Jeecg-Boot 技术文档 V1.1

项目介绍

> Jeecg-boot 一个全新的版本，采用前后端分离方案，提供强大代码生成器的快速开发平台
>
> 前端页面代码和后端功能代码一键生成，不需要写任何代码，保持jeecg一贯的强大！！
>
> 技术交流QQ群： 284271917
>
> 在线演示： http://boot.jeecg.org
>
> 源码下载：https://github.com/zhangdaiscott/jeecg-boot
>
> 视频教程：https://pan.baidu.com/s/1Il0TS50I70vH1AG1y40wtw 提取码：hok5

技术架构：

```
后端技术： SpringBoot_2.0.3.RELEASE + Mybatis-plus_3.0.6 + Shiro_1.4.0-RC2 +
Jwt_3.4.1
        + Swagger-ui + Redis
前端技术： Ant-design-vue + Vue + Webpack
其他技术： Druid（数据库连接池）、Logback（日志工具）、poi（Excel工具）、
        Quartz（定时任务）、lombok（简化代码）
项目构建： Maven、Jdk8
```

前端开发必读文档：

前端UI组件： Ant Design of Vue

https://vuecomponent.github.io/ant-design-vue/docs/vue/introduce-cn

报表UI组件：viser-vue

https://viserjs.github.io/demo.html#/viser/bar/basic-bar

VUE基础知识：

https://cn.vuejs.org/v2/guide

捐赠

如果觉得还不错，请作者喝杯咖啡吧

图片地址：https://static.oschina.net/uploads/img/201903/08155608_0EFX.png

效果抢先看:

1. 系统效果

图片地址：https://static.oschina.net/uploads/img/201902/25154007_icdX.png

图片地址：https://static.oschina.net/uploads/img/201904/14155402_AmlV.png

图片地址：https://static.oschina.net/uploads/img/201904/14160623_8fwk.png

图片地址：https://static.oschina.net/uploads/img/201904/14160633_u59G.png

图片地址：https://static.oschina.net/uploads/img/201904/14160643_kCJ7.png

图片地址：https://static.oschina.net/uploads/img/201904/14160650_fcgw.png

图片地址：https://static.oschina.net/uploads/img/201904/14160657_cHwb.png

图片地址：https://static.oschina.net/uploads/img/201904/14160705_NAJn.png

图片地址：https://static.oschina.net/uploads/img/201904/14160751_bsO9.png

图片地址：https://static.oschina.net/uploads/img/201904/14160801_2AhS.png

图片地址：https://static.oschina.net/uploads/img/201904/14160813_KmXS.png

图片地址：https://static.oschina.net/uploads/img/201904/14160828_pkFr.png

图片地址：https://static.oschina.net/uploads/img/201904/14160834_Lo23.png

图片地址：https://static.oschina.net/uploads/img/201904/14160842_QK7B.png

图片地址：https://static.oschina.net/uploads/img/201904/14160849_GBm5.png

图片地址：https://static.oschina.net/uploads/img/201904/14160858_6RAM.png

图片地址：https://static.oschina.net/uploads/img/201904/14160905_RGJ5.png

图片地址：https://static.oschina.net/uploads/img/201904/14160917_9Ftz.png

图片地址：https://static.oschina.net/uploads/img/201904/14160926_PUDV.png

图片地址：https://static.oschina.net/uploads/img/201904/14160935_Nibs.png

图片地址：https://static.oschina.net/uploads/img/201904/14160947_gfoN.png

图片地址：https://static.oschina.net/uploads/img/201904/14160957_hN3X.png

图片地址：https://static.oschina.net/uploads/img/201904/14161004_bxQ4.png

图片地址：https://static.oschina.net/uploads/img/201904/14161013_zW5n.png

1. 代码参考

图片地址：https://static.oschina.net/uploads/img/201901/07154721_WS18.png

图片地址：https://static.oschina.net/uploads/img/201901/07154651_bTji.png

新手学习思路【零基础入门】

Jeecg-Boot学习思路（跟着我们零基础学习）

> 1. 开发环境搭建 http://jeecg-boot.mydoc.io/?t=344848
> 开发工具 ： https://pan.baidu.com/s/1kFIJcn5GSlSJWAQKeCowrg 提取码：ilmc
> 2. 项目如何启动 http://jeecg-boot.mydoc.io/?t=344849

3. JeecgBoot学习视频 https://pan.baidu.com/s/1Il0TS50I70vH1AG1y40wtw 提取码：hok5

4. 代码生成器使用 http://jeecg-boot.mydoc.io/?t=344854

5. 常见问题贴：

http://www.jeecg.org/forum.php?mod=viewthread&tid=7816&page=1&extra=#pid21237

> 6. 开发必看技术文档
>
> Ant Design of Vue | https://vue.ant.design/docs/vue/introduce-cn
>
> Vue | https://cn.vuejs.org/v2/guide
>
> 源码下载 | https://github.com/zhangdaiscott/jeecg-boot
>
> Jeecg-Boot | http://jeecg-boot.mydoc.io
>
> 7. 基础知识学习（Vue全家桶、Springboot） 参考下面《基础知识学习视频》

《基础知识学习视频》

两套Springboot视频(建议第一套)
链接：https://pan.baidu.com/s/1KlfH96wGNRN8waPIurQ6yg 提取码：tu6m
链接：https://pan.baidu.com/s/11Z0iLW9o-W4-4tYHXlDO9A 提取码：7kz2

两套vue视频，前后端分离

1.Vue基础知识视频
链接：https://pan.baidu.com/s/1r69bFZ0_N2-g4XNxEqDtfg 提取码：gt81

2.Vue高级视频教程
链接：https://pan.baidu.com/s/1W361MyZDe0ntzJVo6lw5rQ 提取码：16ne
链接：https://pan.baidu.com/s/1MqN5Ith1nUbp6T-jRDhgIA 提取码：i3m6

开发环境准备

技术点

需要掌握的基础知识

序号	知识点	资料

1	Npm 命令	http://www.runoob.com/nodejs/nodejs-npm.html
2	Node.js 入门	http://www.runoob.com/nodejs/nodejs-tutorial.html
3	Vue	https://cn.vuejs.org/
4	ES6	https://blog.csdn.net/itzhongzi/article/details/73330681
5	Vue全家桶	Webpack 、axios、Vue router、Vuex、Vue Loader、Vue cli
6	Springboot	
7	Mybatis-plus	https://mp.baomidou.com
8	Shiro	
9	Yarn	建议，比npm更快

开发工具安装

目录索引：

JeecgBoot采用前后端分离的架构，官方推荐开发工具
前端开发： Webstrom 或者 IDEA
后端开发： Eclipse安装lombok插件 或者 IDEA
开发工具下载：https://pan.baidu.com/s/1tZmFuViGz5IwHhzmA-FN6A 提取码：frya

图片地址：https://static.oschina.net/uploads/img/201904/14220711_9Wvq.png

图片地址：https://static.oschina.net/uploads/img/201902/25100424_D6wm.png

后端开发工具

序号	工具	参考
1	eclipse安装lombok插件	https://blog.csdn.net/qq_25646191/article/details/79639633
2	Eclipse自定义皮肤主题	https://blog.csdn.net/StillOnMyWay/article/details/79109741
3	Eclipse常用快捷键	https://blog.csdn.net/zhangdaiscott/article/details/52790087

前端开发工具

序号	工具	描述	参考
1	Nodejs/Npm安装	JavaScript运行环境，此处使用到它的包管理器npm	http://www.jianshu.com/p/03a76b2e7e00
2	Yarn安装	下载包工具	https://yarnpkg.com/zh-Hans/docs/install
3	WebStorm安装与使用	WEB前端开发工具	https://blog.csdn.net/u011781521/article/details/53558979

配置Nodejs镜像

```
npm config set registry https://registry.npm.taobao.org --global
npm config set disturl https://npm.taobao.org/dist --global

yarn config set registry https://registry.npm.taobao.org --global
yarn config set disturl https://npm.taobao.org/dist --global
```

WebStorm-2018.1.3 开发工具入门配置

序号	标题	链接
1	WebStorm安装与使用	https://blog.csdn.net/u011781521/article/details/53558979
2	webstorm 2018 激活破解	https://blog.csdn.net/q3585914/article/details/79853404
3	修改webstorm主题	https://blog.csdn.net/master_yao/article/details/50675454
4	Webstorm切换快捷键风格（Webstorm快捷键与eclipse对比介绍）	https://blog.csdn.net/gsying1474/article/details/52036443
5	WebStorm SVN用法	https://blog.csdn.net/hysh_keystone/article/details/52013789
6	‘svn’不是内部或外部命令问题解决	https://blog.csdn.net/mitea90/article/details/19075673
7	设置webstorm的vue新建文件模板(后面篇章)	https://blog.csdn.net/diligentkong/article/details/75040651
8	WebStorm卡顿拉取svn慢解决	https://blog.csdn.net/WYA1993/article/details/84671501

前端Webstorm开发界面：

图片地址：https://static.oschina.net/uploads/img/201901/07163141_8U71.png

后端Eclipse开发界面：

图片地址：https://static.oschina.net/uploads/img/201901/07163150_Oeie.png

开发环境搭建

目录索引：

第一部分： 前端开发环境搭建

一、安装开发工具

安装nodejs、webstrom、yarn,安装方法参照【开发环境准备】-【开发工具】(http://jeecg-boot.mydoc.io/?t=344847)

二、导入项目

1、使用webstrom导入项目

（1）前端工程ant-design-jeecg-vue

（2）Webstrom打开项目

图片地址：https://static.oschina.net/uploads/img/201904/14220126_qkgS.png

2、本地开发构建运行

（1）执行命令 yarn install 下载项目依赖

图片地址：https://static.oschina.net/uploads/img/201904/14220404_newr.png

（2）项目依赖的模块下载完成，则项目构建完成

第二部分： 后端开发环境搭建

一、安装开发工具

安装jdk、eclipse、redis等，安装方法参照【开发环境准备】-【开发工具】(http://jeecg-boot.mydoc.io/?t=344847)

二、导入项目

1、使用eclipse导入项目

进入eclipse资源管理库，添加后台项目jeecg-boot svn地址：

下载项目

图片地址：https://static.oschina.net/uploads/img/201901/30121652_3lNW.png

图片地址：https://static.oschina.net/uploads/img/201901/30121701_2mDO.png

下载后项目是java的工程需要移除有再重新import 成maven工程

图片地址：https://static.oschina.net/uploads/img/201901/30121847_Q3oG.png

重新导入项目后，maven会自动下载项目依赖，至此后台项目环境搭建完成

如何启动项目

目录索引：

一、后端项目启动

（1）执行数据库初始化sql

sql文件地址：jeecg-boot/docs/db/jeecg-boot_1.1.0-20190415.sql

（2）开发模式配置（/src/main/resources/application-dev.yml）

项目名称、端口号配置（请默认即可不需要修改）

```
server:
```

```
    port: 8080
    servlet:
      context-path: /jeecg-boot
```

端口号是8080，项目名称是jeecg-boot

本地后台接口地址：http://localhost:8080/jeecg-boot

不能直接访问，会提示TOEKN无效错误。

数据库配置 ：

```
spring:
  datasource:
    dynamic:
      datasource:
          #主数据源
          master:
            url: jdbc:mysql://127.0.0.1:3306/jeecg-boot?characterEncoding=UTF-
8&useUnicode=true&useSSL=false
            username: root
            password: root
            driver-class-name: com.mysql.jdbc.Driver
```

redis配置：（配置redis的host和port）

```
  #redis 配置
  redis:
    database: 0
    host: 127.0.0.1
    lettuce:
      pool:
        max-active: 8   #最大连接数据库连接数,设 0 为没有限制
        max-idle: 8     #最大等待连接中的数量,设 0 为没有限制
        max-wait: -1ms  #最大建立连接等待时间。如果超过此时间将接到异常。设为-1表示无
限制。
        min-idle: 0     #最小等待连接中的数量,设 0 为没有限制
      shutdown-timeout: 100ms
    password: ''
    port: 6379
```

（3）启动redis服务

（4）以上配置完成后，即可启动后台项目

本地启动：

找到类/src/main/java/org/jeecg/JeecgApplication.java，右键执行

二、前台项目启动

（1）使用命令yarn install 下载项目依赖

（2）配置后台接口地址

[1]. public/index.html （开发环境、正式发布）

```
window._CONFIG['domianURL'] = 'http://localhost:8080/jeecg-boot';
window._CONFIG['imgDomainURL'] = 'http://localhost:8080/jeecg-
boot/sys/common/view';
```

[2]. vue.config.js（仅开发环境配置）

图片地址：https://static.oschina.net/uploads/img/201904/18151657_IdiE.png

（3）启动项目

调出Show npm Scripts 功能

找到项目目录下文件package.json文件，鼠标右键选择Show npm Scripts

图片地址：https://static.oschina.net/uploads/img/201901/30121009_NDhQ.png

图片地址：https://static.oschina.net/uploads/img/201901/30121019_YZdg.png

点击命令：serve　即可启动项目

图片地址：https://static.oschina.net/uploads/img/201901/30150226_PVpI.png

看到如下日志 则启动成功

图片地址：https://static.oschina.net/uploads/img/201901/30150342_nREr.png

通过 http://localhost:3000 访问项目即可进入系统，默认账号密码： admin/123456

Maven私服设置

找到 maven老家 conf/settings.xml，

在<mirrors>标签内增加阿里云maven镜像 最终结果见下面:

```
<mirrors>
      <mirror>
          <id>nexus-aliyun</id>
          <mirrorOf>*,!jeecg,!jeecg-snapshots</mirrorOf>
          <name>Nexus aliyun</name>
          <url>http://maven.aliyun.com/nexus/content/groups/public</url>
       </mirror>
 </mirrors>
```

然后执行maven命令,享受一下mvn时飞的感觉；

注意： jeecg导入项目提示jar未下载来，直接选中项目右键 tomcat7:run，相关的依赖会自动下载。

重要说明：

> Jeecg有自己的私服，如果配置了阿里镜像，默认所有的依赖都只会去阿里镜像下载，所以这里jeecg采用mirrorOf进行了排除，如果你这边也有自己的私服，可以参考jeecg进行配置，不然私有依赖会出现下载失败情况。

websorm svn下载项目

图片地址：https://static.oschina.net/uploads/img/201901/30115917_NdpC.png

输入svn地址：

图片地址：https://static.oschina.net/uploads/img/201901/30115926_vP6z.png

选择要checkout的工作目录：

图片地址：https://static.oschina.net/uploads/img/201901/30115934_1uWq.png

点击OK，下载项目。

快速开始

Hello World

前后端分离框如何快速进入开发，请参照下面hello world实现demo

一、后台请求实现

```
@RestController
@RequestMapping("/test/jeecgDemo")
@Slf4j
public class JeecgDemoController {

/**
 * hello world
 *
 * @param id
 * @return
 */
@GetMapping(value = "/hello")
public Result<String> hello() {
Result<String> result = new Result<String>();
result.setResult("Hello World!");
result.setSuccess(true);
return result;
}
}
```

请求 /test/jeecgDemo/hello 返回响应

```
{
"success": true,
"message": null,
"code": null,
"result": "Hello World!",
"timestamp": 1548833208562
}
```

二、前台vue页面实现

（1）创建vue页面src/views/jeecg/helloworld.vue

调用后台请求，获取返回的Hello World! 输出到页面，页面代码如下：

```
<template>
  <div>
    {{ msg }}
  </div>
</template>

<script>
 import {getAction} from '@/api/manage'
 export default {
   data () {
     return {
       msg: ""
     }
   },
   methods: {
     hello () {
       var url = "/test/jeecgDemo/hello"
       getAction(url).then((res) => {
         if (res.success) {
           this.msg = res.result;
         }
       })
     }
   },
   created() {
     this.hello();
   }
```

```
  }
</script>
```

代码说明：

1、data() 方法中定义数据对象msg

2、数据对象msg输出到页面，表达式如下：

```
{{ msg }}
```

3、定义一个方法，发起请求获取后台响应

后台实现的是get方法，引入getAction方法

import {getAction} from '@/api/manage'

定义方法调用：

```
    hello () {
      var url = "/test/jeecgDemo/hello"
      getAction(url).then((res) => {
        if (res.success) {
          this.msg = res.result;
        }
      })
    }
```

4、Vue生命周期 created 中调用方法

created() {

```
  this.hello();
}
```

hello方法中

this.msg = res.result;

把请求返回的Hello World! 赋值给msg数据对象，msg值改变则页面显示也改变。

三、配置菜单

配置helloword菜单【系统管理】-【菜单管理】

图片地址：https://static.oschina.net/uploads/img/201901/30170002_FM0S.png

- 其中前端组件配置相对src/views/目录下的　目录名+文件名
- 例如页面src/views/jeecg/helloworld.vue　前端组件配置 jeecg/helloworld

图片地址：https://static.oschina.net/uploads/img/201901/30170031_O53Y.png

用户角色授权【系统管理】-【角色管理】-授权

图片地址：https://static.oschina.net/uploads/img/201901/30170110_TIZa.png

图片地址：https://static.oschina.net/uploads/img/201901/30170127_4JzF.png

点击菜单访问页面展示Hello World!

上线发布

简易发布方案

正式环境部署

> 部署方案采用nginx+tomcat部署方案
>
> 后端服务通过JAR方式运行
>
> 前端项目build后的静态资源，部署到nginx中

一、后台项目jeecg-boot打jar包

1、修改数据库连接 application-prod.yml

2、修改缓存redis配置 application-prod.yml

3、修改上传附件配置 application-prod.yml

图片地址：https://static.oschina.net/uploads/img/201904/14221455_vj7T.png

4、切换配置为线上配置 application.yml

图片地址：https://static.oschina.net/uploads/img/201904/14221536_ylwz.png

然后 maven package 打jar包

二、后台项目jeecg-boot启动

通过命令启动项目

```
Window启动命令：
java -jar D:\jeecg-boot-1.0.1.jar

Linux下后台进程启动命令：
nohup java -jar jeecg-boot-1.0.1.jar >catalina.out 2>&1 &

关掉项目：
ps -ef|grep java
```

```
kill 进程号
```

三、前台项目build

1、修改后台接口服务地址 public/index.html

```
 //图片预览请求地址
 window._CONFIG['domianURL'] = 'http://localhost:8080/jeecg-boot';
 window._CONFIG['imgDomainURL'] = 'http://localhost:8080/jeecg-
boot/sys/common/view';
```

2、build项目

使用build命令打包项目

图片地址：https://static.oschina.net/uploads/img/201901/30163255_iNMZ.png

build完成后台会生成一个dist的目录该目录下即为build后的文件。

3、nginx部署前端项目

拷贝dist下的代码到nginx安装目录下html目录中，即可

四、nginx配置（conf/nginx.conf）

nginx监听80端口

```
server {
listen       80;
server_name  你的域名;

#后台服务配置，配置了这个location便可以通过http://域名/jeecg-boot/xxxx 访问
location ^~ /jeecg-boot {
proxy_pass              http://127.0.0.1:8080/jeecg-boot/;
proxy_set_header        Host 127.0.0.1;
proxy_set_header        X-Real-IP $remote_addr;
proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
}
#解决Router(mode: 'history')模式下，刷新路由地址不能找到页面的问题
location / {
root   html;
index  index.html index.htm;
if (!-e $request_filename) {
rewrite ^(.*)$ /index.html?s=$1 last;
break;
}
}
```

```
}
```

配置后启动nginx

通过：http://你的域名 访问项目，出现如下页面，使用账户/密码：admin/123456 登录成功即可

图片地址：https://static.oschina.net/uploads/img/201901/30165034_CqVQ.png

WAR发布方案

正式环境部署

> 部署方案采用nginx+tomcat部署方案
>
> 后端服务发布部署到tomcat中
>
> 前端项目由于build后都是静态问题，部署到nginx中

一、后台项目jeecg-boot打war包

（1）后台项目jeecg-boot打war包之前要进行如下改动

1、pom.xml文件中项目打包格式设置为war

<packaging>war</packaging>

具体配置如下：

```
  <modelVersion>4.0.0</modelVersion>
<groupId>org.jeecgframework.boot</groupId>
<artifactId>jeecg-boot</artifactId>
<version>1.1.0</version>
<packaging>war</packaging>
```

2、增加项目web容器部署的支持：

修改类/src/main/java/org/jeecg/JeecgApplication.java

代码如下：

```
package org.jeecg;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

import springfox.documentation.swagger2.annotations.EnableSwagger2;
```

```
@SpringBootApplication
@EnableSwagger2
public class JeecgApplication extends SpringBootServletInitializer {

@Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application)
{
        return application.sources(JeecgApplication.class);
    }


    public static void main(String[] args) {
    System.setProperty("spring.devtools.restart.enabled", "true");
    SpringApplication.run(JeecgApplication.class, args);
    }
}
```

1、修改数据库连接 application-prod.yml

2、修改缓存redis配置 application-prod.yml

3、修改上传附件配置 application-prod.yml

图片地址：https://static.oschina.net/uploads/img/201904/14221455_vj7T.png

4、切换配置为线上配置 application.yml

图片地址：https://static.oschina.net/uploads/img/201904/14221536_ylwz.png

然后maven install 或者 maven package 打war包

二、后台项目jeecg-boot部署tomcat

1、设置tomcat端口号 8080，设置tomcat编码 URIEncoding="UTF-8"

2、部署项目到tomcat安装目录webapps/jeecg-boot工程目录下

部署完后通过http://localhost:8080/jeecg-boot 可以访问项目，提示token错误说明部署成功！！

三、前台项目build

1、修改 public/index.html

```
 //图片预览请求地址
 window._CONFIG['domianURL'] = 'http://localhost:8080/jeecg-boot';
 window._CONFIG['imgDomainURL'] = 'http://localhost:8080/jeecg-
boot/sys/common/view';
```

2、后台接口服务项目名默认是jeecg-boot，如果需要个性化可以修改src/utils/request.js 中baseURL参数

（一般情况下默认不需要修改）

具体代码如下：

```
// 创建 axios 实例
const service = axios.create({
  baseURL: '/jeecg-boot/', // api base_url
  timeout: 6000 // 请求超时时间
})
```

3、build项目

使用build命令打包项目

图片地址：https://static.oschina.net/uploads/img/201901/30163255_iNMZ.png

build完成后台会生成一个dist的目录该目录下即为build后的文件。

4、nginx部署前端项目

拷贝dist下的代码到nginx安装目录下html目录中，即可

四、nginx配置（conf/nginx.conf）

nginx监听80端口

```
server {
listen       80;
server_name  你的域名;

#后台服务配置，配置了这个location便可以通过http://域名/jeecg-boot/xxxx 访问
location ^~ /jeecg-boot {
proxy_pass              http://127.0.0.1:8080/jeecg-boot/;
proxy_set_header        Host 127.0.0.1;
proxy_set_header        X-Real-IP $remote_addr;
proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
}
#解决Router(mode: 'history')模式下，刷新路由地址不能找到页面的问题
location / {
root   html;
index  index.html index.htm;
if (!-e $request_filename) {
rewrite ^(.*)$ /index.html?s=$1 last;
break;
}
}
```

```
}
```

配置后启动tomcat，启动nginx

通过http://你的域名/ 访问项目，出现如下页面，使用账户/密码：admin/123456 登录成功即可

图片地址：https://static.oschina.net/uploads/img/201901/30165034_CqVQ.png

代码生成器用法

如何使用代码生成器？

功能介绍：

> jeecg-boot代码生成器非常强大，支持单表、一对多模型生成，生成的代码包括前台和后台，生成后直接使用，无需修改。
>
> 功能说明： 一键生成的代码（包括：controller、service、dao、mapper、entity、vue）
>
> 模板位置： src/main/resources/jeecg/code-template

1. 单表GUI代码生成工具

找到jeecg-boot/src/main/java/org/jeecg/JeecgOneGUI.java，右键执行

图片地址：https://static.oschina.net/uploads/img/201904/14222638_Svth.png

1. 一对多代码生成工具

jeecg-boot/src/main/java/org/jeecg/JeecgOneToMainUtil.java

直接在此代码里面配置参数，右键执行就会生成对应代码

代码生成器配置：

1、代码生成器数据库配置文件

配置文件: src/main/resources/jeecg/jeecg_database.properties

图片地址：https://static.oschina.net/uploads/img/201904/16150206_W4mQ.jpg

2、代码生成器基础配置 (项目路径、根业务包路径、模板路径)

配置文件： src/main/resources/jeecg/jeecg_config.properties

3、Jeecg-boot采用前后端分离架构，vue页面需要手工复制到前端代码里面

```
* 1. 页面生成路径： src/main/java/{业务包根路径}/{子业务包}/vue/
* 2. 使用方法，手工复制到webstorm项目下面
* 3. 配置访问菜单
```

代码生成器模板

功能介绍：

> 目前代码生成器提供了四套模板，单表两套、一对多两套。

如何切换模板，修改配置文件src/main/resources/jeecg/jeecg_config.properties 里的参数templatepath，即可切换模板。

模板	路径	描述
单表业务分层模板	src/main/resources/jeecg/code-template/one	control上层分子业务包
单表代码分层模板	src/main/resources/jeecg/code-template/one2	control下层分子业务包
一对多默认风格模板	src/main/resources/jeecg/code-template/onetomany	一个表单维护
一对多ERP风格模板	src/main/resources/jeecg/code-template/onetomany2	子表数据分开维护

效果图：

1.单表业务分层模板

图片地址：https://static.oschina.net/uploads/img/201903/01122156_83aL.png

2.单表代码分层模板

图片地址：https://static.oschina.net/uploads/img/201903/01122205_LQcZ.png

3.一对多默认风格模板

图片地址：https://static.oschina.net/uploads/img/201904/14223818_Oqgj.png

4.一对多ERP风格模板

图片地址：https://static.oschina.net/uploads/img/201904/14223829_lhYI.png

5.单表代码生成器，表单风格提供两种选择

a. 默认弹窗模式表单
b. 抽屉风格表单

图片地址：https://static.oschina.net/uploads/img/201904/14224047_fKg6.png

如果需要抽屉风格，把Style#Drawer后缀删掉，覆盖原生产的页面即可。

UI前端开发技巧

源码解读

1. 登录页面代码位置

```
src\components\layouts\UserLayout.vue
src/views/user/Login.vue
```

1. 首页logo修改

```
src/components/tools/Logo.vue
```

1. 图片预览路径

```
src/api/api.js
 //图片预览请求地址
 const imgView = "http://127.0.0.1:8080/jeecg-boot/sys/common/view/";
 图文访问路径： http://127.0.0.1:8080/jeecg-boot/sys/common/view/user/h.jpg
```

4.首页报表

```
src/views/dashboard/*
src/views/dashboard/Analysis.vue
```

1. 登录退出逻辑

```
1.登录页面： src/views/user/Login.vue
2.相关API定义位置： src/api/index.js（很多无用的删掉）
                src/api/index.js
                src/api/login.js
                src/api/manage.js
3.左侧菜单加载页面：src/components/menu
            src/utils/util.js
            src/permission.js
4.隐藏路由配置
  用途： 如果那个组件不想在菜单上配置，但有需要路由跳转，则需要在这个地方配置路由。
  src/config/router.config.js
  对象： constantRouterMap
5. 接口:  /sys/login       登录接口
        /sys/permission/queryByUser  获取用户信息接口（首页菜单）
```

6.首页风格设置

src/defaultSettings.js

常用命令

npm install ｜ 下载依赖

Form 表单开发特殊性

v-decorator 属性

针对特殊控件： select、radio、checkbox

```
<a-radio-group buttonStyle="solid" v-decorator="[ 'status', {'initialValue':0}]">
    <a-radio-button :value="0">正常</a-radio-button>
    <a-radio-button :value="-1">停止</a-radio-button>
</a-radio-group>
```

注意： 此处的默认值只能通过{'initialValue':0} 这样的设置，不能通过属性。

表单编辑赋值操作：

```
this.$nextTick(() => {
    this.form.setFieldsValue(pick(this.model,'description','status'));
});
```

报表开发技术点

采用： viser-vue： ^2.4.4

文档： https://viserjs.github.io

示例： https://viserjs.github.io/demo.html#/viser/components/special-data-and-range-mark

全局配置文件

升级日志： 20190324

前台全局配置文件

配置内容：后台域名、图片服务器域名配置

文件位置：public/index.html

好处： 前端build完也可以直接修改index.html配置内容

```
<!-- 全局配置 -->
  <script>
    window._CONFIG = {};
    window._CONFIG['domianURL'] = 'http://localhost:8080/jeecg-boot';
    window._CONFIG['imgDomainURL'] = 'http://localhost:8080/jeecg-
boot/sys/common/view';
  </script>
```

用法：

参数	写法	描述

后台服务域名	window._CONFIG['domianURL']	-
图片服务器域名	window._CONFIG['imgDomainURL']	-

自定义组件

JgEditableTable 帮助文档

JgEditableTable 帮助文档

参数配置

参数	类型	必填	说明
columns	array		表格列的配置描述，具体项见下表
dataSource	array		表格数据
loading	boolean		是否正在加载，加载中不会显示任何行
actionButton	boolean		是否显示操作按钮，包括"新增"、"删除"
rowNumber	boolean		是否显示行号
rowSelection	boolean		是否可选择行

columns 参数详解

参数	类型	必填	说明

title	string		表格列头显示的问题
key	string		列数据在数据项中对应的 key，必须是唯一的
type	string		表单的类型，可以通过 `JgEditableTableUtil.Types`赋值
width	string		列的宽度，可以是百分比，也可以是 `px`或其他单位，建议设置为百分比，且每一列的宽度加起来不应超过 100%，否则可能会不能达到预期的效果
placeholder	string		表单预期值的提示信息，可以使用 `${...}`变量替换文本（使用方式见下方）
defaultValue	string		默认值，在新增一行时生效
validateRules	array		表单验证规则，配置方式见下表

当 type=checkbox 时所需的参数

参数	类型	必填	说明
defaultChecked	boolean		默认值是否选中

customValue	array		自定义值，checkbox需要的是boolean值，如果数据是其他值（例如`'Y' or 'N'`）时，就会导致错误，所以提供了该属性进行转换，例：`customValue: ['Y','N']`，会将`true`转换为`'Y'`，`false`转换为`'N'`，反之亦然

当 type=select 时所需的参数

参数	类型	必填	说明
options	array		下拉选项列表，详见下表

options 所需参数

参数	类型	必填	说明
title	string		显示标题
value	string		真实值

validateRules 配置规则

validateRules 需要的是一个数组，数组里每项都是一个规则，规则是object类型，规则的各个参数如下

- required 是否必填，可选值为trueorfalse
- pattern 正则表达式验证，只有成功匹配该正则的值才能成功通过验证
- message 当验证未通过时显示的提示文本，可以使用${...}变量替换文本（使用方式见下方）
- 点我查看示例(#示例二)

${...} 变量使用方式

在placeholder和message这两个属性中可以使用${...}变量来替换文本

在示例二(#示例二)中，配置了title为字段名称的一列，而placeholder配置成了请输入${title}，那么最终显示效果为请输入字段名称

这就是${...}变量的使用方式，在${}中可以使用的变量有title、key、defaultValue这三个属性的值

FAQ

如何获取表单的值？

在示例一(#示例一)中，设定了一个 ref="editableTable" 的属性，

那么在vue中就可以使用this.$refs.editableTable获取到该表格的实例，

并调取其中的方法，在示例三(#示例三)中就是以这种做法获取到的值

如何进行表单验证？

在获取值的时候默认会进行表单验证操作，用户在输入的时候也会对正在输入的表单进行验证，只要配置好规则就可以了

如何添加或删除一行？

该功能已封装到组件中，你只需要将 actionButton 设置为 true 即可

示例一

```
<jg-editable-table
    ref="editableTable"
    :loading="loading"
    :columns="columns"
    :dataSource="dataSource"
    :rowNumber="true"
    :rowSelection="true"
    :actionButton="true"
    style="margin-top: 8px;"
    @selectRowChange="handleSelectRowChange"/>
```

示例二

```
columns:[
    {
        title: '字段名称',
```

```
        key: 'fieldName',
        type: Types.input,
        placeholder: '请输入${title}',
        defaultValue: '称名段字',
        // 表单验证规则
        validateRules: [
          {
            required: true, // 必填
            message: '请输入${title}' // 提示的文本
          },
          {
            pattern: /^[a-z|A-Z][a-z|A-Z\d_-]{0,}$/, // 正则
            message: '${title}必须以字母开头，可包含数字、下划线、横杠'
          }
        ]
    }
]
```

示例三

```
// 获取被逻辑删除的字段id
let deleteIds = this.$refs.editableTable.getDeleteIds();
// 获取所有表单的值，并进行验证
this.$refs.editableTable.getValues((error, values) => {
    // 错误数 = 0 则代表验证通过
    if (error === 0) {
        this.$message.success('验证通过')
        // 将通过后的数组提交到后台或自行进行其他处理
        console.log(deleteIds, values)
    } else {
        this.$message.error('验证未通过')
    }
})
```

DictSelectTag字典标签

> 针对字典的使用，目前提供了一个标签和函数。
>
> DictSelectTag 标签： 用于表单的标签使用，比如通过性别字典编码：sex，可以直接渲染出下拉组件。
>
> DictSelectUtil.js函数： 用于列表数据展示，针对列表字段字典值替换成字典文本，进行展示

。

[1].DictSelectTag 字典标签用法

示例：

```
<DictSelectTag  v-model="queryParam.sex" placeholder="请输入用户性别"
dictCode="sex"/>
```

v-decorator用法：

```
<j-dict-select-tag  v-decorator="['sex', {}]" :triggerChange="true" placeholder="请输入
用户性别"
                dictCode="sex"/>
```

[2].DictSelectUtil.js 列表字典函数用法

示例：

第一步： 引入依赖方法

```
import {initDictOptions, filterDictText} from '@/components/dict/DictSelectUtil'
```

第二步： 在created()初始化方法执行字典配置方法

```
 this.initDictConfig();
```

第三步： 实现initDictConfig方法，加载列表所需要的字典(列表上有多个字典项，就执行多次initDictOptions方法)

```
//sexDictOptions 自行定义
      initDictConfig() {
        //初始化字典 - 性别
        initDictOptions('sex').then((res) => {
          if (res.success) {
            this.sexDictOptions = res.result;
          }
        });
      },
```

第四步：实现字段的customRender方法

```
customRender: (text, record, index) => {
       //字典值替换通用方法
       return filterDictText(this.sexDictOptions, text);
```

高级查询组件文档和示例

components包下文件描述

1._util

存放自定义函数 详细见代码注释

2.AvatarList

显示头像群并支持tip，用法参考srcviewsHome.vue（如下图）

https://static.oschina.net/uploads/img/201904/12181253_O0Xi.png

3.chart

存放各种图表相关的组件,条形图柱形图折线图等等 具体用法参考首页

4.countDown

一个倒计时组件，用法参考home页,简单描述,该组件有3个属性,

target(时间/毫秒数)必填，

format(function,该方法接收一个毫秒数的参数,用于格式化显示当前倒计时时间)非必填,

onEnd倒计时结束触发函数

https://static.oschina.net/uploads/img/201904/12182046_mwqJ.png

5.dict

数据字典专用，用法参考文件夹下readme文件

6.Ellipsis

字符串截取组件,可以指定字符串的显示长度,并将全部内容显示到tip中,简单使用参考srcviewssystemPermissionList.vue

7.jeecg

该包下自定义了很多列表/表单中用到的组件 参考包下readme文件

8.jeecgbiz

该包下定义了一些业务相关的组件，比如选择用户弹框,根据部门选择用户等等

9.layouts+page

系统页面布局相关组件，比如登陆进去之后页面顶部显示什么，底部显示什么，菜单点击触发多个tab的布局等等 一般情况不需要修改

10.menu

菜单组件，俩个，一个折叠菜单一个正常显示的菜单

11.NumberInfo

数字信息显示组件 如下图

https://static.oschina.net/uploads/img/201904/12185858_uvJ5.png

12.online

该包下封装了online表单的相关组件,用于展示表单各种控件,验证表单等等,相关用法参考readme

13.setting

该包下封装了首页风格切换等功能如下图

https://static.oschina.net/uploads/img/201904/12190520_jySG.png

14.table

一个二次封装的table组件,用于展示列表，参考readme

15.tools

Breadcrumb.vue：面包屑二次封装,支持路由跳转

DetailList.vue：详情展示用法参考 srcviewsprofileadvancedAdvanced.vue(效果如下图)

https://static.oschina.net/uploads/img/201904/12193954_Uar6.png

个人认为该页面代码有两点值得学习：

1.vue provide/inject的使用

2.该页面css定义方式,只定义一个顶层class,其余样式都定义在其下,这样只要顶层class不和别的页面冲突,整个页面的样式都是唯一生效的

FooterToolBar.vue:fixed定位的底部，通过是否定义内部控件的属性slot="extra"决定是左浮动或是右浮动

HeaderNotice.vue:首页通知(如下图)

https://static.oschina.net/uploads/img/201904/12195340_fPe0.png

HeaderInfo.vue:上下文字布局（如下图）

https://static.oschina.net/uploads/img/201904/12195638_dG5o.png

Logo.vue:首页左上侧的log图

https://static.oschina.net/uploads/img/201904/12200908_ihv3.png

UserMenu.vue:首页右上侧的内容

https://static.oschina.net/uploads/img/201904/12201226_laQK.png

16.trend

趋势显示组件参考首页（如下图）

https://static.oschina.net/uploads/img/201904/12201600_Wo8K.png

更多组件统一文档

JDate 日期组件 使用文档

说明: antd-vue日期组件需要用moment中转一下，用起来不是很方便，特二次封装，使用时只需要传字符串即可

参数配置

参数	类型	必填	说明
placeholder	string		placeholder
readOnly	boolean		true/false 默认 false
value	string		绑定v-model或是v-decorator后不需要设置
showTime	boolean		是否展示时间 true/false 默认 false
dateFormat	string		日期格式 默认 'YYYY-MM-DD' 若showTime设置为true则需要将其设置成对应的时间格式(如:YYYY-MM-DD HH:mm:ss)
triggerChange	string		触发组件值改变的事件是否是change,当使用v-decorator时且没有设置decorator的option.trigger为input需要设置该值为true

使用示例

1.组件带有v-model的使用方法

```
<j-date v-model="dateStr"></j-date>
```

2.组件带有v-decorator的使用方法

a).设置trigger-change属性为true

```
    <j-date :trigger-change="true" v-decorator="['dateStr',{}]"></j-date>
```

b).设置decorator的option.trigger为input

```
    <j-date v-decorator="['dateStr',{trigger:'input'}]"></j-date>
```

3.其他使用

添加style

```
<j-date v-model="dateStr" style="width:100%"></j-date>
```

添加placeholder

```
<j-date v-model="dateStr" placeholder="请输入dateStr"></j-date>
```

添加readOnly

```
<j-date v-model="dateStr" :read-only="true"></j-date>
```

备注:

script内需引入jdate

```
<script>
 import JDate from '@/components/jeecg/JDate'
 export default {
   name: "demo",
   components: {
     JDate
   }
   //...
 }
</script>
```

JSuperQuery 高级查询 使用文档

参数配置

参数	类型	必填	说明

fieldList	array		需要查询的列集合 示例如下，type类型有 :date/datetime/string/int/number
callback	array		回调函数名称(非必须)默认 handleSuperQuery

fieldList结构示例：

```
 const superQueryFieldList=[{
   type:"date",
   value:"birthday",
   text:"生日"
 },{
   type:"string",
   value:"name",
   text:"用户名"
 },{
   type:"int",
   value:"age",
   text:"年龄"
 }]
```

页面代码概述:

1.import之后再components之内声明

```
import JSuperQuery from '@/components/jeecg/JSuperQuery.vue';
 export default {
   name: "JeecgDemoList",
   components: {
     JSuperQuery
   },
```

2.页面引用

```
  <!-- 高级查询区域 -->
  <j-super-query :fieldList="fieldList" ref="superQueryModal"
@handleSuperQuery="handleSuperQuery"></j-super-query>
```

3.list页面data中需要定义三个属性：

```
  fieldList:superQueryFieldList,
  superQueryFlag:false,
  superQueryParams:""
```

4.list页面声明回调事件handleSuperQuery(与组件的callback对应即可)

```
//高级查询方法
handleSuperQuery(arg) {
  if(!arg){
    this.superQueryParams=''
    this.superQueryFlag = false
  }else{
    this.superQueryFlag = true
    this.superQueryParams=JSON.stringify(arg)
  }
  this.loadData()
},
```

5.改造list页面方法

```
  // 获取查询条件
  getQueryParams() {
    let sqp = {}
    if(this.superQueryParams){
      sqp['superQueryParams']=encodeURI(this.superQueryParams)
    }
    var param = Object.assign(sqp, this.queryParam, this.isorter);
    param.field = this.getQueryField();
    param.pageNo = this.ipagination.current;
    param.pageSize = this.ipagination.pageSize;
    return filterObj(param);
  },
```

6.打开弹框调用show方法：

```
this.$refs.superQueryModal.show();
```

JEllipsis 字符串超长截取省略号显示

说明: 遇到超长文本展示，通过此标签可以截取省略号显示，鼠标放置会提示全文本

参数配置

参数	类型	必填	说明
value	string	必填	字符串文本
length	number	非必填	默认25

使用示例

1.组件带有v-model的使用方法

```
<j-ellipsis :value="text"/>


# Modal弹框实现最大化功能

1.定义modal的宽度：
```vue
  <a-modal
    :width="modalWidth"


    />
```

2.自定义modal的title,居右显示切换图标

```
  <template slot="title">
    <div style="width: 100%;">
      <span>{{ title }}</span>
      <span style="display:inline-block;width:calc(100% - 51px);padding-right:10px;text-
align: right">
        <a-button @click="toggleScreen" icon="appstore"
style="height:20px;width:20px;border:0px"></a-button>
      </span>
    </div>
  </template>
```

3.定义toggleScreen事件,用于切换modal宽度

```
  toggleScreen(){
      if(this.modaltoggleFlag){
        this.modalWidth = window.innerWidth;
      }else{
```
```

```
      this.modalWidth = 800;
    }
    this.modaltoggleFlag = !this.modaltoggleFlag;
  },
```

4.data中声明上述用到的属性

```
  data () {
    return {
      modalWidth:800,
      modaltoggleFlag:true,
```

表单字段重复校验简易工具类

重复校验效果：

图片地址：https://static.oschina.net/uploads/img/201904/19191836_eGkQ.png

• 编码排重使用示例

1.引入排重接口,代码如下:

```
import { duplicateCheck } from '@/api/api'
```

2.找到编码必填校验规则的前端代码,代码如下:

```
<a-input placeholder="请输入编码" v-decorator="['code', validatorRules.code ]"/>

code: {
          rules: [
            { required: true, message: '请输入编码!' },
            {validator: this.validateCode}
          ]
        },
```

3.找到rules里validator对应的方法在哪里,然后使用第一步中引入的排重校验接口.

以用户online表单编码为示例,其中四个必传的参数有:

```
  {tableName:表名,fieldName:字段名,fieldVal:字段值,dataId:表的主键},
```

具体使用代码如下:

```
  validateCode(rule, value, callback){
      let pattern = /^[a-z|A-Z][a-z|A-Z|\d|_|-]{0,}$/;
      if(!pattern.test(value)){
        callback('编码必须以字母开头，可包含数字、下划线、横杠');
      } else {
```

```
        var params = {
          tableName: "onl_cgreport_head",
          fieldName: "code",
          fieldVal: value,
          dataId: this.model.id
        };
        duplicateCheck(params).then((res)=>{
          if(res.success){
           callback();
          }else{
            callback(res.message);
          }
        })
      }
    },
```

后端开发技巧

常用问题汇总

1、Druid监控

```
访问：http://localhost:8080/jeecg-boot/druid/，
登录名：admin，密码123456
```

2、在线接口文档swagger

http://localhost:8080/jeecg-boot/swagger-ui.html#/

3、项目根路径如何修改

目前项目后台访问默认路径是： http://localhost:8080/jeecg-boot

默认端口： 8080

默认项目名： jeecg-boot

如果需要自定义可以修改配置文件： src/main/resources/application.yml

```
server:
    port: 8080
    servlet:
      context-path: /jeecg-boot
```

4、获取登录用户信息

```
SysUser sysUser = (SysUser)SecurityUtils.getSubject().getPrincipal();
```

菜单路由配置

菜单配置说明

字段名称	说明
菜单类型	一级菜单：配置一级菜单；子菜单：配置下级菜单；按钮：配置页面按钮权限
菜单名称	定义菜单名称
上级菜单	菜单类型为子菜单时，选择关联的上级菜单
菜单路径	定义菜单的路径，通常为：/包名/文件名 具体参见【菜单路径配置说明】
前端组件	定义菜单访问的组件名称，有两种类型，一种为通用组件，一种为具体的页面， 具体参见【前端组件配置说明】
菜单图标	菜单树展示的图标
排序	菜单展示的先后顺序
隐藏路由	不展示为菜单，但是在页面中跳转，弹出的页面路由菜单
聚合路由	多个下级菜单路由在一个页面聚合展示

• 前端组件配置说明：

1、非叶子菜单（即没有下级的菜单）配置固定 前端组件layouts/RouteView

2、普通的叶子菜单（即具体的页面） 配置相对于src/views目录的路径

例如src/views/jeecg/helloworld.vue 这个页面

配置菜单时 前端组件为 jeecg/helloworld

3、需要跳转到第三页面的菜单 前端组件固定为：layouts/IframePageView

（比如跳转百度：https://www.baidu.com）

• 菜单路径配置说明

1、非叶子菜单（即没有下级的菜单），URL配置规则：按照功能模块定义的关键根路径即可，不能重复，需以“/”开头

2、普通的叶子菜单（即具体的页面），URL和前端组件配置保持一致即可，需在前端组件值前加“/”

3、需要跳转到第三页面的菜单，菜单路径配置第三方调整的地址即可，例如
http://www.baidu.com

路由菜单规则

> 菜单配置就是配置前端所需要的路由
>
> 菜单路径： 对应页面访问请求URL （系统唯一，不能有重复URL）
>
> 前端组件： 对应前端页面组件（路径+名字，无.vue后缀）

路由name取值规则：

```
通过菜单URL，生成路由name（去掉URL前缀斜杠，替换内容中的斜杠‘/’为-）
举例： URL = /account/settings/base
     RouteName = account-settings-base
```

前端页面跳转用法:

```
<router-link :to="{ name: 'account-settings-base' }">
  基本设置
 </router-link>
```

Online报表菜单如何配置？

带参数路由菜单如何配置？

> 什么是带参数菜单？
>
> 就是菜单URL是动态的，带有参数，根据参数不同页面不同的渲染效果。

目前JEECG已经实现了此功能，具体配置可以参考Online报表：

此配置分两部分：

第一部分： 动态路由的配置（路由）

第二部分： 具体菜单配置（非路由）

第一部分：动态路由的配置

1. 菜单URL示例：

```
/online/cgreport/:code
```

1. 菜单类型是否路由： 选择是

图片地址：https://static.oschina.net/uploads/img/201904/19173821_TZ0T.png

第二部分：具体菜单配置

[1]. 把URL的动态参数：code，改成具体值

例如：

```
/online/cgreport/87b55a515d3441b6b98e48e5b35474a6
```

[2].菜单的路由类型设置为非路由（很重要）

图片地址：https://static.oschina.net/uploads/img/201904/19174358_jhbi.png

自定义注解用法

字典翻译注解@Dict

一、功能说明

将数据库某一列的值按照字典配置翻译成对应的文字描述

比如： 用户表有一字段:性别,数据库存储的1,2分别表示男,女,当数据被查询展示在列表上时,就需要将1,2翻译成男女,这就要用到@Dict

二、使用说明（以用户管理翻译性别列为例说明）

1.配置字典

图片地址：https://static.oschina.net/uploads/img/201904/12160354_iYbF.png

图片地址：https://static.oschina.net/uploads/img/201904/12160430_ErxX.png

2.后端实体属性上加注解(此处dicCode 对应上述字典编码)

```
/**
 * 性别（1：男 2：女）
 */
@Dict(dicCode = "sex")
 private Integer sex;
```

3.前端定义column(**此处dataIndex原字段名为sex ,这里需要定义为sexdictText,即原字段名+'_dictText'**)

```
columns: [
   //...省略其他列
   {
    title: '性别',
    align: "center",
    width: 80,
    dataIndex: 'sex_dictText'
    }]
```

日志记录注解@AutoLog

数据权限注解@PermissionData

用于数据权限使用示例见【后端开发技巧】-->【系统权限】-->【数据权限用法】

Spring缓存注解@Cacheable

数据库设计规范

建表规范

- 主键必须是ID,字符串类型，32位长度，唯一索引;
- 建表标准字段，必须有：创建人、创建时间、修改人、修改时间等标准字段;

```
ALTER TABLE `表名`
ADD COLUMN `create_by`  varchar(32) NULL COMMENT '创建人',
ADD COLUMN `create_time`  datetime NULL COMMENT '创建时间' AFTER `create_by`,
ADD COLUMN `update_by`  varchar(32) NULL COMMENT '修改人' AFTER `create_time`,
ADD COLUMN `update_time`  datetime NULL COMMENT '修改时间' AFTER `update_by`;
```

- 表字段注释，每个字段必须设置注释说明;
- 表字段注释，状态类型的字段必须说明取值规则（比如性别sex取值规则）

```
        比如：'性别  0/男,1/女'
```

- 索引，查询频率高的字段加索引（单字段索引 、组合索引）;
- 类型字段，尽量用字符串varchar类型1-2长度，少用int类型，避免不必要的问题。

事务如何使用？

> jeecg-boot 采用注解事务方式，事务控制在service层面。
>
> 注解： @Transactional

如何加事务？=> 在service对应的方法加上注解@Transactional即可，具体参考一下代码：

```
    /**
 * 事务控制在service层面
 * 加上注解：@Transactional，声明的方法就是一个独立的事务（有异常DB操作全部回滚）
 */
@Transactional
public void testTran() {
JeecgDemo pp = new JeecgDemo();
pp.setAge(1111);
pp.setName("测试事务  小白兔 1");
```

```
jeecgDemoMapper.insert(pp);

JeecgDemo pp2 = new JeecgDemo();
pp2.setAge(2222);
pp2.setName("测试事务  小白兔 2");
jeecgDemoMapper.insert(pp2);

Integer.parseInt("hello");//自定义异常

JeecgDemo pp3 = new JeecgDemo();
pp3.setAge(3333);
pp3.setName("测试事务  小白兔 3");
jeecgDemoMapper.insert(pp3);
return ;
}
```

系统日志怎么插入？

> jeecg-boot 提供了在线日志管理功能，可以在线实时查看系统登录更新的所有操作。
>
> jeecg-boot提供两种方式，写入系统日志

方式一： 自定义注解@AutoLog

在Control的方法上，加上注解@AutoLog("操作内容描述")

参考：

```
    /**
 *   添加
 * @param jeecgDemo
 * @return
 */
@RequestMapping(value = "/add", method = RequestMethod.POST)
@AutoLog(value = "添加测试DEMO")
public Result<JeecgDemo> add(@RequestBody JeecgDemo jeecgDemo) {
Result<JeecgDemo> result = new Result<JeecgDemo>();
try {
jeecgDemoService.save(jeecgDemo);
result.success("添加成功！");
} catch (Exception e) {
e.printStackTrace();
log.info(e.getMessage());
result.error500("操作失败");
```

```
}
return result;
}
```

方式二：调用共通API插入日志

a. 引入共通service

```
@Autowired
private ISysBaseAPI sysBaseAPI;
```

b. 调用插入日志方法

```
sysBaseAPI.addLog("登录失败，用户名:"+username+"不存在！",
CommonConstant.LOG_TYPE_1, null);
```

定时任务如何开发？

> 采用Quartz分布式集群调度，支持在线配置定时任务

如何使用呢？

第一步： 自定义job（实现类org.quartz.Job）

```
/**
 * 示例不带参定时任务
 *
 * @author Scott
 */
@Slf4j
public class SampleJob implements Job {

@Override
public void execute(JobExecutionContext jobExecutionContext) throws
JobExecutionException {
   log.info(String.format(" Jeecg-Boot 普通定时任务 SampleJob！ 时间:" +
DateUtils.getTimestamp()));
}
}
```

第二步：在线配置定时任务

图片地址：https://static.oschina.net/uploads/img/201901/19140027_UGlu.png

第三步：支持在线管理，启停

图片地址：https://static.oschina.net/uploads/img/201901/19140154_YlMm.png

redis 如何使用？

jeecg-boot集成了redis

用法分三种：

1. 通过Jeecg自封装工具类

```
  //封装了redis操作各种方法
  @Autowired
  private RedisUtil redisUtil;
```

2.通过注解

参考链接： https://www.cnblogs.com/fashflying/p/6908028.html

```
  //key的定义参考官方文档
 @Cacheable(cacheNames="jeecgDemo", key="#id")

 示例：
/**
 * 缓存注解测试： redis
 */
@Cacheable(cacheNames="jeecgDemo", key="#id")
public JeecgDemo getByIdCacheable(String id) {
JeecgDemo t = jeecgDemoMapper.selectById(id);
System.err.println(t);
return t;
}
```

3.通过原生工具service

```
 @Autowired
 private RedisTemplate<String, Object> redisTemplate;
 @Autowired
 private StringRedisTemplate stringRedisTemplate;
```

其他技巧：

@CacheEvict用来标注在需要清除缓存元素的方法或类上的

参考链接： https://www.cnblogs.com/fashflying/p/6908028.html

```
@CacheEvict(value="dictCache", allEntries=true)
public Result<SysDict> delete(@RequestParam(name="id",required=true) String id) {
```

动态数据源使用

动态数据源使用

Druid 动态数据源

一、动态数据源配置

/src/main/resources/application.yml

```
datasource:
  datasource:
        master:
          url: jdbc:mysql://127.0.0.1:3306/jeecg-boot?characterEncoding=UTF-
8&useUnicode=true&useSSL=false
          username: root
          password: root
          driver-class-name: com.mysql.jdbc.Driver
        multi-datasource1:
          url: jdbc:mysql://localhost:3306/jeecg-
boot2?useUnicode=true&characterEncoding=utf8&autoReconnect=true&zeroDateTi
meBehavior=convertToNull&transformedBitIsBoolean=true
          username: root
          password: root
          driver-class-name: com.mysql.jdbc.Driver
```

master 为主数据源，系统默认数据源

multi-datasource1 ：自定义的第三方数据源，multi-datasource1名称随便定义

二、动态数据源使用

使用 @DS 切换数据源。

@DS 可以注解在方法上和类上，同时存在方法注解优先于类上注解。

注解在service实现或mapper接口方法上，但强烈不建议同时在service和mapper注解。 (可能会有问题)

注解	结果
没有@DS	默认数据源
@DS("dsName")	dsName可以为组名也可以为具体某个库的名称

代码示例：

```
@Service
@DS("multi-datasource1")
public class JeecgDemoServiceImpl implements JeecgDemoService {
```

```
  @Autowired
  private JdbcTemplate jdbcTemplate;

  public List<Map<String, Object>> selectAll() {
    return  jdbcTemplate.queryForList("select * from user");
  }

  @Override
  @DS("multi-datasource2")
  public List<Map<String, Object>> selectByCondition() {
    return  jdbcTemplate.queryForList("select * from user where age >10");
  }
}
```

数据日志如何用？

> 如果你需要记录某个表单的每次详细更新记录，就需要用到这个
>
> 此功能会将表单每次变更的数据内容以版本方式存储，提供对比功能，查看每个字段变更情况

```
//业务表单记录数据变更日志
sysDataLogService.addDataLog(String tableName, String dataId, String dataContent)
```

查询过滤器用法

• 查询过滤器用法

```
QueryWrapper<?> queryWrapper = QueryGenerator.initQueryWrapper(?,
req.getParameterMap());
```

代码示例：

```
@GetMapping(value = "/list")
public Result<IPage<JeecgDemo>> list(JeecgDemo jeecgDemo,
@RequestParam(name = "pageNo", defaultValue = "1") Integer pageNo,
                                    @RequestParam(name = "pageSize", defaultValue = "10")
Integer pageSize,
HttpServletRequest req) {
Result<IPage<JeecgDemo>> result = new Result<IPage<JeecgDemo>>();

//调用QueryGenerator的initQueryWrapper
QueryWrapper<JeecgDemo> queryWrapper =
```

```
QueryGenerator.initQueryWrapper(jeecgDemo, req.getParameterMap());

Page<JeecgDemo> page = new Page<JeecgDemo>(pageNo, pageSize);
IPage<JeecgDemo> pageList = jeecgDemoService.page(page, queryWrapper);
result.setSuccess(true);
result.setResult(pageList);
return result;
}
```

• 查询规则 (本规则不适用于高级查询,高级查询有自己对应的查询类型可以选择)

查询模式	用法	说明
模糊查询	支持左右模糊和全模糊 需要在查询输入框内前或后带*或是前后全部带*	
取非查询	在查询输入框前面输入! 则查询该字段不等于输入值的数据(数值类型不支持此种查询,可以将数值字段定义为字符串类型的)	
\> \>= < <=	同取非查询 在输入框前面输入对应特殊字符即表示走对应规则查询	
in查询	若传入的数据带,(逗号) 则表示该查询为in查询	
多选字段模糊查询	上述4 有一个特例，若某一查询字段前后都带逗号 则会将其视为走这种查询方式 ,该查询方式是将查询条件以逗号分割再遍历数组 将每个元素作like查询 用or拼接,例如 现在name传入值 ,a,b,c, 那么结果sql就是 name like '%a%' or name like '%b%' or name like '%c%'	

AutoPOI EXCEL导出导入工具

AutoPOI (Excel和 Word简易工具类 EasyPOI衍变升级版)

AutoPOI 功能如同名字auto，追求的就是自动化，让一个没接触过poi的人员，可以傻瓜式半智能化的快速实现Excel导入导出、Word模板导出、可以仅仅5行代码就可以完成Excel的导入导出。

AutoPOI的主要特点

1.设计精巧,使用简单

2.接口丰富,扩展简单

3.默认值多,write less do more

4.AbstractView 支持,web导出可以简单明了

AutoPOI的几个入口工具类

1.ExcelExportUtil Excel导出(普通导出,模板导出)

2.ExcelImportUtil Excel导入

3.WordExportUtil Word导出(只支持docx ,doc版本poi存在图片的bug,暂不支持)

关于Excel导出XLS和XLSX区别

1.导出时间XLS比XLSX快2-3倍

2.导出大小XLS是XLSX的2-3倍或者更多

3.导出需要综合网速和本地速度做考虑^~^

几个工程的说明

1.autopoi-parent 父包--作用大家都懂得

2.autopoi 导入导出的工具包,可以完成Excel导出,导入,Word的导出,Excel的导出功能

3.autopoi-web 耦合了spring-mvc 基于AbstractView,极大的简化spring-mvc下的导出功能

4.sax 导入使用xercesImpl这个包(这个包可能造成奇怪的问题哈),word导出使用poi-scratchpad,都作为可选包了

maven

```
<dependency>
 <groupId>org.jeecgframework</groupId>
 <artifactId>autopoi-web</artifactId>
 <version>1.0.2</version>
</dependency>
```

AutoPoi 模板 表达式支持

- 空格分割
- 三目运算 {{test ? obj:obj2}}
- n: 表示 这个cell是数值类型 {{n:}}
- le: 代表长度{{le:()}} 在if/else 运用{{le:() > 8 ? obj1 : obj2}}
- fd: 格式化时间 {{fd:(obj;yyyy-MM-dd)}}
- fn: 格式化数字 {{fn:(obj;###.00)}}
- fe: 遍历数据,创建row
- !fe: 遍历数据不创建row
- $fe: 下移插入,把当前行,下面的行全部下移.size()行,然后插入
- !if: 删除当前列 {{!if:(test)}}
- 单引号表示常量值 '' 比如'1' 那么输出的就是 1

AutoPoi导出实例

1.注解,导入导出都是基于注解的,实体上做上注解,标示导出对象,同时可以做一些操作

```
@ExcelTarget("courseEntity")
public class CourseEntity implements java.io.Serializable {
/** 主键 */
private String id;
/** 课程名称 */
@Excel(name = "课程名称", orderNum = "1", needMerge = true)
private String name;
/** 老师主键 */
@ExcelEntity(id = "yuwen")
@ExcelVerify()
private TeacherEntity teacher;
/** 老师主键 */
@ExcelEntity(id = "shuxue")
private TeacherEntity shuxueteacher;
```

```
@ExcelCollection(name = "选课学生", orderNum = "4")
private List<StudentEntity> students;
```

2.基础导出

传入导出参数,导出对象,以及对象列表即可完成导出

```
HSSFWorkbook workbook = ExcelExportUtil.exportExcel(new ExportParams(
"2412312", "测试", "测试"), CourseEntity.class, list);
```

3.基础导出,带有索引

在到处参数设置一个值,就可以在导出列增加索引

```
ExportParams params = new ExportParams("2412312", "测试", "测试");
params.setAddIndex(true);
HSSFWorkbook workbook = ExcelExportUtil.exportExcel(params,
TeacherEntity.class, telist);
```

4.导出Map

创建类似注解的集合,即可完成Map的导出,略有麻烦

```
List<ExcelExportEntity> entity = new ArrayList<ExcelExportEntity>();
entity.add(new ExcelExportEntity("姓名", "name"));
entity.add(new ExcelExportEntity("性别", "sex"));

List<Map<String, String>> list = new ArrayList<Map<String, String>>();
Map<String, String> map;
for (int i = 0; i < 10; i++) {
map = new HashMap<String, String>();
map.put("name", "1" + i);
map.put("sex", "2" + i);
list.add(map);
}

HSSFWorkbook workbook = ExcelExportUtil.exportExcel(new ExportParams(
"测试", "测试"), entity, list);
```

5.模板导出

根据模板配置,完成对应导出

```
TemplateExportParams params = new TemplateExportParams();
params.setHeadingRows(2);
params.setHeadingStartRow(2);
Map<String,Object> map = new HashMap<String, Object>();
```

```
    map.put("year", "2013");
    map.put("sunCourses", list.size());
    Map<String,Object> obj = new HashMap<String, Object>();
    map.put("obj", obj);
    obj.put("name", list.size());
 params.setTemplateUrl("org/jeecgframework/poi/excel/doc/exportTemp.xls");
Workbook book = ExcelExportUtil.exportExcel(params, CourseEntity.class, list,
map);
```

6.导入

设置导入参数,传入文件或者流,即可获得相应的list

```
ImportParams params = new ImportParams();
params.setTitleRows(2);
params.setHeadRows(2);
//params.setSheetNum(9);
params.setNeedSave(true);
long start = new Date().getTime();
List<CourseEntity> list = ExcelImportUtil.importExcel(new File(
"d:/tt.xls"), CourseEntity.class, params);
```

7.SpringMvc的无缝融合

简单几句话,Excel导出搞定

```
@RequestMapping(value = "/exportXls")
public ModelAndView exportXls(HttpServletRequest request, HttpServletResponse
response) {
ModelAndView mv = new ModelAndView(new JeecgEntityExcelView());
List<JeecgDemo> pageList = jeecgDemoService.list();
//导出文件名称
mv.addObject(NormalExcelConstants.FILE_NAME,"导出Excel文件名字");
//注解对象Class
mv.addObject(NormalExcelConstants.CLASS,JeecgDemo.class);
//自定义表格参数
mv.addObject(NormalExcelConstants.PARAMS,new ExportParams("自定义导出Excel模板
内容标题", "自定义Sheet名字"));
//导出数据列表
mv.addObject(NormalExcelConstants.DATA_LIST,pageList);
return mv;
}
```

自定义视图	用途	描述	

JeecgMapExcelView	实体对象导出视图	例如：List<JeecgDemo>	
JeecgEntityExcelView	Map对象导出视图	List<Map<String, String>> list	
JeecgTemplateExcelView	Excel模板导出视图	-	
JeecgTemplateWordView	Word模板导出视图	-	

8.Excel导入校验,过滤不符合规则的数据,追加错误信息到Excel,提供常用的校验规则,已经通用的校验接口

```
/**
    * Email校验
    */
   @Excel(name = "Email", width = 25)
   @ExcelVerify(isEmail = true, notNull = true)
   private String email;
   /**
    * 手机号校验
    */
   @Excel(name = "Mobile", width = 20)
   @ExcelVerify(isMobile = true, notNull = true)
   private String mobile;

   ExcelImportResult<ExcelVerifyEntity> result =
ExcelImportUtil.importExcelVerify(new File(
           "d:/tt.xls"), ExcelVerifyEntity.class, params);
   for (int i = 0; i < result.getList().size(); i++) {
      System.out.println(ReflectionToStringBuilder.toString(result.getList().get(i)));
   }
```

9.导入Map

设置导入参数,传入文件或者流,即可获得相应的list,自定义Key,需要实现IExcelDataHandler接口

```
ImportParams params = new ImportParams();
List<Map<String,Object>> list = ExcelImportUtil.importExcel(new File(
```

```
"d:/tt.xls"), Map.class, params);
```

10.字典用法

在实体属性注解excel中添加dicCode="",此处dicCode即为jeecg系统中数据字典的Code

```
@Excel(name="性别",width=15,dicCode="sex")
private java.lang.String sex;
```

11.字典表用法

此处dictTable为数据库表名，dicCode为关联字段名，dicText为excel中显示的内容对应的字段

```
@Excel(name="部门",dictTable="t_s_depart",dicCode="id",dicText="departname")
 private java.lang.String depart;
```

12.Replace用法

若数据库中存储的是0/1 ，则导出/导入的excel单元格中显示的是女/男

```
@Excel(name="测试替换",width=15,replace={"男_1","女_0"})
private java.lang.String fdReplace;
```

13.高级字段转换用法

• exportConvert：在导出的时候需要替换值则配置该值为true，同时增加一个方法，方法名为原get方法名前加convert。

• importConvert：在导入的时候需要替换值则配置该值为true，同时增加一个方法，方法名为原set方法名前加convert。

```
@Excel(name="测试转换",width=15,exportConvert=true,importConvert=true)
private java.lang.String fdConvert;

/**
 * 转换值示例： 在该字段值的后面加上元
 * @return
 */
public String convertgetFdConvert(){
 return this.fdConvert+"元";
}
```

```
/**
 * 转换值示例： 替换掉excel单元格中的"元"
 * @return
 */
public void convertsetFdConvert(String fdConvert){
  this.fdConvert = fdConvert.replace("元","");
}
```

Excel 注解说明

@Excel

属性	类型	默认值	功能
name	String	null	列名,支持name_id
needMerge	boolean	fasle	是否需要纵向合并单元格(用于含有list中,单个的单元格,合并list创建的多个row)
orderNum	String	"0"	列的排序,支持name_id
replace	String[]	{}	值得替换 导出是{a_id,b_id} 导入反过来
savePath	String	"upload"	导入文件保存路径,如果是图片可以填写,默认是upload/className/ IconEntity这个类对应的就是upload/Icon/
type	int	1	导出类型 1 是文本 2 是图片,3 是函数,10 是数字 默认是文本

width	double	10	列宽
height	double	10	列高,后期打算统一使用@ExcelTarget的height,这个会被废弃,注意
isStatistics	boolean	fasle	自动统计数据,在追加一行统计,把所有数据都和输出 这个处理会吞没异常,请注意这一点
isHyperlink	boolean	FALSE	超链接,如果是需要实现接口返回对象
isImportField	boolean	TRUE	校验字段,看看这个字段是不是导入的Excel中有,如果没有说明是错误的Excel,读取失败,支持name_id
exportFormat	String	""	导出的时间格式,以这个是否为空来判断是否需要格式化日期
importFormat	String	""	导入的时间格式,以这个是否为空来判断是否需要格式化日期
format	String	""	时间格式,相当于同时设置了exportFormat 和 importFormat

databaseFormat	String	"yyyyMMddHHmmss"	导出时间设置,如果字段是Date类型则不需要设置 数据库如果是string 类型,这个需要设置这个数据库格式,用以转换时间格式输出
numFormat	String	""	数字格式化,参数是Pattern,使用的对象是DecimalFormat
imageType	int	1	导出类型 1 从file读取 2 是从数据库中读取 默认是文件 同样导入也是一样的
suffix	String	""	文字后缀,如% 90变成90%
isWrap	boolean	TRUE	是否换行 即支持\n
mergeRely	int[]	{}	合并单元格依赖关系,比如第二列合并是基于第一列 则{0}就可以了
mergeVertical	boolean	fasle	纵向合并内容相同的单元格
fixedIndex	int	-1	对应excel的列,忽略名字
isColumnHidden	boolean	FALSE	导出隐藏列

@ExcelCollection

属性	类型	默认值	功能
id	String	null	定义ID
name	String	null	定义集合列名,支持nanm_id

orderNum	int	0	排序,支持name_id
type	Class<?>	ArrayList.class	导入时创建对象使用

单表导出实体注解源码

```
public class SysUser implements Serializable {

    /**id*/
    private String id;

    /**登录账号 */
    @Excel(name = "登录账号", width = 15)
    private String username;

    /**真实姓名*/
    @Excel(name = "真实姓名", width = 15)
    private String realname;

    /**头像*/
    @Excel(name = "头像", width = 15)
    private String avatar;

    /**生日*/
    @Excel(name = "生日", width = 15, format = "yyyy-MM-dd")
    private Date birthday;

    /**性别（1：男 2：女）*/
    @Excel(name = "性别", width = 15,dicCode="sex")
    private Integer sex;

    /**电子邮件*/
    @Excel(name = "电子邮件", width = 15)
    private String email;

    /**电话*/
    @Excel(name = "电话", width = 15)
    private String phone;

    /**状态(1：正常  2：冻结 ）*/
    @Excel(name = "状态", width = 15,replace={"正常_1","冻结_0"})
    private Integer status;
```

一对多导出实体注解源码

```
@Data
public class JeecgOrderMainPage {

/**主键*/
private java.lang.String id;
/**订单号*/
@Excel(name="订单号",width=15)
private java.lang.String orderCode;
/**订单类型*/
private java.lang.String ctype;
/**订单日期*/
@Excel(name="订单日期",width=15,format = "yyyy-MM-dd")
private java.util.Date orderDate;
/**订单金额*/
@Excel(name="订单金额",width=15)
private java.lang.Double orderMoney;
/**订单备注*/
private java.lang.String content;
/**创建人*/
private java.lang.String createBy;
/**创建时间*/
private java.util.Date createTime;
/**修改人*/
private java.lang.String updateBy;
/**修改时间*/
private java.util.Date updateTime;

@ExcelCollection(name="客户")
private List<JeecgOrderCustomer> jeecgOrderCustomerList;
@ExcelCollection(name="机票")
private List<JeecgOrderTicket> jeecgOrderTicketList;
}
```

系统权限用法

页面按钮权限用法

1.前端页面通过使用指令 v-has

```
<a-button @click="handleAdd" v-has="'user:add'" type="primary" icon="plus">添加
```

```
用户</a-button>
```

2.后台进入菜单管理页面配置按钮权限菜单

图片地址：https://static.oschina.net/uploads/img/201903/18181604_piv1.png

3.进入角色管理授权按钮（授权后即可看见按钮）

图片地址：https://static.oschina.net/uploads/img/201904/12141144_Ft4J.png

后台访问级别权限控制

1.后台请求权限控制，通过Shiro注解 @RequiresPermissions

```
@RequestMapping(value = "/add", method = RequestMethod.POST)
@RequiresPermissions("user:add")
public Result<SysUser> add(@RequestBody JSONObject jsonObject) {
```

2.后台进入菜单管理页面配置访问权限标识（选择按钮类型）

（配置方式与按钮权限一样，即同一个授权标识，可以同时控制后台请求和前台按钮显示控制）

图片地址：https://static.oschina.net/uploads/img/201903/18181604_piv1.png

3.进入角色管理授权访问权限（授权后即可访问该请求）

图片地址：https://static.oschina.net/uploads/img/201904/12141144_Ft4J.png

数据权限用法

一、功能说明

列表数据权限，主要通过数据权限控制行数据，让不同的人有不同的查看数据规则；

比如： 销售人员只能看自己的数据；销售经理可以看所有下级销售人员的数据；财务只看金额大于5000的数据等等；

二、使用说明（*有两种使用方法，以下说明以用户管理列表查询示例'配置数据规则:只查询用户账号带1的用户*）

方法A步骤如下:

A-1.新增权限菜单：进入【系统管理】-->【菜单管理】界面 新增一个权限菜单(如下图)

图片地址：https://static.oschina.net/uploads/img/201904/12110437_Eu7N.png

A-2.配置数据权限规则：找到上述1新增的菜单,点击操作列更多中的数据规则,配置,*只查询用户账号带1的用户*(如下图)

图片地址：https://static.oschina.net/uploads/img/201904/12111148_Atxy.png

A-3.角色授权：进入【系统管理】-->【角色管理】界面找到当前用户对应的角色，点击 更多->授权 操作，右侧弹出框中找到上述1菜单，点击后勾选权限规则,保存(如下图)

图片地址：https://static.oschina.net/uploads/img/201904/12112938_3kpp.png

A-4.在后台请求方法上加注解 @PermissionData 在方法上加注解是为了提高系统运行效率,这样就可以指定请求走权限过滤的逻辑,而非一棍子打死,让所有请求都去筛选一下权限(如下图)

图片地址：https://static.oschina.net/uploads/img/201904/12105637_AHvS.png

A-5.测试,访问用户管理界面发现数据被过滤了,即权限生效!

方法A的问题在于,每个请求都需要配置一个权限菜单,这样其实也很费劲,同时对于菜单管理也不是很好,鉴于此可以考虑使用方法B

方法B基于注解属性pageComponent,步骤如下：

B-1.找到需要配置权限的页面菜单,这里是用户管理菜单，

图片地址：https://static.oschina.net/uploads/img/201904/12115326_syKM.png

直接在该菜单上配置数据规则(如A-2)

B-2.角色授权（如A-3）

B-3.添加注解 (如A-4,不同的是注解上增加了一个属性)
@PermissionData(pageComponent="system/UserList") pageComponent的值和B-1中菜单的前端组件值保持一致

B-4.测试,访问用户管理界面发现数据被过滤了,即权限生效!

规则字段配置说明（非常重要）：
①条件规则：大于/大于等于/小于/小于等于/等于/包含/模糊/不等于/自定义SQL
②规则值：指定值（固定值/系统上下文变量）

三、数据权限规范说明

1.系统上下文变量

注意：数据权限配置，规则值可以填写系统上下文变量（当前登录人信息），从而根据当前登录人信息进行权限控制。

编码	描述	
sys_user_code	当前登录用户登录账号	
sys_user_name	当前登录用户真实名称	
sys_date	当前系统日期	
sys_time	当前系统时间	
sys_company_code	当前登录用户公司编号	
sys_org_code	当前登录用户部门编号	

规则值，配置写法如下：#{sys*user*code}

2.建表规范（系统标准字段）

如果需要通过当前登录人，进行数据权限控制，则业务表必须有以下系统标准字段；数据添加和编辑，jeecg会通过拦截器自动注入操作人的信息。

比如：创建人，创建时间，创建人所属部门、创建人所属公司，有了这些标准字段，就可以通过当前登录人进行数据隔离控制；

字段英文名	字段中文名
CREATE_BY	系统用户登录账号
CREATE_NAME	系统用户真实名字
SYS_ORG_CODE	登录用户所属部门
SYS_COMPANY_CODE	登录用户所属公司

3.组织机构邮编规则

JEECG组织机构支持无限层级，上下级关系通过组织机构编码实现，组织机构编码规则类似邮编方式，看下图；

邮编规则优势： 邮编规则，上下级编码固定规律，便于定位下级和上级；

图片地址：https://static.oschina.net/uploads/img/201804/16113903_qZFJ.png

Online开发

Online图表

Online表单

online 自定义按钮

1.功能简述：

通过自定义按钮功能，可以为智能表单列表添加按钮，实现扩展功能。

2.操作截图

图片地址：https://static.oschina.net/uploads/img/201904/18160618_WzDN.png

图片地址：https://static.oschina.net/uploads/img/201904/18160941_dBVA.png

3.按钮配置说明（很重要）

按钮编码：该编码在一个智能表单配置中唯一，同时js增强中定义的函数名和该编码的值需要保持一致(详见js增强描述)

按钮名称：按钮上面显示的文本。

按钮样式：可选button/link。

button:即生成的按钮显示在导航工具栏上；

link:显示在每一条数据的操作列。

动作类型：可选action/js。

action:该按钮会触发通用入口，挂接到SQL增强上（前提是SQL增强配置中配置了按钮编码对应的sql语句）。

Js:该按钮会触发JS增强中类型为“list”的配置中编写了函数名为按钮编码的函数。

按钮图标：和antd-vue的icon保持一致 参考：https://vue.ant.design/components/icon-cn/

显示表达式：按钮样式为link时起作用

4.效果演示

图片地址：https://static.oschina.net/uploads/img/201904/18161726_3XW7.png

online JS增强

1.功能简述

通过定义list/form的增强JS，实现原智能表单未实现的功能

2.操作描述

图片地址：https://static.oschina.net/uploads/img/201904/18163049_Gj60.png

图片地址：https://static.oschina.net/uploads/img/201904/18163015_W1wu.png

3.定义规则

js增强方法定义：不要使用function test(){}的形式,一律使用funname(){}的形式

js增强方法名规范：方法名唯一,且需要和自定义按钮的buttonCode保持一致或是和以下列表中的编码值保持一致

编码（方法名）	描述
beforeAdd	在新增之前调用,后续扩展after方法
beforeEdit	在编辑之前调用,该方法可以携带一个参数row，表示当前记录，后续扩展after方法
beforeDelete	在删除之前调用,该方法可以携带一个参数row，表示当前记录,后续扩展after方法
mounted	在对应页面vue钩子函数mounted中调用
created	在对应页面vue钩子函数created中调用

js增强关键字：在任意方法内，可使用that关键字,该关键字指向当前页面的vue实例,那就意味着可以用that调用任何当前页面的实例方法/属性,如加载数据that.loadData(),获取查询对象that.queryParam或是that.getQueryParams()等等。

js增强中发起后台请求： 请用this.postAction,this.getAction,this.deleteAction(参考下例)

备注：什么情况下定义的js增强方法会携带参数row？js增强最终还是挂载在按钮上或是挂在vue钩子函数中，我们列表按钮按按钮样式划分有两种,一种在列表上方，一种在列表操作列下，在操作列下的按钮，其对应的方法都会携带一个参数row,指向当前行记录

4.示例（js增强中发起后台请求）

4-1.后台定义请求方法

图片地址：https://static.oschina.net/uploads/img/201904/18203632_hShZ.png

4-2.定义js增强（此处是直接在created中发起了一个请求）

图片地址：https://static.oschina.net/uploads/img/201904/18203826_S7si.png

4-3.进入页面测试效果如下：

图片地址：https://static.oschina.net/uploads/img/201904/18203943_masP.png

后台也接收到参数

图片地址：https://static.oschina.net/uploads/img/201904/18204035_YufO.png

online SQL增强

1.功能简述

通过增强SQL，可以关联修改业务数据

2.操作截图

图片地址：https://static.oschina.net/uploads/img/201904/18170047_4o7W.png

图片地址：https://static.oschina.net/uploads/img/201904/18170813_ATd0.png

> 注意：
>
> 1.这里选择的按钮一定要是按钮类型是action的,因为js类型的是走的js增强，而按钮样式未作限制
>
> 2.这边我将按钮点击后触发的sql定义为,修改demo表的性别字段为1
>
> 3.#{id}是一种规范,id可以是任何当前表中的字段名
>
> 4.如果数据库定义的字段是数值类型的，这边是不需要加单引号('')的

3.效果演示

操作前

图片地址：https://static.oschina.net/uploads/img/201904/18172333_qPeh.png

操作后

图片地址：https://static.oschina.net/uploads/img/201904/18172404_Byqw.png

4.sql增强中可以定义系统变量(如下)

变量名称	变量释义

#{sys.sys_user_code}	登陆用户的ID
#{sys.sys_org_code}	登陆用户所属机构编码
#{sys.sys_company_code}	登陆用户所属公司编码
#{sys.sys_date}	系统日期"yyyy-MM-dd"
#{sys.sys_time}	系统时间"yyyy-MM-dd HH:mm"
#{sys.sys_user_name}	登录用户真实姓名

示例SQL： update demo set content= '#{sys.sys*usem*ame}' where id = '#{id}' (设置个人简介的内容为当前用户真实姓名)

online java增强

1.功能简述：

通过Java增强可在表单的增加、修改、和删除数据时实现额外的功能，类似spring中的aop

2.操作截图：

先定义一个类再绑定该类到java增强按钮上

图片地址：https://static.oschina.net/uploads/img/201904/18174611_Ri7X.png

图片地址：https://static.oschina.net/uploads/img/201904/18174431_lkzL.png

图片地址：https://static.oschina.net/uploads/img/201904/18174933_bG7R.png

> 注意：
>
> 1.自定义的java增强类需要实现接口implements CgformEnhanceJavaInter，并且重写方法execute的
>
> 2.如果选择spring-key 则需要在类上加上对应的注解并填入注解value,如果选择java-class则需要填写该类的路径
>
> 3.java增强是一个类似aop的功能,也就是说如果一个按钮配置了sql增强，还是可以再在这个按钮上配置java增强的,这样其实两者都会执行（上述截图就是在sql增强的按钮上配置了java增强）

执行效果如下：

图片地址：https://static.oschina.net/uploads/img/201904/18175851_zHZp.png