Jeecg-Boot 技术文档 V2.0

项目介绍

2.4+ 新版本文档请移步到：http://doc.jeecg.com

(重要的事情说三遍，注意注意注意啦！！！)

* ---

图片地址：https://static.oschina.net/uploads/img/201905/24164523_XDhg.png

> Jeecg-Boot 是一款基于SpringBoot+代码生成器的快速开发平台！采用前后端分离架构：SpringBoot，Mybatis，Shiro，JWT，Vue&Ant Design。强大的代码生成器让前端和后台代码一键生成，不需要写任何代码，保持jeecg一贯的强大，绝对是全栈开发福音！！
>
> JeecgBoot在提高UI能力的同时，降低了前后分离的开发成本，JeecgBoot还独创在线开发模式（No代码概念），一系列在线智能开发：在线配置表单、在线配置报表、在线图表设计、在线设计流程等等。
>
> JEECG宗旨是: 简单功能由Online Coding配置实现（在线配置表单、在线配置报表、在线图表设计、在线设计流程、在线设计表单），复杂功能由代码生成器生成进行手工Merge，既保证了智能又兼顾了灵活;
>
> 业务流程采用工作流来实现、扩展出任务接口，供开发编写业务逻辑，表单提供多种解决方案：表单设计器、online配置表单、编码表单。同时实现了流程与表单的分离设计（松耦合）、并支持任务节点灵活配置，既保证了公司流程的保密性，又减少了开发人员的工作量。

- 官方网站： http://www.jeecg.com
- 源码下载： https://github.com/zhangdaiscott/jeecg-boot
- QQ交流群：②769925425、①284271917、③816531124
- 在线演示： http://boot.jeecg.com
- 版本日志： http://www.jeecg.com/doc/log

- 新手帮助： [快速入门](http://www.jeecg.com/doc/quickstart)	[常见问题](http://www.jeecg.com/doc/qa)	[视频教程](https://www.bilibili.com/video/BV1Y541147m1)	[寻求帮助](https://github.com/zhangdaiscott/jeecg-boot/issues)

技术架构：

后端技术： SpringBoot_2.1.3.RELEASE + Mybatis-plus_3.1.2 + Shiro_1.4.0 + Jwt_3.7.0 + 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://www.antdv.com/docs/vue/introduce

报表UI组件：viser-vue

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

VUE基础知识：

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

系统截图

大屏模板

图片地址：https://static.oschina.net/uploads/img/201912/25133248_Ag1C.jpg

图片地址：https://static.oschina.net/uploads/img/201912/25133301_k9Kc.jpg

PC端

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

图片地址
：https://oscimg.oschina.net/oscnet/ba807921197596ba56f495d4b22ee3280ca.jpg

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

图片地址：https://static.oschina.net/uploads/img/201904/14160813_KmXS.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/201904/14161004_bxQ4.png

在线接口文档

图片地址：https://static.oschina.net/uploads/img/201908/27095258_M2Xq.png

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

报表

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

流程

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

图片地址：https://static.oschina.net/uploads/img/201904/14160917_9Ftz.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/201907/05165142_yyQ7.png

手机端

图片地址
：https://oscimg.oschina.net/oscnet/da543c5d0d57baab0cecaa4670c8b68c521.jpg

图片地址：https://oscimg.oschina.net/oscnet/fda4bd82cab9d682de1c1fbf2060bf14fa6.jpg

PAD端

图片地址：https://oscimg.oschina.net/oscnet/e90fef970a8c33790ab03ffd6c4c7cec225.jpg

图片地址：https://oscimg.oschina.net/oscnet/d78218803a9e856a0aa82b45efc49849a0c.jpg

图片地址：https://oscimg.oschina.net/oscnet/0404054d9a12647ef6f82cf9cfb80a5ac02.jpg

图片地址：https://oscimg.oschina.net/oscnet/59c23b230f52384e588ee16309b44fa20de.jpg

Online初体验

一分钟快速学习 (https://segmentfault.com/a/1190000019448442)

欢迎吐槽，欢迎star~

![GitHub stars(https://img.shields.io/github/stars/zhangdaiscott/jeecg-boot.svg?style=social&label=Stars)](https://github.com/zhangdaiscott/jeecg-boot)

![GitHub forks(https://img.shields.io/github/forks/zhangdaiscott/jeecg-boot.svg?style=social&label=Fork)](https://github.com/zhangdaiscott/jeecg-boot)

新手入门教程

2.4+ 新版本文档请移步到：http://doc.jeecg.com

(重要的事情说三遍，注意注意注意啦！！！)

* ---

Jeecg-Boot入门教程必看（新手学习）

> 我们精心制作本教程，方便开源用户和新入职成员快速掌握Jeecg-Boot开发。根据我们的实际经验，通过此教程学习，刚入行的毕业生也可以顺利的进行开发，所以请耐心完成本套教程的学习（有一定工作经验的，可以忽略自己熟悉的章节）。
>
> 1. 开发环境搭建　　http://doc.jeecg.com/2043873
> 2. 项目如何启动　　http://doc.jeecg.com/2043874
> 3. JeecgBoot学习视频　http://www.jeecg.com/doc/video
> 4. 代码生成器使用　　http://doc.jeecg.com/2043916
> 5. 常见问题贴：http://jeecg.com/doc/qa
> 6. 必看学习资料
>
> ES6　　| http://es6.ruanyifeng.com
>
> Vue　　| https://cn.vuejs.org/v2/guide
>
> Ant Design of Vue | https://vue.ant.design/docs/vue/introduce-cn
>
> Jeecg-Boot文档　| http://doc.jeecg.com
>
> 7. 基础知识学习（Vue全家桶、Springboot） 参考下面《基础知识学习视频》
> 8. 如果你发现bug，请点击这里反馈 https://github.com/zhangdaiscott/jeecg-boot/issues

《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/1Yv1ttP1_b6ORrTGLPQ1n1g 提取码：gcjo
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/1wL09TPdDlFtKrLTeKSv_Yw　提取码：lw5z
链接：https://pan.baidu.com/s/1-Kdlthotm27dJUijwm4sSQ　提取码：03sv

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

> 链接：https://pan.baidu.com/s/16z9qNtyk24bsrZxRFBHP2w 提取码：pagv

- node-v12.13.1-x64.msi
- yarn-1.21.1.msi
- ideaIU-2019.2.3.exe
- jdk-8u191-windows-x64.exe
- apache-maven-3.5.4.zip
- redis64-3.0.501
- mysql-5.7.26-winx64.zip

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	https://my.oschina.net/jeecg/blog/4277939
2	Yarn安装	下载包工具	https://my.oschina.net/jeecg/blog/4278012
3	WebStorm安装与使用	WEB前端开发工具	https://blog.csdn.net/u011781521/article/details/53558979

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

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/dujing_15620553271/article/details/79126676
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=345669)

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=345669)

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：

• 前台项目启动

1. 安装依赖包

2. 配置后台接口服务地址

3. 启动项目

一、后端项目启动

jeecg-boot 从v2.0版本，重构成maven多模块项目，启动项目：jeecg-boot-module-system

项目结构说明：
├─jeecg-boot-parent（父POM： 项目依赖、modules组织）
│ ├─jeecg-boot-base-common（共通Common模块： 底层工具类、注解、接口）
│ ├─jeecg-boot-module-system （系统管理模块： 系统管理、权限等功能） -- 默认作为启动项目
│ ├─jeecg-boot-module-{?}（自己扩展新模块项目，启动的时候，在system里面引用即可）

（1）执行数据库初始化sql

sql文件地址：jeecg-boot/db/jeecg-boot-mysql.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

注意： 不能直接访问后台，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

注意：（如果启动报错的话，请升级node版本，把依赖node_modules删了，重新yarn install）

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

> JEECG存在自定义JAR包，放在自己的Maven私服上面，所以有的时候会遇到下载失败。
>
> 一般遇到下载失败的情况，是因为用户设置了本地镜像，导致无法从JEECG私服下载资源，参考下面的方式进行镜像排除配置即可。

找到 maven老家 conf/settings.xml，

在<mirrors>标签内增加下面方式的阿里云maven镜像（删除自己的镜像配置），最终结果见下面：

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

然后执行maven命令，依赖就会顺利下载；此配置重点在这句话<mirrorOf>*,!jeecg,!jeecg-snapshots</mirrorOf>

如果不加这句话，默认所有的依赖都会去阿里云仓库下载，加上后jeecg的依赖包就可以从jeecg私服下载了。

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。

数据库切换Oracle/SqlServer/Postgresql

Oracle数据

1.添加oracle驱动，修改pom.xml

```
<!-- oracle驱动 -->
<dependency>
<groupId>com.oracle</groupId>
<artifactId>ojdbc6</artifactId>
<version>11.2.0.3</version>
</dependency>
```

2.修改数据库连接

```
修改druid配置
validationQuery: SELECT 1 FROM DUAL

driver-class-name: oracle.jdbc.OracleDriver
url: jdbc:oracle:thin:@192.168.1.200:1521:ORCL
username: jeecgboot
password: jeecgboot
```

SQL server数据

1.添加SQL server驱动，修改pom.xml

```
<!--  sqlserver-->
<dependency>
<groupId>com.microsoft.sqlserver</groupId>
<artifactId>sqljdbc4</artifactId>
<version>4.0</version>
<scope>runtime</scope>
</dependency>
```

2.修改数据库连接

```
修改druid配置
validationQuery: SELECT 1
filters: stat,slf4j


driver-class-name: com.microsoft.sqlserver.jdbc.SQLServerDriver
url: jdbc:sqlserver://192.168.1.200:1433;SelectMethod=cursor;DatabaseName=jeecg-boot
username: sa
password: SA
```

postgresql数据库

1.添加postgresql驱动，修改pom.xml

(根据不同版本的数据库引入对应版本的驱动，下载地址：https://jdbc.postgresql.org/download.html)

```
<!--  postgresql-->
<dependency>
  <groupId>postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>42.2.5</version>
</dependency>
```

2.修改数据库连接

```
增加spring下的配置
spring:
  #postgresql 报错问题
  jpa:
      properties:
          hibernate:
              temp:
                  use_jdbc_metadata_defaults: false

修改druid配置
validationQuery: SELECT 1


url: jdbc:postgresql://localhost:5432/postgres
```

```
username: postgres
password: root
driver-class-name: org.postgresql.Driver
```

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;
}
}
```

直接访问请求http://localhost:8080/jeecg-boot/test/jeecgDemo/hello 会提示token无效，所以需要配置下拦截器ShiroConfig排除。

```
配置文件： jeecg-boot-module-system/src/main/java/org/jeecg/config/ShiroConfig.java
加入配置：filterChainDefinitionMap.put("/test/jeecgDemo/hello", "anon");
```

图片地址：https://static.oschina.net/uploads/img/201908/08231321_wyjh.png

再访问请求http://localhost:8080/jeect-boot/test/jeecgDemo/hello，会返回结果如下：

```
{
"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

JAR部署方案

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

> 目前 jeecg-boot-module-system 作为启动和打包项目。

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

5、修改pom.xml加上打包插件（如果已经有了，就不需要加了）

```
<build>
   <plugins>
<plugin>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
  </plugins>
</build>
```

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-module-system-2.0.0.jar

Linux下后台进程启动命令：
nohup java -jar jeecg-boot-module-system-2.0.0.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

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

重要提示：

```
    1. 后台服务接口地址，一定要配置外网的IP或者域名，配置内网域名后台访问不到的。
    2. 后台启动默认名字jeecg-boot，不建议修改。如果需要修改，请自行替换此文中所有提
到项目名jeecg-boot的地方；
        同时修改前端代码 src/utils/request.js，里面的项目名字。
```

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

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;
}
}
}
```

五、nginx 开启压缩，提高首页访问效率

https://github.com/zhangdaiscott/jeecg-boot/issues/88

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部署方案

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（jeecg-boot-module-system）

（1）后台项目jeecg-boot-module-system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-module-system</artifactId>
<version>2.0.0</version>
<packaging>war</packaging>
```

2、pom.xml文件删除插件spring-boot-maven-plugin

下面配置删除

```
<build>
  <plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
  </plugins>
</build>
```

3、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);
    }
}
```

4、修改配置文件（数据库和redis配置）

- 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

极简部署方案

基于 spring boot 特性

1、首先修改配置，去掉项目名 jeecg-boot

```
ant-design-jeecg-vue/src/utils/request.js
ant-design-jeecg-vue/public/index.html
```

图片地址：https://static.oschina.net/uploads/img/201905/29171344_KCyn.png

图片地址：https://static.oschina.net/uploads/img/201905/29171615_uhYZ.png

2、修改路由History 模式为“hash”

```
src/router/index.js
```

图片地址：https://static.oschina.net/uploads/img/201905/29165605_bIXK.png

然后 ant-design-jeecg-vue 运行 build

```
npm run build
```

3、然后将编译之后dist下的文件复制到 jeecg-boot-module-system 项目的 /src/main/resources/static 目录下。

4、修改springboot项目启动，根路径访问页面为 index.html

```
jeecg-boot-module-system\src\main\java\org\jeecg\config\WebMvcConfiguration
```

```
/**
 * 访问根路径默认跳转 index.html页面 （简化部署方案： 可以把前端打包直接放到项目的 webapp，上面的配置）
 */
@Override
public void addViewControllers(ViewControllerRegistry registry) {
registry.addViewController("/").setViewName("index.html");
}
```

5、修改springboot项目的启动名字，去掉项目名 jeecg-boot

```
jeecg-boot-module-system/src/main/resources/application-dev.yml
```

图片地址：https://static.oschina.net/uploads/img/201905/29171247_YzQ5.png

6、重新启动项目，访问 http://localhost:8080/ 就可以看到效果

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-module-system/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-module-system/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: jeecg-boot-module-system/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： jeecg-boot-module-system/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。
>
> 代码生成器工具和模板在项目 jeecg-boot-module-system 中。
>
> 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

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';
    window._CONFIG['pdfDomainURL'] = 'http://localhost:8080/jeecg-
boot/sys/common/pdf/pdfPreviewIframe';
    window._CONFIG['casPrefixUrl'] = 'http://cas.example.org:8443/cas';
  </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']	-
pdf文件预览地址	window._CONFIG['pdfDomainURL']	-
CAS服务器地址	window._CONFIG['casPrefixUrl']	-

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

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

yarn install ｜ 下载依赖(推荐)

cnpm install ｜ 下载依赖

npm install ｜ 下载依赖(不建议用，容易出各种诡异问题&慢)

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'));
});
```

自定义组件

常用组件文档

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,
```

<a-select/> 下拉选项滚动错位的解决方法

问题描述

当使用了 a-modal 或其他带有滚动条的组件时，使用a-select组件并打开下拉框时滚动滚动条，就会导致错位的问题产生。

解决方法

大多数情况下，在 a-select 上添加一个 getPopupContainer 属性，值为node => node.parentNode即可解决。

但是如果遇到 a-select 标签层级过深的情况，可能仍然会显示异常，只需要多加几个.parentNode（例：node => node.parentNode.parentNode.parentNode）多尝试几次直到解决问题即可。

代码示例

```
<a-select
    placeholder="请选择展示模板"
    :options="dicts.displayTemplate"
    :getPopupContainer="node => node.parentNode"
/>
```

JAsyncTreeList 异步数列表组件使用说明

引入组件

```
import JTreeTable from '@/components/jeecg/JTreeTable'
export default {
  components: { JTreeTable }
}
```

所需参数

参数	类型	必填	说明
rowKey	String	非必填	表格行 key 的取值，默认为"id"
columns	Array	必填	表格列的配置描述，具体见Antd官方文档
url	String	必填	数据查询url
childrenUrl	String	非必填	查询子级时的url，若不填则使用url参数查询子级

queryKey	String	非必填	根据某个字段查询，如果传递 id 就根据 id 查询，默认为 parentId
queryParams	Object	非必填	查询参数，当查询参数改变的时候会自动重新查询，默认为{}
topValue	String	非必填	查询顶级时的值，如果顶级为0，则传0，默认为null
tableProps	Object	非必填	自定义给内部table绑定的props

代码示例

```
<template>
 <a-card :bordered="false">
  <j-tree-table :url="url" :columns="columns" :tableProps="tableProps"/>
 </a-card>
</template>

<script>
 import JTreeTable from '@/components/jeecg/JTreeTable'

 export default {
  name: 'AsyncTreeTable',
  components: { JTreeTable },
  data() {
   return {
    url: '/api/asynTreeList',
    columns: [
     { title: '菜单名称', dataIndex: 'name' },
     { title: '组件', dataIndex: 'component' },
     { title: '排序', dataIndex: 'orderNum' }
    ],
    selectedRowKeys: []
   }
  },
```

```
    computed: {
      tableProps() {
        let _this = this
        return {
          // 列表项是否可选择
          // 配置项见：https://vue.ant.design/components/table-cn/#rowSelection
          rowSelection: {
            selectedRowKeys: _this.selectedRowKeys,
            onChange: (selectedRowKeys) => _this.selectedRowKeys = selectedRowKeys
          }
        }
      }
    }
  }
</script>
```

JCheckbox 使用文档

说明: antd-vue checkbox组件处理的是数组，用起来不是很方便，特二次封装，使用时只需处理字符串即可

参数配置

参数	类型	必填	说明
options	array		checkbox需要配置的项，是个数组，数组中每个对象包含两个属性:label(用于显示)和value(用于存储)

使用示例

```
<template>
  <a-form :form="form">
    <a-form-item label="v-model式用法">
      <j-checkbox v-model="sport" :options="sportOptions"></j-checkbox><span>{{ sport }}</span>
    </a-form-item>
```

```
    <a-form-item label="v-decorator式用法">
      <j-checkbox v-decorator="['sport']" :options="sportOptions"></j-
checkbox><span>{{ getFormFieldValue('sport') }}</span>
    </a-form-item>
  </a-form>
</template>

<script>
  import JCheckbox from '@/components/jeecg/JCheckbox'
  export default {
    components: {JCheckbox},
    data() {
      return {
        form: this.$form.createForm(this),
        sport:'',
        sportOptions:[
          {
            label:"足球",
            value:"1"
          },{
            label:"篮球",
            value:"2"
          },{
            label:"乒乓球",
            value:"3"
          }]
      }
    },
    methods: {
     getFormFieldValue(field){
       return this.form.getFieldValue(field)
     }
    }
  }
</script>
```

JCodeEditor 使用文档

说明: 一个简易版的代码编辑器，支持语法高亮

参数配置

参数	类型	必填	说明
language	string		表示当前编写代码的类型 javascript/html/css/sql
placeholder	string		placeholder
lineNumbers	Boolean		是否显示行号
fullScreen	Boolean		是否显示全屏按钮
zIndex	string		全屏以后的z-index

使用示例

```
<template>
  <div>
    <j-code-editor
      language="javascript"
      v-model="editorValue"
      :fullScreen="true"
      style="min-height: 100px"/>
    {{ editorValue }}
  </div>
</template>

<script>
 import JCodeEditor from '@/components/jeecg/JCodeEditor'
 export default {
    components: {JCodeEditor},
    data() {
      return {
        form: this.$form.createForm(this),
        editorValue:'',
      }
    }
 }
</script>
```

JFormContainer 使用文档

说明: 暂用于表单禁用

使用示例

```
<!-- 在form下直接写这个组件，设置disabled为true就能将此form中的控件禁用 -->
  <a-form layout="inline" :form="form" >
    <j-form-container disabled>
      <!-- 表单内容省略..... -->
    </j-form-container>
  </a-form>
```

JImportModal 使用文档

说明: 用于列表页面导入excel功能

使用示例

```
<template>
  <!--  此处省略部分代码..... -->
  <a-button @click="handleImportXls" type="primary" icon="upload">导入</a-
button>
  <!--  此处省略部分代码..... -->
  <j-import-modal ref="importModal" :url="getImportUrl()" @ok="importOk"></j-
import-modal>
  <!--  此处省略部分代码..... -->
</template>

<script>
  import JCodeEditor from '@/components/jeecg/JCodeEditor'
  export default {
    components: {JCodeEditor},
    data() {
      return {
        //省略代码.....
      }
    },
```

```
    methods:{
      //省略部分代码.....
      handleImportXls(){
        this.$refs.importModal.show()
      },
      getImportUrl(){
         return '你自己处理上传业务的后台地址'
      },
      importOk(){
        this.loadData(1)
      }
    }
  }
</script>
```

JSelectMultiple 多选下拉组件

online用 实际开发请使用components/dict/JMultiSelectTag

JSlider 滑块验证码

使用示例

```
<template>
  <div style="width: 300px">
    <j-slider @onSuccess="sliderSuccess"></j-slider>
  </div>
</template>

<script>
  import JSlider from '@/components/jeecg/JSlider'
  export default {
    components: {JSlider},
    data() {
      return {
        form: this.$form.createForm(this),
        editorValue:'',
      }
    },
    methods:{
      sliderSuccess(){
```

```
      console.log("验证完成")
    }
  }
}
</script>
```

JTreeSelect 树形下拉组件

异步加载的树形下拉组件

参数配置

参数	类型	必填	说明
placeholder	string		placeholder
dict	string		表名,显示字段名,存储字段名拼接的字符串
pidField	string		父ID的字段名
pidValue	string		根节点父ID的值 默认'0' 不可以设置为空,如果想使用此组件，而数据库根节点父ID为空，请修改之

使用示例

```
<template>
 <a-form>
  <a-form-item label="树形下拉测试" style="width: 300px">
   <j-tree-select
    v-model="departId"
    placeholder="请选择部门"
    dict="sys_depart,depart_name,id"
    pidField="parent_id">
   </j-tree-select>
   {{ departId }}
```

```
    </a-form-item>
  </a-form >
</template>

<script>
  import JTreeSelect from '@/components/jeecg/JTreeSelect'
  export default {
    components: {JTreeSelect},
    data() {
      return {
        departId:""
      }
    }
  }
</script>
```

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（可以作废了，建议@Dict注解进行翻译）
>
> @Dict用法
>
> 字典翻译注解@Dict，用于列表字段字典翻译（比如字段sex存的值是1，会自动生成一个翻译字段 sex_dictText 值是‘男’ ）。详细文档：http://jeecg-boot.mydoc.io/?t=345678

[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"/>
```

- 从数据库表获取字典数据，dictCode格式说明: 表名,文本字段,取值字段

```
<j-dict-select-tag v-model="queryParam.username" placeholder="请选择用户名称"
                   dictCode="sys_user,realname,id"/>
```

注意：

当从数据库表获取字典数据的时候支持写查询条件进行过滤数据：

```
<j-dict-select-tag v-model="queryParam.username" placeholder="请选择用户名称"
dictCode="sys_user,realname,id,sex = '2'"/>
```

上述dictCode = "sys*user,realname,id,sex = '2'"表示从sys*user表中查询数据且只查询sex='2'的数据

[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/JDictSelectUtil'
```

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);
```

JEditableTable 帮助文档

JEditableTable 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，默认false
actionButton	boolean		是否显示操作按钮，包括"新增"、"删除"，默认false
rowNumber	boolean		是否显示行号，默认false
rowSelection	boolean		是否可选择行，默认false
dragSort	boolean		是否可拖动排序，默认false
dragSortKey	string		拖动排序存储的Key，无需定义在columns内也能在getValues()时获取到值，默认orderNum
maxHeight	number		设定最大高度(px)，默认400
disabledRows	object		设定禁用的行，被禁用的行无法被选择和编辑，配置方法可以查看示例
disabled	boolean		是否禁用所有行，默认false

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 `JEditableTableUtil.FormTypes`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。留空会自动计算百分比
placeholder	string		表单预期值的提示信息，可以使用 `${...}`变量替换文本（详见`${...} 变量使用方式`）
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 [validateRules 配置规则](#validaterules-配置规则)

props	object		设置添加给表单元素的自定义属性，例如:`props:{title: 'show title'}`
disabled	boolean		是否禁用当前列，默认false

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
allowInput	boolean		是否允许用户输入内容，并创建新的内容

dictCode	String		数据字典Code，若options也有值，则拼接在options后面

options 所需参数

参数	类型	必填	说明
text	string		显示标题
value	string		真实值
~~title~~	~~string~~		~~显示标题（已废弃，若同时填写了 title 和 text 那么优先使用 text）~~

当 type=upload 时所需的参数

参数	类型	必填	说明
action	string		上传文件路径
token	boolean		上传的时候是否传递token
responseName	string		若要从上传成功后从response中取出返回的文件名，那么这里填后台返回的包含文件名的字段名

当 type=slot 时所需的参数

参数	类型	必填	说明
slotName	string		slot的名称

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
- handler 自定义函数校验，使用方法请见示例五(#示例五)
- message 当验证未通过时显示的提示文本，可以使用${...}变量替换文本（详见${...} 变量使用方式）
- 配置示例请看示例二(#示例二)

事件

事件名	触发时机	参数
added	当添加行操作完成后触发	
deleted	当删除行操作完成后触发（批量删除操作只会触发一次）	`deleteIds` 被逻辑删除的id
selectRowChange	当行被选中或取消选中时触发	`selectedRowIds` 被选中行的id
valueChange	当数据发生改变的时候触发的事件	`{ type, row, column, value, target }` Event对象

方法

关于方法的如何调用的问题，请在**FAQ**中查看方法如何调用(#方法如何调用)

initialize

用于初始化表格（清空表格）

- 参数: 无
- 返回值: 无

resetScrollTop

重置滚动条Top位置

- 参数:

参数名	类型	必填	说明

top	number		新top位置，留空则滚动到上次记录的位置，用于解决切换tab选项卡时导致白屏以及自动将滚动条滚动到顶部的问题

• 返回值: 无

add

主动添加行，默认情况下，当用户的滚动条已经在底部的时候，会将滚动条固定在底部，即添加后无需用户手动滚动，而会自动滚动到底部

• 参数:

参数名	类型	必填	说明
num	number		添加几行，默认为1
forceScrollToBottom	boolean		是否在添加后无论用户的滚动条在什么位置都强制滚动到底部，默认为false

• 返回值: 无

removeRows

主动删除一行或多行

• 参数:

参数名	类型	必填	说明
id	string 或 array		被删除行的id。如果要删除一个，可以直接传id，如果要删除多个，需要将多个id封装成一个数组传入

• 返回值: 无

removeSelectedRows

主动删除被选中的行

- 参数: 无
- 返回值: 无

getValues

用于获取表格里所有表单的值，可进行表单验证

- 参数:

参数名	类型	必填	说明
callback	function		获取值的回调方法，会传入`error`和`values`两个参数。`error`：未通过验证的数量，当等于`0`时代表验证通过；`values`：获取的值（即使未通过验证该字段也有数据）
validate	boolean		是否进行表单验证，默认为`true`，设为`false`则代表忽略表单验证
rowIds	array		默认返回所有行的数据，如果传入了`rowIds`，那么就会只返回与该`rowIds`相匹配的数据，如果没有匹配的数据，就会返回空数组

- 返回值: 无

getValuesSync

getValues的同步版，会直接将获取到的数据返回

- 参数:

参数名	类型	必填	说明

options	object		选项，详见下方所需参数

- - options 所需参数

参数名	类型	必填	说明
validate	boolean		是否进行表单验证，默认为`true`，设为`false`则代表忽略表单验证
rowIds	array		默认返回所有行的数据，如果传入了`rowIds`，那么就会只返回与该`rowIds`相匹配的数据，如果没有匹配的数据，就会返回空数组

- 返回值: object
- error 未通过验证的数量，当等于0时代表验证通过
- values 获取的值（即使未通过验证该字段也有数据）
- 使用示例

```
let { error, values } = this.$refs.editableTable.getValuesSync({ validate: true, rowIds: ['rowId1', 'rowId2'] })
if (error === 0) {
    console.log('表单验证通过，数据：', values);
} else {
    console.log('未通过表单验证，数据：', values);
}
```

getValuesPromise

getValues的promise版，会在resolve中传入获取到的值，会在reject中传入失败原因，例如VALIDATE*NO*PASSED

- 参数:

参数名	类型	必填	说明

validate	boolean		同`getValues`的`validate`参数
rowIds	array		默认返回所有行的数据，如果传入了`rowIds`，那么就会只返回与该`rowIds`相匹配的数据，如果没有匹配的数据，就会返回空数组

- 返回值: Promise

getDeleteIds

用于获取被逻辑删除的行的id，返回一个数组，用户可将该数组传入后台，并进行批量删除

- 参数: 无
- 返回值: array

getAll

获取所有的数据，包括values、deleteIds

会在resolve中传入获取到的值：{values, deleteIds}

会在reject中传入失败原因，例如VALIDATE_NO_PASSED

- 参数:

参数名	类型	必填	说明
validate	boolean		同`getValues`的`validate`参数

- 返回值: Promise

setValues

主动设置表格中某行某列的值

- 参数:

参数名	类型	必填	说明

values	array		传入一个数组，数组中的每项都是一行的新值，具体见下面的示例

- 返回值: 无
- 示例：

```
setValues([
    {
        rowKey: id1, // 行的id
        values: { // 在这里 values 中的 name 是你 columns 中配置的 key
            'name': 'zhangsan',
            'age': '20'
        }
    },
    {
        rowKey: id2,
        values: {
            'name': 'lisi',
            'age': '23'
        }
    }
])
```

${...} 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

JEditableTableUtil 使用说明

在之前配置columns时提到过JEditableTableUtil这个工具类，那么如果想要知道详细的使用说明就请看这里

export 的常量

FormTypes

这是配置columns.type时用到的常量值，其中包括

- normal 默认，直接显示值，不渲染表单
- input 显示输入框
- inputNumber 显示数字输入框
- checkbox 显示多选框
- select 显示选择器（下拉框）
- date 日期选择器
- datetime 日期时间选择器
- upload 上传组件（文件域）
- slot 自定义插槽

VALIDATE*NO*PASSED

在判断表单验证是否通过时使用，如果 reject 的值 === VALIDATE*NO*PASSED 则代表表单验证未通过，你可以做相应的其他处理，反之则可能是发生了报错，可以使用 console.error 输出

封装的方法

validateTables

当你的页面中存在多个JEditableTable实例的时候，如果要获取每个实例的值、判断表单验证是否通过，就会让代码变得极其冗余、繁琐，于是我们就将该操作封装成了一个函数供你调用，它可以同时获取并验证多个JEditableTable实例的值，只有当所有实例的表单验证都通过后才会返回值，否则将会告诉你具体哪个实例没有通过验证。具体使用方法请看下面的示例

- 参数:

参数名	类型	必填	说明
cases	array		传入一个数组，数组中的每项都是一个JEditableTable的实例

- 返回值: Promise
- 示例：

```
import { validateTables, VALIDATE_NO_PASSED } from '@/utils/JEditableTableUtil'
// 封装cases
let cases = []
cases.push(this.$refs.editableTable1)
cases.push(this.$refs.editableTable2)
cases.push(this.$refs.editableTable3)
cases.push(this.$refs.editableTable4)
```

```
cases.push(this.$refs.editableTable5)
// 同时验证并获取多个实例的值
validateTables(cases).then((all) => {
    // all 是一个数组，每项都对应传入cases的下标，包含values和deleteIds
    console.log('所有实例的值：', all)
}).catch((e = {}) => {
    // 判断表单验证是否未通过
    if (e.error === VALIDATE_NO_PASSED) {
        console.log('未通过验证的实例下标:', e.index)
    } else {
        console.error('发生异常:', e)
    }
})
```

FAQ

方法如何调用？

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。

假如我要调取initialize方法，就可以这么写：this.$refs.editableTable.initialize()

如何获取表单的值？

使用getValue方法进行获取，详见示例三(#示例三)

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，当然你也可以在代码中主动调用新增方法或修改，具体见上方的方法介绍。

为什么使用了ATab组件后，切换选项卡会导致白屏或滚动条位置会归零？

在ATab组件中确实会导致滚动条位置归零，且不会触发onscroll方法，所以无法动态加载行，导致白屏的问题出现。

解决方法是在ATab组件的onChange事件触发时执行实例提供的resetScrollTop()方法即可，但是需要注意的是：代码主动改变ATab的activeKey不会触发onChange事件，还需要你手动调用下。

• 示例

```
<template>
  <a-tabs @change="handleChangeTab">
    <a-tab-pane tab="表格1" :forceRender="true" key="1">
      <j-editable-table
        ref="editableTable1"
        :loading="tab1.loading"
        :columns="tab1.columns"
        :dataSource="tab1.dataSource"/>
    </a-tab-pane>
    <a-tab-pane tab="表格2" :forceRender="true" key="2">
      <j-editable-table
        ref="editableTable2"
        :loading="tab2.loading"
        :columns="tab2.columns"
        :dataSource="tab2.dataSource"/>
    </a-tab-pane>
  </a-tabs>
</template>
```

```
/*--- 忽略部分代码片段 ---*/
methods: {

    /** 切换tab选项卡的时候重置editableTable的滚动条状态 */
    handleChangeTab(key) {
      this.$refs[`editableTable${key}`].resetScrollTop()
    }

}
/*--- 忽略部分代码片段 ---*/
```

slot(自定义插槽)如何使用？

代码示例请看：示例四(slot)(#示例四(slot))

示例一

```
<j-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

```
import { FormTypes } from '@/utils/JEditableTableUtil'

/*--- 忽略部分代码片断 ---*/
columns: [
    {
        title: '名称',
        key: 'name',
        type: FormTypes.input,
        placeholder: '请输入${title}',
        defaultValue: '称名',
        // 表单验证规则
        validateRules: [
            {
                required: true, // 必填
                message: '${title}不能为空' // 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'
            }
        ]
    },
    {
        title: '年龄',
        key: 'age',
        type: FormTypes.inputNumber,
        placeholder: '请输入${title}',
        defaultValue: 18,
        validateRules: [{required: true, message: '${title}不能为空'}]
    }
```

```
]
/*--- 忽略部分代码片断 ---*/
```

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')
    }
})
```

示例四(slot)

```
<template>
    <j-editable-table :columns="columns" :dataSource="dataSource">
        <!-- 定义插槽 -->
        <!-- 这种定义插槽的写法是vue推荐的新版写法
（https://cn.vuejs.org/v2/guide/components-slots.html#具名插槽），旧版已被废弃的写法不再支持 -->
        <!-- 若webstorm这样写报错，请看这篇文章
：https://blog.csdn.net/lxq_9532/article/details/81870651 -->
        <template v-slot:action="props">
            <a @click="handleDelete(props)">删除</a>
        </template>
    </j-editable-table>
</template>
<script>
    import { FormTypes } from '@/utils/JEditableTableUtil'
    import JEditableTable from '@/components/jeecg/JEditableTable'
    export default {
        components: { JEditableTable },
        data() {
```

```
        return {
            columns: [
                // ...
                {
                    title: '操作',
                    key: 'action',
                    width: '8%',
                    type: FormTypes.slot, // 定义该列为 自定义插值列
                    slotName: 'action' // slot 的名称，对应 v-slot 冒号后面和等号前面的内容
                }
            ]
        }
    },
    methods: {
        /* a 标签的点击事件，删除当前选中的行 */
        handleDelete(props) {
            // 参数解释
            // props.index ：当前行的下标
            // props.text ：当前值，可能是defaultValue定义的值，也可能是从dataSource中取出的值
            // props.rowId ：当前选中行的id，如果是新增行则是临时id
            // props.column ：当前操作的列
            // props.getValue ：这是一个function，执行后可以获取当前行的所有值（禁止在template中使用）
            //                  例：const value = props.getValue()
            // props.target ：触发当前事件的实例，可直接调用该实例内的方法（禁止在template中使用）
            //                  例：target.add()

            // 使用实例：删除当前操作的行
            let { rowId, target } = props
            target.removeRows(rowId)
        }
    }
}
</script>
```

示例五

```
// 该示例是自定义函数校验
columns: [
```

```
{
    title: '字段名称',
    key: 'dbFieldName',
    type: FormTypes.input,
    defaultValue: '',
    validateRules: [
        {
            // 自定义函数校验 handler
            handler(type, value, row, column, callback, target) {
                // type 触发校验的类型（input、change、blur）
                // value 当前校验的值
                // callback(flag, message) 方法必须执行且只能执行一次
                //          flag = 是否通过了校验，不填写或者填写 null 代表不进行任何操作
                //          message = 提示的类型，默认使用配置的 message
                // target 行编辑的实例对象

                if (type === 'blur') {

                    if (value === 'abc') {
                        callback(false, '${title}不能是abc') // false = 未通过，可以跟自定义提示
                        return
                    }

                    let { values } = target.getValuesSync({ validate: false })
                    let count = 0
                    for (let val of values) {
                        if (val['dbFieldName'] === value) {
                            if (++count >= 2) {
                                callback(false, '${title}不能重复')
                                return
                            }
                        }
                    }
                    callback(true) // true = 通过验证
                } else {
                    callback() // 不填写或者填写 null 代表不进行任何操作
                }
            },
            message: '${title}默认提示'
        }
    ]
},
```

```
]
```

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

JSearchSelectTag 字典表的搜索组件

下拉搜索组件,支持异步加载,异步加载用于大数据量的字典表

1.参数配置

参数	类型	必填	说明
placeholder	string		placeholder
disabled	Boolean		是否禁用
dict	string		表名,显示字段名,存储字段名拼接而成的字符串,如果提供了dictOptions参数则此参数可不填
dictOptions	Array		多选项,如果dict参数未提供,可以设置此参数加载多选项
async	Boolean		是否支持异步加载,设置成true,则通过输入的内容加载远程数据,否则在本地过滤数据,默认false

2.使用示例

https://static.oschina.net/uploads/img/201905/25160321_fPFb.png

```
<template>
  <a-form>
    <a-form-item label="下拉搜索" style="width: 300px">
      <j-search-select-tag
        placeholder="请做出你的选择"
        v-model="selectValue"
        :dictOptions="dictOptions">
      </j-search-select-tag>
      {{ selectValue }}
    </a-form-item>
```

```
    <a-form-item label="异步加载" style="width: 300px">
      <j-search-select-tag
        placeholder="请做出你的选择"
        v-model="asyncSelectValue"
        dict="sys_depart,depart_name,id"
        :async="true">
      </j-search-select-tag>
      {{ asyncSelectValue }}
    </a-form-item>
  </a-form >
</template>

<script>
 import JSearchSelectTag from '@/components/dict/JSearchSelectTag'
 export default {
   components: {JSearchSelectTag},
   data() {
     return {
       selectValue:"",
       asyncSelectValue:"",
       dictOptions:[{
         text:"选项一",
         value:"1"
       },{
         text:"选项二",
         value:"2"
       },{
         text:"选项三",
         value:"3"
       }]
     }
   }
 }
</script>
```

JMultiSelectTag 多选组件

下拉/checkbox

1.参数配置

参数	类型	必填	说明

placeholder	string		placeholder
disabled	Boolean		是否禁用
type	string		多选类型 select/checkbox 默认是select
dictCode	string		数据字典编码或者表名,显示字段名,存储字段名拼接而成的字符串,如果提供了options参数 则此参数可不填
options	Array		多选项,如果dictCode参数未提供,可以设置此参数加载多选项

2.使用示例

https://static.oschina.net/uploads/img/201905/25160051_NI20.png

```
<template>
  <a-form>
    <a-form-item label="下拉多选" style="width: 300px">
      <j-multi-select-tag
        v-model="selectValue"
        :options="dictOptions"
        placeholder="请做出你的选择">
      </j-multi-select-tag>
     {{ selectValue }}
    </a-form-item>

    <a-form-item label="checkbox">
      <j-multi-select-tag
        v-model="checkboxValue"
        :options="dictOptions"
        type="checkbox">
      </j-multi-select-tag>
     {{ checkboxValue }}
```

```
    </a-form-item>
  </a-form >
</template>

<script>
 import JMultiSelectTag from '@/components/dict/JMultiSelectTag'
 export default {
   components: {JMultiSelectTag},
   data() {
     return {
       selectValue:"",
       checkboxValue:"",
       dictOptions:[{
         label:"选项一",
         value:"1"
       },{
         label:"选项二",
         value:"2"
       },{
         label:"选项三",
         value:"3"
       }]
     }
   }
 }
</script>
```

JPopup 弹窗选择组件

1.参数配置

参数	类型	必填	说明
placeholder	string		placeholder
code	string		online报表编码
orgFields	string		online报表中显示的列,多个以逗号隔开

destFields	string		回调对象的属性,多个以逗号隔开,其顺序和orgFields一一对应
field	string		v-model模式专用,表示从destFields中选择一个属性的值返回给当前组件
triggerChange	Boolean		v-decorator模式下需设置成true
callback(事件)	function		回调事件,v-decorator模式下用到,用于设置form控件的值

2.使用示例

https://static.oschina.net/uploads/img/201905/25161102_kY8F.png

```
<template>
  <a-form :form="form">
    <a-form-item label="v-model模式指定一个值返回至当前组件" style="width: 300px">
      <j-popup
        v-model="selectValue"
        code="user_msg"
        org-fields="username,realname"
        dest-fields="popup,other"
        field="popup"/>
      {{ selectValue }}
    </a-form-item>

    <a-form-item label="v-decorator模式支持回调多个值至当前表单" style="width: 300px">
      <j-popup
        v-decorator="['one']"
        :trigger-change="true"
        code="user_msg"
        org-fields="username,realname"
```

```
        dest-fields="one,two"
        @callback="popupCallback"/>
      {{ getFormFieldValue('one') }}
    </a-form-item>

    <a-form-item label="v-decorator模式被回调的值" style="width: 300px">
      <a-input v-decorator="['two']"></a-input>
    </a-form-item>


  </a-form >
</template>

<script>
 import JPopup from '@/components/jeecgbiz/JPopup'
 export default {
   components: {JPopup},
   data() {
     return {
       form: this.$form.createForm(this),
       selectValue:"",
     }
   },
   methods:{
     getFormFieldValue(field){
       return this.form.getFieldValue(field)
     },
     popupCallback(row){
       this.form.setFieldsValue(row)
     }
   }
 }
</script>
```

JSelectDepart 部门选择组件

选择部门组件,存储部门ID,显示部门名称

1.参数配置

参数	类型	必填	说明
modalWidth	Number		弹框宽度 默认500

multi	Boolean		是否多选 默认false
rootOpened	Boolean		是否展开根节点 默认true
disabled	Boolean		是否禁用 默认false

2.使用示例

https://static.oschina.net/uploads/img/201905/25161735_xJTT.png

```
<template>
  <a-form :form="form">
    <a-form-item label="部门选择v-decorator" style="width: 300px">
      <j-select-depart v-decorator="['bumen']"/>
     {{ getFormFieldValue('bumen') }}
    </a-form-item>

    <a-form-item label="部门选择v-model" style="width: 300px">
      <j-select-depart v-model="bumen"/>
     {{ bumen }}
    </a-form-item>

    <a-form-item label="部门多选v-model" style="width: 300px">
      <j-select-depart v-model="bumens" :multi="true"/>
     {{ bumens }}
    </a-form-item>

  </a-form >
</template>

<script>
 import JSelectDepart from '@/components/jeecgbiz/JSelectDepart'
 export default {
   components: {JSelectDepart},
   data() {
     return {
       form: this.$form.createForm(this),
       bumen:"",
       bumens:""
     }
```

```
  },
  methods:{
    getFormFieldValue(field){
      return this.form.getFieldValue(field)
    }
  }
 }
</script>
```

JSelectMultiUser 用户多选组件

使用示例

https://static.oschina.net/uploads/img/201905/25162051_Gqcp.png

```
<template>
 <a-form :form="form">
  <a-form-item label="用户选择v-decorator" style="width: 500px">
   <j-select-multi-user v-decorator="['users']"/>
   {{ getFormFieldValue('users') }}
  </a-form-item>

  <a-form-item label="用户选择v-model" style="width: 500px">
   <j-select-multi-user v-model="users" ></j-select-multi-user>
   {{ users }}
  </a-form-item>

 </a-form >
</template>

<script>
 import JSelectMultiUser from '@/components/jeecgbiz/JSelectMultiUser'
 export default {
  components: {JSelectMultiUser},
  data() {
   return {
    form: this.$form.createForm(this),
    users:"",
   }
  },
```

```
  methods:{
    getFormFieldValue(field){
      return this.form.getFieldValue(field)
    }
  }
 }
</script>
```

JSelectUserByDep 根据部门选择用户

1.参数配置

参数	类型	必填	说明
modalWidth	Number		弹框宽度 默认1250
disabled	Boolean		是否禁用

2.使用示例

https://static.oschina.net/uploads/img/201905/25162329_Hu1T.png

```
<template>
 <a-form :form="form">
  <a-form-item label="用户选择v-decorator" style="width: 500px">
   <j-select-user-by-dep v-decorator="['users']"/>
   {{ getFormFieldValue('users') }}
  </a-form-item>

  <a-form-item label="用户选择v-model" style="width: 500px">
   <j-select-user-by-dep v-model="users" ></j-select-user-by-dep>
   {{ users }}
  </a-form-item>

 </a-form >
</template>

<script>
 import JSelectUserByDep from '@/components/jeecgbiz/JSelectUserByDep'
 export default {
  components: {JSelectUserByDep},
```

```
  data() {
    return {
      form: this.$form.createForm(this),
      users:"",
    }
  },
  methods:{
    getFormFieldValue(field){
      return this.form.getFieldValue(field)
    }
  }
 }
</script>
```

JTreeDict 分类字典树形下拉组件

1.参数配置

参数	类型	必填	说明
placeholder	string		placeholder
disabled	Boolean		是否禁用
parentCode	string		指定一个节点的编码,加载该节点下的所有字典数据,若不指定，默认加载所有数据
field	string		指定当前组件需要存储的字段 可选: id(主键)和code(编码) 默认是id
async	Boolean		是否异步加载,对于数据量大的字典数据,建议设置成true

2.使用示例

图片地址：https://static.oschina.net/uploads/img/201906/04153943_2JYe.png

```
<template>
```

```
  <div>
    <j-tree-dict v-model="editorValue" field="code" :async="false" style="width:
300px"/>
    {{ editorValue }}
  </div>
</template>

<script>
 import JTreeDict from '@/components/jeecg/JTreeDict'
 export default {
   components: {JTreeDict},
   data() {
     return {
       form: this.$form.createForm(this),
       editorValue:'',
     }
   }
 }
</script>
```

duplicateCheck 表单字段重复校验通用JS

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);
        }
      })
    }
  },
```

JTreeSelect树形下拉框 (异步加载)

1.参数配置

参数	类型	必填	说明
placeholder	string		placeholder
disabled	Boolean		是否禁用
dict	string		表名,显示字段,存储字段 拼接的字符串
pidField	string		指定父级节点的字段 默认pid

pidValue	string		指定父级节点的id值 不指定则默认为 0 不可以为空，若为空请自行修改数据库
condition	string(json字符串)		支持自定义查询条件，进行过滤数据，请按此标准示例赋值：condition='{"create_by":"admin"}'

2.使用示例

图片地址：https://static.oschina.net/uploads/img/201906/11173700_JGRP.png

```
<template>
  <a-form :form="form">
    <a-form-item>
      <j-tree-select
        style="width: 300px"
        v-model="treeValue"
        dict="sys_depart,depart_name,id"
        pid-field="parent_id">
      </j-tree-select>
     {{ treeValue }}
    </a-form-item>

    <a-form-item>
      <j-tree-select
        style="width: 300px"
        v-decorator="['demoTree']"
        dict="sys_depart,depart_name,id"
        pid-field="parent_id">
      </j-tree-select>
     {{ getTreeFieldValue() }}
    </a-form-item>
  </a-form>
</template>

<script>
  import JTreeSelect from '@/components/jeecg/JTreeSelect'
```

```
  export default {
    components: { JTreeSelect },
    data() {
      return {
        form: this.$form.createForm(this),
        treeValue:'',
      }
    },
    methods:{
      getTreeFieldValue(){
        return this.form.getFieldValue("demoTree")
      }
    }

  }
</script>
```

报表开发

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

自定义报表组件

报表组件文档

柱状图

引用方式

```
import Bar from '@/components/chart/Bar'
```

参数列表

参数名	类型	必填	说明
title	string		报表标题
dataSource	array		报表数据源
height	number		报表高度，默认254

dataSource 示例

```
[
  {
    "x": "1月",
    "y": 320
  },
  {
    "x": "2月",
    "y": 457
  },
  {
    "x": "3月",
    "y": 182
  }
]
```

代码示例

```
<template>
  <bar title="柱状图" :dataSource="dataSource" :height="420"/>
</template>

<script>
  import Bar from '@/components/chart/Bar'

  export default {
    name: 'ChartDemo',
    components: {
      Bar
    },
    data() {
```

```
        return {
            dataSource: [
                {
                    "x": "1月",
                    "y": 320
                },
                {
                    "x": "2月",
                    "y": 457
                },
                {
                    "x": "3月",
                    "y": 182
                }
            ]
        }
    }
  }
</script>

<style></style>
```

多列柱状图

引用方式

```
import BarMultid from '@/components/chart/BarMultid'
```

参数列表

参数名	类型	必填	说明
title	string		报表标题
fields	array		主列字段列表
dataSource	array		报表数据源
height	number		报表高度，默认254

fields 示例

```
["Jan.", "Feb.", "Mar.", "Apr.", "May", "Jun.", "Jul.", "Aug."]
```

dataSource 示例

```
[
  {
    "type": "Jeecg", // 列名
    "Jan.": 18.9,
    "Feb.": 28.8,
    "Mar.": 39.3,
    "Apr.": 81.4,
    "May": 47,
    "Jun.": 20.3,
    "Jul.": 24,
    "Aug.": 35.6
  },
  {
    "type": "Jeebt",
    "Jan.": 12.4,
    "Feb.": 23.2,
    "Mar.": 34.5,
    "Apr.": 99.7,
    "May": 52.6,
    "Jun.": 35.5,
    "Jul.": 37.4,
    "Aug.": 42.4
  }
]
```

迷你柱状图

不带标题和数据轴的柱状图

引用方式

```
import MiniBar from '@/components/chart/MiniBar'
```

参数列表

参数名	类型	必填	说明
width	number		报表宽度度，默认自适应宽度
height	number		报表高度，默认200
dataSource	array		报表数据源

dataSource 示例

```
[
  {
    "x": "1月",
    "y": 320
  },
  {
    "x": "2月",
    "y": 457
  },
  {
    "x": "3月",
    "y": 182
  }
]
```

面积图

引用方式

```
import AreaChartTy from '@/components/chart/AreaChartTy'
```

参数列表

参数名	类型	必填	说明
title	string		报表标题
dataSource	array		报表数据源

height	number		报表高度，默认254
lineSize	number		线的粗细，默认2

dataSource 示例

```
[
  {
    "x": "1月",
    "y": 320
  },
  {
    "x": "2月",
    "y": 457
  },
  {
    "x": "3月",
    "y": 182
  }
]
```

多行折线图

引用方式

```
import LineChartMultid from '@/components/chart/LineChartMultid'
```

参数列表

参数名	类型	必填	说明
title	string		报表标题
fields	array		主列字段列表
dataSource	array		报表数据源
height	number		报表高度，默认254

fields 示例

```
["jeecg", "jeebt"]
```

dataSource 示例

```
[
   {
      "type": "Jan", // 列名
      "jeecg": 7,
      "jeebt": 3.9
   },
   { "type": "Feb", "jeecg": 6.9, "jeebt": 4.2 },
   { "type": "Mar", "jeecg": 9.5, "jeebt": 5.7 },
   { "type": "Apr", "jeecg": 14.5, "jeebt": 8.5 },
   { "type": "May", "jeecg": 18.4, "jeebt": 11.9 },
   { "type": "Jun", "jeecg": 21.5, "jeebt": 15.2 },
   { "type": "Jul", "jeecg": 25.2, "jeebt": 17 },
   { "type": "Aug", "jeecg": 26.5, "jeebt": 16.6 },
   { "type": "Sep", "jeecg": 23.3, "jeebt": 14.2 },
   { "type": "Oct", "jeecg": 18.3, "jeebt": 10.3 },
   { "type": "Nov", "jeecg": 13.9, "jeebt": 6.6 },
   { "type": "Dec", "jeecg": 9.6, "jeebt": 4.8 }
]
```

饼状图

引用方式

```
import Pie from '@/components/chart/Pie'
```

参数列表

参数名	类型	必填	说明
dataSource	array		报表数据源
height	number		报表高度，默认 254

dataSource 示例

```
[
  // 所有的 percent 相加等于 100
  { "item": "一月", "percent": 40 },
  { "item": "二月", "percent": 21 },
  { "item": "三月", "percent": 17 },
  { "item": "四月", "percent": 13 },
  { "item": "五月", "percent": 9 }
]
```

雷达图

引用方式

```
import Radar from '@/components/chart/Radar'
```

参数列表

参数名	类型	必填	说明
dataSource	array		报表数据源
height	number		报表高度，默认 254

dataSource 示例

```
[
  // score 最小值为 0，最大值为 100
  { "item": "一月", "score": 40 },
  { "item": "二月", "score": 20 },
  { "item": "三月", "score": 67 },
  { "item": "四月", "score": 43 },
  { "item": "五月", "score": 90 }
]
```

进度条

引用方式

```
import MiniProgress from '@/components/chart/MiniProgress'
```

参数列表

参数名	类型	必填	说明
percentage	number		当前进度百分比，默认0，最高100
target	number		目标值，默认10
height	number		进度条高度，默认10
color	string		进度条颜色，默认#13C2C2

仪表盘

引用方式

```
import DashChartDemo from '@/components/chart/DashChartDemo'
```

参数列表

参数名	类型	必填	说明
title	string		报表标题
value	number		当前值，默认6.7，最大为9
height	number		报表高度，默认254

排名列表

引用方式

```
import RankList from '@/components/chart/RankList'
```

参数列表

参数名	类型	必填	说明
title	string		报表标题
list	array		排名列表数据
height	number		报表高度，默认自适应高度

list 示例

```
[
  {
    "name": "北京朝阳 1 号店",
    "total": 1981
  },
  { "name": "北京朝阳 2 号店", "total": 1359 },
  { "name": "北京朝阳 3 号店", "total": 1354 },
  { "name": "北京朝阳 4 号店", "total": 263 },
  { "name": "北京朝阳 5 号店", "total": 446 },
  { "name": "北京朝阳 6 号店", "total": 796 }
]
```

Icon图标扩展方法

编写中...

vue路由带参总结

1. params

<router-link :to="{name:'test', params: {id:1}}">

配置路由格式要求: path: "/test/:id"

js参数获取：this.$route.params.id

2.query

<router-link :to="{name:'test', query: {id:1}}">

配置路由:无要求

js参数获取：this.$route.query.id

备注：

router-link是html写法，JS中语法如下：

this.$router.push({name:'test',query: {id:'1'}})

this.$router.push({name:'test',params: {id:'1'}})

列表自定义列实现

功能说明：

> 页面自定义设置列表需要选择的列，设置组件集成的两种方法，一个是在列表外增加设置组件，一个是在列表表头增加设置组件
> 具体代码案例参照【常用示例-单表模型示例】功能

功能预览：

图片地址：https://static.oschina.net/uploads/img/201906/14115727_mdM3.png

图片地址：https://static.oschina.net/uploads/img/201906/14115739_9e8f.png

图片地址：https://static.oschina.net/uploads/img/201906/14115753_6Sbd.png

实现方法：

一. 增加初始化配置

1 . data() 方法中配置

```
 //表头
 columns:[],
 //列设置
settingColumns:[],
//列定义
defColumns: [{
            title: '#',
```

```
        dataIndex: '',
        key: 'rowIndex',
        width: 60,
        align: "center",
        customRender: function (t, r, index) {
          return parseInt(index) + 1;
        }
      },
      {
        title: '姓名',
        align: "center",
        dataIndex: 'name'
      },
      .......
      .......
    ]
```

说明：
columns：列表展示的列，初始为空。
settingColumns：保存勾选的列设置
defColumns：定义列表可以展示的列信息

2. 增加设置按钮，两种实现方式任选其一即可

（1）第一种在列表外增加设置按钮

```
<a-popover title="自定义列" trigger="click" placement="leftBottom">
        <template slot="content">
          <a-checkbox-group @change="onColSettingsChange" v-
model="settingColumns" :defaultValue="settingColumns">
            <a-row>
              <template v-for="(item,index) in defColumns">
                <template v-if="item.key!='rowIndex'&& item.dataIndex!='action'">
                  <a-col :span="12"><a-checkbox :value="item.dataIndex">{{ item.title
}}</a-checkbox></a-col>
                </template>
              </template>
            </a-row>
          </a-checkbox-group>
        </template>
        <a><a-icon type="setting" />自定义列</a>
      </a-popover>
```

（2）第二种在表头列中扩展按钮

在操作列定义中增加插槽设置

```
{
          title: '操作',
          dataIndex: 'action',
          align: "center",
          scopedSlots: {
            filterDropdown: 'filterDropdown',
            filterIcon: 'filterIcon',
            customRender: 'action'},
        }
```

<a-table></a-table> 中增加插槽代码

```
<div slot="filterDropdown">
        <a-card>
          <a-checkbox-group @change="onColSettingsChange" v-
model="settingColumns" :defaultValue="settingColumns">
            <a-row>
              <template v-for="(item,index) in defColumns">
                <template v-if="item.key!='rowIndex'&& item.dataIndex!='action'">
                  <a-col :span="12"><a-checkbox :value="item.dataIndex">{{ item.title
}}</a-checkbox></a-col>
                </template>
              </template>
            </a-row>
          </a-checkbox-group>
        </a-card>
      </div>
      <a-icon slot="filterIcon" type='setting' :style="{ fontSize:'16px',color:  '#108ee9'
}" />
```

3. 实现checkbox @change

```
//列设置更改事件
    onColSettingsChange (checkedValues) {
      var key = this.$route.name+":colsettings";
      Vue.ls.set(key, checkedValues, 7 * 24 * 60 * 60 * 1000)
      this.settingColumns = checkedValues;
      const cols = this.defColumns.filter(item => {
        if(item.key =='rowIndex'|| item.dataIndex=='action'){
```

```
          return true
        }
        if (this.settingColumns.includes(item.dataIndex)) {
          return true
        }
        return false
      })
      this.columns =  cols;
    },
```

4. 页面加载时实现列的初始化方法

```
initColumns(){
      //权限过滤（列权限控制时打开，修改第二个参数为授权码前缀）
      //this.defColumns = colAuthFilter(this.defColumns,'testdemo:');

      var key = this.$route.name+":colsettings";
      let colSettings= Vue.ls.get(key);
      if(colSettings==null||colSettings==undefined){
        let allSettingColumns = [];
        this.defColumns.forEach(function (item,i,array ) {
          allSettingColumns.push(item.dataIndex);
        })
        this.settingColumns = allSettingColumns;
        this.columns = this.defColumns;
      }else{
        this.settingColumns = colSettings;
        const cols = this.defColumns.filter(item => {
          if(item.key =='rowIndex'|| item.dataIndex=='action'){
            return true;
          }
          if (colSettings.includes(item.dataIndex)) {
            return true;
          }
          return false;
        })
        this.columns =  cols;
      }
    }
```

created中调用：

```
created() {
      this.initColumns();
    },
```

国际化改造方案

国际化改造大致分两部分:
（1）antd UI组件国际化
（2）业务文案的国际化需求

一、antd UI 组件国际化

文档参见【LocaleProvider 国际化(https://vue.ant.design/components/locale-provider-cn/)】
在项目App.vue页面增加国际化代码

```
<template>
  <a-locale-provider :locale="zh_CN">
    <App />
  </a-locale-provider>
</template>
<script>
import zh_CN from 'ant-design-vue/lib/locale-provider/zh_CN';
import moment from 'moment';
import 'moment/locale/zh-cn';

moment.locale('zh-cn');
export default {
  data() {
    return {
      zh_CN,
    }
  }
}
</script>
```

增加国际化设置功能切换locale国际化的类型值

二、业务文案的国际化需求

(1) 安装 vue-i18n：

```
$ npm install vue-i18n
```

(2) 在 src/components/lang/ 中创建语言js，例如 en-US.js 和 zh-CN.js

在 en-US.js 和 zh-CN.js 中分别作了如下配置：

```
// zh-CN.js

export default {
  lang: '中文',
}

// en-US.js

export default {
  lang: 'English',
}
```

(3) 在main.js实例化组件

注意:Vue.use(VueI18n) 要放在实例化之前，不然会报错

```
import VueI18n from 'vue-i18n'
Vue.use(VueI18n)

// 注册i18n实例并引入语言文件，文件格式等下解析
const i18n = new VueI18n({
  locale: Vue.ls.get("language", "zh-CN")),
  messages: {
    'zh-CN': require('@/components/lang/zh-CN.js'),
    'en-US': require('@/components/lang/en-US.js')
  }
})
//将i18n注入到vue实例中
new Vue({
  el: '#app',
  router,
  store,
  i18n,
  components: { App },
  template: '<App/>'
```

```
})
```

在组件中这样使用

```
<template>
  <div>
    {{ $t('lang') }}
  </div>
</template>
```

如果想传参，例如zh-CN.js中这样配置：

```
welcome: "你好，{name}"
```

页面组件中这样使用：

```
<template>
  <div>
    {{ $tc('welcome', {'name':'jeecg' }) }}
  </div>
</template>
```

页面输出 ： 你好，jeecg

常用方法：

```
$t(path, locale, option) // text 文本替换，locale可单独设置语言，option可传参数替换模板
$tc(path, choice, locale, option) // text+choice 比$t多一个choice，可以选择模板内容（模板内容间用 | 分隔，如 香蕉|苹果|梨，最多只能使用三个选项，下标从0开始，当选项为2个时下标从1开始~~ ）
$te(path) // text+exist 判断当前语言包中path是否存在
$d(number|Date, path, locale) // date 时间格式化
$n(number, path, locale) // number  数字格式化（货币等）

// 更多细节参考官方文档：https://kazupon.github.io/vue-i18n/api/#vue-injected-methods
```

如果某些js中含有字符需要切换语言（包括rules），需要写在computed中。

```
computed: {
  rules1 () {
```

```
    return {
      username: [
        {required: true, message: this.$t('username.required'), trigger: 'blur'}
      ]
    }
  }
}
```

（3）国际化语言切换实现

```
handleSetLanguage(lang) {
    this.$i18n.locale = lang; //改变当前语言
    Vue.ls.set("language", lang);//将lang 语言存在localStorage里，main里面就会根据属性值进行判断 locale: Vue.ls.get("language", "zh-CN")
},
```

vuex 使用详解

一、什么是vuex

vuex是一个专门为vue.js设计的集中式状态管理架构。它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。。

二、vuex使用场景

vuex主要是是做数据交互，父子组件传值可以很容易办到，但是兄弟组件间传值（兄弟组件下又有父子组件），或者大型spa单页面框架项目，页面多并且一层嵌套一层的传值，异常麻烦，用vuex来维护共有的状态或数据会更便捷

场景说明：

• 由于vuex中的state存放的数据在页面刷新时会丢失，vuex只能用于单个页面中不同组件（例如兄弟组件）的数据流通

三、Vuex核心概念

• 1、store：类似容器，包含应用的大部分状态

一个页面只能有一个容器
状态存储是响应式的
不能直接改变store中的状态，唯一途径显示地提交mutations
在actions里面，也不能直接更改state里面的状态值，必须先定义一个mutations，然后在

actions里面commit这个mutations，从而来更改state的状态值；
如果要再次请求异步，那么就是dispatch一个actions

- 2、State：包含所有应用级别状态的对象
- 3、Getters：在组件内部获取store中状态的函数，类似组件的计算属性computed
- 4、Mutations：唯一修改状态的事件回调函数，默认是同步的，如果要异步就使用Actions
- 5、Actions：包含异步操作、提交mutations改变状态
- 6、Modules：将store分割成不同的模块

四、引入Vuex

1、安装vuex

```
npm install vuex --save
```

2、新建一个store文件夹（这个不是必须的），并在文件夹下新建index.js文件，文件中引入我们的vue和vuex

图片地址：https://static.oschina.net/uploads/img/201908/22145513_qBt5.png

3、在main.js 中引入新建的store

```
import store from './store/'

 new Vue({
     el: '#app',
     router,
     store,//使用store
     template: '<App/>',
     components: { App }
   })
```

五、页面使用

1.读取store里的值：

this.$store.state.字段名
如果有 moudle 的话，假设叫 user ,那么取值又要变了，加上 module 名

this.$store.state.user.字段名

2.发起操作请求:

this.$store.dispatch('action中的方法名'，'参数')；

参数你可以随便传json

3.getters的用法

this.$store.getters.filterDoned
filterDoned 是在 todo 里写的一个 getters 方法，就这么调用噢

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

http://localhost:8080/jeecg-boot/doc.html

登录名：jeecg，密码jeecg1314

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

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

5、手工编码校验token有效性

```
//校验Token有效性
String token = request.getParameter("token");
SysUser sysUser = shiroRealm.checkUserTokenIsEffect(token);
```

6、想要设定查询模式为模糊查询，怎么作全局修改？

修改后端/jeecg-boot-base-common/src/main/java/org/jeecg/common/system/query/QueryGenerator.java,找到installMplus方法下述代码修改即可

```
//根据参数值带什么关键字符串判断走什么类型的查询
QueryRuleEnum rule = convert2Rule(value);
value = replaceValue(rule,value);

// add -begin 添加判断若为字符串类型则设置为模糊查询
if("class java.lang.String".equals(type)) {
// 可以设置左右模糊或全模糊，因人而异
rule = QueryRuleEnum.LIKE;
}
// add -end 添加判断若为字符串类型则设置为模糊查询

addEasyQuery(queryWrapper, name, rule, value);
```

7、更多常见问题，实时更新

更多常见问题，实时更新，点击我(http://www.jeecg.org/forum.php?mod=viewthread&tid=7816&page=1&extra=#pid21237)

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
是否路由	此处很重要，默认是路由；如果是非的话，访问404
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（具体参考底部截图）

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（具体参考底部截图）

（比如跳转百度：https://www.baidu.com）

• 4、java后台请求的菜单

需要以{{ window._CONFIG['domianURL'] }}开头（具体参考底部截图）

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、需要跳转到第三方页面的菜单，菜单路径配置第三方跳转的地址即可，例如 http://www.baidu.com

参考示例：

• 1.路由菜单配置截图

图片地址：https://static.oschina.net/uploads/img/201905/21115338_Z1Os.png

• 2.外部链接菜单

图片地址：https://static.oschina.net/uploads/img/201905/21115431_ZsNq.png

• 3.后台链接菜单

图片地址：https://static.oschina.net/uploads/img/201905/21115546_ZVvn.png

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

> 字典翻译注解@Dict： 用于列表字段字典翻译（比如字段sex存的值是1，会自动生成一个翻译字段 sex_dictText 值是‘男’ ）

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'
    }]
```

4.字典表翻译用法

```
@Dict(dicCode = "id",dictTable="sys_user",dicText="realname")
```

截图：

图片地址：https://static.oschina.net/uploads/img/201906/10163148_JVFh.png

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

参考： http://jeecg-boot.mydoc.io/?t=345704

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`,
ADD COLUMN `del_flag` tinyint(1) NULL COMMENT '删除标识0-正常,1-已删除' AFTER
`update_time`;
```

- 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");
  }
}
```

数据快照功能如何用？

> 数据快照功能： 记录每次单据变更内容，保存版本号
>
> 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

1. 记录数据快照接口

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

1. 对比数据快照变更内容

图片地址：https://static.oschina.net/uploads/img/201905/16155629_zDPt.png

图片地址：https://static.oschina.net/uploads/img/201905/16155718_Mspf.png

查询过滤器用法

目录索引：

查询过滤器

一、功能描述

查询过滤器可以帮助快速生成查询条件，不需要编码通过配置实现，支持模糊查询、匹配查询、范围查询、不匹配查询等规则。

二、查询条件如何实现

第一步：页面实现查询条件

在线列表的查询区域，增加需要的查询字段，如下图所示。

图片地址：https://static.oschina.net/uploads/img/201907/24163522_rb24.png

效果:

图片地址：https://static.oschina.net/uploads/img/201907/24163640_wrvU.png

第二步：controller层处理

Controller中对应的处理逻辑中追加如下代码：

```
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;
}
```

三、查询规则

说明：页面查询字段，需跟后台Controller中Page的字段对应一致，后台不需写代码自动生成查询条件SQL;

> 默认查询条件是全匹配，想实现模糊查询需求在查询值的前后加: *;

查询匹配方式规则：

[1].全匹配查询：查询数据没有特殊格式，默认为全匹配查询

[2].模糊查询：查询数据格式需加星号：{ * }

例如：

> 格式一： 张*　（后模糊匹配）
> 格式二： *张　（前模糊匹配）
> 格式三： *张*　（全模糊匹配）
> 格式四： *张*三*（更高级匹配）

[3].包含查询：查询数据格式采用逗号分隔：{ , }

例如：

> 格式：　张三,李四

(含义：In('张三','李四'))

[4].不匹配查询：查询数据格式需要加叹号前缀：{ ! }

例如：

格式： !张三
(含义：不等于'张三')
特殊说明：查询不为Null的语法：!null(大小写没关系);
查询不为空字符串的方法：!(只有一个叹号);

[5].范围查询，支持数字，时间的范围查询，针对范围查询页面会生成两个查询控件

1. 如果是单一匹配方式，则页面查询控件的name， 跟实体字段命名一样
2. 如果是范围匹配方式，则页面查询控件需要变成两个分别名 {*}_begin，{*}_end

{*}_begin： 表示查询范围开始值
{*}_end: 表示查询范围结束值

举例：
字段名称 orderDate
查询开始时间 : orderDate_begin
查询结束时间 : orderDate_end

四、更多查询规则参考

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)
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	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%'	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,

• 高级值规则用法 (查询内容，带有查询规则符号)

查询模式	用法	举例
<	小于查询。 查询内容值规则："lt+ 空格 + 内容"	输入值： “lt 100”
<=	小于等于查询。 查询内容值规则："le+ 空格+ 内容"	输入值： “le 100”
\>	大于查询。 查询内容值规则："gt+ 空格+ 内容"	输入值： “gt 100”
\>=	大于等于查询。 查询内容值规则："ge+ 空格+ 内容"	输入值： “ge 100”

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	(作废参数)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;
}
```

自定义sql分页实现

（1）mapper 接口以及 xml

```
/**
 * @Description: 系统通告表
 * @Author: jeecg-boot
 * @Date:  2019-01-02
 * @Version: V1.0
 */
public interface SysAnnouncementMapper extends BaseMapper<SysAnnouncement>
{



List<SysAnnouncement> querySysCementListByUserId(Page<SysAnnouncement>
page, String userId,String msgCategory);

}
```

这里要注意的是，这个 Page page 是必须要有的，否则 Mybatis-Plus 无法为你实现分页。

```
<resultMap id="SysAnnouncement"
type="org.jeecg.modules.system.entity.SysAnnouncement" >
<result column="id" property="id" jdbcType="VARCHAR"/>
<result column="titile" property="titile" jdbcType="VARCHAR"/>
<result column="msg_content" property="msgContent" jdbcType="VARCHAR"/>
<result column="start_time" property="startTime" jdbcType="TIMESTAMP"/>
<result column="end_time" property="endTime" jdbcType="TIMESTAMP"/>
<result column="sender" property="sender" jdbcType="VARCHAR"/>
<result column="priority" property="priority" jdbcType="VARCHAR"/>
<result column="msg_category" property="msgCategory" jdbcType="VARCHAR"/>
```

```
<result column="msg_type" property="msgType" jdbcType="VARCHAR"/>
<result column="send_status" property="sendStatus" jdbcType="VARCHAR"/>
<result column="send_time" property="sendTime" jdbcType="VARCHAR"/>
<result column="cancel_time" property="cancelTime" jdbcType="VARCHAR"/>
<result column="del_flag" property="delFlag" jdbcType="VARCHAR"/>
<result column="create_by" property="createBy" jdbcType="VARCHAR"/>
<result column="create_time" property="createTime" jdbcType="TIMESTAMP"/>
<result column="update_by" property="updateBy" jdbcType="VARCHAR"/>
<result column="update_time" property="updateTime" jdbcType="TIMESTAMP"/>
<result column="user_ids" property="userIds" jdbcType="VARCHAR"/>
</resultMap>


<select id="querySysCementListByUserId" parameterType="String"
resultMap="SysAnnouncement">
  select sa.* from sys_announcement sa,sys_announcement_send sas
  where sa.id = sas.annt_id
  and sa.send_status = '1'
  and sa.del_flag = '0'
  and sas.user_id = #{userId}
  and sa.msg_category = #{msgCategory}
  and sas.read_flag = '0'
</select>
```

（3） service 实现

```
@Override
public Page<SysAnnouncement>
querySysCementPageByUserId(Page<SysAnnouncement> page, String userId,String
msgCategory) {
 return page.setRecords(sysAnnouncementMapper.querySysCementListByUserId(page,
userId, msgCategory));
}
```

（4） controller 实现

```
@RequestMapping(value = "/list", method = RequestMethod.GET)
public Result<Page<SysAnnouncement>> queryPageList(SysAnnouncement
sysAnnouncement,
  @RequestParam(name="pageNo", defaultValue="1") Integer pageNo,
```

```
  @RequestParam(name="pageSize", defaultValue="10") Integer pageSize,
  HttpServletRequest req) {
Result<Page<SysAnnouncement>> result = new
Result<Page<SysAnnouncement>>();
Page<SysAnnouncement> pageList = new
Page<SysAnnouncement>(pageNo,pageSize);
pageList =
sysAnnouncementService.querySysCementPageByUserId(pageList,"","1");//通知公告消息
log.info("查询当前页："+pageList.getCurrent());
log.info("查询当前页数量："+pageList.getSize());
log.info("查询结果数量："+pageList.getRecords().size());
log.info("数据总数："+pageList.getTotal());
result.setSuccess(true);
result.setResult(pageList);
return result;
}
```

新建maven模块项目？

> 团队唐根博客
>
> jeecg-boot新建module模块：
> https://my.oschina.net/u/3903209/blog/3083399?tdsourcetag=s*pctim*aiomsg

1、复制下面的pom内容新建一个maven项目（jeecg-boot-module-demo）

http://www.jeecg.org/forum.php?mod=viewthread&tid=8015&extra=

2、创建完成，在父pom里面引用此模块

父POM: jeecg-boot-frameworkpom.xml

```
<modules>
<module>jeecg-boot-base-common</module>
<module>jeecg-boot-module-system</module>
<module>jeecg-boot-module-demo</module>
</modules>
```

3、底层共通业务接口说明（新创建Maven模块项目）

> jeecg-boot-module-system 项目禁止其他独立模块直接引用，如果需要共通API或者调用系统信息接口请使用

• 统一的业务接口： jeecg-boot-base-common/org.jeecg.common.system.api.ISysBaseAPI(所有子模块需要的接口都在这里定义)

• 接口实现类在system里： jeecg-boot-module-system/org.jeecg.modules.system.service.impl.SysBaseApiImpl

• jeecg-boot-module-system不让直接引用的目的是，防止相互调用，另外让子模块项目更轻量，如果调用system，随着项目累计，会导致乱也不好维护。

4、 启动项目

jeecg-boot-module-system项目作为启动项目，修改system的pom，加入新创建的maven模块项目

```
<dependency>
  <groupId>org.jeecgframework.boot</groupId>
  <artifactId>新的模块名</artifactId>
  <version>版本号</version>
</dependency>
```

针对敏感数据，加密传递方案

针对敏感数据，加密传递方案

第一步：

在vue页面引入aesEncrypt.js encryption方法。示例代码：

```
import { encryption } from '@/utils/encryption/aesEncrypt'
```

第二步：

请求后台获取 /sys/getEncryptedString 接口，以此获取加密所需要的key和iv

第三步：

使用引入的 encryption方法进行对敏感数据进行加密。方法第一个参数是所需要加密的字符串
第二个参数和第三个参数是第二步从后台获取的key和iv

第四步（后台解密）：

用AesEncryptUtil类的 desEncrypt方法对加密后的字符串进行解密。示例代码：

```
AesEncryptUtil.desEncrypt(sysLoginModel.getPassword()).trim();
```

注意事项：

加密解密所需要的key和iv在EncryptedString类中。长度为16个字符

系统权限用法

页面表单权限

显示隐藏控制

一、用法

```
v-has="'name'"
```

代码示例：

```
<a-form-item v-has="'name'"
  :labelCol="labelCol"
  :wrapperCol="wrapperCol"
  label="请假人">
  <a-input placeholder="请输入请假人" v-decorator="['name', {}]" />
</a-form-item>
```

二、权限配置：

图片地址：https://static.oschina.net/uploads/img/201905/16150058_JC6X.png

三、使用说明

- v-has="'name'" 指令值“name”为授权标识，可对该授权标识进行“显示/访问”控制
- 权限编码在【系统管理--菜单管理】中配置，添加按钮类型的菜单数据，授权标识配置值“name”，策略选择显示/访问，状态选择有效
- 控制规则：
- （1）使用v-has指令后，菜单权限中若没有对应指令编码的配置，则不显示控件，
- （2）权限配置无效状态时，则不进行权限控制，有效状态时进行控制
- （3）策略：显示/访问，未授权时不显示，授权后显示

四、流程节点权限

（1）说明：

- 节点权限配置优先级高于菜单权限配置
- 节点权限应用于使用组件方式加载的附加表单页面，并对附加表单页面进行权限控制
- 显示控制用法见上面用法描述

- 节点权限是通过 props: ['formData'],来传递给节点表单页面的，因此页面一定要定义这个，否则，节点配置的权限不生效
- 权限配置无效状态时，则不进行权限控制，有效状态时进行控制

（2）权限配置：

在【流程管理-流程设计】中找到需要配置的流程，进入【流程配置-流程节点】选择需要进行权限控制的节点，

点击【更多-权限设置】，新增/编辑 来配置权限。

图片地址：https://static.oschina.net/uploads/img/201904/29111317_wzTj.png

禁用控制用法一

一、用法

（1）页面引入混入代码

```
import {DisabledAuthFilterMixin} from '@/mixins/DisabledAuthFilterMixin'
```

```
mixins: [DisabledAuthFilterMixin],
```

图片地址：https://static.oschina.net/uploads/img/201905/16154133_smhI.png

（2）权限控制代码示例：

```
<a-input-number :disabled="isDisabledAuth('name')"  v-decorator="[ 'days', {}]" />
```

二、权限配置：

图片地址：https://static.oschina.net/uploads/img/201905/16150845_JvPu.png

三、使用说明

- :disabled="isDisabledAuth('name')" 调用方法disabledAuth，方法参数“name”为授权标识，该方法根据授权规则返回true/false，控制是否禁用
- 权限编码在【系统管理--菜单管理】中配置，添加按钮类型的菜单数据，授权标识配置值“name”，策略选择可编辑，状态选择有效
- 控制规则：
- （1）菜单权限中若没有对应指令编码的配置，则不进行禁用控制，
- （2）权限配置无效状态时，则不进行权限控制，有效状态时进行控制
- （3）策略：可编辑，未授权时控件禁用，授权后可编辑

四、流程节点权限

（1）说明：

- 节点权限配置优先级高于菜单权限配置
- 节点权限应用于使用组件方式加载的附加表单页面，并对附加表单页面进行权限控制
- 显示控制用法见上面用法描述
- 节点权限是通过 props: ['formData'],来传递给节点表单页面的，因此页面一定要定义这个，否则，节点配置的权限不生效，节点表单开发方法见【流程节点对接表单页面开发方法】
- 权限配置无效状态时，则不进行权限控制，有效状态时进行控制

（2）权限配置：

在【流程管理-流程设计】中找到需要配置的流程，进入【流程配置-流程节点】选择需要进行权限控制的节点，

点击【更多-权限设置】，新增/编辑 来配置权限。

图片地址：https://static.oschina.net/uploads/img/201904/29111317_wzTj.png

禁用控制用法二

一、用法

（1）页面引入工具js

```
import { disabledAuthFilter } from "@/utils/authFilter"
```

(2)methods方法中实现：

```
isDisabledAuth(code){
      return disabledAuthFilter(code);
    },
```

图片地址：https://static.oschina.net/uploads/img/201905/16152021_GhSq.png

图片地址：https://static.oschina.net/uploads/img/201905/28140915_NrKk.png

（2）权限控制代码示例：

```
<a-input-number :disabled="isDisabledAuth('name')"   v-decorator="[ 'days', {}]" />
```

二、权限配置：

图片地址：https://static.oschina.net/uploads/img/201905/16150845_JvPu.png

三、使用说明

- :disabled="isDisabledAuth('name')" 调用方法disabledAuth，方法参数“name”为授权标识，该方法根据授权规则返回true/false，控制是否禁用
- 权限编码在【系统管理--菜单管理】中配置，添加按钮类型的菜单数据，授权标识配置值“name”，策略选择可编辑，状态选择有效
- 控制规则：
- （1）菜单权限中若没有对应指令编码的配置，则不进行禁用控制，
- （2）权限配置无效状态时，则不进行权限控制，有效状态时进行控制
- （3）策略：可编辑，未授权时控件禁用，授权后可编辑

四、流程节点权限

（1）说明：

- 节点权限配置优先级高于菜单权限配置
- 节点权限应用于使用组件方式加载的附加表单页面，并对附加表单页面进行权限控制
- 显示控制用法见上面用法描述
- 节点权限是通过 props: ['formData'],来传递给节点表单页面的，因此页面一定要定义这个，否则，节点配置的权限不生效，节点表单开发方法见【流程节点对接表单页面开发方法】
- 权限配置无效状态时，则不进行权限控制，有效状态时进行控制

-

(2)methods方法中实现：

```
isDisabledAuth(code){
      return disabledAuthFilter(code,this.formData);
    },
```

（2）权限配置：

在【流程管理-流程设计】中找到需要配置的流程，进入【流程配置-流程节点】选择需要进行权限控制的节点，

点击【更多-权限设置】，新增/编辑 来配置权限。

图片地址：https://static.oschina.net/uploads/img/201904/29111317_wzTj.png

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

JAVA访问权限控制

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

数据权限规则篇

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、数据权限分两大类型

序号	类型	规则字段区别	说明	
1	编码方式	规则字段是驼峰写法，对应mybatis实体的字段	编码模式（通过代码生成器生成代码）	
2	Online方式	规则字段是下划线写法，对应表的字段	Online模式（在线表单模式，无代码）	

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
②规则值：指定值（固定值/系统上下文变量）

三、数据权限规则篇

1.当前用户上下文变量

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_org_code	当前登录用户部门编号	
sysMultiOrgCode	当前登录用户拥有的所有机构编码，逗号分隔	

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

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/201905/13210323_zYDm.png

系统数据权限用法

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_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

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

列表列权限控制

> 针对数据列表的列进行权限控制，控制列的展示与不展示，需要菜单的权限配置与页面代码配置使用
>
> 控制规则：
>
> 增加权限控制配置与代码后，配置有效的状态未授权时隐藏，授权时显示

举例： 针对常用示例列表，的用户名字段

图片地址：https://static.oschina.net/uploads/img/201908/12145605_LjT5.png

权限控制步骤

1. 针对列表列配置权限

在配置前需要对需要控制的列表权限编码定义一个前缀，规则自己设计，不同的列表定义不同前缀最好不要重复
例如：定义前缀“testdemo:” 则需要对列表中的name列进行控制，权限编码为，前缀+列字段名（ “testdemo:name” ）

在对应的列表页面菜单下配置权限：

图片地址：https://static.oschina.net/uploads/img/201906/14113738_SuNj.png

配置说明：

- 菜单类型：选择“按钮/权限”
- 授权标识：前缀+列字段名（ “testdemo:name” ）
- 授权策略：选择“显示/访问(授权后显示/可访问)”
- 状态：选择“有效”

2. 增加页面控制代码

（1）引入工具方法

```
import { colAuthFilter } from "@/utils/authFilter"
```

（2）created方法中增加方法调用，根据权限过滤展示的列

```
created() {
```

```
    this.columns = colAuthFilter(this.columns,'testdemo:');
    this.loadData();
  },
```

说明：
colAuthFilter方法：
第一个参数：列表定义的列信息
第二个参数：列权限控制定义的权限编码前缀“testdemo:”

聚合路由的使用

> 聚合路由， 配置后子菜单路由不显示，子菜单之间的跳转通过页面上的路由链接进行跳转

图片地址：https://static.oschina.net/uploads/img/201906/21183237_TDHp.png

图片地址：https://static.oschina.net/uploads/img/201906/21183251_hsZ8.png

图片地址：https://static.oschina.net/uploads/img/201906/21183354_M651.png

备注：父级菜单配置如下
（1）配置聚合路由，选择是；
（2）菜单地址配置其子菜单中的一个菜单地址，作为默认跳转的地址；

高级实战技巧

WebSocket的集成

> JEECG BOOT 增加websocket 旨在服务端主动向客户端推送数据，实现系统向在线用户推送消息，可群发，可对指定用户发送

jeecg boot 集成 websocket 步骤

（1）maven依赖

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>
```

（2）WebSocket配置类

```
package org.jeecg.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.socket.server.standard.ServerEndpointExporter;

@Configuration
public class WebSocketConfig {
    /**
     * 注入ServerEndpointExporter，
     * 这个bean会自动注册使用了@ServerEndpoint注解声明的Websocket endpoint
     */
    @Bean
    public ServerEndpointExporter serverEndpointExporter() {
        return new ServerEndpointExporter();
    }

}
```

(3) WebSocket操作类

> 通过该类WebSocket可以进行群推送以及单点推送

```
package org.jeecg.modules.message.websocket;

import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.CopyOnWriteArraySet;

import javax.websocket.OnClose;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.PathParam;
import javax.websocket.server.ServerEndpoint;

import org.springframework.stereotype.Component;

import lombok.extern.slf4j.Slf4j;
```

```
@Component
@Slf4j
@ServerEndpoint("/websocket/{userId}")
public class WebSocket {

    private Session session;

    private static CopyOnWriteArraySet<WebSocket> webSockets =new
CopyOnWriteArraySet<>();
    private static Map<String,Session> sessionPool = new HashMap<String,Session>();

    @OnOpen
    public void onOpen(Session session, @PathParam(value="userId")String userId) {
        try {
this.session = session;
webSockets.add(this);
sessionPool.put(userId, session);
log.info("【websocket消息】有新的连接，总数为:"+webSockets.size());
} catch (Exception e) {
}
    }

    @OnClose
    public void onClose() {
        try {
webSockets.remove(this);
log.info("【websocket消息】连接断开，总数为:"+webSockets.size());
} catch (Exception e) {
}
    }

    @OnMessage
    public void onMessage(String message) {
    log.info("【websocket消息】收到客户端消息:"+message);
    }

    // 此为广播消息
    public void sendAllMessage(String message) {
    log.info("【websocket消息】广播消息:"+message);
        for(WebSocket webSocket : webSockets) {
            try {
            if(webSocket.session.isOpen()) {
```

```
            webSocket.session.getAsyncRemote().sendText(message);
            }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    // 此为单点消息
    public void sendOneMessage(String userId, String message) {
        Session session = sessionPool.get(userId);
        if (session != null&&session.isOpen()) {
            try {
            log.info("【websocket消息】 单点消息:"+message);
                session.getAsyncRemote().sendText(message);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    // 此为单点消息(多人)
    public void sendMoreMessage(String[] userIds, String message) {
    for(String userId:userIds) {
    Session session = sessionPool.get(userId);
            if (session != null&&session.isOpen()) {
                try {
                log.info("【websocket消息】 单点消息:"+message);
                    session.getAsyncRemote().sendText(message);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
    }

    }

}
```

前端中VUE使用WebSocket

```
<script>
  import store from '@/store/'

  export default {
    data() {
      return {
      }
    },
    mounted() {
        //初始化websocket
        this.initWebSocket()
    },
    destroyed: function () { // 离开页面生命周期函数
        this.websocketclose();
    },
    methods: {
      initWebSocket: function () {
        // WebSocket与普通的请求所用协议有所不同，ws等同于http，wss等同于https
        var userId = store.getters.userInfo.id;
        var url =
window._CONFIG['domianURL'].replace("https://","ws://").replace("http://","ws://")+"/w
ebsocket/"+userId;
        this.websock = new WebSocket(url);
        this.websock.onopen = this.websocketonopen;
        this.websock.onerror = this.websocketonerror;
        this.websock.onmessage = this.websocketonmessage;
        this.websock.onclose = this.websocketclose;
      },
      websocketonopen: function () {
        console.log("WebSocket连接成功");
      },
      websocketonerror: function (e) {
        console.log("WebSocket连接发生错误");
      },
      websocketonmessage: function (e) {
        var data = eval("(" + e.data + ")");
        //处理订阅信息
        if(data.cmd == "topic"){
          //TODO 系统通知

        }else if(data.cmd == "user"){
          //TODO 用户消息
```

```
            }
          },
          websocketclose: function (e) {
            console.log("connection closed (" + e.code + ")");
          }
        }
      }
    }
</script>
```

Websocket业务对接

> 项目对接完Websocket后，实现业务对接，通过后台业务对接，推送相对应的业务消息给客户端，客户端处理对应业务的消息

系统通告业务对接示例

系统通告管理中，管理员可发两种类型的消息，全体用户和指定用户发送

发送的消息是一个json串

（1）调用WebSocket 服务

```
@Resource
private WebSocket webSocket;
```

（2）方法中调用

> cmd为业务类型，例如topic表示系统消息，user表示用户消息，可以自定义cmd类型，客户端根据返回的cmd类型处理不同的业务响应

全体发送

```
//创建业务消息信息
JSONObject obj = new JSONObject();
obj.put("cmd", "topic");//业务类型
obj.put("msgId", sysAnnouncement.getId());//消息id
obj.put("msgTxt", sysAnnouncement.getTitile());//消息内容
//全体发送
webSocket.sendAllMessage(obj.toJSONString());
```

单个用户发送

```
//创建业务消息信息
JSONObject obj = new JSONObject();
obj.put("cmd", "user");//业务类型
obj.put("msgId", sysAnnouncement.getId());//消息id
obj.put("msgTxt", sysAnnouncement.getTitile());//消息内容
//单个用户发送 (userId为用户id)
webSocket.sendOneMessage(userId, obj.toJSONString());
```

多个用户发送

```
//创建业务消息信息
JSONObject obj = new JSONObject();
obj.put("cmd", "user");//业务类型
obj.put("msgId", sysAnnouncement.getId());//消息id
obj.put("msgTxt", sysAnnouncement.getTitile());//消息内容
//多个用户发送 (userIds为多个用户id，逗号‘,’分隔)
webSocket.sendMoreMessage(userIds, obj.toJSONString());
```

（3）vue 客户端根据返回的cmd类型处理不同的业务响应

```
websocketonmessage: function (e) {
        console.log("-----接收消息-------",e.data);
        var data = eval("(" + e.data + ")"); //解析json对象
        if(data.cmd == "topic"){
          //TODO 系统通知

        }else if(data.cmd == "user"){
          //TODO 用户消息

        }

},
```

HBuilder打包APP手机端安装配置教程

> jeecgBoot前端UI项目，可以打成app安装手机上，采用HBuilder工具，详细步骤如下

第一步：

在vue.config.js中加入 baseUrl: './', 代码示例如下图：

图片地址：https://oscimg.oschina.net/oscnet/effc05ee4842024533a64a0ac7cc6091141.jpg

修改public目录下index.html文件把全局配置的地址改成域名。如下图

图片地址：https://oscimg.oschina.net/oscnet/a52e1112a2f5909f7d33382a916d8b517a2.jpg

修改src/utils/request.js

把axios 的baseURL修改一下,如下示例图：

图片地址：https://oscimg.oschina.net/oscnet/17664cd8cad55936303f601c1e650df311c.jpg

第二步：

在HBuilder中选择文件—>打开目录，选择build后的dist文件，如下图：

图片地址：https://oscimg.oschina.net/oscnet/f4b8fa307d96034ec562df3818d911239cd.jpg

第三步：

选中项目，右击选择转换成移动App

第四步：

选中项目，右击选择发行，弹出来菜单，选择第一项。出现一个窗口如下图： 点击打包，即可完成打包

图片地址：https://oscimg.oschina.net/oscnet/f7f90a6ffa39d8f29270a541c4b8c7dc537.jpg

下面是苹果打包截图：

图片地址
：https://oscimg.oschina.net/oscnet/d1e9b848e1d81381b6362e3a42a5e92e6e9.jpg

如果打包的app访问不到后台数据，请修改HBuilder中的配置文件，如下图

图片地址
：https://oscimg.oschina.net/oscnet/67bb79e1414272d6149aaf9ad9e1a3b8532.jpg

HBuilderX打包APP

1、Build JEECG BOOT项目

> 打包APP之前首先Build项目，并对响应的配置做更改

一、修改配置信息

（1）修改后端服务域名地址

• 修改文件public/index.html

图片地址：https://static.oschina.net/uploads/img/201908/15112301_Q9Ey.png

• 修改文件src/utils/request.js

图片地址：https://static.oschina.net/uploads/img/201908/15112449_xrsU.png

（2）修改路由模式

• 修改文件src/router/index.js

mode 值改为 hash

图片地址：https://static.oschina.net/uploads/img/201908/15112718_dk3b.png

（3）build之前配置publicPath

• 修改文件vue.config.js

增加配置 publicPath:'./',

图片地址：https://static.oschina.net/uploads/img/201908/15112943_7lSM.png

以上配置完成后即可执行 npm run build 命令进行编译，编译之后文件会在dist目录下

2、HBuilderX 打包APP

> 第一步build的文件（dist目录下的文件）使用HBuilderX 打包成APP
>
> 使用 HBuiderX 打包。（工具下载地址：http://www.dcloud.io/）

一、创建项目

【文件-新建-项目】 选择项目类型和填写项目名称

图片地址：https://static.oschina.net/uploads/img/201908/15113942_Oh6w.png

创建后会生成如下目录

图片地址：https://static.oschina.net/uploads/img/201908/15115203_512u.png

二、dist目录下的文件拷贝到该工程

（1）包里工程路面下unpackge目录和mainifest.json文件 其他文件目录全部删除

（2）拷贝dist目录下的文件到该工程目录

拷贝后目录结构大概如下：

图片地址：https://static.oschina.net/uploads/img/201908/15115936_vORZ.png

三、打包前项目配置

双击文件mainifest.json 进入文件配置界面

（1）基础配置

根据具体情况填写

图片地址：https://static.oschina.net/uploads/img/201908/15120436_ZINc.png

（2）图标配置

配置logo

图片地址：https://static.oschina.net/uploads/img/201908/15120639_3yrT.png

（3）启动图配置

图片地址：https://static.oschina.net/uploads/img/201908/15120859_RPfD.png

（3） SDK配置 （暂时不用选择）

（4）模块权限配置

图片地址：https://static.oschina.net/uploads/img/201908/15121140_DDEa.png

打包后app点击返回键直接退出处理

在index.html中增加如下代码

```
<script type="text/javascript">
   //如下代码主要是解决打包后的app点击返回键直接退出
   document.addEventListener('plusready', function(a) { //等待plus ready后再调用5+
API：
            //// 在这里调用5+ API
            var first = null;
            plus.key.addEventListener('backbutton', function() { //监听返回键
                  //首次按键，提示 ‘再按一次退出应用’
                  if (!first) {
                     first = new Date().getTime(); //获取第一次点击的时间戳
                     // console.log('再按一次退出应用');//用自定义toast提示最好
                     // toast('双击返回键退出应用'); //调用自己写的吐丝提示 函数
                     plus.nativeUI.toast("双击退出", {duration:'short'}); //通过H5+ API 调用
Android 上的toast 提示框
                     setTimeout(function() {
                        first = null;
                     }, 1000);
                  } else {
                     if (new Date().getTime() - first < 1000) { //获取第二次点击的时间戳, 两
次之差 小于 1000ms 说明1s点击了两次,
                        plus.runtime.quit(); //退出应用
                     }
                  }
               }, false);
         });
</script>
```

四、打包APP

【发行-原生APP-云打包】

图片地址：https://static.oschina.net/uploads/img/201908/15121507_IOdE.png

打包后控制台如下打印

图片地址：https://static.oschina.net/uploads/img/201908/15121636_LVsf.png

根据下载地址下载APP,手机安装即可！！

CAS单点登录对接

1、CAS单点登录服务端准备

> 搭建CAS服务端，如果已经部署服务端，此步骤可以省略！！

CAS单点登录对接前期准备

（1）单点登录的介绍

单点登录（Single Sign On ,简称SSO）是目前比较流行的服务于企业业务整合的解决方案之一，SSO 使得在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。

CAS（Central Authentication Service）是 Yale大学发起的一个企业级的、开源的项目，旨在为 Web 应用系统提供一种可靠的单点登录解决方法

(2)CAS服务端搭建

> 此处不做详细搭建部署介绍，可以网络查询资料部署, 官网地址 https://github.com/apereo/cas-overlay-template

当前我对接的CAS版本是5.2X，也可以使用 5.3X

注意！注意！注意！ 重要的事情说三遍！

CAS服务端必须要开启rest支持

添加如下依赖即可：

```
<!--开启cas server的rest支持-->
<dependency>
    <groupId>org.apereo.cas</groupId>
    <artifactId>cas-server-support-rest</artifactId>
    <version>${cas.version}</version>
</dependency>
```

服务端部署完成后能正常使用系统用户进行登录登出即可

图片地址：https://static.oschina.net/uploads/img/201908/02195957_kM3e.png

图片地址：https://static.oschina.net/uploads/img/201908/02200015_C5n8.png

2、Jeecg Boot 后端对接CAS步骤

> Jeecg Boot后端加入对应代码，支持单点登录（实际是支持通过ticket获取用户信息，进行模拟登录）

1.从网盘下载代码添加到后端项目工程中

```
代码下载地址：
链接: https://pan.baidu.com/s/1WsW3pKBOeEjy6LDFYWikfQ 提取码: bu2v
```

2、解压后拷贝cas目录下代码到工程目录/org/jeecg/modules下

3、application.yml 进行如下配置

```
cas:
  # 配置CAS服务地址，cas为工程目录，部署到ROOT目录下https://cas.test.com:8443即可
  prefixUrl: https://cas.test.com:8443/cas
```

4、ShiroConfig中对该请求进行权限排除

/src/main/java/org/jeecg/config/ShiroConfig.java 类中方法 shiroFilter

进行如下配置：

```
//cas验证登录
filterChainDefinitionMap.put("/cas/client/validateLogin", "anon");
```

以上完成Jeecg Boot后端对接CAS所有操作，启动项目即可

3、Jeecg Boot 前端项目对接CAS步骤

> Jeecg Boot前端加入对应代码，支持单点登录（这块已经做了封装，集成非常简单）
>
> 思路： 前端vue层面判断登录状态，未登录直接在vue层面跳转到CAS服务器，CAS登录成功调回vue页面带着参数ticket，vue再用ticket进行模拟登录。

1、从网盘下载代码，拷贝到前端工程目录下

代码下载地址：
链接: https://pan.baidu.com/s/1uylDHry0WWgkEvvSqD8ITA 提取码: 6iya

2、下载代码解压后，把sso目录下代码拷贝到前端工程/src目录下

3、index.html页面增加CAS服务地址配置

该地址与第二步与后端对接时配置的地址一致

```
<script>
 window._CONFIG = {};
 window._CONFIG['casPrefixUrl'] = 'https://cas.test.com:8443/cas';
</script>
```

4、src/store/modules/user.js文件中增加验证登录方法

（1）js引入

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

(2)actions中增加如下方法

```
// CAS验证登录
    ValidateLogin({ commit }, userInfo) {
      return new Promise((resolve, reject) => {
        getAction("/cas/client/validateLogin",userInfo).then(response => {
          console.log("----cas 登录--------",response);
          if(response.success){
            const result = response.result
            const userInfo = result.userInfo
            Vue.ls.set(ACCESS_TOKEN, result.token, 7 * 24 * 60 * 60 * 1000)
            Vue.ls.set(USER_NAME, userInfo.username, 7 * 24 * 60 * 60 * 1000)
            Vue.ls.set(USER_INFO, userInfo, 7 * 24 * 60 * 60 * 1000)
            commit('SET_TOKEN', result.token)
            commit('SET_INFO', userInfo)
            commit('SET_NAME', { username: userInfo.username,realname:
userInfo.realname, welcome: welcome() })
            commit('SET_AVATAR', userInfo.avatar)
            resolve(response)
          }else{
```

```
          resolve(response)
        }
      }).catch(error => {
        reject(error)
      })
    })
  },
```

（3）登出方法改造

```
// 登出
  Logout({ commit, state }) {
    return new Promise((resolve) => {
      let logoutToken = state.token;
      commit('SET_TOKEN', '')
      commit('SET_PERMISSIONLIST', [])
      Vue.ls.remove(ACCESS_TOKEN)
      //console.log('logoutToken: '+ logoutToken)
      logout(logoutToken).then(() => {
        resolve()
      }).catch(() => {
        resolve()
      })
    })
  },
```

改造为：

```
// 登出
  Logout({ commit, state }) {
    return new Promise((resolve) => {
      let logoutToken = state.token;
      commit('SET_TOKEN', '')
      commit('SET_PERMISSIONLIST', [])
      Vue.ls.remove(ACCESS_TOKEN)
      //console.log('logoutToken: '+ logoutToken)
      logout(logoutToken).then(() => {
        var sevice = "http://"+window.location.host+"/";
        var serviceUrl = encodeURIComponent(sevice);
        window.location.href =
window._CONFIG['casPrefixUrl']+"/logout?service="+serviceUrl;
        //resolve()
      }).catch(() => {
```

```
          resolve()
        })
      })
    },
```

5、改造src/main.js代码，增加如下代码

（1）引入js

```
import SSO from '@/sso/sso.js'
```

（2）改造下面代码

```
new Vue({
    router,
    store,
    mounted () {
      store.commit('SET_SIDEBAR_TYPE', Vue.ls.get(SIDEBAR_TYPE, true))
      store.commit('TOGGLE_THEME', Vue.ls.get(DEFAULT_THEME, config.navTheme))
      store.commit('TOGGLE_LAYOUT_MODE', Vue.ls.get(DEFAULT_LAYOUT_MODE,
config.layout))
      store.commit('TOGGLE_FIXED_HEADER', Vue.ls.get(DEFAULT_FIXED_HEADER,
config.fixedHeader))
      store.commit('TOGGLE_FIXED_SIDERBAR', Vue.ls.get(DEFAULT_FIXED_SIDEMENU,
config.fixSiderbar))
      store.commit('TOGGLE_CONTENT_WIDTH',
Vue.ls.get(DEFAULT_CONTENT_WIDTH_TYPE, config.contentWidth))
      store.commit('TOGGLE_FIXED_HEADER_HIDDEN',
Vue.ls.get(DEFAULT_FIXED_HEADER_HIDDEN, config.autoHideHeader))
      store.commit('TOGGLE_WEAK', Vue.ls.get(DEFAULT_COLOR_WEAK,
config.colorWeak))
      store.commit('TOGGLE_COLOR', Vue.ls.get(DEFAULT_COLOR, config.primaryColor))
      store.commit('SET_TOKEN', Vue.ls.get(ACCESS_TOKEN))
      store.commit('SET_MULTI_PAGE',Vue.ls.get(DEFAULT_MULTI_PAGE,true))
    },
    render: h => h(App)
  }).$mount('#app')
```

改造为：

```
SSO.init(() => {
  main();
});
```

```
function main() {
  new Vue({
    router,
    store,
    mounted () {
      store.commit('SET_SIDEBAR_TYPE', Vue.ls.get(SIDEBAR_TYPE, true))
      store.commit('TOGGLE_THEME', Vue.ls.get(DEFAULT_THEME, config.navTheme))
      store.commit('TOGGLE_LAYOUT_MODE', Vue.ls.get(DEFAULT_LAYOUT_MODE,
config.layout))
      store.commit('TOGGLE_FIXED_HEADER', Vue.ls.get(DEFAULT_FIXED_HEADER,
config.fixedHeader))
      store.commit('TOGGLE_FIXED_SIDERBAR', Vue.ls.get(DEFAULT_FIXED_SIDEMENU,
config.fixSiderbar))
      store.commit('TOGGLE_CONTENT_WIDTH',
Vue.ls.get(DEFAULT_CONTENT_WIDTH_TYPE, config.contentWidth))
      store.commit('TOGGLE_FIXED_HEADER_HIDDEN',
Vue.ls.get(DEFAULT_FIXED_HEADER_HIDDEN, config.autoHideHeader))
      store.commit('TOGGLE_WEAK', Vue.ls.get(DEFAULT_COLOR_WEAK,
config.colorWeak))
      store.commit('TOGGLE_COLOR', Vue.ls.get(DEFAULT_COLOR, config.primaryColor))
      store.commit('SET_TOKEN', Vue.ls.get(ACCESS_TOKEN))
      store.commit('SET_MULTI_PAGE',Vue.ls.get(DEFAULT_MULTI_PAGE,true))
    },
    render: h => h(App)
  }).$mount('#app')
}
```

6、组织机构登录切换 src/components/tools/UserMenu.vue 增加 mounted()

```
mounted(){
      let depart = this.userInfo().orgCode;
      if(!depart){
        this.updateCurrentDepart();
      }
  },
```

以上步骤完成后，CAS对接完成，启动前端项目 访问http://localhost:3000 即可

Online开发

Online开发初体验—Jeecg-Boot 在线配置图表

> **Online开发——初体验（在线配置图表）**
>
> - 01 通过JSON数据，快速配置图形报表
>
> - 02 通过SQL数据，快速配置图形报表
>
> - 03 图表模板配置，实现不同数据源图表合并展示
>
> - 04 图表布局，支持单排、双排、组合、TAB

图片地址：https://oscimg.oschina.net/oscnet/220b8d5cbb4df49218db7ecb6bc6348cce5.jpg

> 演示如何通过JSON数据，快速的配置一个图形报表，支持曲线、柱状图、饼状图等

图片地址：http://jeecg.org/gif/online-graphic-02.gif

> 演示如何通过SQL查询数据库，快速的配置一个图形报表，支持曲线、柱状图、饼状图等

图片地址：https://oscimg.oschina.net/oscnet/bb180e13a62abd3d3b7cece57900a6d774e.jpg

> 演示如何实现一个复杂的报表模板，从不同的数据源取数据，展示不同的图表内容，组合展示

图片地址：https://oscimg.oschina.net/oscnet/8cc56a48abc7fabae0257da9d81cbae9934.jpg

> 演示如何实现图表的多种排版布局，支持单排、双排、组合、TAB

Online表单

online 基础篇-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.操作截图

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

online 基础篇-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： 和前端开发保持一致,使用postAction,getAction,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/201904/25115420_JSp2.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/201904/25115023_4L8K.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 基础篇-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_user_code}	登陆用户的ID

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

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

online 基础篇-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

Online之JS增强-关联修改控件值【单(主)表】

功能说明：在online表单中，当改变表单某一控件值的时候关联改变其他控件的值,【仅限单表/主表】

示例说明: 出生日期(birthday)被改变的时候改变对应的年龄(age),步骤如下

1.配置form的增强JS:（JS增强定义见online基础篇）(http://jeecg-boot.mydoc.io/?t=345710)

图片地址：https://static.oschina.net/uploads/img/201904/28181449_syHc.png

代码如下：

```
onlChange(){
  return {
   birthday(){
     let value = event.value
     let currBirthday = new Date(value.replace(/-/g, "\/"));
     let d = new Date();
     let age = d.getFullYear()-currBirthday.getFullYear()
     let values = {'age':age}
     that.triggleChangeValues(values)
   }
 }
}
```

2.测试

图片地址：https://static.oschina.net/uploads/img/201904/28181640_PnOC.png

Summary：配置form表单JS增强 -单表实现关联change有如下几点需要注意（如1图例子）

1.方法名规则是：表名+'_onlChange'或是直接写onlChange也行(单/主表才支持这种命名规则)

2.上述方法会返回一个对象，对象中的属性名和数据库的字段名保持一致，上述1定义表示：当birthday字段内容改变的时候会触发birthday方法改变age字段的值

3.每个字段方法内有两个内置参数that和event，that指向当前页面的vue实例对象,event对象包含属性如下表

4.使用getAction发起请求

5.使用that.triggleChangeValues(values)改变其他控件的值，values是一个对象可以配置多个控件的值

> event对象描述如下

属性名	描述
row	当前表单的数据,编辑页面通过row.id可以获取当前表单的id值
column	当前列的配置信息,通过column.key 获取当前字段名称
value	当前控件的值

Online之JS增强-关联修改控件值【从表】

功能说明：在online主从表表单中，当表表单某一控件值改变从的时候关联改变其他控件的值,【以

下说明仅限于从表】

示例说明：当改变学校字段(school)的时候对应的描述字段(note)跟着改变 ,步骤如下

1.配置form的增强JS:（JS增强定义见online基础篇）(http://jeecg-boot.mydoc.io/?t=345710)

图片地址：https://static.oschina.net/uploads/img/201904/28184524_WcRs.png

代码如下：

```
a_sub_kua_onlChange(){
    return {
        school(){

            let id = event.row.id
            let value = event.value
            let targrt = event.target
            let columnKey = event.column.key

            getAction('/test/jeecgDemo/getNote',{school:value}).then(res=>{
                let otherValues = {'note':res}

                that.triggleChangeValues(otherValue,id,targrt)
            })

        }
    }
}
```

2.编写后台代码接收请求

图片地址：https://static.oschina.net/uploads/img/201904/25172905_Bfer.png

代码如下：

```
  @Autowired
  private ISysDictService sysDictService;

  @GetMapping("/getNote")
  public String getNote(@RequestParam(name="school") String school) {
    //TODO 业务自定义 此处下拉框的value是字典项的编码需要将其转换成名称
    String schoolName = sysDictService.queryDictTextByKey("school", school);
    return schoolName+"是一所学校";
  }
```

3.测试：

图片地址：https://static.oschina.net/uploads/img/201904/25173031_lnVa.png

Summary：配置form表单JS增强 -主子表实现子表的关联change有如下几点需要注意（如1图例子）

1. 方法名规则是：表名+'_onlChange',每张表对应一个一个方法

2. 上述方法会返回一个对象，对象中的属性名和数据库的字段名保持一致，上述1定义表示：当school字段内容改变的时候会触发school方法

3. 每个字段方法内有两个内置参数that和event，that指向当前页面的vue实例对象,event对象包含属性如下表

4. 使用getAction发起请求

5. 使用that.triggleChangeValues(values,id,target)改变其他控件的值，values是一个对象可以配置多个控件的值

> event对象描述如下

属性名	描述
type	当前操作控件的类型
row	当前行的数据,通过row.id可以获取当前行的id值
column	当前列的配置信息,通过column.key 获取当前字段名称
value	当前控件的值
target	当前控件所在table的target对象,调用triggleChangeValues用到

Online之JS增强+自定义按钮

> 以下内容基于online自定义按钮和online js增强,如遇到规则定义不明确或是代码不理解的情况请先阅读基础篇的相关文档

1. 通过自定按钮+js增强实现点击按钮控制表单控件的显示/隐藏（以online demo表为例演示控制姓名控件的显示/隐藏）

• 1-1.进入online表单开发页面,选择demo表记录，点击自定义按钮，新增两个按钮，如下图

图片地址：https://static.oschina.net/uploads/img/201905/08160359_MF7I.png

• 1-2. 进入online表单开发页面,选择demo表记录，点击JS增强，编写js，按钮编码对应方法名，如下

图片地址：https://static.oschina.net/uploads/img/201905/08161327_yRfD.png

• 1-3. 表单页效果如下：

https://static.oschina.net/uploads/img/201905/08161826_RaDv.png

2.【主子表】通过自定按钮+js增强实现一键增加子表的多条数据/清除子表的多条数据

子表表名：a*sub*kua

子表数据字段如下(省略部分字段)

字段名	描述
study_begin	开始时间
study_end	结束时间
school	学校
note	备注

• 2-1. 进入online表单开发页面,选择一张主表记录，点击自定义按钮，新增两个按钮

图片地址：https://static.oschina.net/uploads/img/201905/10150021_NWPs.png

• 2-2. 进入online表单开发页面,选择上述主表记录，点击JS增强，编写js，按钮编码对应方法名，如下

图片地址：https://static.oschina.net/uploads/img/201905/10152506_No0O.png

> 注意：
>
> 1.新增数据,此处是自定义一个测试的数据对象，实际开发中可以向后台发起请求获取数据,然后添加到子表
>
> 2.此处有几个内置方法可以调用

方法名	参数	描述
clearSubRows	tbname	传入参数子表名，用于清除子表数据。(示例如上图)
addSubRows	tbname,rows	传入参数子表名和数据对象，用于新增子表的数据，rows可以是一个对象，也可以是一个数组(一次性新增多条记录)(示例如上图)
clearThenAddRows	tbname,rows	传入参数子表名和数据对象，用于新增子表的数据，在新增前先清除子表现有数据，参数用法同addSubRows

• 2-3. 新增测试结果：

图片地址：https://static.oschina.net/uploads/img/201905/10154840_0Mh9.png

• 2-4. 删除测试结果

图片地址：https://static.oschina.net/uploads/img/201905/10154942_x8WA.png

图片地址：https://static.oschina.net/uploads/img/201905/10154959_HgND.png

Online之popup使用

• 1 . 功能说明：online-demo表单在简介字段上配置popup关联online报表,popup选择完成后,将姓名、性别赋值到表单上

• 2 . 配置online报表（如图2-1）

图片地址：https://static.oschina.net/uploads/img/201905/09204132_71Hf.png

• 3 . 配置online表单-demo表的简介字段的页面属性为popup弹出框

图片地址：https://static.oschina.net/uploads/img/201905/09203707_ZXiO.png

• 4 . 配置online表单-demo表的简介字段的> 校验字段，如下图

图片地址：https://static.oschina.net/uploads/img/201905/09203827_3Uyw.png

注意：校验字段定义规则

> 1.字典table定义成online报表的编码如图2-1
>
> 2.字典code定义成online报表的结果字段，需要使用哪些字段的值就拼接哪些字段，字段之间用逗号分隔
>
> 3.字典text定义成online表单的字段，需要设置哪些字段的值就拼接哪些字段，字段之间用逗号分隔，这和online报表的字段一一对应，所以字段顺序很重要
>
> 4 . 上述字典text配置online表单的字段，第一个字段一定是当前字段，那么对应的字典code的第一个字段的取值即为当前字段的值

• 5 . 测试表单

图片地址：https://static.oschina.net/uploads/img/201905/09204748_BfCS.png

Online表单数据权限

以online表单的demo表为例演示数据权限的配置

1.进入online的demo表的信息维护界面

图片地址：https://static.oschina.net/uploads/img/201905/07181450_6GMU.png

图片地址：https://static.oschina.net/uploads/img/201905/07181922_viNE.png

2.拿到上图ID：4028f6816a4f0a52016a4f0a52cc0000 配置权限菜单：

图片地址：https://static.oschina.net/uploads/img/201905/07182120_pPdA.png

> **此处有两点注意（如上图圈圈）：**

1.菜单类型选择：按钮/权限 类型

2.菜单路径配置为： /online/cgform/api/getData/ + 图1-2中的ID字符串

3.配置菜单数据规则（此处配置一个的数据规则：筛选性别为男性的数据）

图片地址：https://static.oschina.net/uploads/img/201905/07182714_WtcL.png

图片地址：https://static.oschina.net/uploads/img/201905/07182759_xqjO.png

4.菜单授权（先勾选菜单保存后再操作菜单的数据权限）

图片地址：https://static.oschina.net/uploads/img/201905/07183050_o6QE.png

5.重新进入demo测试页，测试结果如下：

图片地址：https://static.oschina.net/uploads/img/201905/07183230_kiBy.png

Online表单权限配置按钮/列的显示或隐藏

一、列的显示/隐藏

> 以online的demo表为例，控制demo列表的字段{名字} {年龄} {工资}显示与隐藏。

- 1、进入online的demo表的信息维护界面，查看数据列（如下图1-2）

图片地址：https://static.oschina.net/uploads/img/201905/07181450_6GMU.png

需要控制以下三个字段

图片地址：https://static.oschina.net/uploads/img/201905/09095043_r0hb.png

- 2、配置权限菜单

图片地址：https://static.oschina.net/uploads/img/201905/08140902_ISha.png

图片地址：https://static.oschina.net/uploads/img/201905/08142214_LRj9.png

> 注意：
>
> 1.Online表单字段权限菜单统一配置在Online表单权限这个菜单下
>
> 2.每个表单独配置一个按钮类型的父菜单如图2-1
>
> 3.每个字段权限的菜单必须填写授权标识，此标识声明规则为：online:表名:字段名 如图2-2

• 3、 角色授权

图片地址：https://static.oschina.net/uploads/img/201905/08141857_9DeR.png

> 注意，以下两种情况的配置均会使列隐藏：
>
> 1.配置的权限菜单但是没有角色授权
>
> 2.配置了权限菜单且角色授权了，但是授权策略配置（见图2-2）没有选择显示

• 4、 针对上述两种情况配置年龄/名称字段使其隐藏，分别将年龄字段的角色授权取消、名称字段授权策略配置成不显示

图片地址：https://static.oschina.net/uploads/img/201905/08143524_9fk6.png

图片地址：https://static.oschina.net/uploads/img/201905/08143436_cVA8.png

5.查看测试结果：对比图1-2可知权限控制有效

图片地址：https://static.oschina.net/uploads/img/201905/08143728_ZI8N.png

二、按钮的显示/隐藏

• 控制online列表页面新增/编辑 按钮权限，每个按钮对应一个唯一的编码。

• 权限配置规则 online:表名:按钮编码

• 自定义按钮对应其自定义的按钮编码,其余操作同上篇列的显示/隐藏。

按钮名称	按钮编码	权限授权标识
新增	add	online:{表名}:add
编辑	update	online:{表名}:update
删除	delete	online:{表名}:delete
批量删除	batch_delete	online:{表名}:batch_delete
导入	import	online:{表名}:import
导出	export	online:{表名}:export

Online表单 扩展控件类型

> **需求描述：**现有的online表单配置的页面属性-控件类型不满足实际开发 需要扩展自定义的控件
>
> **示例说明：**扩展一个下拉多选控件

步骤如下：

• 1、新增控件类型

> 找到前端vue文件 srcviewsmodulesonlinecgformtablesPageAttributeTable.vue 修改fieldShowType配置，增加一个option，代码如下

图片地址：https://static.oschina.net/uploads/img/201905/14111139_gxyM.png

• 2、新增Widget

> 在文件夹srccomponentsonlineautoformview下新建一个vue文件,SelectMultiWidget 【文件取名建议：以Widget结尾】页面代码如下：

图片地址：https://static.oschina.net/uploads/img/201905/14110811_5oGR.png

• 3、注册Widget

> 在srccomponentsonlineautoformindex.js中新增代码如下：

图片地址：https://static.oschina.net/uploads/img/201905/14111611_UAQD.png

• 4.功能测试

• 4-1 配置表单的某一字段的控件类型

图片地址：https://static.oschina.net/uploads/img/201905/14112504_DRU4.png

• 4-2 配置表单的某一字段的字典信息

图片地址：https://static.oschina.net/uploads/img/201905/14112528_9WIs.png

• 4-3 测试表单显示结果如下

图片地址：https://static.oschina.net/uploads/img/201905/14112610_ZK6U.png

Online树形列表配置介绍

> 1.表配置

图片地址：https://static.oschina.net/uploads/img/201906/10152627_E3j1.png

> 2.数据库字段配置

图片地址：https://static.oschina.net/uploads/img/201906/10153128_uaDd.png

> 3.页面属性配置

图片地址：https://static.oschina.net/uploads/img/201906/10153215_aDxX.png

“

4.测试效果如下：

图片地址：https://static.oschina.net/uploads/img/201906/10153331_WO6q.png

Online表单配置

Online表单配置参数说明

Online表单配置

Online 表单支持：单表模型、一对多模型、一对一模型、树模型。

一、表单参数介绍

1.主参数说明

配置	配置说明
表名	数据库表名
表描述	数据库表描述
表类型	表的模型，分单表、一对多、树
表单分类	区分表种类的（不重要）
主键策略	主键策略：UUID/NATIVE(自增)/SEQUENCE（适合oracle）此处暂时统一使用UUID
PC表单风格	PC端，表单添加页面和修改页面的风格 分为1/2/3/4列展示
查询模式	暂时系统内置为多字段查询
显示复选框	生成的表单数据列表，是否带着checkbox 系统内置为是
是否分页	生成的表单数据列表，是否分页展示 系统内置为是
是否树:	控制表单类型，树类型表单需要选择是

树形表单父id	树类型表单，用于控制上下级父子关系字段
是否有子节点字段	树类型表单，用于控制树节点是否能展开
树开表单列	树类型表单，列表页面用于折叠展示的字段

2.TAB明细页功能说明

序号	名称	功能说明
1	数据库属性	对应数据库表字段的配置
2	页面属性	1.对应表单字段展示控件效果，可定义控件类型，控件长度; 2. 查询条件配置; 3. 控制字段是否显示; 4. 表单字段支持各种控件
3	校验字段	1. 用于设置表单字段对应的字典：比如性别男女，popup对应的报表；2.设置字段的校验规则： 手机号，邮箱等等3.设置列表字段，弹出页面链接
4	外键	外键设置，目前外键关系是在附表中设置
5	索引	对应数据库表索引，支持单字段索引，多字段索引索引类型: 普通索引\唯一索引

详细说明:

• 页面属性的字段控件支持
：text/password/select/radio/checkbox/date/datetime/file/textarea/UE编辑器/popup等

图片地址：https://static.oschina.net/uploads/img/201906/14172200_thPk.png

Online表单标准字段说明

Online标准字段

Online表单建表默认标准字段，不要轻易删除，方便数据权限管理；

约定字段名称	约定字段释义

create_time	创建日期
create_by	创建人用户账号
update_time	修改日期
update_by	更新人用户账号
sys_org_code	创建人所属部门

在表单配置中如果存在上述字段，则表单在进行保存或者更新时，会默认填充为系统变量的值。
示例中配置了系统变量的字段。
图片地址：https://static.oschina.net/uploads/img/201906/14181011_KyfT.png

Online表单一对多表单开发

一对多类型表单
Online 支持一对多，一对一模型表单配置。

1.创建主表/附表

主表创建，设置表单类型为主表；
图片地址：https://static.oschina.net/uploads/img/201906/14175619_A9eu.png

附表创建，设置表单类型为附表，配置外键字段
图片地址：https://static.oschina.net/uploads/img/201906/14175740_no1W.png

附表外键设置，设置该字段对应的主表名、主表字段
图片地址：https://static.oschina.net/uploads/img/201906/14175816_TWuy.png

主子表关系会自动展示在主表的附表字段（多个附表会以逗号分割）
主表和附表创建完后，分别点击同步数据库，创建表。
说明： 附表不能单独维护数据；

> 一对一和一对多配置，通过表类型设置，子表tab展示顺序通过序号控制。

图片地址：https://static.oschina.net/uploads/img/201804/16141616_IYFO.png

2.测试功能

主表作为统一表单功能测试入口，附表不提供功能测试入口；
点击主表，功能测试链接进入表单列表, 打开表单界面
图片地址：https://static.oschina.net/uploads/img/201906/14175956_UZMb.png

Online表单树型表单开发

树表类型表单

表单创建，基础配置如下：

1.设置表单类型为：单表;

2.是否树选择：是;

3.设置特殊字段：【树形表单父id】【是否有子节点字段】【树开表单列】

图片地址：https://static.oschina.net/uploads/img/201906/14180639_bdLJ.png

效果如：

图片地址：https://static.oschina.net/uploads/img/201906/14180707_NKpI.png

Online表单删除说明

表单删除

表单删除分两种模式：删除移除

删除： 删除online配置表的同时会删除数据库物理表。

移除： 只删除online配置表，数据库物理表保留。

图片地址：https://static.oschina.net/uploads/img/201906/14181149_PPJD.png

Online表单控件

下拉框、多选框、单选框配合数据字典使用

数据字典

表单字段，通过系统配置字典，展示下拉、Checkbox、Radio等

首选配置表单字段页面控件类型：

图片地址：https://static.oschina.net/uploads/img/201906/14183821_T47p.png

配置字字典：

图片地址：https://static.oschina.net/uploads/img/201906/14183755_CqXD.png

字段code对应： 系统管理 -> 数据字典

图片地址：https://static.oschina.net/uploads/img/201906/14183929_dBT5.png

演示：

图片地址：https://static.oschina.net/uploads/img/201906/14184017_qPRS.png

下拉多选

数据字典

表单字段，通过系统配置字典，展示下拉，支持多选

首选配置表单字段页面控件类型：

图片地址：https://static.oschina.net/uploads/img/201906/14190227_RmIo.png

配置字字典：

图片地址：https://static.oschina.net/uploads/img/201906/14190316_JUDl.png

字段code对应： 系统管理 -> 数据字典

图片地址：https://static.oschina.net/uploads/img/201906/14190356_UUwh.png

演示：

图片地址：https://static.oschina.net/uploads/img/201906/14190420_yhRs.png

下拉搜索

• 1.页面属性配置控件类型

图片地址：https://static.oschina.net/uploads/img/201906/14191139_vmAm.png

• 2.校验字段配置字典表

图片地址：https://static.oschina.net/uploads/img/201906/14191242_VUg1.png

• 3.演示，默认加载所有数据(下图1),可输入j进行筛选(下图2)

图片地址：https://static.oschina.net/uploads/img/201906/14191342_Rzkn.png

图片地址：https://static.oschina.net/uploads/img/201906/14191439_OES9.png

Popup控件

Popup控件

popup选择框的使用依赖于Online报表

（1）创建一个Online报表来提供弹出数据列表的数据集

图片地址：https://static.oschina.net/uploads/img/201906/14184337_JO1E.png

（2）选择控件类型为popup弹出框

图片地址：https://static.oschina.net/uploads/img/201906/14184421_GorQ.png

（3）字典Table、字典Code、字典Text项填写对应的Online报表信息

> 字典Table :填写online报表编码
> 字典Code:填写 需要写入表单中的字段名
> 字典Text: 填写online报表列表字段名
> 如下设置：
> 把报表user_msg查出的字段 realname,username 选择后分别写入表单中popupok和popback

图片地址：https://static.oschina.net/uploads/img/201906/14184634_8YNp.png

（4）展示效果

图片地址：https://static.oschina.net/uploads/img/201906/14184901_gZl1.png

点击输入框弹出报表列表

图片地址：https://static.oschina.net/uploads/img/201906/14185014_GywK.png

选择后数据带入表单

图片地址：https://static.oschina.net/uploads/img/201906/14185025_xh6U.png

时间控件

时间控件

目前支持两种时间类型：年月日/年月日 时分秒

图片地址：https://static.oschina.net/uploads/img/201906/14185303_cM4D.png

富文本控件

富文本控件

目前采用tinyMCE编辑器，作为系统默认富文本编辑

图片地址：https://static.oschina.net/uploads/img/201906/14185527_cjmm.png

演示：

图片地址：https://static.oschina.net/uploads/img/201906/14185554_tcnO.png

字段显示隐藏控制

字段显示隐藏

Online可控制字段在列表中和表单中是否显示；

图片地址：https://static.oschina.net/uploads/img/201906/14185749_EX0U.png

字段排列顺序调整

字段排列顺序

表单和列表字段显示的先后顺序是按照表单设计顺序来展示的

图片地址：https://static.oschina.net/uploads/img/201906/14185855_6EXp.png

字段校验规则使用

字段校验规则

校验规则可扩展，支持非空，邮箱，数字、手机号正则等等校验规则。

如果选项里没有你需要的规则，你还可以通过输入的正则表达式的方式来进行配置校验规则（详细配置方式可见下方的演示）。

需要注意点：如果同时校验数字，并且校验必须输入，请选择校验必填字段。

图片地址：https://static.oschina.net/uploads/img/201906/14185956_3WUr.png

配置自定义校验规则演示

下图中配置了一段正则表达式：**^123$** ，意思是只有输入了 "**123**" 的时候才通过校验，你也可以输入与你需求相匹配的正则表达式，关于正则表达式，可以点我查看教程 (https://www.runoob.com/regexp/regexp-tutorial.html)

图片地址
：http://www.jeecg.org/data/attachment/forum/201908/16/160834bn2rv4uczuuzm2pa.gif

用户选择器

• 1.页面属性配置控件类型

图片地址：https://static.oschina.net/uploads/img/201906/14191654_Y9eP.png

• 2.演示

• 2-1. 表单页面控件显示

图片地址：https://static.oschina.net/uploads/img/201906/14191803_lve3.png

• 2-2. 点击选择用户按钮弹框可选择用户

图片地址：https://static.oschina.net/uploads/img/201906/14191851_VQFI.png

• 2-3. 弹框选择数据回填至表单

图片地址：https://static.oschina.net/uploads/img/201906/14191930_hwo9.png

部门选择器

• 1.配置页面属性的控件类型

图片地址：https://static.oschina.net/uploads/img/201906/14192119_ijCt.png

Online报表

online报表 动态参数

概述：online报表可以在sql中写入参数，然后访问这个报表数据的时候，在访问地址中给这个参数赋值，实现online报表的动态参数。

示例步骤：

• 1.定义一个online报表，如下图

图片地址：https://static.oschina.net/uploads/img/201905/18175558_1095.png

> 备注：SQL解析成功则会在动态报表配置明细中新增多行配置信息，若不能正常解析，检查自

己的SQL语句，是否写入了sql函数并且在函数中定义了动态参数 如`

name like concat('%','${username}','%')

`

这样是不支持的，但是想要在sql中这么用怎么办？方法如下

1.将该条件去掉，解析sql

2.待sql解析完成再拼上去

3.在报表参数tab下，新增一个参数名为username 对应${username}

• 2.进入online报表信息的列表页面，找到上述新增的报表记录,在更多项中点击配置地址，在弹出框中点击复制，即可拿到访问该报表的地址

图片地址：https://static.oschina.net/uploads/img/201905/18175704_LXNg.png

图片地址：https://static.oschina.net/uploads/img/201905/18175950_p71m.png

• 3.此处拿到的地址为：/online/cgreport/aa48a426ed5b46dd9224e68acf4a90d8?sex=${sex}，只需要将${sex}替换成需要传入的值即可，如下例

图片地址：https://static.oschina.net/uploads/img/201905/18180435_VvIX.png

online报表 使用步骤

• 1.进入online报表配置页面,新增一个报表

图片地址：https://static.oschina.net/uploads/img/201905/24104953_wfWf.png

• 2.填写编码、名称、报表SQL,点击sql解析，解析完成会根据sql查询的字段生成配置明细

图片地址：https://static.oschina.net/uploads/img/201905/24105255_Brul.png

• 3.可以配置字段文本、字段类型、是否显示、是否查询、查询模式、字典信息等等，配置完成，点击确定，保存信息。

图片地址：https://static.oschina.net/uploads/img/201905/24105950_Agp8.png

• 4.进入列表页面,在记录的操作列【更多】选项中，点击功能测试，进入报表页面,也可以点击配置地址，复制出访问链接访问报表页面,如下：

图片地址：https://static.oschina.net/uploads/img/201905/24110112_w6s5.png

Online图表

Online图表配置手册

Online 图表配置文档

一、配置单个数据源的图表

• 配置地址: /online/graphreport

具体步骤

1. 在页面中点击 新增 按钮
2. 在打开的弹窗中输入你的图表信息。其中，必填项有：

- 图表名称
- 编码（编码是唯一的）
- X轴字段（数据源中被当做 X 轴的字段）
- Y轴字段（数据源中被当做 Y 轴的字段）
- 查询SQL/数据JSON

1. 其中有几个动态的内容区域，分别是：

- 当 `数据类型` 字段选为 `JSON` 后，`查询SQL` 字段会被替换成 `数据JSON` 字段，该字段会验证你的JSON字符串格式是否正确，反之则不变

1. 配置列表字段

- 列表字段前两项是配置 数据表格 的 `列（columns）`
  - `字段名` 是必填的，对应 `column.dataIndex`
  - `字段文本` 是对字段名的描述，对应 `column.title`，不填则不显示
- `是否显示` 默认勾选，如果去掉勾选则不显示此列
- `计算总计` 默认不勾选，如果勾选上则会对当前列所有的数据进行求和，如果存在非数字的内容，则拒绝计算并提示"包含非数字内容"
- `是否查询` 默认不勾选，如果勾选上则会在图表最上方显示一个表单，用于筛选表格的数据
- `字段类型` 默认为空，可选择查询条件表单的类型，可选值有：数值类型、日期类型、字符类型、长整型
- `查询模式` 默认为空，可选择查询条件的筛选方式，如果选择了范围查询，则会显示两个表单，一个是开始值，一个是结束值，共同完成筛选

1. 点击右下角的确定按钮完成添加操作

使用方法

• 在 操作 列中，选中 更多，点击 功能测试 可以查看你配置的效果。
• 效果会根据不同的 展示模板 显示不同的布局

配置示例：JSON数据格式

假如我有一段JSON，我要将它配置成和下图一样图表，那么需要怎么做呢？

图片地址
：http://www.jeecg.org/data/attachment/forum/201904/24/153759lsqquv6uioutwopt.png

第一步：准备好你需要的JSON

```
[
   {"day": "星期一", "step": 1234, "assess": "良"},
   {"day": "星期二", "step": 1884, "assess": "优"},
   {"day": "星期三", "step": 1671, "assess": "良+"},
   {"day": "星期四", "step": 2197, "assess": "优+"},
   {"day": "星期五", "step": 1342, "assess": "中"},
   {"day": "星期六", "step": 545, "assess": "差"},
   {"day": "星期日", "step": 244, "assess": "极差"}
]
```

第二步：填写JSON

点击“新增”按钮，填写一些基本信息，然后将 数据类型 改为 JSON，然后将JSON填入数据JSON字段中，如下图

图片地址
：http://www.jeecg.org/data/attachment/forum/201904/24/152831h4di2s2zhzsiwr57.png

第三步：配置数据字段

数据字段即X轴字段、Y轴字段和Y轴文字。

X轴字段 顾名思义，就是需要在X轴显示的字段，根据上图示例图表中我们可以发现，X轴方向显示的是星期一到星期日，而在准备的json中，day字段是存储星期信息的，所以我们要将 X轴字段处填写成day。

Y轴字段 也是如此，即对应需要在Y轴上显示的字段，这里我们填写上 step

Y轴文字 是对Y轴数据的一个解释。这里我们填上步数，那么就会在鼠标悬浮在图表上时直观的显示出来，如下图所示。

图片地址
：http://www.jeecg.org/data/attachment/forum/201904/24/154819za9k1uz737c7nak7.png

第四步：配置数据表格的列

在列表字段下面的表格中配置，配置示例如下

图片地址
：http://www.jeecg.org/data/attachment/forum/201904/24/161336nwh2we0y5rrhw2z6.png

这里的配置是配置数据表格的列信息，只有配置上去的字段才会被显示出来。

数据表格可以计算列的总数，当计算总计被勾选上之后，会在数据表格最下面显示一行“总计”，当所有的计算总计都没被勾选的话，那么就不会显示这一行，如果要计算总计的列中某一行包含非数字的值，那么将会计算失败，并显示错误信息（包含非数字内容）

第五步：提交并测试功能

点击右下角的确定按钮并成功保存之后，我们可以在新增加的数据行右侧点击更多 --> 功能测试

图片地址：http://www.jeecg.org/data/attachment/forum/201904/24/160905jbsqsgnlqo9oh624.png

最终显示效果如下：

图片地址：http://www.jeecg.org/data/attachment/forum/201904/24/163104u1cuur0icf0700cm.png

我们发现只有一个柱状图，而刚刚配置的数据表格并没有显示出来，这是因为图表类型只配置了一个柱状图。我们回到列表页面，点击编辑按钮，在图表类型处勾选数据列表，如下图所示

图片地址：http://www.jeecg.org/data/attachment/forum/201904/24/162913n1h1bth7o8co1tdo.png

点击确定保存，再点击功能测试，最终显示效果如下：

图片地址：http://www.jeecg.org/data/attachment/forum/201904/24/162915izno5za14a7p5cyq.png

至此，配置JSON数据格式的图表就已经完成了

配置SQL数据格式

配置SQL数据格式的图表与JSON的步骤类似，只是需要将数据类型 改为 SQL即可，在查询SQL处填写上你的SQL语句，填写好X轴字段、Y轴字段和Y轴文字点击确定保存即可

二、配置多数据源的图表

多数据源图表可以将你配置过的单数据源图表整合到一个页面中，并且可以进行分组、排序

- 配置地址: /online/graphreport/templet（暂定）

具体步骤

1. 点击页面中的 新增 按钮
2. 在打开的弹窗中输入你的图表信息。其中，必填项有：

- 报表名称
- 报表编码（编码是唯一的）
- 报表风格（Tab风格、单排布局、双排布局、组合布局）

1. 图表配置

- 图表 （必填项，选择的是你配置过的`单数据源图表`）
- 图表类型（如果选择`不配置`，那么则应用选择的`单数据源图表`中配置的`图表类型`，如果配

置了则优先显示此处配置的图表类型）
- 组合编码（必填项，只能为数字，数字越小越往前排）

1. 点击右下角的确定按钮完成添加操作

注意事项

1. 如果报表风格配置成了组合布局，那么就会将配置的图表显示在一张图表内，并且图表类型只能配置成柱状图或曲线图，即使配置成了其他的类型，实际运行中也一样不会生效

使用方法

- 在 操作 列中，选中 更多，点击 功能测试 可以查看你配置的效果。
- 效果会根据不同的 报表风格和组合展示风格 显示不同的布局

配置示例

第一步：配置图表名称、编码、风格、多图表组合等

示例配置图如下

图片地址：http://www.jeecg.org/data/attachment/forum/201904/24/164212p0zdvmdq71ev17op.png

第二步：查看效果

在新增加的数据行右侧点击更多 --> 功能测试

图片地址：http://www.jeecg.org/data/attachment/forum/201904/24/160905jbsqsgnlqo9oh624.png

最终显示效果如下：

图片地址：http://www.jeecg.org/data/attachment/forum/201904/24/170233fmgumdu0gu01hgoz.png

图片地址：http://www.jeecg.org/data/attachment/forum/201904/24/170234ijwxdh2wwhx6h6dw.png

图片地址：http://www.jeecg.org/data/attachment/forum/201904/24/170237hyem041em70m6qua.png

图片地址：http://www.jeecg.org/data/attachment/forum/201904/24/170238xi5eza8z5h59ds5n.png

edu

Online表单专题课程

lesson one

一、online表单开发简单介绍

1.在线开发,通过在线配置,实现单表、树形列表、一对多表的建表及信息维护

2.除了基本的增删改查业务操作,提供更为强大的增强功能（sql增强、java增强、js增强）

3.遇到在线开发解决不了的超级复杂业务,系统提供了online代码生成器

4.降低开发成本提高开发效率

二、online表单创建、菜单授权、访问测试

先创建一个简单的表单、然后录入菜单、进行访问测试

录入菜单组件地址：
modules/online/cgform/auto/OnlCgformAutoList

---

lesson two

online表单配置讲解

表单控件：文本框、下拉框、下拉多选框、字典搜索框、popup弹出框、日期、文件、用户选择器、部门选择器

数据字典、表单校验、是否查询、范围查询

外键、索引

---

lesson three

java增强

详见：online基础篇-java增强(http://jeecg-boot.mydoc.io/?v=46121&t=345713)

---

lesson four

自定义按钮

按钮样式	说明	
link	挂在列表中操作列【更多】中	
button	列表上方新增按钮所在行	
form	表单页面右侧折叠处	

按钮类型	说明	
js	触发自定义js事件(编写js增强)	
action	调用后台请求 /online/cgform/api/doButton 触发sql增强	

详见：online基础篇-自定义按钮(http://jeecg-boot.mydoc.io/?v=46121&t=345710)

---

lesson five

sql增强

详见：online基础篇-sql增强(http://jeecg-boot.mydoc.io/?v=46121&t=345712)

---

lesson six

js增强

详见：online基础篇-js增强(http://jeecg-boot.mydoc.io/?v=46121&t=345711)

---

lesson seven

自定义按钮+sql增强+js增强简单运用示例：

1.link+js ：打印当前操作行的数据

2.link+action：执行sql增强 ：比如将操作行的年龄字段值加1

3.button+js:打印选中行的记录

4.button+action:执行sql增强 ：比如将选中行的年龄字段值加1

---

lesson eight

自定义按钮+js增强操作form显示/隐藏表单控件

详见:【Online之JS增强+自定义按钮 一】(http://jeecg-boot.mydoc.io/?v=46121&t=345716)

---

lesson nine

自定义按钮+js增强操作form实现一键增加子表的多条数据/清除子表的多条数据

详见:【Online之JS增强+自定义按钮 二】(http://jeecg-boot.mydoc.io/?v=46121&t=345716)

---

lesson ten

js增强操作form 增加控件change事件关联修改其他控件值(单/主表)

详见:Online之JS增强-关联修改控件值【单(主)表】(http://jeecg-

boot.mydoc.io/?v=46121&t=345714)

---

lesson eleven

js增强操作form 增加控件change事件关联修改其他控件值(从表)

详见:Online之JS增强-关联修改控件值【从表】(http://jeecg-boot.mydoc.io/?v=46121&t=345715)

---

lesson twelve

online列表/表单列的查看权限

规则详见： Online表单权限配置按钮/列的显示或隐藏(http://jeecg-boot.mydoc.io/?v=46121&t=345719)

---

lesson thirteen

online按钮权限

规则详见： Online表单权限配置按钮/列的显示或隐藏(http://jeecg-boot.mydoc.io/?v=46121&t=345719)

---

lesson fourteen

online数据权限

规则详见： Online表单数据权限(http://jeecg-boot.mydoc.io/?v=46121&t=345718)

---

lesson fifteen

online代码生成器生成代码、菜单配置、访问测试

工作流开发

Online表单对接流程

（1）设计流程：【流程管理-流程设计-新建流程】创建流程

图片地址：https://static.oschina.net/uploads/img/201904/29112127_MERJ.png

（2）Online表单增加字段bpm_status,string类型，设置默认值1，字典code设置`bpm_status`

图片地址：https://static.oschina.net/uploads/img/201905/13105615_67iA.png

图片地址：https://static.oschina.net/uploads/img/201905/13105710_uTZH.png

（3）表单挂接：流程设计完成后，选择流程【流程配置-业务管理-新增】增加表单关联

表单类型：选择Online表单，
表名：填写对接的Online表单表名
唯一编码：自动生成，不能修改
流程状态列表：默认值`bpm_status`,不能修改
标题表达式：通过表达式`${}`获取流程变量的值，组成一个标题

图片地址：https://static.oschina.net/uploads/img/201905/13105841_QLTG.png

（4）表单流程发起：

流入申请单，点击提交流程按钮发起流程，在【个人办公-我的任务中进行流程审批办理】

自定义开发表单与流程对接

（1）设计流程：【流程管理-流程设计-新建流程】创建流程

图片地址：https://static.oschina.net/uploads/img/201904/29112127_MERJ.png

（2）表单挂接：流程设计完成后，选择流程【流程配置-业务关联-新增】增加表单关联

图片地址：https://static.oschina.net/uploads/img/201905/13112345_Ns8g.png

说明：

- 表单类型：选择自定义开发

• 表名：填写需要的对接的表单对应业务的表名，存在多种表时填写主表名

• 唯一编码：流程业务对接唯一编码，流程发起时需要用到该值，该值定义整个系统中不能重复,会根据表名生成一个默认编码，可自己修改， 例如：dev*ext*biz*leave*001

• 流程状态列名：记录流程状态的字段，在关联的Online表名指定的表中存在的字段，该字段用于接收，流程处理过程中回填过来的处理状态

• 标题表达式：流程提交后在，任务列表中展示的业务标题，例如：测试新流程【${name}】，其中表达式${name}，取流程变量中的值

（3）表单流程发起代码实现：

申请单，进行流程发起

• 1、申请单列表增加按钮 <a @click="startProcess(record)">提交流程 控制待提交状态下展示 v-if="record.bpmStatus === '1'"

• 2、提交流程，功能实现

```
1、data()中定义

flowCode:"dev_ext_biz_leave_001",
url: {
  startProcess: "/process/extActProcess/startMutilProcess",
},

2、方法实现
startProcess: function(record){
    var that = this;
    this.$confirm({
      title:"提示",
      content:"确认提交流程吗?",
      onOk: function(){
        var param = {
          flowCode:that.flowCode,
          id:record.id,
          formUrl:"modules/extbpm/biz/modules/ExtBizLeaveForm",
          formUrlMobile:"modules/extbpm/biz/modules/ExtBizLeaveForm"
        }
        postAction(that.url.startProcess,param).then((res)=>{
          if(res.success){
            that.$message.success(res.message);
            that.loadData();
            that.onClearSelected();
          }else{
            that.$message.warning(res.message);
```

```
        }
      });
    }
  });
},
```

说明：提交流程是需要传递四个参数

- flowCode：流程业务对接唯一编码
- id:表单数据id
- formUrl：pc端流程审批，默认展示的表单组件
- formUrlMobile：移动端流程审批，默认展示的表单组件

（4）流程对接完成，发布流程，然后申请单提交流程即可。

流程节点个性化表单设置

一、表单设置说明

流程节点个性化表单设置，解决不同节点审批人查看或者处理不同表单的需求，具体设置【流程配置--流程节点】，选择需要个性化设置的节点编辑，配置“PC表单地址”

图片地址：https://static.oschina.net/uploads/img/201904/29154859_O8oe.png

- 配置规则：
- （1）PC表单地址可配置组件或者第三表单地址（http/https开头）
- （2）配置组件地址：可对接自己开发的表单页面组件，也可以对接Online表单页面组件

自己开发的表单页面组件：开发方法见【流程节点对接表单页面开发方法】
选择views目录下开发页面
例如页面src/views/modules/extbpm/biz/modules/ExtBizLeaveForm.vue 则组件名为 modules/extbpm/biz/modules/ExtBizLeaveForm
Online表单页面组件：
不带操作按钮的页面：modules/bpm/task/form/OnlineFormDetail
带操作按钮的页面：modules/bpm/task/form/OnlineFormOpt

- （3）配置第三表单地址（http/https开头），例如：http://www.test.com/formtest?dataId=${BPM*DATA*ID},其中${BPM*DATA*ID}去流程变量中的值。

流程变量固定值说明：
BPM_DATA_ID:表单数据id
BPM_FORM_KEY：表名

applyUserId：流程发起人
JG_LOCAL_PROCESS_ID：当前流程实例id
其他属性对应表单字段数据：例如表字段id，name，流程变量中也会存在

流程节点对接表单页面开发方法

流程节点对接表单页面开发方法

（1）在普通的列表表单页面改造，去掉<a-modal> 模态框

（2）页面引入

```
import { httpAction,getAction } from '@/api/manage'
```

（3）定义props

```
props: ['formData'],
```

（4）定义data

```
url: {
        queryById: "",//填写表单的queryById请求地址
      },
```

(5)created方法实现

```
created () {
    console.log("form start");
    console.log("formdata",this.formData);
    var params = {id:this.formData.dataId};//查询条件
    getAction(this.url.queryById,params).then((res)=>{
      if(res.success){
        console.log("获取流程节点信息",res);
        this.edit (res.result);
      }
    })
  },
```

（6）'formData'对象说明：

节点对接页面组件时通过定义的props: ['formData'],传入数据，该对象为流程节点和页面的数据传输对象，可根据其中的数据展示表单

```
formData对象属性如下：
```

dataId:表单数据id
taskId:当前环节任务id
taskDefKey:当前环节任务key
procInsId:流程实例id
tableName:表名
permissionList:节点权限配置
vars:流程变量
页面接收该对象数据后可根据数据信息通过接口获取相关的业务数据展示表单。

业务表单视角流程对接

1、功能说明

功能说明：

1、以业务表单为视角的流程发起、办理功能，业务单据对接流程后，申请人和审批人在相同的页面进行业务的申请和审批办理，业务表单页面会展示不同业务列下的数据和操作。

待我审批：查询指派给我的任务，并进行办理审批操作

图片地址：https://static.oschina.net/uploads/img/201905/23164616_wk6U.png

图片地址：https://static.oschina.net/uploads/img/201905/23164808_AGhL.png

我发起的申请：查询我创建的申请单，可进行“提交流程”、“查看审批进度”操作

图片地址：https://static.oschina.net/uploads/img/201905/23165038_AFZP.png

2、流程对接前置条件

流程对接前置条件

表单对应的数据库表必须字段存在三个字段

id:表主键id，string类型，UUID生成策略
bpm_status:流程流转状态，不能为空，默认值为1，状态码值见【数据字典-bpm_status】
create_by:创建人，记录单据创建者，用于查询我发起的单据

3、表单与流程业务关联配置

表单与流程业务关联配置

设计好流程后，在【流程配置-业务关联】配置业务关联

图片地址：https://static.oschina.net/uploads/img/201905/13112345_Ns8g.png

```
说明：
表单类型：选择自定义开发
表名：填写需要的对接的表单对应业务的表名，存在多种表时填写主表名
唯一编码：流程业务对接唯一编码，流程发起时需要用到该值，该值定义整个系统中不能重复,会根据表名生成一个默认编码，可自己修改， 例如：`dev_ext_biz_leave_001`
流程状态列名：记录流程状态的字段，在关联的Online表名指定的表中存在的字段，该字段用于接收，流程处理过程中回填过来的处理状态
标题表达式：流程提交后在，任务列表中展示的业务标题，例如：测试新流程【${name}】，其中表达式${name}，取流程变量中的值
```

4、单据列表页面扩展流程业务功能

单据列表页面改造，扩展流程业务相关功能

（1）修改代码混入引用

```
import { JeecgListMixin } from '@/mixins/JeecgListMixin'

mixins:[JeecgListMixin],
```

修改为：

```
import {JeecgBpmListMixin} from '@/mixins/JeecgBpmListMixin'

mixins: [JeecgBpmListMixin],
```

（2）列表增加流程状态

```
引入：
import {initDictOptions, filterDictText} from '@/components/dict/JDictSelectUtil'

data ()中定义变量：
bpmStatusDictOptions:[],

methods个增加方法：
 initDictConfig() {
```

```
      //初始化字典
      initDictOptions('bpm_status').then((res) => {
        if (res.success) {
          this.bpmStatusDictOptions = res.result;
        }
      });
    },
```

数据列表增加列：

```
{
          title: '流程状态',
          align:"center",
          dataIndex: 'bpmStatus',
          customRender: (text, record, index) => {
            //字典值替换通用方法
            return filterDictText(this.bpmStatusDictOptions, text);
          }
        },
```

（3）列表操作功能扩展

按钮扩展

```
<span slot="action" slot-scope="text, record">
        <template v-if="record.bpmStatus === '1'">
          <a @click="handleEdit(record)">编辑</a>
          <a-divider type="vertical"/>
          <a @click="startProcess(record)">提交流程</a>
          <a-divider type="vertical"/>
        </template>
        <template v-else-
if="showBtn(record.bpmStatus)&&queryParam.bizTaskType=='1'">
          <a @click="handleProcess(record)">办理</a>
          <a-divider type="vertical"/>
        </template>
        <a-dropdown>
          <a class="ant-dropdown-link">更多 <a-icon type="down" /></a>
          <a-menu slot="overlay">
            <a-menu-item v-if="queryParam.bizTaskType=='2'">
              <a href="javascript:;" @click="handleDetail(record)">详情</a>
            </a-menu-item>
            <a-menu-item v-if="record.bpmStatus === '1'">
              <a-popconfirm title="确定删除吗?" @confirm="() =>
```

```
handleDelete(record.id)">
              <a>删除</a>
            </a-popconfirm>
          </a-menu-item>
          <!--<a-menu-item v-else @click="handlePreviewPic(record)">审批进度</a-
menu-item>-->
          <a-menu-item v-else @click="handleTrack(record)">审批进度</a-menu-
item>
        </a-menu>
      </a-dropdown>
    </span>
```

组件扩展

```
<bpm-process-track-modal ref="bpmProcessTrackModal"></bpm-process-track-
modal>
<bpm-biz-task-deal-modal ref="taskDealModal" :path="path" :formData="formData"
@ok="handleOk"></bpm-biz-task-deal-modal>

引入组件
import BpmBizTaskDealModal from
"@/views/modules/bpm/common/BpmBizTaskDealModal.vue";
import BpmProcessTrackModal from "../../bpm/common/BpmProcessTrackModal.vue";

components: {
    BpmProcessTrackModal,
    BpmBizTaskDealModal
  },
```

data ()流程参数定义

```
flowCode:"TEST001",//唯一编码
formUrl:"modules/extbpm/biz/modules/ExtBizLeaveForm",
formUrlMobile:"modules/extbpm/biz/modules/ExtBizLeaveForm",
```

formUrl表单组件页面开发参见：流程节点对接表单页面开发方法(http://jeecg-boot.mydoc.io/?t=345727)

增加过滤字段

```
<a-form-item label="类型">
          <a-radio-group v-model="queryParam.bizTaskType"
@change="onBizTaskTypeChange">
            <a-radio value="1">待我审批</a-radio>
            <a-radio value="2">我发起的申请</a-radio>
```

```
        </a-radio-group>
      </a-form-item>
```

5、后台代码扩展

后台代码扩展

列表list请求方法中增加如下代码即可：

```
//-------------------------我的审批任务过滤---开始-----------------------------
String userid = JwtUtil.getUserNameByToken(req); // 获取当前登录用户
String bizTaskType = req.getParameter("bizTaskType");//业务类型1待我审批2我发起的申请
JoaUtil.filterRunningTask(bizTaskType,queryWrapper,userid, "TEST001");
//-------------------------我的审批任务过滤---结束-----------------------------
```

```
JoaUtil.filterRunningTask(bizTaskType,queryWrapper,userid, "TEST001");
说明：
方法：filterRunningTask(String bizTaskType,QueryWrapper queryWrapper,String userName,String flowCode)
bizTaskType ：业务类型1待我审批2我发起的申请
queryWrapper：查询器
userName ：用户账号
flowCode 流程唯一编码
```

代码截图：

图片地址：https://static.oschina.net/uploads/img/201905/23174019_eOEE.png

表单设计器

基础操作手册

表单设计器基础

介绍

表单设计器是一款可以“所见即的”式的设计表单的工具。操作很简单，只需要将你需要的**组件拖动到页面中**即可。

我们支持的组件有很多，具体如下所示

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184308az1d2d93k2u4dd9u.png

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184311hmhm2iqioipispwy.png

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184313ss4p5l5oas6dl0zd.png

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184314l7nk7b42os85cosz.png

基本使用

如下图所示，只需要将你需要的**组件拖动到页面中**即可。

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184315l08u8p4rz0mozeo0.gif

配置字段属性

以单行文本为例，配置它的字段属性

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184321q0wnnsw2nkzo019k.png

标题

用于配置标签的标题，例如我修改成“姓名”就会显示成下面这种效果

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184322niih5nzi56ifmggi.png

宽度

宽度，顾名思义，就是设置组件的宽度，但是就算两个组件的宽度均不大于50%，也无法做到显示在同一行，想要显示在同一行，只能通过布局组件来实现，例如栅格布局组件。

占位内容

即当用户没有输入任何内容时的提示内容，如下图所示

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184323qeiezpf7paqqifvw.png

默认值

即默认显示的内容

CSS类名

用于高级的CSS增强，在高级操作手册里有详细讲解

数据绑定Key

即存储到后台的数据key，如果在表单属性里设置了Online表单，就可选择online表单里的字段，否则就只能手动输入。

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184325et0uikgffxsjkthg.png

操作属性

可设置组件的一些属性，例如只读、禁用等

校验

可设置一些校验属性，例如常用的有必填，如果设置了必填，则效果如下

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184327cl5k15e0pazlueu0.png

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/184329fd716rzxex6fi66r.png

配置表单属性

表单属性用于对当前表单做出一些全局的修改

图片地址：http://www.jeecg.org/data/attachment/forum/201908/06/185030bsbwnswtqxqsbqj2.png

具体配置如下：

Online表单

可以对接配置的Online表单，只需要下拉选择你要对接的Online表单即可，选择之后会将数据同步到Online表单数据库里

标签对齐方式

可选择文本标签的对齐方式，默认为顶部对齐

图片地址

：http://www.jeecg.org/data/attachment/forum/201908/06/184317qtjgzozj7g77p792.gif

表单字段宽度

可以控制表单标签所占的宽度，默认100，最大200，最小0。标签对齐方式为顶部对齐时无效

图片地址
：http://www.jeecg.org/data/attachment/forum/201908/06/184318jk5n22nan6sjs56w.gif

组件尺寸

可选择组件的尺寸大小，可选值有：medium、small、mini

图片地址
：http://www.jeecg.org/data/attachment/forum/201908/06/184319s4hezl1y3t1a1lkh.gif

弹窗宽度

可更改弹窗的宽度，可以在**预览**界面查看效果

弹窗顶部距离

可更改弹窗的顶部距离，可以在**预览**界面查看效果

表单内边距

可更改表单的内边距，可以在**预览**界面查看效果

例如进行如下配置

图片地址
：http://www.jeecg.org/data/attachment/forum/201908/06/184331jumt0sudbckziiib.png

运行效果如下

图片地址
：http://www.jeecg.org/data/attachment/forum/201908/06/184332sat6kzgnd3k7byyd.png

JS增强

用于写更高级的全局JS增强，在高级操作手册里有详细讲解

CSS增强

用于写更高级的全局CSS增强，在高级操作手册里有详细讲解

高级操作手册

表单设计器进阶

本进阶文档适用于有前端编程经验的人

一、全局CSS增强

可以在**表单属性**里最下方找到**CSS增强**的输入框，在这里可以写css样式，在预览的时候会自动应用你写的样式。

配合几乎每个组件都有**CSS类名**属性，就能做到单独对某一个或某些组件写样式修改了。

CSS增强例子

例如我向页面中拖入了一个**单行文本**组件，并且想要把这个输入框的文字颜色改为**红色**，需要有以下步骤：

1. 修改组件的CSS类名属性，增加一个类名，例如叫input-red

```
![image](http://www.jeecg.org/data/attachment/forum/201908/07/120529rs08sk7ttu2wr78i.png)
```

1. 在表单属性里的CSS增强中修改样式，如下所示：

```
![image](http://www.jeecg.org/data/attachment/forum/201908/07/120532tspzoznqqet6qqz6.png)
```

1. 接着点预览查看效果

```
![image](http://www.jeecg.org/data/attachment/forum/201908/07/120533fxajo9i8o5xf2xcm.png)
```

我们可以发现输入框的颜色并没有变成红色，这是因为element-ui将真正的input组件放到了更深层的dom里，我们可以用chrome的devTools工具查看一下

1. 找到真实的input类名，我们可以发现真实的input类名叫做el-inputinner

```
![image](http://www.jeecg.org/data/attachment/forum/201908/07/120535huauuttisyrxrhca.png)
```

1. 再次修改CSS增强代码，如下图所示

```
![image](http://www.jeecg.org/data/attachment/forum/201908/07/120537gmamava8218an7nx.png)
```

1. 再点击预览查看一下效果，发现输入框的颜色已经变成红色了，并且还不会影响到其他组件。

```
![image](http://www.jeecg.org/data/attachment/forum/201908/07/120538godnnnl1zgoo1mwn.png)
```

二、全局JS增强

可以在**表单属性**里最下方找到**JS增强**的输入框，在这里可以写js代码，在预览的时候会自动执行你写的JS代码。

在JS增强输入框里除了可以用挂载到window中的全局变量之外，还可以用我们给你封装好的几个变量以及方法，他们分别是：

- vm Vue实例，可以调用Vue的一系列的方法，例如 vm.$nextTick();
- event Event对象，可以调用 event.type 来判断当前是什么增强类型（全局还是按钮）
- api 封装了一下api，具体如下
- getFormData(key) 获取form表单的值
- setFormData(key, value) 设置form表单的值
- setFormOptions(key, optionsKey, optionsValue) 设置 组件 的options
- watch(watchItems) 设置监听 models 值的变化
- 示例：

```
// 与vue的watch用法相同，可参见vue的官方文档
api.watch([
    name(val, oldVal){
        // name 发生了变化
    },
    info: {
        deep: true,
        handler(val, oldVal){
            // info 发生了变化
        }
    }
]);
```

```
- `get(url, parameter)` 发送Get请求
- `post(url, parameter)` 发送Post请求
- `put(url, parameter)` 发送Put请求
- `request(url, parameter, method)` 发送请求
```

三、按钮点击事件JS增强

除了全局JS增强，还支持按钮的JS增强，就是会在点击按钮的时候执行的JS代码

图片地址
：http://www.jeecg.org/data/attachment/forum/201908/07/120540lkg2iik2e529re10.png

上图中，红框里的代码是按钮的默认点击事件，可以在预览界面点击界面看到输出。

与全局JS增强一样，按钮JS增强也封装好了几个变量以及方法，并且与全局JS增强完全一致。

示例：点击按钮获取输入框的值

如果想做到点击按钮的时候获取输入框的值，我们需要下面这几步：

1. 在页面中拖入一个输入框，并且将它的数据绑定Key改为name

```
![image](http://www.jeecg.org/data/attachment/forum/201908/07/120541l77rsn7t211h5gwg.png)
```

1. 在页面中拖入一个按钮，并且将它的点击事件改为下面这样：

```
![image](http://www.jeecg.org/data/attachment/forum/201908/07/120543kxuz7aa6si6p6fa7.png)

```js
// 可以用 api 里的 getFormData 方法获取
var name = api.getFormData("name");
alert("你填写的姓名是：" + name);
```
```

1. 点击预览，查看效果

```
![image](http://www.jeecg.org/data/attachment/forum/201908/07/120545ahmwwmm1jj6j3zxq.png)
```

三、自定义接收URL

目的是为了让用户自定义接收数据的后台接口地址，你可以在你自己定义的接口里做数据处理，并保存到数据库、

配置方式：

在“表单属性”最下方有一个“自定义接收URL”输入框，你可以输入你的后台API地址

上下文变量规则

上下文变量规则

介绍

在表单设计器中，我们允许使用 {{ 变量名}} 形式的“上下文变量”来动态的展示你的表单。

用法

只需要在特定的属性中输入 {{ 变量名 }} 就能使用“上下文变量”，允许拼接多个变量，允许拼接其他字符

支持的范围

目前支持上下文变量的组件有“文本组件”的“文本内容”属性和所有组件的“默认值”属性

支持的变量

变量名	解释	值示例
sysUserCode	当前登录用户登录账号	zhangsan
sysUserName	当前登录用户真实名称	张三
sysOrgCode	当前登录用户部门编号	A001
sysDate	当前系统日期	2019-7-26
sysTime	当前系统时间	10:43:42

使用示例

假如说我们要在界面中显示系统时间和用户的信息，可以在页面上配置如下信息

图片地址：http://www.jeecg.org/data/attachment/forum/201907/26/110443ygmqcfbujqg5mvdb.png

图片地址：http://www.jeecg.org/data/attachment/forum/201907/26/110452xlz33c48arcd3ha4.png

图片地址：http://www.jeecg.org/data/attachment/forum/201907/26/110501fltzqpwl59q4ljwl.png

那么最终的运行结果如下

图片地址：http://www.jeecg.org/data/attachment/forum/201907/26/110433p5mbxgy5kjckyxx5.png

数据绑定映射

数据绑定映射使用文档

介绍

数据绑定映射是允许将含有**多个返回值**的组件映射到数据对象中的一种方式。

例如用户组件，默认情况下选择了一个用户以后，我们是将id存储到了数据库，但是我同时也想将username存储到数据库里去，就需要用到数据绑定映射了

用法

目前仅用户组件和部门组件支持使用数据绑定映射

首先我们需要找到数据绑定映射属性，并点击“新增映射”

图片地址
：http://www.jeecg.org/data/attachment/forum/201907/26/114508rrsvszq70scj23r2.png

其中有两个输入框，分别是From和To，From的意思是从组件中取出某个特定的值，映射到To中填写的字段中。

各个组件支持填写的From值

用户组件

变量名	解释	值示例
id	用户ID	e9ca23d68d884d4ebb19d07889727dae
username	用户名	zhangsan
realname	真实姓名	张三
avatar	头像地址	xxx.png
birthday	生日	1990-7-11
sex	性别(1=男 2=女)	1
sex_dictText	性别字典文本	男
email	邮箱地址	zhangsan@xx.com

phone	电话号码	150xxxxxxxx
orgCode	机构编码	A001
status	状态	1
status_dictText	状态字典文本	正常
createTime	用户创建时间	2018-12-21 17:54:10

部门组件

变量名	解释	值示例
id	机构/部门ID	c6d7cb4deeac411cb3384b1b31278596
departName	机构/部门名称	北京国炬信息技术有限公司
departNameAbbr	缩写	
departNameEn	英文名	
departOrder	排序序号	0
description	描述	
fax	传真	
memo	备注	
mobile	手机号	
address	地址	
orgCode	机构编码	A01
orgType	机构类型 1一级部门 2子部门	1
parentId	父机构ID	
createTime	创建时间	2019-02-11 14:21:51

使用示例

待续

上传组件配置

文件上传配置

本文档会带着你配置**上传组件**，使其上传到你自己的服务器上（仅限七牛云）

如果已有七牛云账号，可以跳过第1步

1. 注册七牛云账号，并确保通过了实名认证

2. 开通七牛云的 **对象存储** 服务，配置信息如下图

> ![image](http://www.jeecg.org/data/attachment/forum/201908/13/184854kthhhtngyhyqramy.png)

1. 创建成功后，鼠标移动到右上角的头像，并点击 **密钥管理** 菜单

> ![image](http://www.jeecg.org/data/attachment/forum/201908/13/184851ma1i0rabtiizii55.png)

1. 在打开的页面里创建一对密钥(Access/Secret Key)，*若已创建可忽略此步骤*

2. 保证你的密钥状态是“**使用中**”

> ![image](http://www.jeecg.org/data/attachment/forum/201908/13/184853vkkds6ck2c29cuu6.png)

1. 将AK和SK配置到后台 **QiniuConfig.java** 文件中

> ![image](http://www.jeecg.org/data/attachment/forum/201908/13/184856fya21r1u23ru2dc9.png)

1. 空间域名可以在**内容管理**页面进行查看

> ![image](http://www.jeecg.org/data/attachment/forum/201908/13/184857tz639hyi9oh43qq3.png)
> **注意：** 如果你没有绑定自定义域名的话，那这个测试域名很快就会被收回，你就用不了了，所以请尽快绑定自定义域名，关于如何绑定，请查看七牛云的官方文档（<https://developer.qiniu.com/kodo/kb/5158/how-to-transition-from-test-domain-name-to-a-custom-domain-name>）

1. 至此，重新启动后台项目即可完成配置。

配置到菜单

配置到菜单

如果要将表单配置到菜单，需要先获取表单的地址，我们可以在表单列表右侧点击**配置地址**，如下图所示：

图片地址：http://www.jeecg.org/data/attachment/forum/201908/26/192125oxbd9hwplz8werrw.png

在弹出的弹窗里，复制**数据列表地址**里的内容，如下图所示：

图片地址：http://www.jeecg.org/data/attachment/forum/201908/26/192129mqi9xpsp59zi22ih.png

然后打开**菜单管理**，并点击新增按钮

图片地址：http://www.jeecg.org/data/attachment/forum/201908/26/192132hl4bkn2h2sezhlvd.png

在弹出的弹窗里填写你所需要的内容，菜单类型

- **菜单类型：** 可以选择“一级菜单”，也可以选择“子菜单”，但“一级菜单”的排序级别不建议低于首页。
- **菜单名称：** 菜单显示名称
- **菜单路径** 填写刚刚复制的路径
- **前端组件** 固定为modules/online/desform/auto/AutoDesignDataForm
- **排序** 排序数字越高，显示顺序越靠下
- **是否路由菜单** 默认为“是”，这里一定要点成“否”

图片地址：http://www.jeecg.org/data/attachment/forum/201908/26/192135k6pnngpplnyrpip6.png

全部填写完成并保存成功之后，需要设置一下权限，点开**角色管理**页面，并找到你当前登录的用户，点击右侧的**授权**，如下图所示：

图片地址：http://www.jeecg.org/data/attachment/forum/201908/26/192137t9t69l79tk9aulil.png

在弹出的弹窗里找到你刚刚添加的菜单，并且打上勾，点击提交

图片地址：http://www.jeecg.org/data/attachment/forum/201908/26/192142wnv1blq1wiimm3mi.png

保存成功之后就可以刷新一下页面，菜单就显示出来了，效果如下图所示：

图片地址：http://www.jeecg.org/data/attachment/forum/201908/26/192145tx75vyrn44u5574h.png

代码实现新增自定义组件

用代码方式新增一个自定义组件的方法

第一步：显示在拖动候选栏里

首先打开 src/components/componentsConfig.js 文件，所有组件的基础配置都是在这个文件里写的。

图片地址：http://www.jeecg.org/data/attachment/forum/201909/02/183445vdwdwgcmc1g0hgim.png

在这个文件里新增一段固定格式的JSON，包含新组件的信息

图片地址：http://www.jeecg.org/data/attachment/forum/201909/02/183418e85sopyts5e5stnx.png

options 里的属性根据组件的要求按需整改。

保存后打开页面就可以发现已经添加到 **基础字段** 里了

图片地址：http://www.jeecg.org/data/attachment/forum/201909/02/183423snbyzj18m2ibsi0s.png

虽然可以拖动到设计器中，但是不会有任何显示，因为我们没有定义组件的实现

图片地址：http://www.jeecg.org/data/attachment/forum/201909/02/183424euh0v7374g4gbguq.png

第二步：编写组件实现

首先新建一个vue文件：src/components/jeecg/JeecgInput.vue，暂时先写上下图的这些代码

图片地址：http://www.jeecg.org/data/attachment/forum/201909/02/183439lyqji0iyrspqm5ia.png

然后在 src/components/WidgetFormItem.vue 文件里引用一下

图片地址：http://www.jeecg.org/data/attachment/forum/201909/02/183420i1a811s8c1s88wbg.png

并且在页面里同步引用，加个 v-if 判断，只有当当前组件是 jeecg-input 的时候才显示

图片地址：http://www.jeecg.org/data/attachment/forum/201909/02/183421vkt9lnl5waq4lnto.png

再回到页面上，就可以看到能正常显示出来了

图片地址：http://www.jeecg.org/data/attachment/forum/201909/02/183425ugykn1xvxzqvok8m.png

但是点击预览仍然是显示不出来组件的，因为设计和预览用的是两个不同的组件

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183427in491h498f632bfa.png

我们还需要打开 src/components/GenerateFormItem.vue，用通用的方式再引用一下刚刚新建的组件

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183428wugg3i3uh4h4lziu.png

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183429h42c5kfhb55c82oc.png

第三步：用户自定义组件属性

组件的属性可以在 src/components/WidgetConfig.vue 文件里整改，打开这个文件，新增以下代码

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183431wmfzffikg9f7ll77.png

回到页面里就可以看到效果

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183432d4574x4c8888zxkk.png

现在我们给 defaultValue 和 placeholder 这两个属性开放给用户修改

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183434oz3vjbsmtz2td36m.png

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183436h1zg13zu3kv0t11l.png

效果如下：

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183437hygcc4biymccxdao.png

还需要组件内部配合修改下，修改成下图这样

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183440sfnuf2owfn3n1uvf.png

在 src/components/WidgetFormItem.vue 和 src/components/GenerateFormItem.vue 组件里也要修改一下传值

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183441f7l2lq64heeva077.png

图片地址
：http://www.jeecg.org/data/attachment/forum/201909/02/183444azy4b7y7gsye5yyy.png

点击预览就可以看到效果了

图片地址：http://www.jeecg.org/data/attachment/forum/201909/02/183443siobhvmh6mh6zyik.png