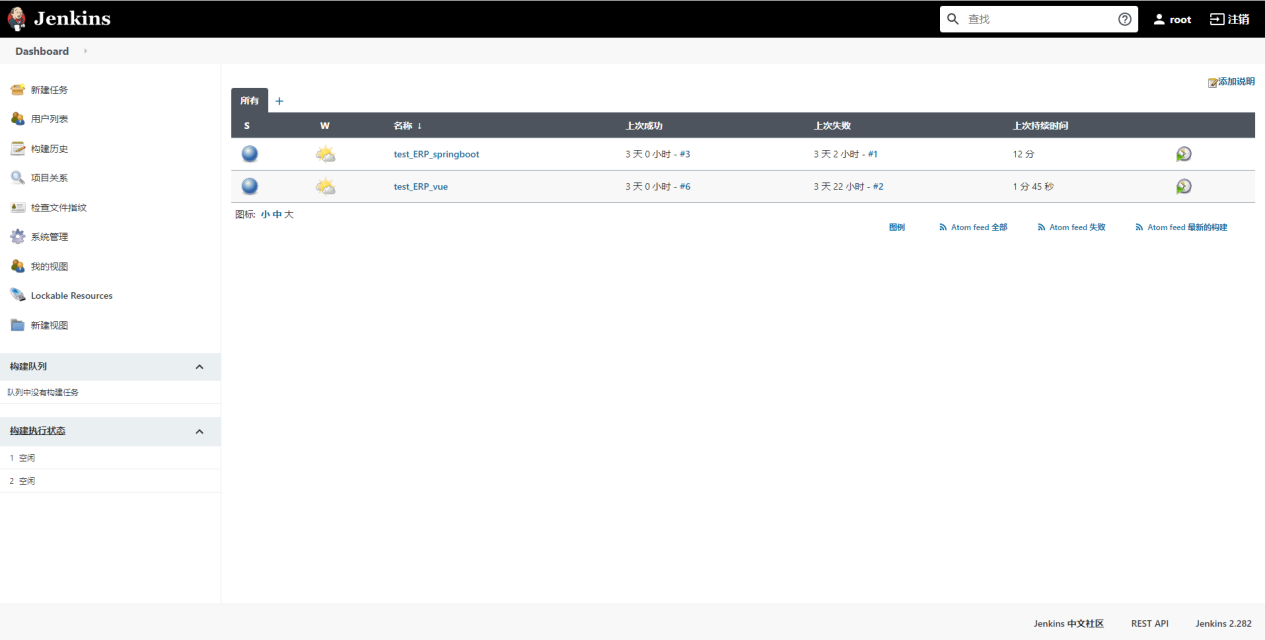
##### 登录



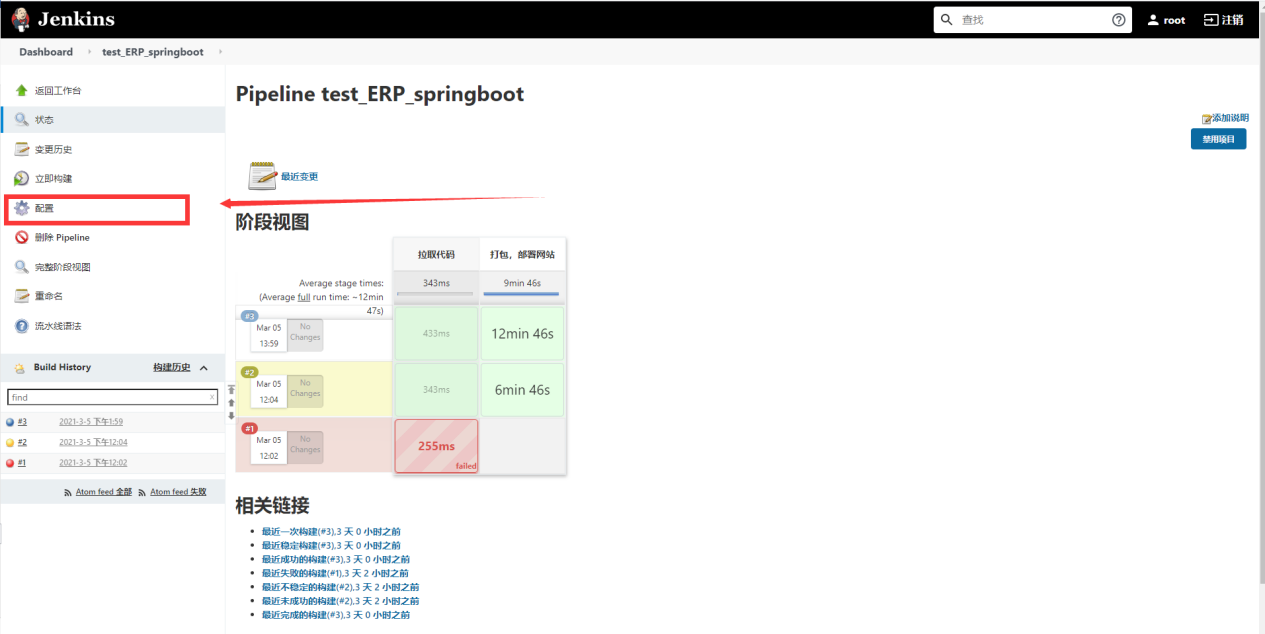
访问链接到达登录页面后，输入账号密码进行登录，登录时勾选保持登录状态

##### 项目

登录之后进入主页面，主页面中包含功能列表与项目列表。



在右侧项目列表中点击项目的名称即可进入项目内部。点击左侧功能列表中的【配置】按钮即可进入项目的详细配置。



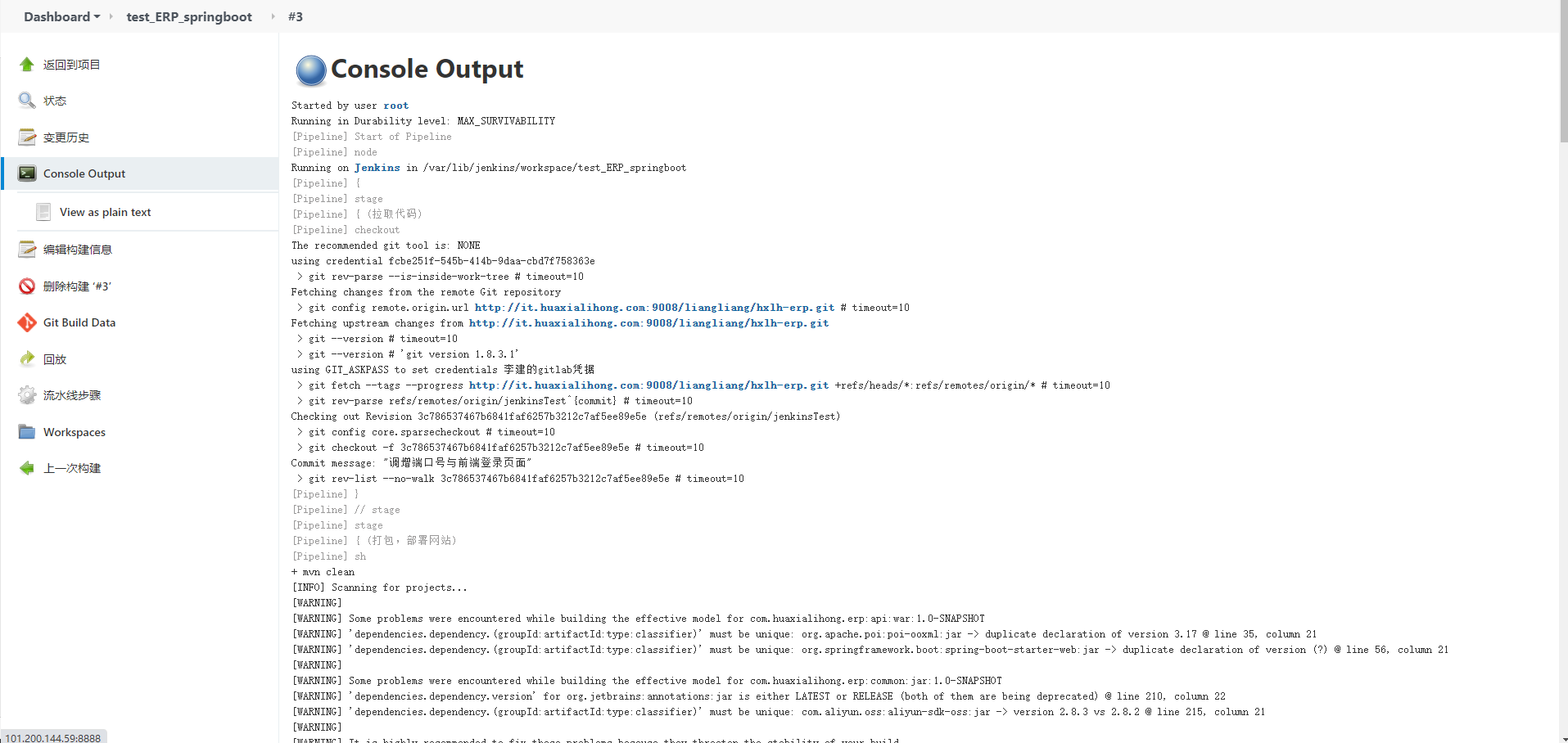
##### 执行

进入项目内部后，点击【立即构建】按钮即可开启全套构建过程（拉取gitlab---编译打包项目----将目标服务器内部旧项目存储到指定位置----将打包好的新项目发送到目标服务器----重启docker容器）。过程完成之后可以在jenkins项目内部主页面查看。



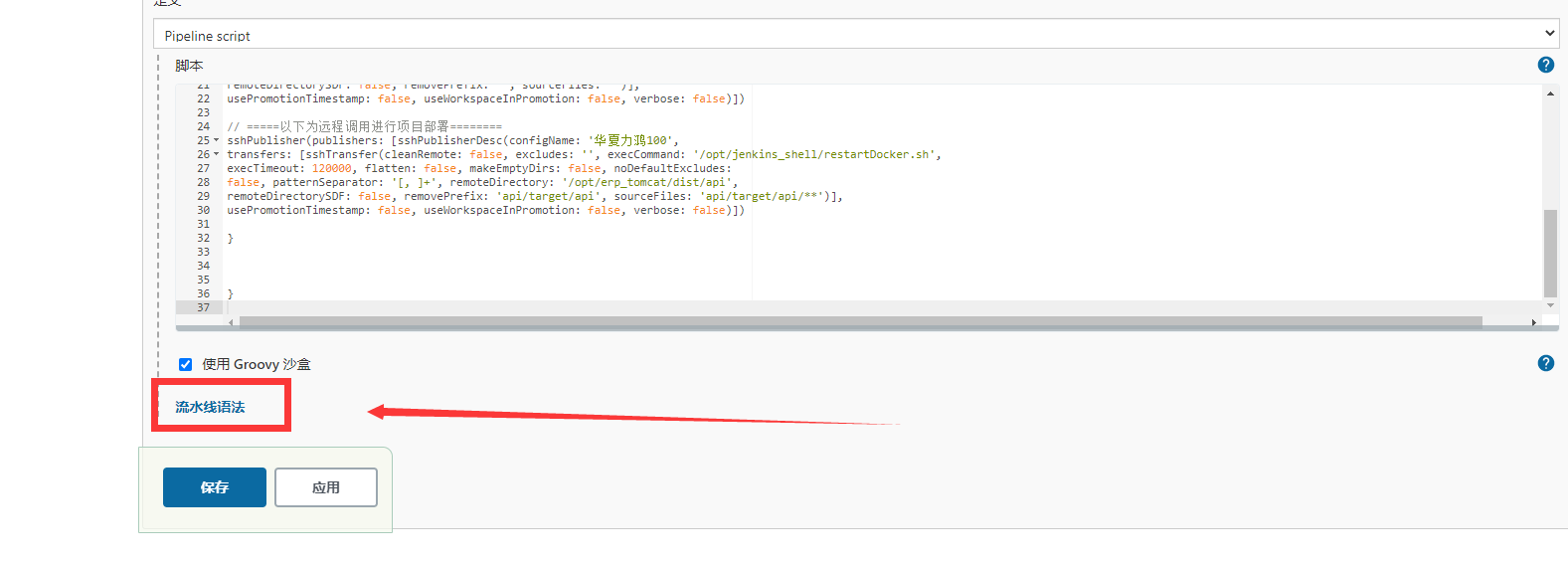
当然，部署过程的日志可以在【构建历史】中点击当前构建过程，

进入后点击【Console Output】查看详细的构建过程日志。



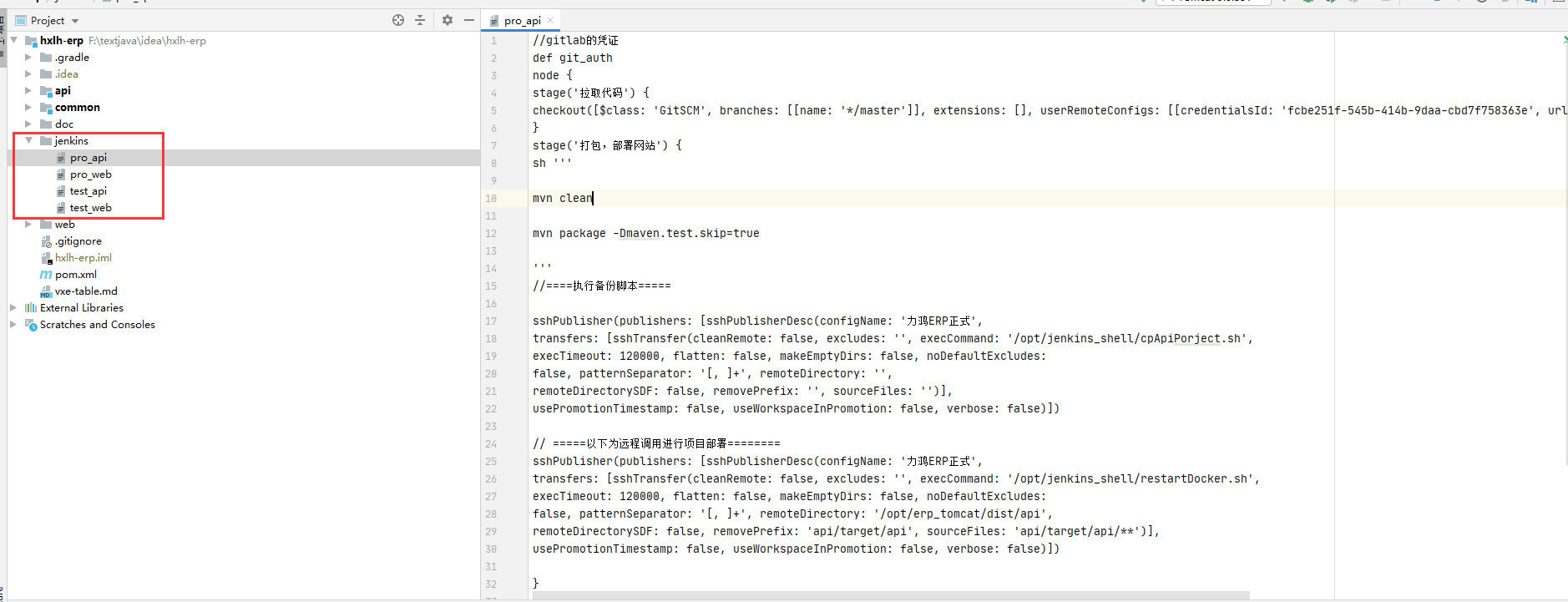
##### Pipeline脚本

Pipeline脚本可自行编写也可以采用自带语句生成器来编写。



#### Jenkins项目配置

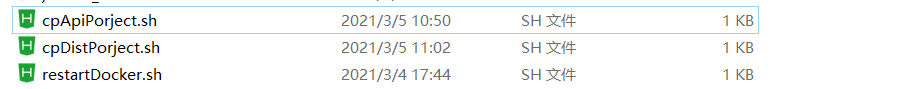
##### 项目中添加jenkins可执行的pipeline脚本



脚本中包含了拉取gitlab中的代码已经编译打包部署的过程。【sshPublisher

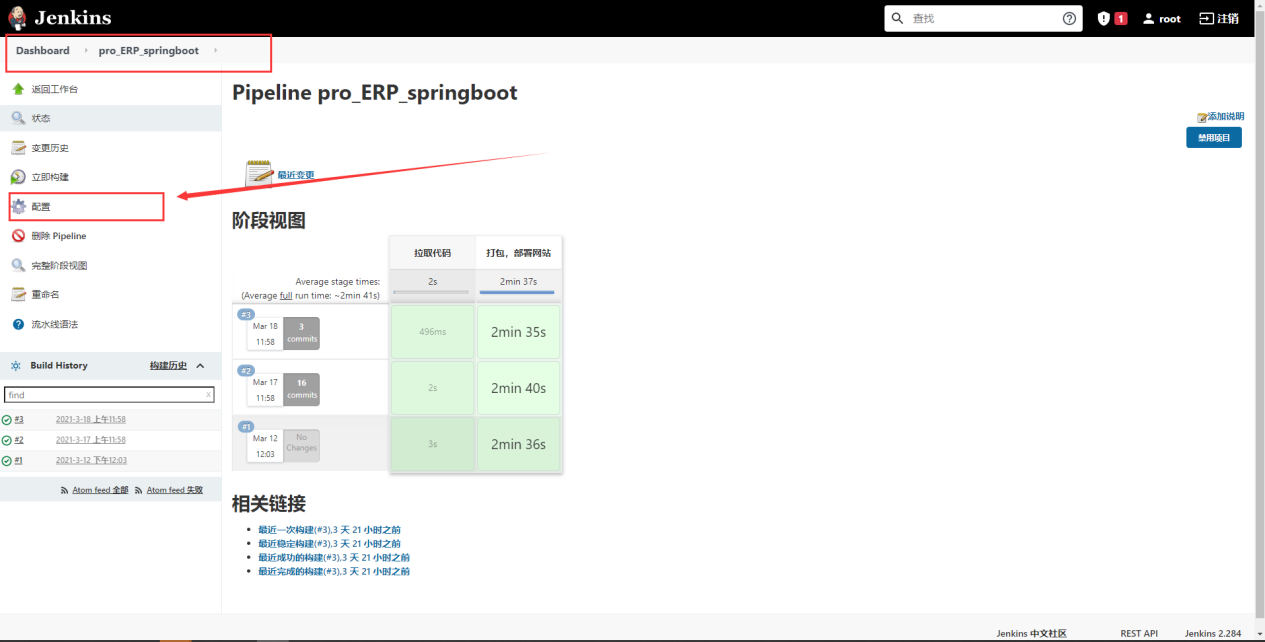
】函数可以对目标服务器进行SSH数据传送已经执行目标Linux中的.sh可执行文件。

##### 将.sh可执行文件放到Linux的一个目录中



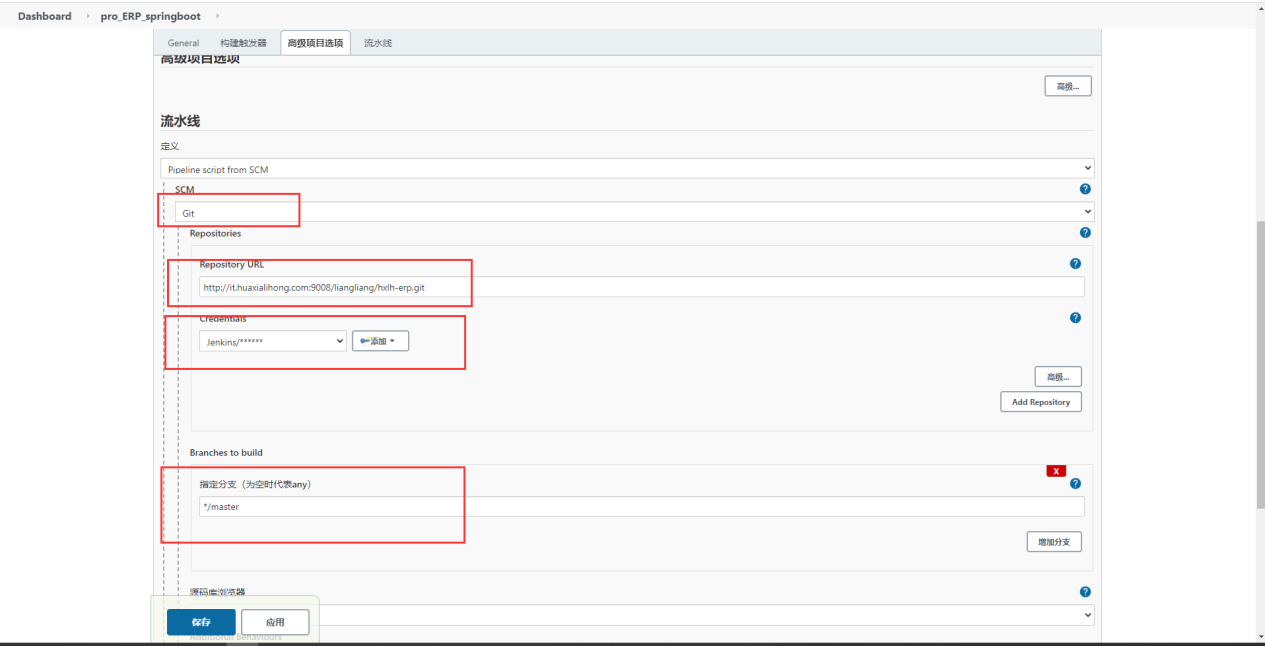
.sh文件中包含备份原打包项目以及重启docker的命令。

##### 在jenkins的控制页面中进行项目配置



如上图进入项目的配置页面。（创建新项目时需要选择流水线工程）

进入配置页面后在流水线标签下配置在第1步骤中项目工程的gitlab地址以及获取pipeline脚本的路径。



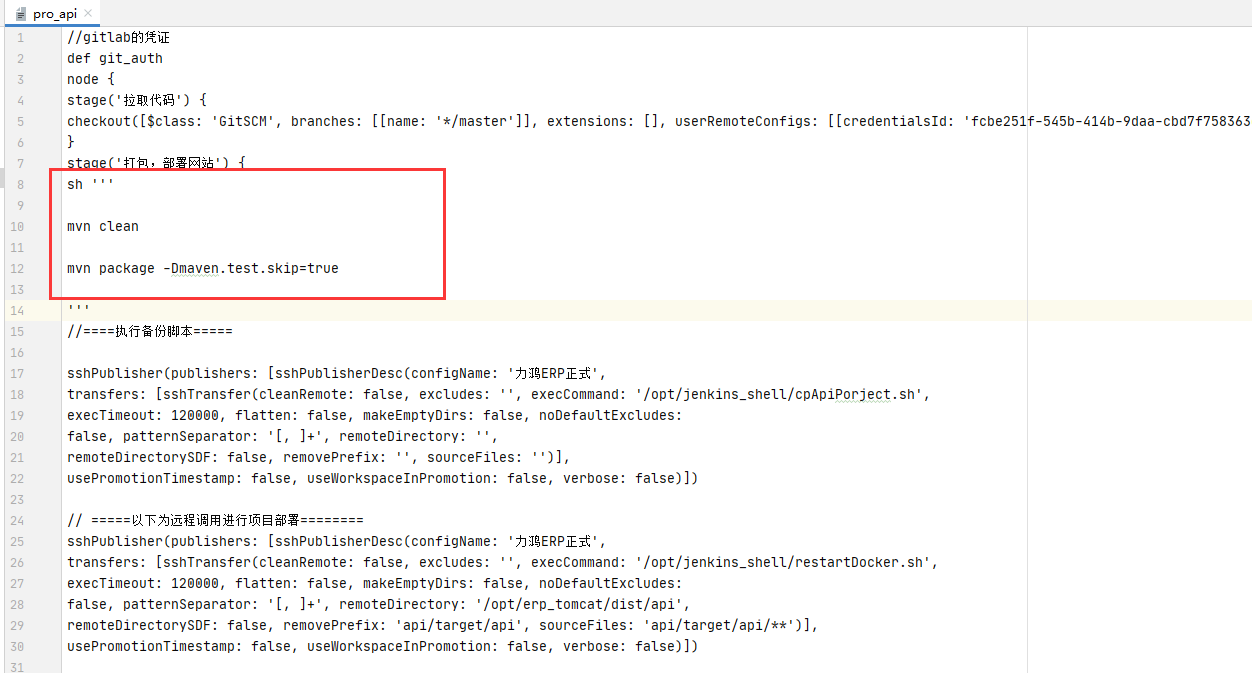


配置完成后点击【保存】按钮，即可完成项目配置。

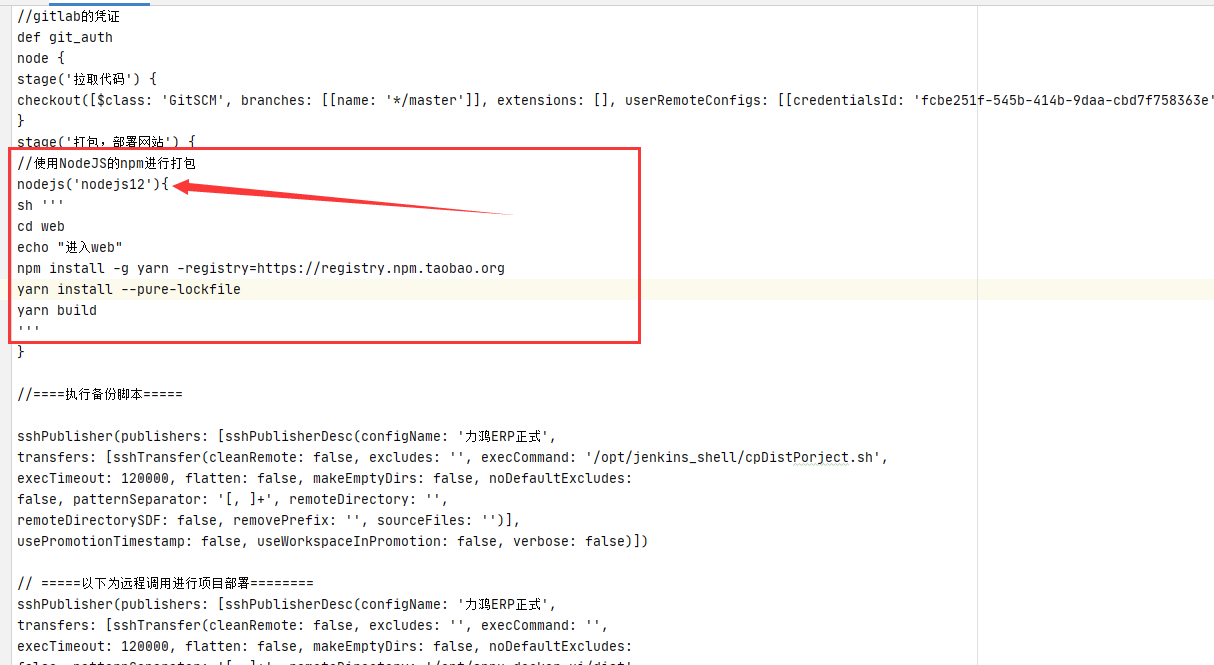
#### Jenkins注意事项

1. Pipeline脚本中，打包springboot项目使用的是jenkins服务器自带的Maven环境和mvn命令而打包vue时需要选择配置node环境中执行npm命令,不能直接填写命令。（如下图）

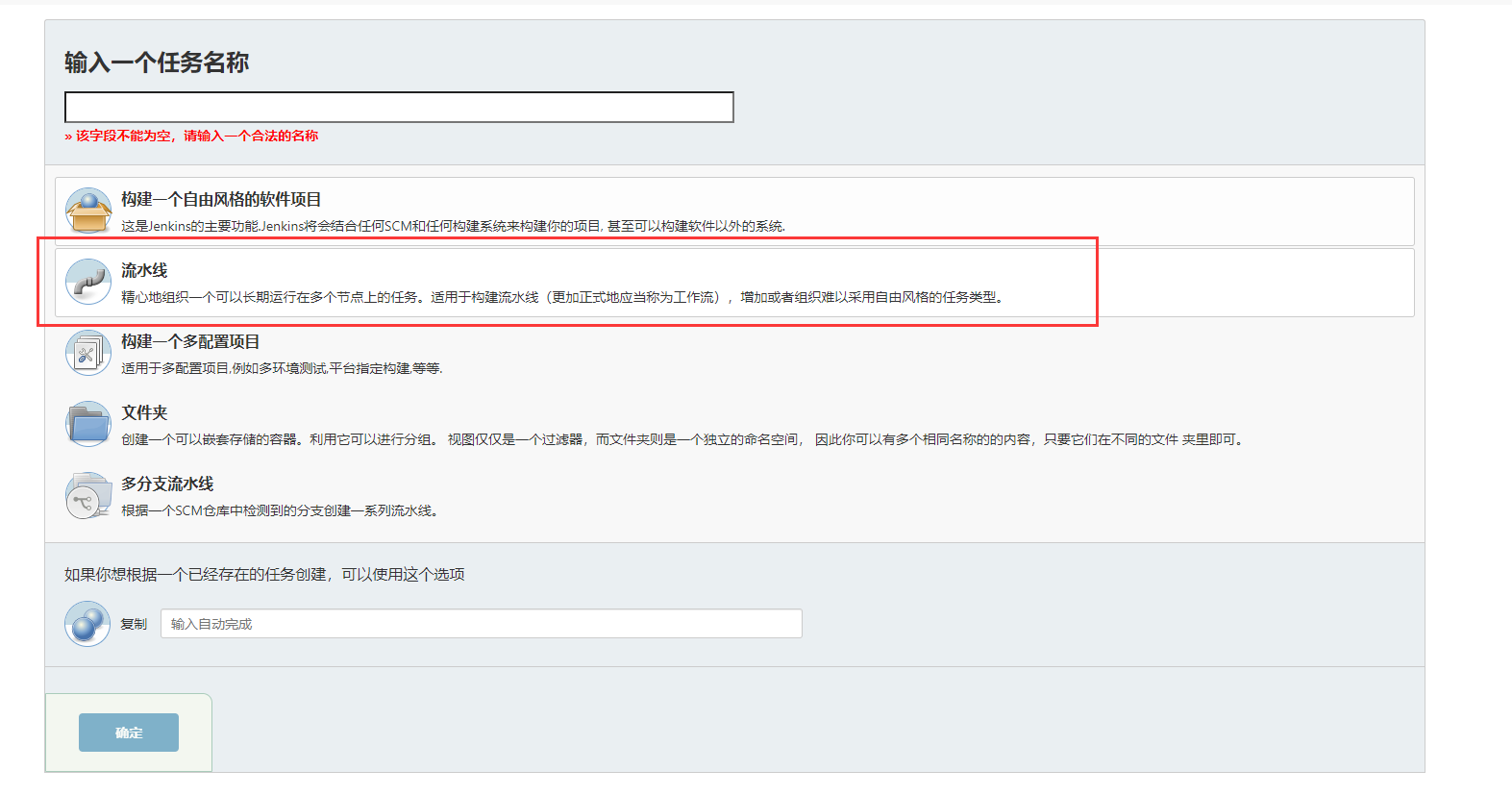
后端



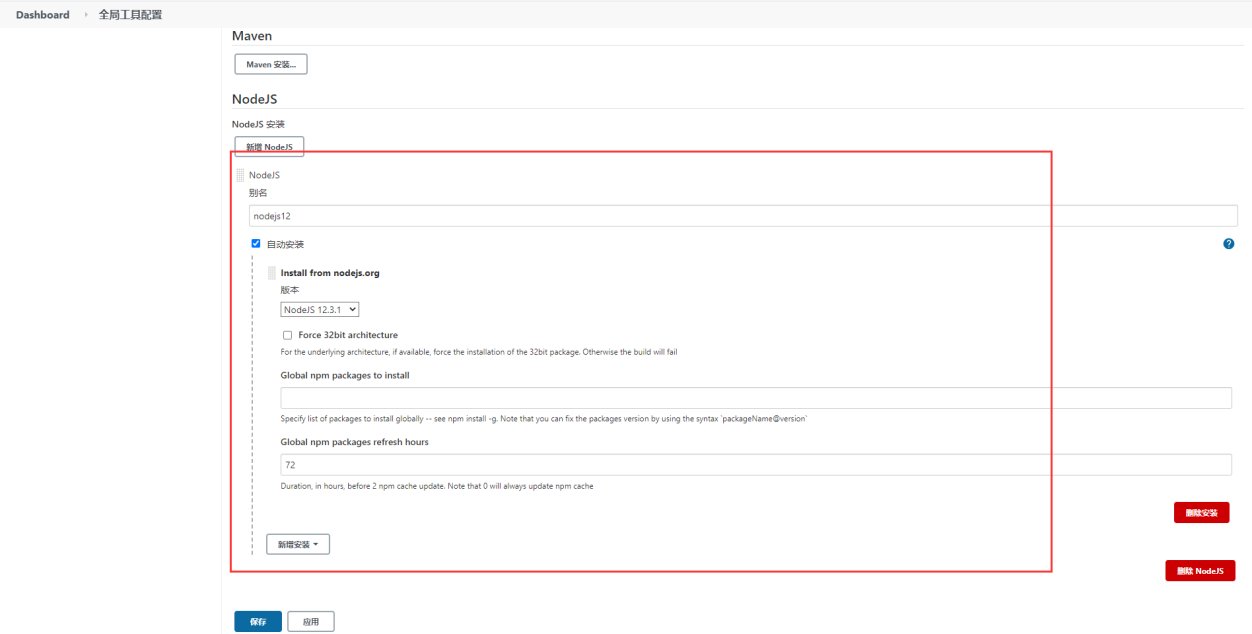
前端

2. 【sshPublisher】函数中，使用execCommand参数可以执行远程服务器上的.sh文件。值得注意的是如果一个【sshPublisher】函数中既使用了execCommand又使用了传输数据的remoteDirectory与sourceFiles，则先执行数据传输在调用.sh文件。

3. 新建项目时请选择流水线工程。



1. 使用nodeJS等引用环境时，需要在全局工具配置中率先配置上nodeJS环境



#### Pipeline脚本注释

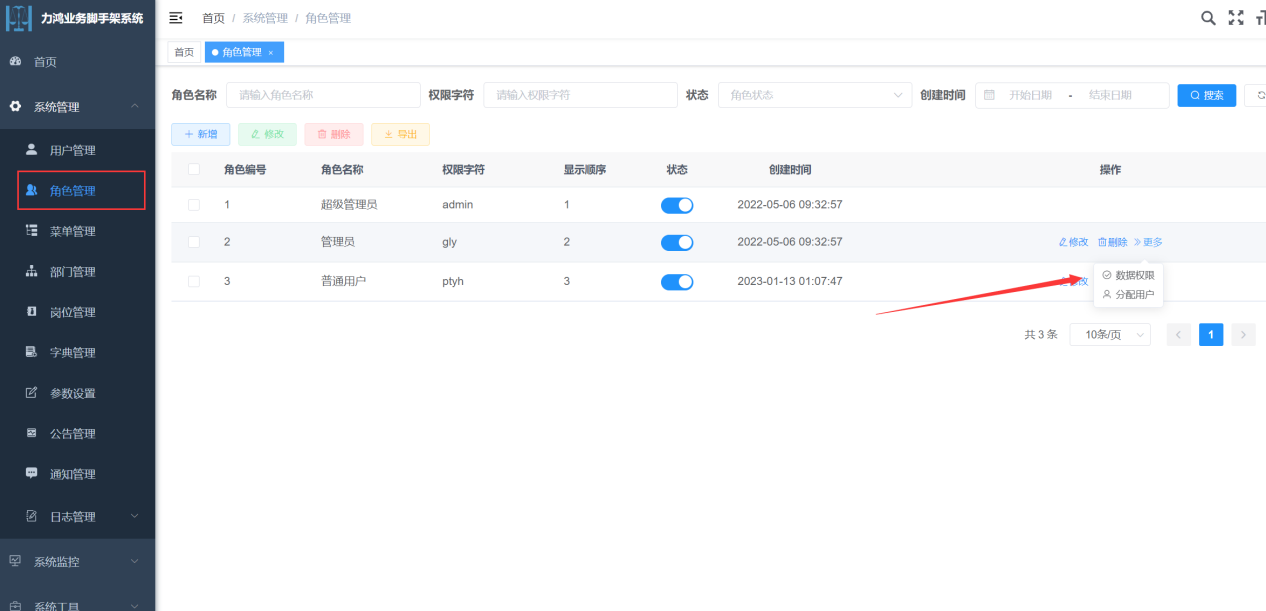
|  |
| --- |
| //定义变量  def git\_auth  //执行一套流水的区块 node {  //执行单元，会按照顺序执行 stage('拉取代码') {  //拉取Git代码  //其中credentialsID为在jenkins中配置的凭据id，需要要访问该Git项目的权限 checkout([$class: 'GitSCM', branches: [[name: '\*/master']], extensions: [], userRemoteConfigs: [[credentialsId: 'fcbe251f-545b-414b-9daa-cbd7f758363e', url: 'http://it.huaxialihong.com:9008/liangliang/jxys-erp.git']]]) }  //执行单元，会按照顺序执行 stage('打包，部署网站') { //使用NodeJS的npm进行打包 nodejs('nodejs12'){ sh ''' cd web echo "进入web" npm install -g yarn -registry=https://registry.npm.taobao.org yarn install --pure-lockfile yarn build ''' }  //====执行备份脚本===== //远程连续命令，其中configName为在jenkins中配置的远程服务器名称。//execCommand为执行目标服务器中.sh脚本  sshPublisher(publishers: [sshPublisherDesc(configName: '力鸿ERP正式', transfers: [sshTransfer(cleanRemote: false, excludes: '', execCommand: '/opt/jenkins\_shell/cpDistPorject.sh', execTimeout: 120000, flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '[, ]+', remoteDirectory: '', remoteDirectorySDF: false, removePrefix: '', sourceFiles: '')], usePromotionTimestamp: false, useWorkspaceInPromotion: false, verbose: false)])  // =====以下为远程调用进行项目部署========  //远程连续命令，其中sourceFiles为需要复制的打包后的文件。removePrefix为去//掉复制的路径。remoteDirectory为发送到目标服务器的具体路径 sshPublisher(publishers: [sshPublisherDesc(configName: '力鸿ERP正式', transfers: [sshTransfer(cleanRemote: false, excludes: '', execCommand: '', execTimeout: 120000, flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '[, ]+', remoteDirectory: '/opt/snnu\_docker\_ui/dist', remoteDirectorySDF: false, removePrefix: 'web/dist', sourceFiles: 'web/dist/\*\*')], usePromotionTimestamp: false, useWorkspaceInPromotion: false, verbose: false)])   } } |

# 定制化相关

由于本项目是基于若依二次开发的，固若依官方文档中的功能本项目也全部支持。这里提供一个[官方文档链接](http://doc.ruoyi.vip/ruoyi-vue/)

以下列出的，则是本项目中特有的定制化功能或者对若依本身架构有影响的部分：

## 数据权限

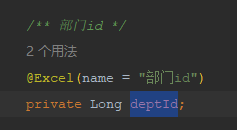


数据权限的配置在导航栏【系统管理】-【角色管理】-【更多】按钮中可以打开，里面设置了默认的几种权限配置，基本上已经能够包括了所有的情况。如果有另外的需求也可以选择自定义权限。



由于原若依框架中只集成了mybatis这一组件，没有集成mybatis-plus、mybatis-plus-join。所以框架之中最初的数据权限仅仅只能对mybatis的方式生效。后来经过改版，目前已经可以对使用mybatis-plus以及mybatis-plus-join框架的来进行数据权限控制。

**在具体使用讲解前，还有几个很重要的注意事项，首先，除了数据权限设置为【仅本人数据权限】之外，其他所以数据权限都是要基于组织机构来划分的，所以当一种数据需要开启数据权限时，则数据库表中必须加入【deptId】字段，同理实体类中也要存在。这个字段不需要service中特别撰写插入逻辑，本框架会自动注入（非开放接口）。**



### Mybatis-Plus和Mybatis-Plus-join

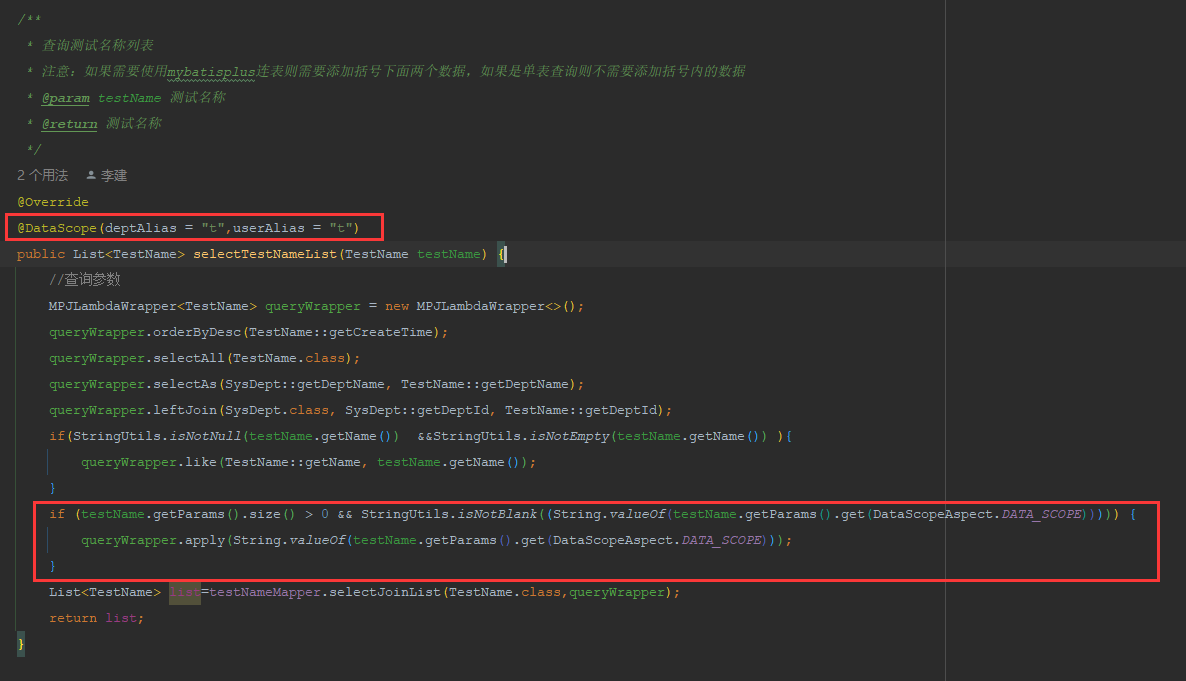
这里分为使用到链表工具（Mybatis-Plus-join）和未使用到链表工具。

* 如果未使用到链表工具，只是查询单表数据，则在service方法上只需要添加【@DataScope】注解即可，不需要指定别名。
* 如果使用到了链表工具，则在在service方法上需要添加【@DataScope(deptAlias = "t",userAlias = "t")】。其中，deptAlias 为需要通过部门数据权限查询的表别名，userAlias 为需要通过用户数据权限查询的表的别名。经过测试，该链表工具生成SQL语句时默认会定义别名为t，而左连接第一个表叫t1，第二个表t2以此类推。

完成以上步骤后，需要在service中添加如下代码即可生效（testName为主要查询实体类的对象，根据查询实体的不同需要自己替换）

|  |
| --- |
| if (*testName*.getParams().size() > 0 && StringUtils.*isNotBlank*((String.*valueOf*(*testName*.getParams().get(DataScopeAspect.*DATA\_SCOPE*))))) {  *queryWrapper*.apply(String.*valueOf*(*testName*.getParams().get(DataScopeAspect.*DATA\_SCOPE*))); } |

完整示例如下：



生成的SQL语句示例（配置为查询仅本部门权限时）：

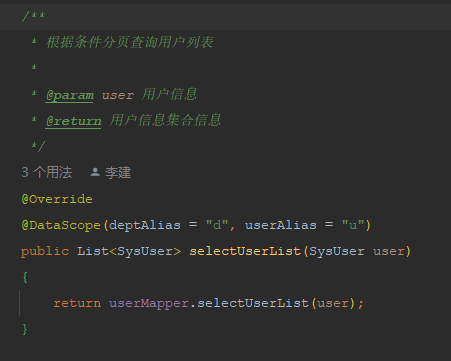
|  |
| --- |
| SELECT count(0) FROM test\_name t LEFT JOIN sys\_dept t1 ON (t1.dept\_id = t.dept\_id) WHERE t.deleted = 0 AND ((t.dept\_id = 4)) |

生成的SQL语句示例（配置为查询仅本人权限时）：

|  |
| --- |
| SELECT count(0) FROM test\_name t LEFT JOIN sys\_dept t1 ON (t1.dept\_id = t.dept\_id) WHERE t.deleted = 0 AND ((t.create\_by = test)) |

### Mybatis(xml方式)

首先，找到需要添加数据权限控制的service实现方法，在其方法上添加注解【@DataScope(deptAlias = "d", userAlias = "u")】。其中，deptAlias 为需要通过部门数据权限查询的表别名，userAlias 为需要通过用户数据权限查询的表的别名，两个参数需要跟xml中定义的表别名一致。

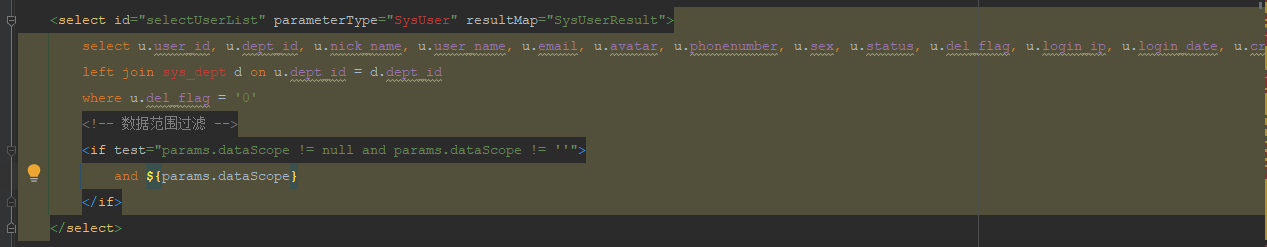


方法添加注解完成之后，xml文件中只需要关联上部门表或者用户表并加入下面这个SQL即可。

|  |
| --- |
| <!-- 数据范围过滤 -->  <if test="params.dataScope != null and params.dataScope != ''">  and ${params.dataScope}  </if> |

如果不需要以部门或者用户区分数据权限的话，也可以在注解中不填写deptAlias 或者userAlias 。比如【@DataScope(deptAlias = "d")】或者【@DataScope(userAlias = "u")】对应的xml中也是对应只关联一个即可。

如下图例子。这里因为本身就是查询用户表，所以直接给sys\_user表起别名为u即可。



完成如上2步操作，即可通过前端页面进行数据权限控制。

### 相关解释

[为了更好的理解该数据权限的实现，这里我做一个简单的解释。本质上，添加【@DataScope】注解是为了声明这个方法会进入的数据权限的控制中，也就是会被【com.jxys.framework.aspectj】目录中DataScopeAspect.java文件拦截到，利用aop的面向切面原理，在组合最后执行的SQL时根据数据权限的配置来添加几个查询参数。](mailto:为了更好的理解该数据权限的实现，这里我做一个简单的解释。本质上，添加【@DataScope】注解是为了声明这个方法会进入的数据权限的控制中，也就是会被【com.jxys.framework.aspectj】目录中DataScopeAspect.java文件拦截到，利用aop的面向切面原理，在组合最后执行的SQL时根据数据权限的配置来添加几个查询参数。)

例如：

查询仅本部门数据时：【@DataScope(deptAlias = "t")】 实际上就是在SQL的最后添加一个【AND ((t.dept\_id = 4))】。其中，4为当前登录角色所属部门

查询仅本人数据时：【@DataScope(userAlias = "t")】实际上就是在SQL的最后添加一个【AND ((t.create\_by = test))】。其中test为当前登录角色的账号

因此，才实现了自动化数据权限，可以在开发中大大简化数据权限控制的代码量和逻辑问题。

## 通知模块

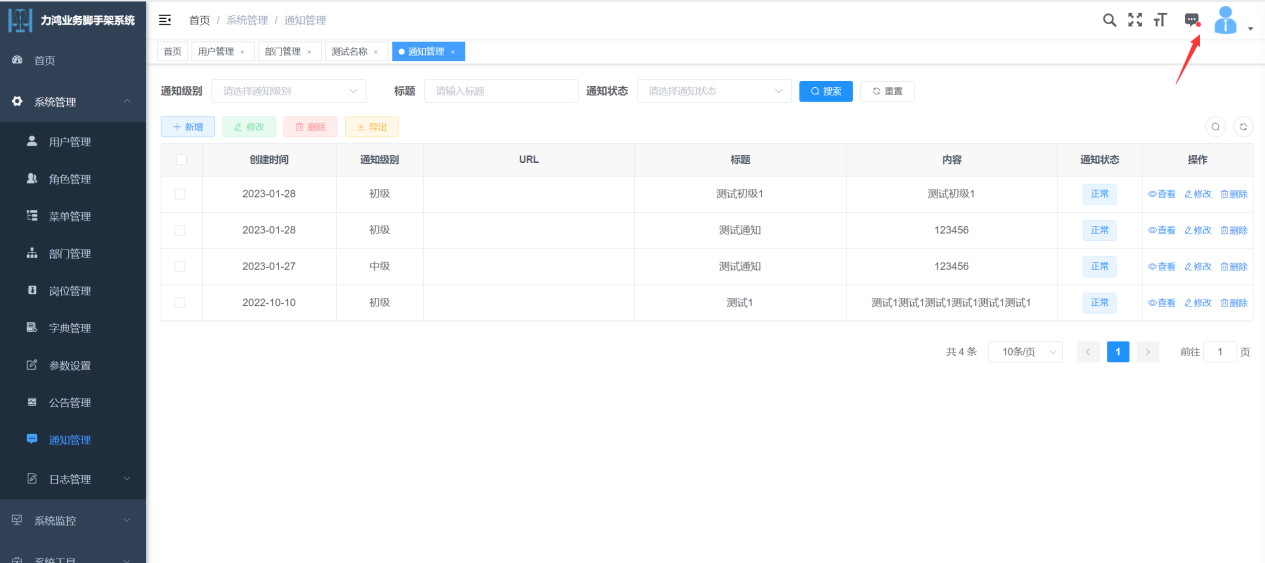
通知模块（notification）是由公司提出并开发的组件，目前是基于一个基础的webSocket来实现的，该模块利用长链接可以与用户前端进行实时交互。

目前通知模块总共定义了三种类型，即：初级、中级、高级。对用户在于前端的感受如下：

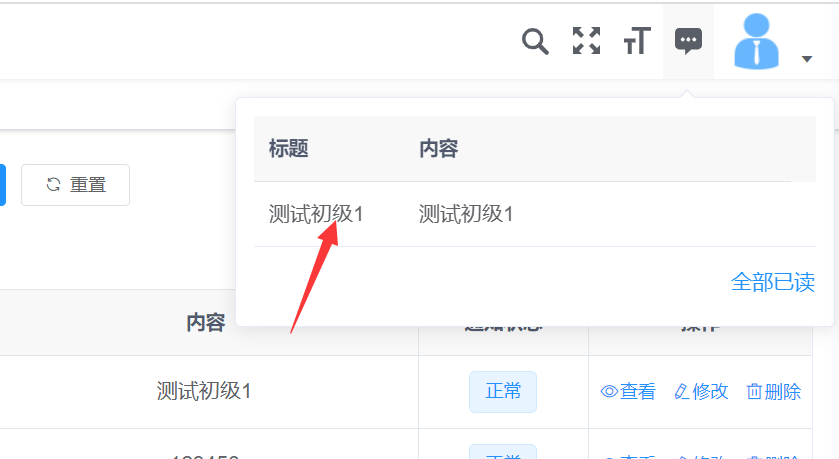
### 用户端三种类型通知的感受与反馈

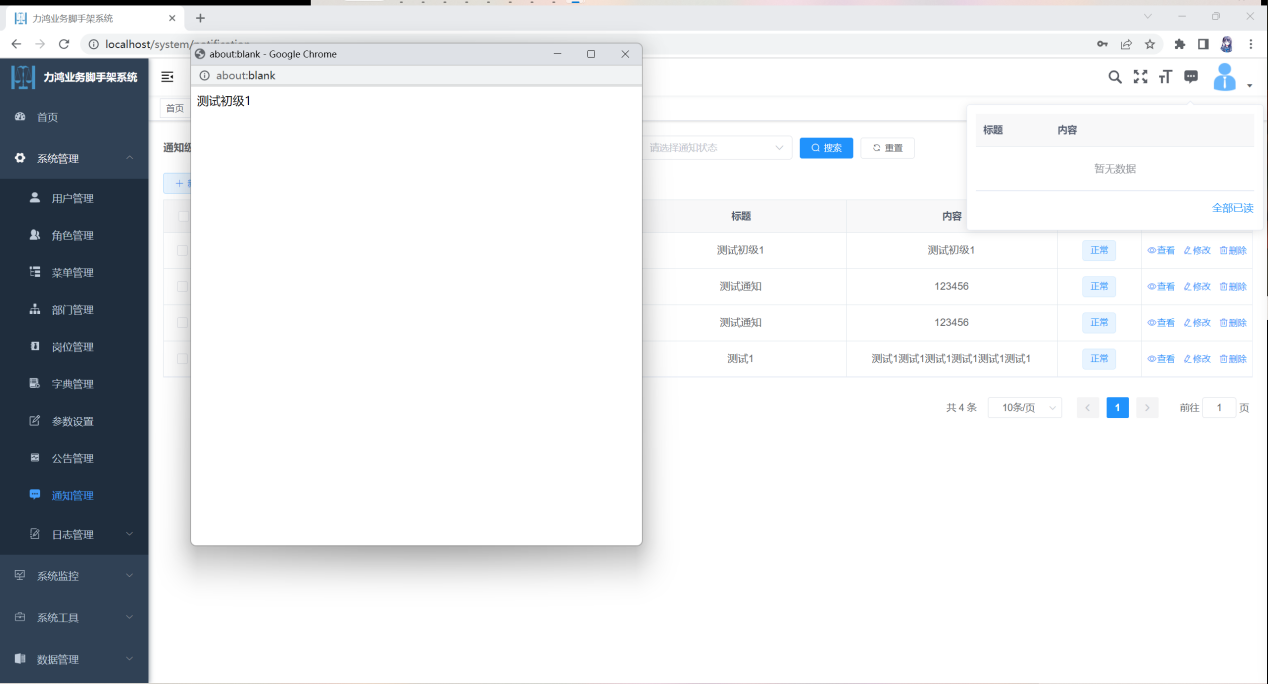
#### 初级别

主体页面不会有任何变化与感受，只有右上角消息窗格内会有红点提示通知，即使长时间不管或者刷新、关闭浏览器等操作，该红点也不会消失。



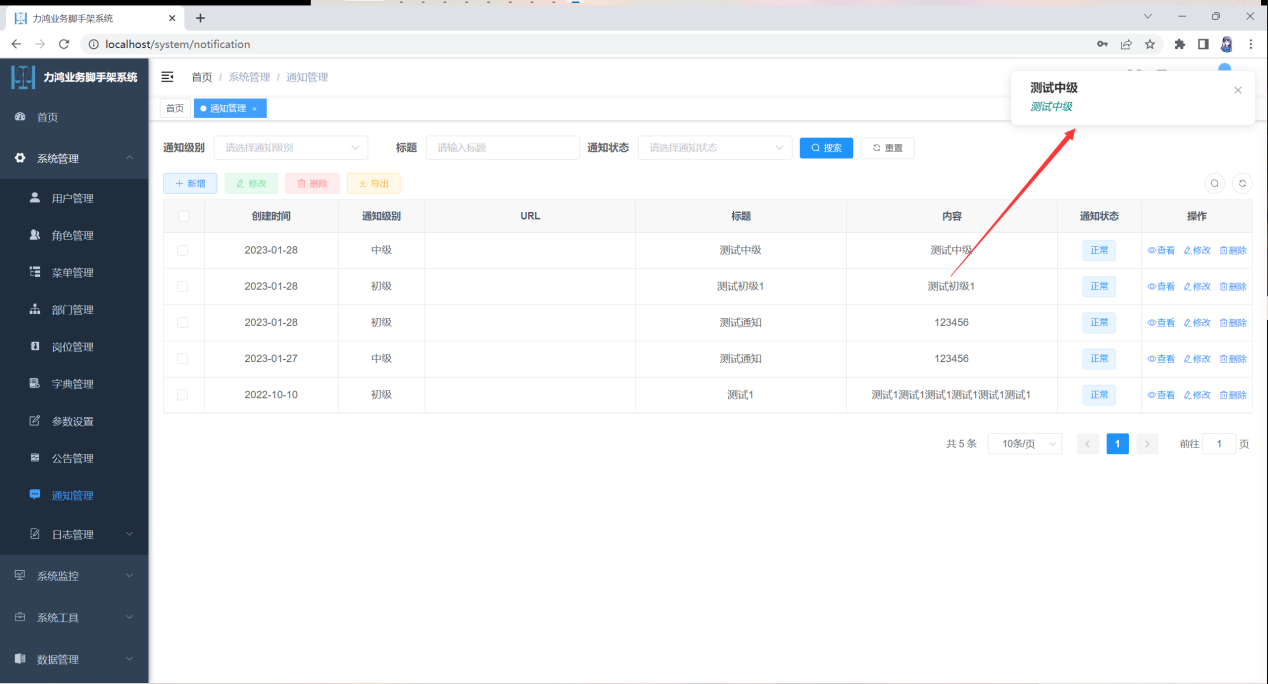
这里需要用户点击消息窗格后，选择该消息点击查看或者点击【全部已读】，消息提示的红点才会彻底消失。注意，在点击该条消息查看时会弹出新的浏览器窗口查看其内容，如果该条消息携带了URL，则还会打开URL链接，同样是以窗口的形式。





#### 中级别

中级别的消息提示会直接在主屏幕中弹出提示标题与内容。



如果用户长时间不管则会在一段时间之后消失，但消息窗格中仍然会保留，此时需要跟【初级别】的消息一样手动点击窗格查看并取消掉红点提示。

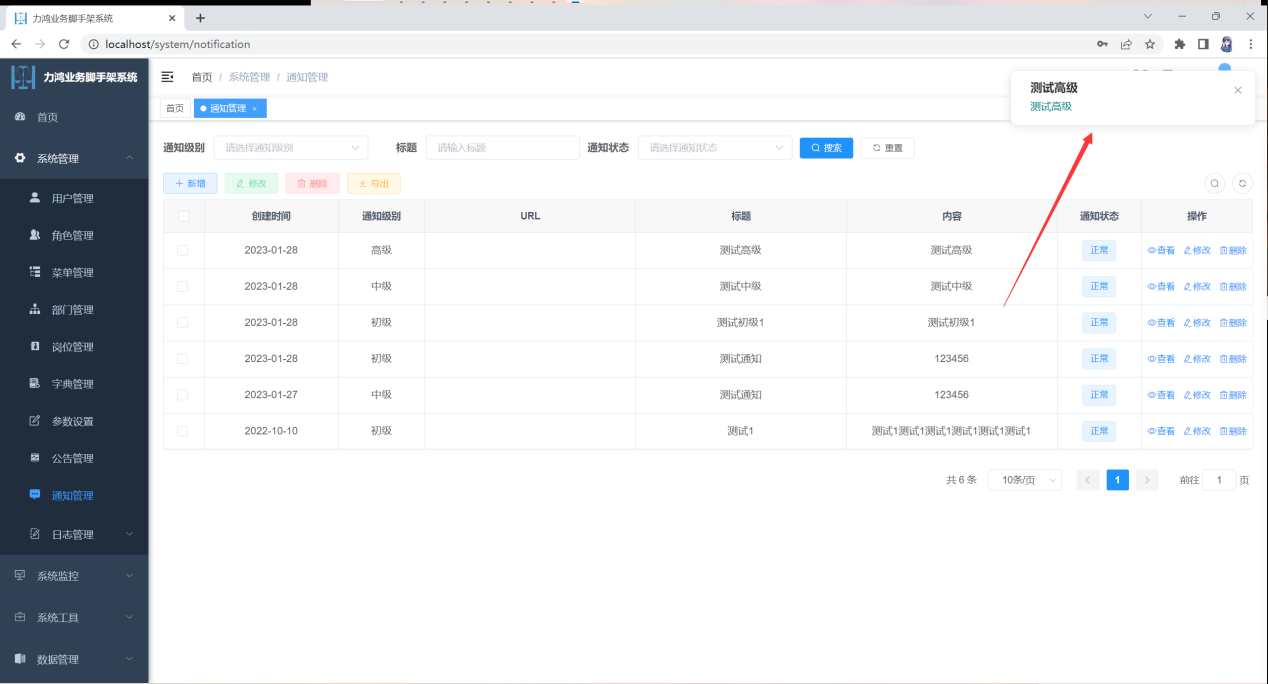


如果用户直接点击弹出的链接则可以当场直接弹出内容进行查看，并清除掉消息窗格中的消息与红点提示。

#### 高级别

【高级别】的提示与【中级别】的类似，弹出与消息窗格规则也几乎一致。唯一不同的是【高级别】的弹出提示框不会自动消失，即使刷新浏览器也不会关闭该弹框，必须要手动点击查看或者点击叉号进行关闭。

如果用户点击了叉号进行关闭，则会保留到消息窗格中，此时不会产生红点提示。

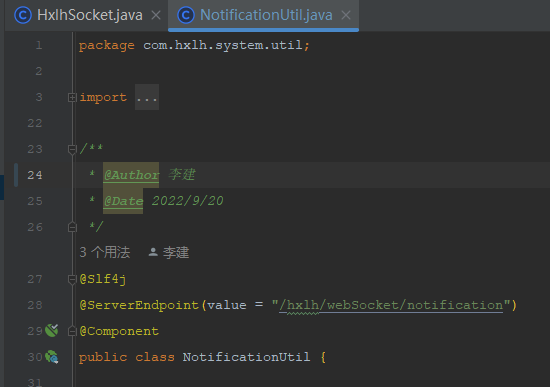


### 关于创建三种通知

创建这三种通知的方式分为两种，一种是由后端代码逻辑依靠系统生成的消息。另一种则是通过前端管理页面来手动控制生成的消息提示。

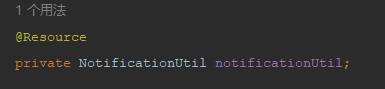
#### 系统自动生成

在建行一世脚手架系统项目中，提供了由直接对接前端的websocket工具【NotificationUtil】。利用该工具，在service的开发时，可以直接引用工具中的sendMessageOne方法或者sendMessageList方法进行对前端的消息发送。



传入的【SysNotification】则是整个消息体的对象类（.java），里面的属性中记录了消息的状态，标题，内容链接和通知级别等。通知级别可以在【Constants】中找到相关定义。

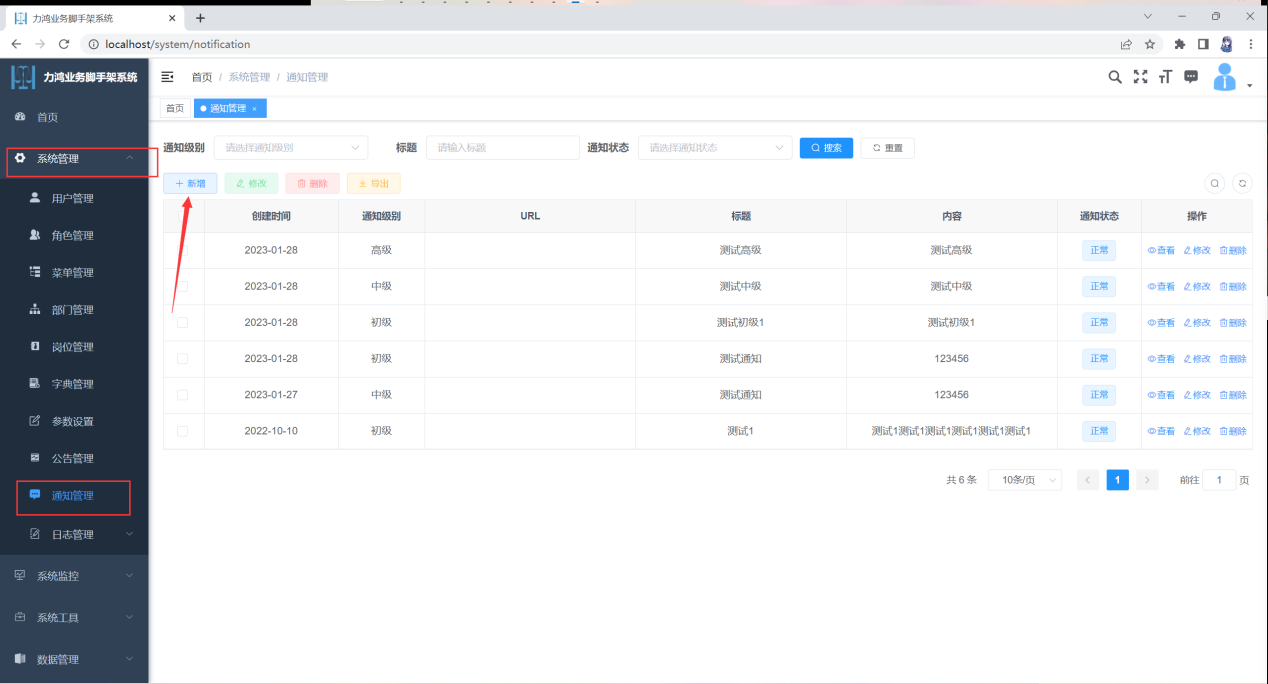
值得注意的是，该工具类不同与常用工具类，需要先注入到service中，在进行使用。



以上，则是最基础的使用方法。在实际使用中，我们肯定需要记录每一次消息通知的状态与是否已读等信息，所以我们还需要配合使用到【SysNotificationUser】类。该类相当于记录一次完整的通知流程，关联【SysNotification】通知内容。记录了通知对象、通知状态、以及是否发送、是否已读。所以我们在使用消息通知体系时，需要在逻辑中插入【SysNotificationUser】类来记录本次通知。

如果以上还有一些不太理解的地方，可以参考【SysNotificationServiceImpl】类中的【insertSysNotification】方法来查看具体使用。

#### 前端手动创建



在前端登录有权限的账后，点击导航栏的【系统管理】-【通知管理】-【新建】按钮来创建一条新的消息通知。在打开的弹出框中，选择通知级别，填写URL、标题、内容等数据，之后选择需要通知到的用户，点击确定即可实时发出。



**注意，在管理页面上：**

**点击查看可以跟踪查看具体对哪些用户发出了通知，哪些用户已读。**

**点击修改后该条通知会重新再次发送一遍。**

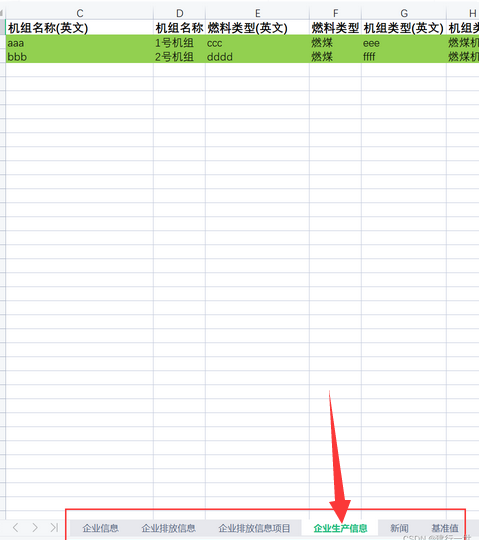
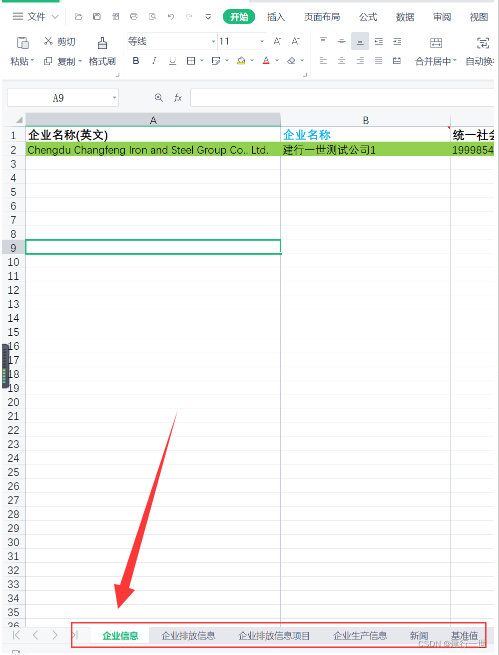
## 多sheet页Excel导出

原若依框架有支持单sheet页的Excel导出，使用【@Excel】注解配合【ExcelUtil.java】工具类即可实现导出功能。

但是，对于多sheet页的导出，ExcelUtil的实现就会显得很吃力。于是在系统中加入了【ExcelExp.java】与【ExcelExportUtil.java】来实现对多sheet页Excel的导出。这里我会举个在双碳项目中使用到的例子，简化一下来介绍该功能。

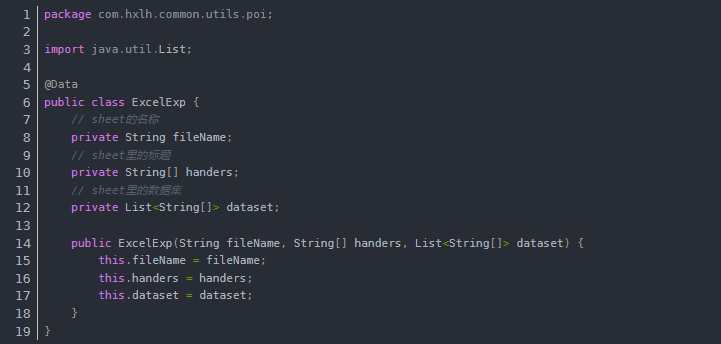
### 实现效果

根据任务需要，导出一份有关企业的Excel数据，而且要通过多sheet页面来完成。这里咱们只认为有两个sheet页面，即为：【企业信息】和【企业生产信息】这两个页面，



### 工具介绍

* 创建一个ExcelExp类，来保存每个sheet也得表头和数据



* 创建ExcelExportUtil类用来制作样式和解析导出包含多ExcelExp的数据集

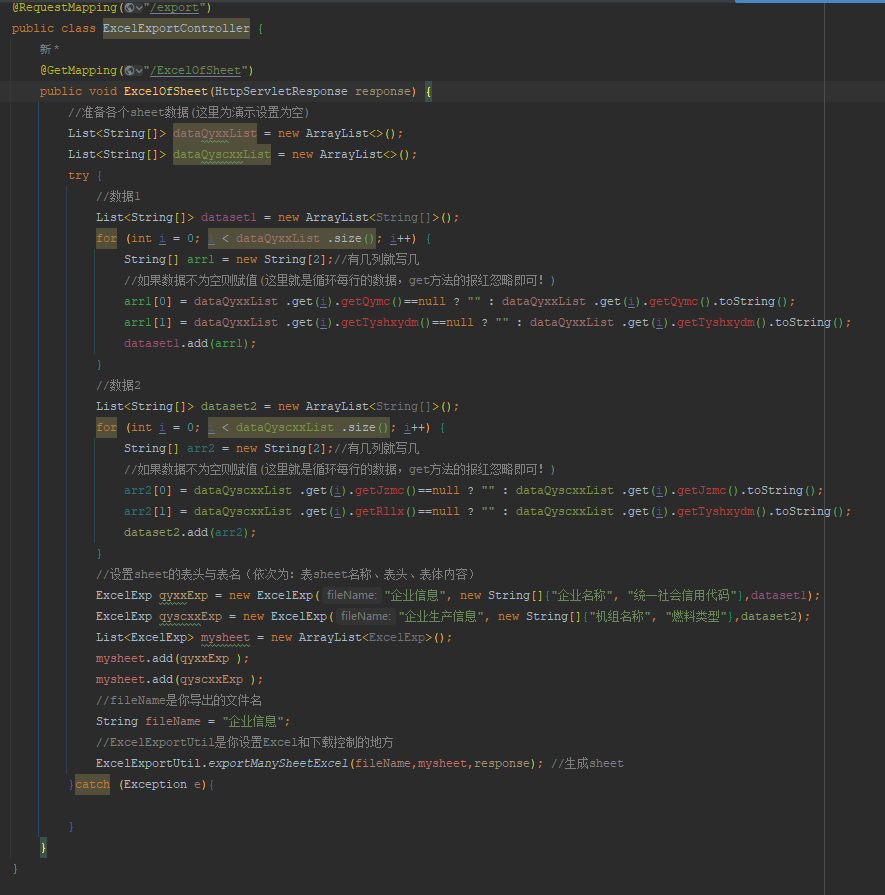


**以上内容均已经集成到了项目中，无需在手动添加！！**

### 整合数据，调用工具导出方法

这里需要准备三组数据，每个sheet的表头、sheet名称、表内容。然后将他们打包到 ExcelExp类中存入 List<ExcelExp>中，之后调用exportManySheetExcel方法进行导出。

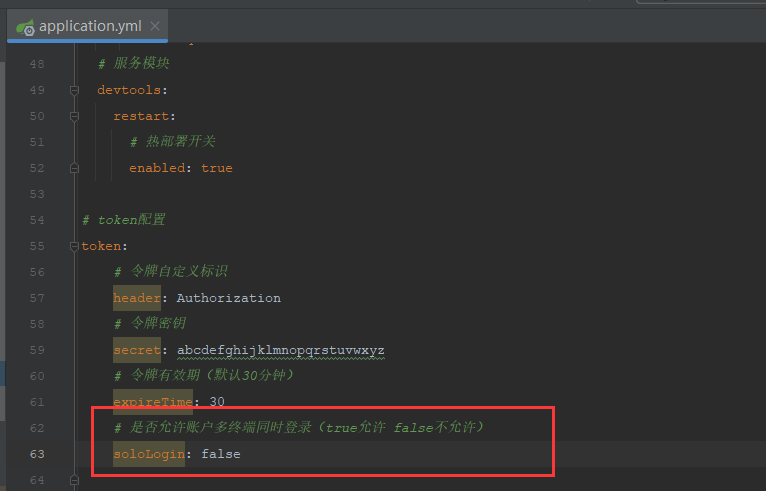
整理的解释我就放在注解中了，这样大家看着也会更直观一些。



到这里一个导出多sheet页Excel的整体后端逻辑就完成了。

## 限制多端登录

多端登录目前已经开发了该功能，现在只需要调整application.yml配置文件中的token.soloLogin即可。

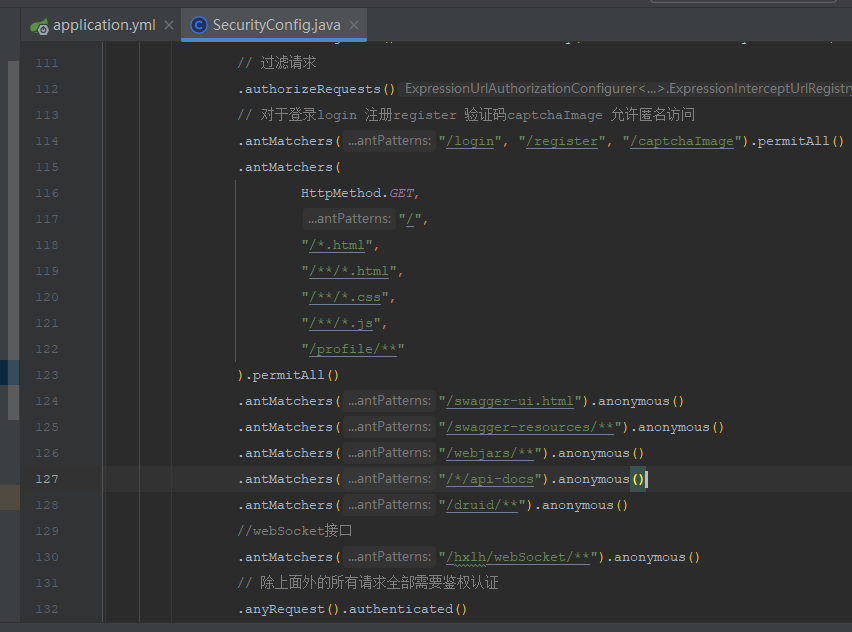


## 设置接口登录直接访问（不鉴权）

### （1）后端

**方法1：**

在 SecurityConfig 中设置httpSecurity 配置匿名访问。其中使用 permitAll() 方法所有人都能访问，包括带上 token 访问，使用 anonymous() 所有人都能访问，但是带上 token 访问后会报错。



**方法2：**

在对应的方法或类上面使用@Anonymous注解。匿名访问的方法上面@PreAuthorize权限注解也需要去掉，因为已经使用匿名访问了，权限自然也不需要去验证了。

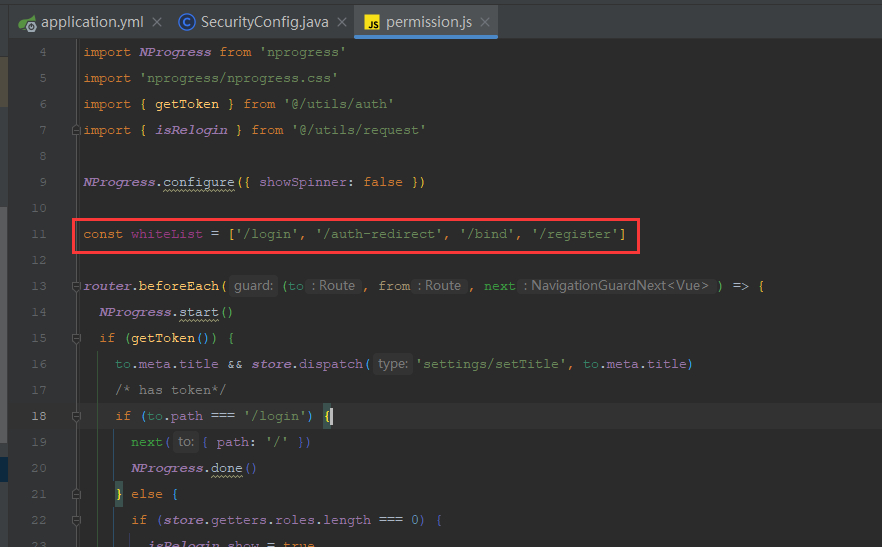


**方法3：**

在 SecurityConfig 中设置httpSecurity 配置匿名访问

### 前端

前端页面的话，尽量不要进行非鉴权的登录，如果有需要利用其他鉴权方式的话。可以直接在【src/permission.js】配置whiteList属性白名单即可



## 定时任务

### 介绍

定时任务模块基于原本的若依框架有着非常强大的管理能力，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。

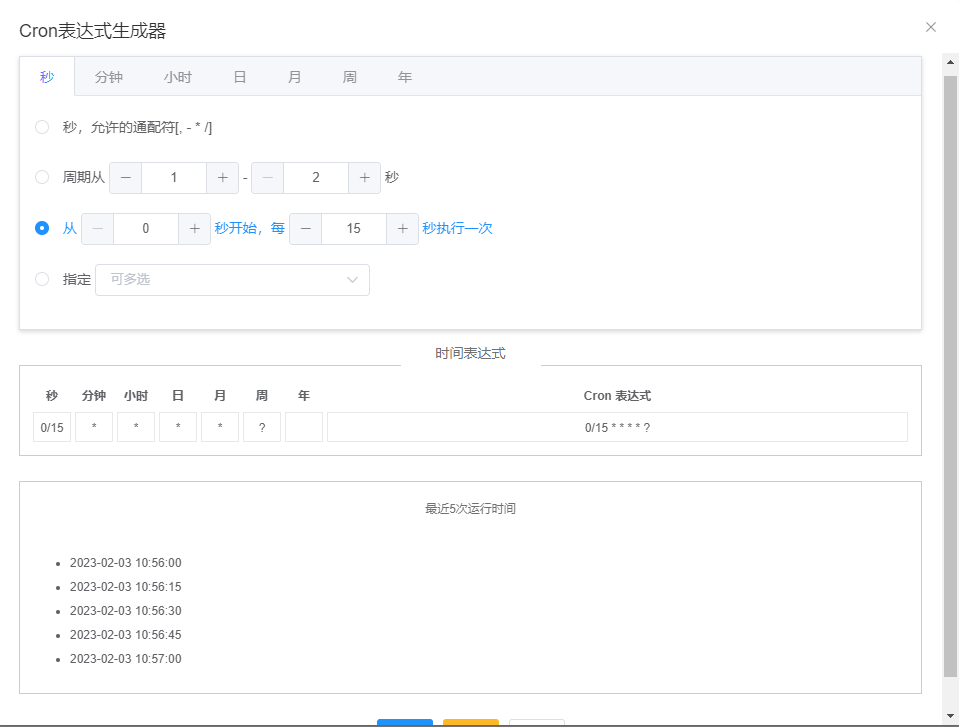
### 使用流程

在原若依框架中提供了两种定义方式，一种是【bean】方式一种是【class】方式，因为建行一世脚手架系统为前后端分离项目，又是基于springboot开发的后端，所以比较推荐使用【bean】方式来完成。下面就介绍具体使用方式。

* 后台添加定时任务处理类，需要添加对应Bean注解@Component或@Service。



* 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：可查询官方cron表达式介绍（也可以直接使用表达式生成工具）



* 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



* 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。
* 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)）

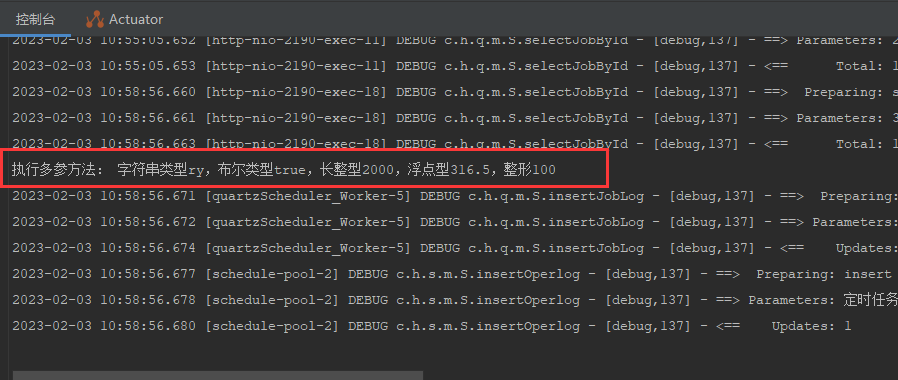
长整型（需要L标识 如：ryTask.ryParams(2000L)）

浮点型（需要D标识 如：ryTask.ryParams(316.50D)）

整型（纯数字即可）

### 执行成功

已多参数请求为例，完成上面流程之后，执行一次多参数的定时任务。成功之后的打印信息：



## 代码生成器

代码生成器的使用也比较广泛，本系统已经做了相对于mybatisplus的适配和集成（原若依只使用mybatis）。其他的只说几个值得注意的事项吧。

1. 代码生成器生成之后目前也会有mapper.xml文件，这个还是选择了保留，给需要的人做个参考。但是基本的增删改查都已经以mybatisplus的方式生成了，所以此文件没有实际意义，可以不放入程序。
2. 在代码生成器导入一个数据库表之后，推荐进入【编辑】选项中的【字段信息里】，补充相关信息与查询条件删减，因为默认几乎以全部的查询条件来生成代码，这里值得注意。
3. 在代码生成器【编辑】-【生成信息】中，【生成模块名】与【生成业务名】，这两个文本框中的内容组合，就是vue实际页面的访问路径，也就是从【view】文件开始的路径。

## 数据及其组件库

### 在系统中，给出了几个相关组件的使用示例。

1. 特效组件：前端页面【系统工具】-【组件工具】
2. 百度搜索：前端页面【系统工具】-【百度相关】
3. Websocket：前端页面【系统工具】-【WebSocket】

如果觉得系统臃肿可以手动删除前端这三个页面文件及其权限，后端需要删除【Jxyssocket.java】文件即可。

以上操作不会对系统其他任何地方造成影响。

### 在系统中，给出了一些数据的示例。

（1）地图数据：前端页面【数据管理】-【地区地图】

（2）图标数据：前端页面【数据管理】-【icon图标】

该示例中的内容均为实际数据，如果不需要，也可以手动去删除后端【jxys-data】整个聚合项目。同时前端需要删除前端页面【系统工具】-【组件工具】中的两个地图view以及前端的权限数据。

以上操作不会对系统其他任何地方造成影响。

# 常用功能及注意点

**在常用功能及注意点中，列出了一些项目开发中，重点使用到的技术问题或者高频使用到容易出错的地方。**

## 系统实体类

1. 若依自带的基础实体类及县对应的表解构，主键并不是id，例如【SysConfig】的主键为configId，【SysUser】的主键为userId。

这里知道这一点就行，不必过多介意，后续开发者自己创建的实体类还是推荐直接使用生成的id为主键。这样更加适配mybatisplus的各种...byId方法。

1. 系统自带的基础实体类相对应的mapper中也并不存在继承【MPJBaseMapper】或者【BaseMapper】，因为操作这些数据时，无法使用mybatisplus的功能。

## 代码生成器

1. 代码生成器的模板存放于【jxys-vue/jxys-generator/src/main/resourcess/vm】中，编辑该模板可以控制生成需要的代码。
2. 如果有某些字段需要定义到实体类的父类【BaseEntity】中，则需要控制代码生成器不生成到实体类里，这时，我们需要编辑【GenTableColumn.java】中的【isSuperColumn()】方法，将父类中定义的字段添加进去。  
    【isSuperColumn()】中的名单用于避免生成多余Domain属性，若某些属性在生成页面时需要用到不能忽略，则同样在【GenTableColumn.java】中的【isUsableColumn()】方法中加入进去即可。
3. 需要修改字段类型对应的前端组件时，可以在【GenUtils.java】中进行配置。

## 系统码表

如果遇到需要使用某些码表的时候，如：性别、民族、各种状态属性。建议在系统的字典管理中创建。也就是前端导航栏的【系统管理】->【字典管理】。在这里定义的码表有两个优势：

一个是能够进入系统自带的完整逻辑中，入载入Redis、可编辑、可管理。

二个是能够配合代码生成器直接生成基于该码表的下拉框。这里使用的组件是前端components中的【DictTag】，该组件可以对应字典管理中创建码表的类型，生成各种不同颜色的组件，数据也会相对变化。十分方便。

当然，如果是需要用户来手动维护的业务类型字典表，就不要创建到这里了，还是单独设置成为页面级的字典表比较合适。

## 验证码使用

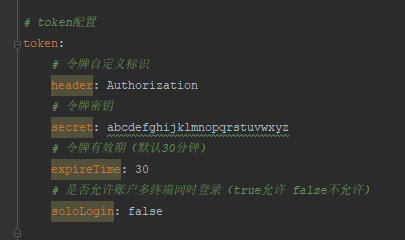
1. 开启验证码需要在前端【系统管理】->【参数设置】中将【sys.account.captchaEnabled】由false改为true。
2. 切换验证码的类型需要在后端配置文件（application-\*\*\*.yml）中，将Jxys:captchaType的数值进行修改：验证码类型 math 数组计算 char 字符验证
3. 验证码默认20s超时，如果需要调整，可以在前端目录（jxys-ui/src/api/login.js）中修改【getCodeImg()】方法

## 登录相关

1. 登录失败有次数限制，密码最大错误次数和密码锁定时间都可以在【jxys-admin】项目中的【application.yml】里进行配置



1. Token过期时间，也就是登录系统后最大的不请求停留时间也是在【jxys-admin】项目中的【application.yml】里进行配置



1. 因为注册功能算是危险操作，需要很强的逻辑判断，固目前取消的注册的使用，如果需要用到，可以在前端【系统管理】->【参数设置】中将【sys.account.registerUser】由false改为true。并在vue页面【login.vue 】data中的register: false改成true。
2. 另外值得注意的一点是，在前端登录的实际调用方法是执行的vuex中的方式，也就是说，实际调用接口的代码在【jxys-ui/src/store/modules/user.js】中，获取用户信息及登出也在这里。

## 流量限制控制

当我们需要对某一个接口进行限流时，例如一分钟请求一百次。则可以在接口处添加【@RateLimiter(count = 100, time = 60)】来进行控制。【count】为允许请求次数。【time 】为这个时间之内。

其他的相关配置参数可以进入到【RateLimiter】的注解中进行修改，位置在【com/ruoyi/common/annotation/RateLimiter.java】中。

## 前端相关

1. 前端已经集成了echarts组件，无需重复集成。只需要在使用到的页面引入【import \* as echarts from 'echarts';】即可。
2. 前端请求工具的核心为【jxys-ui/src/utils/request.js】，在这里你可以配置相关的请求拦截与响应拦截等操作
3. 前端工具类的主入口为【jxys-ui/src/utils/Jxys.js】，在这里你可以配置自己的各种方法工具，当然你也可以写入到其他js中在引入到Jxys.js里，这样会比较方便页面引入与合作开发。
4. 上传头像功能会将头像上传到服务器主机上并不是云端OSS，所以会出现开发环境与生产环境不互通的情况。
5. 前端框架是依据【https://github.com/PanJiaChen/vue-element-admin】开发而来，因为很多使用的内容可以在这里找到。
6. 前端UI框架为【https://github.com/ElemeFE/element】使用文档地址我也提供一下吧【https://element.eleme.cn/2.15/#/zh-CN/component/installation】
7. 如何新增系统图标？如果在【jxys-ui/src/assets/icons/svg】中都没有找到合适的图标的话，可以到 iconfont.cn (https://www.iconfont.cn/)上选择并生成自己的业务图标库，再进行使用。或者其它 svg 图标网站，下载 svg 并放到文件夹之中就可以了。下载完成之后将下载好的 .svg 文件放入 @/icons/svg 文件夹下之后就会自动导入。
8. 通常情况下，在启动本地项目时，默认都是http协议，但是有时候测试网站要求我们的协议是https，那么可以配置vue.config.js（jxys-ui/vue.config.js）中的devServer,让其在启动项目的时候，默认是https协议。

|  |
| --- |
| module.exports = {  ......  devServer: {  https: true,  ......  },  } |