若依开发文档

书栈(BookStack.CN)

目　录

致谢

当前文档 《若依开发文档》 由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2019-08-27。

简介

RuoYi是一款基于SpringBoot+Bootstrap的极速后台开发框架。

- RuoYi 官网地址: http://ruoyi.vip
- RuoYi 在线文档: http://doc.ruoyi.vip
- RuoYi 源码下载: https://gitee.com/y_project/RuoYi
- RuoYi 在线提问: https://gitee.com/y_project/RuoYi/issues
- RuoYi 博客: https://www.oschina.net/p/ruoyi
- QQ 群号: 1389287、1679294、1529866、1772718、1366522、1382251、1145125

RuoYi 是一个 Java EE 企业级快速开发平台，基于经典技术组合（Spring Boot、Apache Shiro、MyBatis、Thymeleaf、Bootstrap、Hplus），内置模块如：部门管理、角色用户、菜单及按钮授权、数据权限、系统参数、日志管理、通知公告等。在线定时任务配置；支持集群，支持多数据源。

主要特性

- 完全响应式布局（支持电脑、平板、手机等所有主流设备）
- 强大的一键生成功能（包括控制器、模型、视图、菜单等）
- 支持多数据源，简单配置即可实现切换。
- 支持按钮及数据权限，可自定义部门数据权限。
- 对常用js插件进行二次封装，使js代码变得简洁，更加易维护
- 完善的XSS防范及脚本过滤，彻底杜绝XSS攻击
- Maven多项目依赖，模块及插件分项目，尽量松耦合，方便模块升级、增减模块。
- 国际化支持，服务端及客户端支持
- 完善的日志记录体系简单注解即可实现

技术选型

1、系统环境

- Java EE 8
- Servlet 3.0

- Apache Maven 3 **2**、主框架

- Spring Boot 2.0

- Spring Framework 5.0

- Apache Shiro 1.4 **3**、持久层

- Apache MyBatis 3.4

- Hibernate Validation 6.0

- Alibaba Druid 1.1 **4**、视图层
- Bootstrap 3.3
- Hplus 4.1
- Thymeleaf 3.0

内置功能

- 用户管理：用户是系统操作者，该功能主要完成系统用户配置。
- 部门管理：配置系统组织机构（公司、部门、小组），树结构展现支持数据权限。
- 岗位管理：配置系统用户所属担任职务。
- 菜单管理：配置系统菜单，操作权限，按钮权限标识等。
- 角色管理：角色菜单权限分配、设置角色按机构进行数据范围权限划分。
- 字典管理：对系统中经常使用的一些较为固定的数据进行维护。
- 参数管理：对系统动态配置常用参数。
- 通知公告：系统通知公告信息发布维护。
- 操作日志：系统正常操作日志记录和查询；系统异常信息日志记录和查询。
- 登录日志：系统登录日志记录查询包含登录异常。
- 在线用户：当前系统中活跃用户状态监控。
- 定时任务：在线（添加、修改、删除)任务调度包含执行结果日志。
- 代码生成：前后端代码的生成（java、html、xml、sql)支持CRUD下载 。
- 系统接口：根据业务代码自动生成相关的api接口文档。
- 服务监控：监视当前系统CPU、内存、磁盘、堆栈等相关信息。
- 在线构建器：拖动表单元素生成相应的HTML代码。
- 连接池监视：监视当期系统数据库连接池状态，可进行分析SQL找出系统性能瓶颈。

更新日志

- v4.0.0 2019-08-08
    - 代码生成支持预览、编辑，保存方案
    - 新增防止表单重复提交注解
    - 新增后端校验（和前端保持一致）
    - 新增同一个用户最大会话数控制
    - Excel导出子对象支持多个字段
    - 定时任务支持静态调用和多参数
    - 定时任务增加分组条件查询
    - 字典类型增加任务分组数据
    - 新增表格是否首次加载数据
    - 新增parentTab选项卡可在同一页签打开
    - 多数据源支持类注解（允许继承父类的注解）
    - 部门及以下数据权限（调整为以下及所有子节点）
    - 新增角色数据权限配（仅本人数据权限）
    - 修改菜单权限显示问题
    - 上传文件修改路径及返回名称

    - 添加报表插件及示例
    - 添加首页统计模板
    - 添加表格拖拽示例
    - 添加卡片列表示例
    - 添加富文本编辑器示例
    - 添加表格动态增删改查示例
    - 添加用户页面岗位选择框提示
    - 点击菜单操作添加背景高亮显示
    - 表格树新增showSearch是否显示检索信息
    - 解决表格列设置sortName无效问题
    - 表格图片预览支持自定义设置宽高
    - 添加表格列浮动提示（单击文本复制）
    - PC端收起菜单后支持浮动显示
    - 详细操作样式调整
    - 修改用户更新描述空串不更新问题
    - 导入修改为模板渲染
    - 修改菜单及部门排序规则
    - 角色导出数据范围表达式翻译
    - 添加summernote富文本字体大小
    - 优化表格底部下边框防重叠&汇总像素问题
    - 树表格支持属性多层级访问
    - 修复IE浏览器用户管理界面右侧留白问题
    - 重置按钮刷新表格
    - 重置密码更新用户缓存
    - 优化验证码属性参数
    - 支持数据监控配置用户名和密码
    - 文件上传修改按钮背景及加载动画
    - 支持配置一级菜单href跳转
    - 侧边栏添加一套浅色主题
    - 树表格添加回调函数（校验异常状态）
    - 用户个人中心适配手机端显示
    - Excel支持设置导出类型&更换样式
    - 检查属性改变修改为克隆方式（防止热部署强转异常）
    - 其他细节优化

- v3.4.0 2019-06-03

    - 新增实例演示菜单及demo
    - 新增页签右键操作
    - 菜单管理新增打开方式
    - 新增点击某行触发的事件
    - 新增双击某行触发的事件
    - 新增单击某格触发的事件
    - 新增双击某格触发的事件
    - 新增是否启用显示细节视图
    - 支持上传任意格式文件
    - 修复角色权限注解失效问题

    - 左侧的菜单栏宽度调整
    - 新增响应完成后自定义回调函数
    - 支持前端及其他模块直接获取用户信息
    - 升级swagger到最新版2.9.2
    - 升级jquery.slimscroll到最新版1.3.8
    - 升级select2到最新版4.0.7
    - 新增角色配置本部门数据权限
    - 新增角色配置本部门及以下数据权限
    - 优化底部操作防止跳到页面顶端
    - 修改冻结列选框无效及样式问题
    - 修复部门四层级修改祖级无效问题
    - 更换开关切换按钮样式
    - 新增select2-bootstrap美化下拉框
    - 添加表格内图片预览方法
    - 修复权限校验失败跳转页面路径错误
    - 国际化资源文件调整
    - 通知公告布局调整
    - 删除页签操作功能
    - 表格树新增查询指定列值
    - 更改系统接口扫描方式及完善测试案例
    - 表格列浮动提示及字典回显默认去背景
    - 修复启用翻页记住前面的选择check没选中问题
    - 去除监控页面底部的广告
    - 日期控件功问题修复及data功能增强
    - 新增角色权限可见性（前端直接调用）
    - 新增获取当前登录用户方法（前端及子模块调用）
    - 修复热部署重启导致菜单丢失问题
    - 优化业务校验失败普通请求跳转页面
    - 操作日志新增状态条件查询
    - 操作类型支持多选条件查询
    - 通知公告防止滚动触底回弹优化
    - 其他细节优化

- v3.3.0 2019-04-01

    - 新增线程池统一管理
    - 新增支持左右冻结列
    - 新增表格字符超长浮动提示
    - 升级datepicker拓展并汉化
    - 升级druid到最新版本v1.1.14
    - 修复个人头像为图片服务器跨域问题
    - 修改上传文件按日期存储
    - 新增表格客户端分页选项
    - 新增表格的高度参数
    - 新增表格销毁方法
    - 新增表格下拉按钮切换方法
    - 新增表格分页跳转到指定页码

    - 新增表格启用点击选中行参数
    - 修复表格数据重新加载未触发部分按钮禁用
    - 使用jsonview展示操作日志参数
    - 新增方法（addTab、editTab）
    - 修改用户管理界面为Tab打开方式
    - 表单验证代码优化
    - 修复@Excel注解 prompt 属性使用报错
    - 修复combo属性Excel兼容性问题
    - 新增@Excel导入导出支持父类字段
    - 修复关闭最后选项卡无法激活滚动问题
    - 增加日期控件显示类型及回显格式扩展选项
    - 修复定时任务执行失败后入库状态为成功状态
    - 支持定时任务并发开关控制
    - 优化权限校验失败普通请求跳转页面
    - 捕获线程池执行任务抛出的异常
    - 修复IE浏览器导出功能报错
    - 新增角色管理分配用户功能
    - 新增表格翻页记住前面的选择
    - 调整用户个人中心页面
    - 修复界面存在的一些安全问题
    - 其他细节优化

- v3.2.0 2019-01-16

    - 部门修改时不允许选择最后节点
    - 修复部门菜单排序字段无效
    - 修复光驱磁盘导致服务监控异常
    - 登录界面去除check插件
    - 验证码文本字符间距修正
    - 升级SpringBoot到最新版本2.1.1
    - 升级MYSQL驱动
    - 修正登录必填项位置偏移
    - Session会话检查优化
    - Excel注解支持多级获取
    - 新增序列号生成方法
    - 修复WAR部署tomcat退出线程异常
    - 全屏操作增加默认确认/关闭
    - 修复个人信息可能导致漏洞
    - 字典数据根据下拉选择新增类型
    - 升级Summernote到最新版本v0.8.11
    - 新增用户数据导入
    - 首页主题样式更换
    - layer扩展主题更换
    - 用户管理移动端默认隐藏左侧布局
    - 详细信息弹出层显示在顶层
    - 表格支持切换状态（用户/角色/定时任务）
    - Druid数据源支持配置继承

    - 修正部分iPhone手机端表格适配问题
    - 新增防止重复提交表单方法
    - 新增表格数据统计汇总方法
    - 支持富文本上传图片文件

- v3.1.0 2018-12-03

    - 新增内网不获取IP地址
    - 新增cron表达式有效校验
    - 定时任务新增详细信息
    - 定时任务默认策略修改（不触发立即执行）
    - 定时任务显示下一个执行周期
    - 支持前端任意日期格式处理
    - 上传头像删除多余提交按钮
    - 表格增加行间隔色配置项
    - 表格增加转义HTML字符串配置项
    - 表格增加显示/隐藏指定列
    - 代码生成优化
    - 操作日志参数格式化显示
    - 页签新增新增全屏显示
    - 新增一键打包部署
    - Excel注解新增多个参数
    - 新增提交静默更新表格方法
    - 新增服务监控菜单

- v3.0.0 2018-10-08

    - 升级poi到最新版3.17
    - 导出修改临时目录绝对路径
    - 升级laydate升级到最新版5.0.9
    - 升级SpringBoot到最新版本2.0.5
    - 优化开始/结束时间校验限制
    - 重置密码参数表中获取默认值
    - 修复头像修改显示问题
    - 新增数据权限过滤注解
    - 新增表格检索折叠按钮
    - 新增清空（登录、操作、调度）日志
    - 固定按钮位置（提交/关闭）
    - 部门/菜单支持（展开/折叠）
    - 部分细节调整优化
    - 项目采用分模块

准备工作

```
1. JDK >= 1.8 (推荐1.8版本)
2. Mysql >= 5.5.0 (推荐5.7版本)
3. Maven >= 3.0
```

运行系统

1、前往Gitee下载页面(https://gitee.com/y_project/RuoYi)下载解压到工作目录2、导入到Eclipse，菜单 File -> Import，然后选择 Maven -> Existing Maven Projects，点击 Next> 按钮，选择工作目录，然后点击 Finish 按钮，即可成功导入Eclipse会自动加载Maven依赖包，初次加载会比较慢（根据自身网络情况而定）3、创建数据库ry并导入数据脚本ry_20190215.sql，quartz.sql4、打开运行 `com.ruoyi.RuoYiApplication.java` 5、打开浏览器，输入：http://localhost:80 （默认账户 `admin/admin123` ）若能正确展示登录页面，并能成功登录，菜单及页面展示正常，则表明环境搭建成功

建议使用Git克隆，因为克隆的方式可以和RuoYi随时保持更新同步。使用Git命令克隆`git clone https://gitee.com/y_project/RuoYi.git`

必要配置

- 修改数据库连接 `编辑resources目录下的application-druid.yml` `url: 服务器地址` `username: 账号` `password: 密码`
- 开发环境配置 `编辑resources目录下的application.yml` `port: 端口` `context-path: 部署路径`

部署系统

`bin/package.bat` 在项目的目录下执行然后会在项目下生成 `target` 文件夹包含 `war` 或 `jar` （多模块生成在ruoyi-admin）

1、jar部署方式 使用命令行执行： `java –jar RuoYi.jar` 或者执行脚本： `bin/run.bat`

2、war部署方式 pom.xml packaging修改为 `war` 放入tomcat服务器webapps

> SpringBoot去除内嵌tomcat

```
1. <!-- 多模块排除内置tomcat -->
2. <dependency>
3.     <groupId>org.springframework.boot</groupId>
4.     <artifactId>spring-boot-starter-web</artifactId>
5.     <exclusions>
6.         <exclusion>
7.             <groupId>org.springframework.boot</groupId>
8.             <artifactId>spring-boot-starter-tomcat</artifactId>
9.         </exclusion>
10.     </exclusions>
```

```
</dependency>

<!-- 单应用排除内置tomcat -->
<exclusions>
    <exclusion>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <groupId>org.springframework.boot</groupId>
    </exclusion>
</exclusions>
```

常见问题

- 如果使用 `Mac` 需要修改 `application.yml` 文件路径 `profile`
- 如果使用 `Linux` 提示表不存在，设置大小写敏感配置在`/etc/my.cnf` 添加 `lower_case_table_names=1`，重启MYSQL服务
- 如果提示当前权限不足，无法写入文件请检查 `profile` 是否可读可写，或者无法访问此目录如遇到问题到Issues反馈

文件结构

```
com.ruoyi
├── common            // 工具类
│       └── annotation                    // 自定义注解
│       └── config                        // 全局配置
│       └── constant                      // 通用常量
│       └── core                          // 核心控制
│       └── enums                         // 通用枚举
│       └── exception                     // 通用异常
│       └── json                          // JSON数据处理
│       └── utils                         // 通用类处理
│       └── xss                           // XSS过滤处理
├── framework         // 框架核心
│       └── aspectj                       // 注解实现
│       └── config                        // 系统配置
│       └── datasource                    // 数据权限
│       └── manager                       // 异步处理
│       └── shiro                         // 权限控制
│       └── util                          // 通用工具
│       └── web                           // 前端控制
├── ruoyi-generator   // 代码生成（可移除）
├── ruoyi-quartz      // 定时任务（可移除）
├── ruoyi-system      // 系统代码
├── ruoyi-admin       // 后台服务
├── ruoyi-xxxxxx      // 其他模块
```

配置文件

通用配置 `application.yml`

```yaml
# 项目相关配置
ruoyi:
  # 名称
  name: RuoYi
  # 版本
  version: 3.2.0
  # 版权年份
  copyrightYear: 2019
  # 文件上传
  profile: D:/profile/
  # 获取ip地址开关
  addressEnabled: true

# 开发环境配置
server:
  # 服务端口
  port: 80
```

```yaml
  servlet:
    # 项目contextPath
    context-path: /
  tomcat:
    # tomcat的URI编码
    uri-encoding: UTF-8
    # tomcat最大线程数，默认为200
    max-threads: 800
    # Tomcat启动初始化的线程数，默认值25
    min-spare-threads: 30

# 日志配置
logging:
  level:
    com.ruoyi: debug
    org.springframework: WARN
    org.spring.springboot.dao: debug

# 用户配置
user:
  password:
    # 密码错误{maxRetryCount}次锁定10分钟
    maxRetryCount: 5

# Spring配置
spring:
  # 模板引擎
  thymeleaf:
    mode: HTML
    encoding: utf-8
    # 禁用缓存
    cache: false
  # 资源信息
  messages:
    # 国际化资源文件路径
    basename: i18n/messages
  jackson:
    time-zone: GMT+8
    date-format: yyyy-MM-dd HH:mm:ss
  profiles:
    active: druid
  # 文件上传
  servlet:
     multipart:
       max-file-size:  30MB
       max-request-size:  30MB
  # 服务模块
  devtools:
    restart:
      # 热部署开关
      enabled: true

# MyBatis
```

```yaml
mybatis:
    # 搜索指定包别名
    typeAliasesPackage: com.ruoyi
    # 配置mapper的扫描，找到所有的mapper.xml映射文件
    mapperLocations: classpath*:mapper/**/*Mapper.xml
    # 加载全局的配置文件
    configLocation: classpath:mapper/mybatis-config.xml

# PageHelper分页插件
pagehelper:
  helperDialect: mysql
  reasonable: true
  supportMethodsArguments: true
  params: count=countSql

# Shiro
shiro:
  user:
    # 登录地址
    loginUrl: /login
    # 权限认证失败地址
    unauthorizedUrl: /unauth
    # 首页地址
    indexUrl: /index
    # 验证码开关
    captchaEnabled: true
    # 验证码类型 math 数组计算 char 字符
    captchaType: math
  cookie:
    # 设置Cookie的域名 默认空，即当前访问的域名
    domain:
    # 设置cookie的有效访问路径
    path: /
    # 设置HttpOnly属性
    httpOnly: true
    # 设置Cookie的过期时间，天为单位
    maxAge: 30
  session:
    # Session超时时间（默认30分钟）
    expireTime: 30
    # 同步session到数据库的周期（默认1分钟）
    dbSyncPeriod: 1
    # 相隔多久检查一次session的有效性，默认就是10分钟
    validationInterval: 10

# 防止XSS攻击
xss:
  # 过滤开关
  enabled: true
  # 排除链接（多个用逗号分隔）
  excludes: /system/notice/*
  # 匹配链接
  urlPatterns: /system/*,/monitor/*,/tool/*
```

数据源配置 `application-druid.yml`

```yaml
# 数据源配置
spring:
    datasource:
        type: com.alibaba.druid.pool.DruidDataSource
        driverClassName: com.mysql.cj.jdbc.Driver
        druid:
            # 主库数据源
            master:
                url: jdbc:mysql://localhost:3306/ry?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&useSSL=true&serverTimezone=GMT%2B8
                username: root
                password: password
            # 从库数据源
            slave:
                # 从数据源开关/默认关闭
                enabled: false
                url:
                username:
                password:
            # 初始连接数
            initialSize: 5
            # 最小连接池数量
            minIdle: 10
            # 最大连接池数量
            maxActive: 20
            # 配置获取连接等待超时的时间
            maxWait: 60000
            # 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒
            timeBetweenEvictionRunsMillis: 60000
            # 配置一个连接在池中最小生存的时间，单位是毫秒
            minEvictableIdleTimeMillis: 300000
            # 配置一个连接在池中最大生存的时间，单位是毫秒
            maxEvictableIdleTimeMillis: 900000
            # 配置检测连接是否有效
            validationQuery: SELECT 1 FROM DUAL
            testWhileIdle: true
            testOnBorrow: false
            testOnReturn: false
            webStatFilter:
                enabled: true
            statViewServlet:
                enabled: true
                # 设置白名单，不填则允许所有访问
                allow:
                url-pattern: /monitor/druid/*
            filter:
                stat:
                    enabled: true
                    # 慢SQL记录
                    log-slow-sql: true
```

```
                    slow-sql-millis: 1000
                    merge-sql: true
                wall:
                    config:
                        multi-statement-allow: true
```

代码生成配置 `generator.yml`

```
# 代码生成
gen:
  # 作者
  author: ruoyi
  # 默认生成包路径 system 需改成自己的模块名称 如 system monitor tool
  packageName: com.ruoyi.system
  # 自动去除表前缀，默认是true
  autoRemovePre: true
  # 表前缀(类名不会包含表前缀)
  tablePrefix: sys_
```

核心技术

SpringBoot框架

1、介绍Spring Boot是一款开箱即用框架，提供各种默认配置来简化项目配置。让我们的Spring应用变的更轻量化、更快的入门。在主程序执行main函数就可以运行。你也可以打包你的应用为jar并通过使用java -jar来运行你的Web应用。它遵循"约定优先于配置"的原则，使用SpringBoot只需很少的配置，大部分的时候直接使用默认的配置即可。可以与Spring Cloud的微服务无缝结合。 `Spring Boot2.0 环境要求必须是jdk8或以上版本，Tomcat8或以上版本`

2、优点

- 使编码变得简单： 推荐使用注解。
- 使配置变得简单： 自动配置、快速构建项目、快速集成新技术能力 没有冗余代码生成和XML配置的要求
- 使部署变得简单： 内嵌Tomcat、Jetty、Undertow等web容器，无需以war包形式部署
- 使监控变得简单： 自带项目监控

Shiro安全控制

1、介绍Apache Shiro是Java的一个安全框架。Shiro可以帮助我们完成：认证、授权、加密、会话管理、与Web集成、缓存等。其不仅可以用在JavaSE环境，也可以用在 JavaEE 环境。

2、优点

- 易于理解的 Java Security API
- 简单的身份认证，支持多种数据源
- 对角色的简单的授权，支持细粒度的授权
- 不跟任何的框架或者容器捆绑，可以独立运行3、特性 `Authentication` 身份认证/登录，验证用户是不是拥有相应的身份 `Authorization` 授权，即验证权限，验证某个已认证的用户是否拥有某个权限，即判断用户是否能做事

情`SessionManagement`会话管理，即用户登录后就是一次会话，在没有退出之前，它的所有信息都在会话中`Cryptography`加密，保护数据的安全性，如密码加密存储到数据库，而不是明文存储`Caching`缓存，比如用户登录后，其用户信息，拥有的角色/权限不必每次去查，提高效率`Concurrency`Shiro支持多线程应用的并发验证，即如在一个线程中开启另一个线程，能把权限自动传播过去`Testing`提供测试支持`RunAs`允许一个用户假装为另一个用户（如果他们允许）的身份进行访问`RememberMe`记住我，这是非常常见的功能，即一次登录后，下次再来的话不用登录了

4、架构`Subject`主体，代表了当前的“用户”，这个用户不一定是一个具体的人，与当前应用交互的任何东西都是Subject，如网络爬虫，机器人等；即一个抽象概念；所有Subject都绑定到SercurityManager，与Subject的所有交互都会委托给SecurityManager；可以把Subject认为是一个门面；SecurityManager才是实际的执行者`SecurityManage`安全管理器；即所有与安全有关的操作都会与SecurityManager交互；且它管理着所有Subject；可以看出它是Shiro的核心，它负责与后边介绍的其他组件进行交互`Realm`域，Shiro从Realm获取安全数据（如用户，角色，权限），就是说SecurityManager要验证用户身份，那么它需要从Realm获取相应的用户进行比较以确定用户身份是否合法；也需要从Realm得到用户相应的角色/权限进行验证用户是否能进行操作；可以有1个或多个Realm，我们一般在应用中都需要实现自己的Realm`SessionManager`如果写过Servlet就应该知道Session的概念，Session需要有人去管理它的生命周期，这个组件就是SessionManager`SessionDAO`DAO大家都用过，数据库访问对象，用于会话的CRUD，比如我们想把Session保存到数据库，那么可以实现自己的SessionDAO，也可以写入缓存，以提高性能`CacheManager`缓存控制器，来管理如用户，角色，权限等的缓存的；因为这些数据基本上很少去改变，放到缓存中后可以提高访问的性能

应用代码通过Subject来进行认证和授权，而Subject又委托给SecurityManager；我们需要给Shrio的SecurityManager注入Realm，从而让SecurityManager能得到合法的用户及其权限进行判断，Shiro不提供维护用户/权限，而是通过Realm让开发人员自己注入。

`Shiro不会去维护用户，维护权限；这些需要自己去设计/提供；然后通过响应的接口注入给Shiro即可`

Thymeleaf模板

1、介绍Thymeleaf是一个用于Web和独立Java环境的模板引擎，能够处理HTML、XML、JavaScript、CSS甚至纯文本。能轻易的与Spring MVC等Web框架进行集成作为Web应用的模板引擎。与其它模板引擎（比如FreeMaker）相比，Thymeleaf最大的特点是能够直接在浏览器中打开并正确显示模板页面，而不需要启动整个Web应用（更加方便前后端分离，比如方便类似VUE前端设计页面），抛弃JSP吧。Thymeleaf 3.0是一个完全彻底重构的模板引擎，极大的减少内存占用和提升性能和并发性，避免v2.1版因大量的输出标记的集合产生的资源占用。Thymeleaf 3.0放弃了大多数面向DOM的处理机制，变成了一个基于事件的模板处理器，它通过处理模板标记或文本并立即生成其输出，甚至在新事件之前响应模板解析器/缓存事件。Thymeleaf是Spring Boot官方的推荐使用模板。

2、优点

- 国际化支持非常简单
- 语法简单，功能强大。内置大量常用功能，使用非常方便
- 可以很好的和Spring集成
- 静态html嵌入标签属性，浏览器可以直接打开模板文件，便于前后端联调
- Spring Boot 官方推荐，用户群广

- 分页实现
- 导入导出
- 上传下载
- 事务管理
- 异常处理
- 系统日志
- 数据权限
- 多数据源
- 代码生成
- 定时任务
- 系统接口
- 国际化支持

分页实现

前端基于Bootstrap的轻量级表格插件 `BootstrapTable` 后端分页组件使用Mybatis分页插件 `PageHelper`

> 分页实现流程

1、前端调用封装好的方法`$.table.init`，传入后台`url`。

```
var options = {
    url: prefix + "/list",
    columns: [{
        field: 'id',
        title: '主键'
    },
    {
        field: 'name',
        title: '名称'
    }]
};
$.table.init(options);
```

2、自定义查询条件参数（特殊情况提前设置查询条件下使用）

```
var options = {
    url: prefix + "/list",
    queryParams: queryParams,
    columns: [{
        field: 'id',
        title: '主键'
    },
    {
        field: 'name',
        title: '名称'
    }]
};
$.table.init(options);

function queryParams(params) {
    var search = $.table.queryParams(params);
    search.userName = $("#userName").val();
    return search;
}
```

3、后台实现查询逻辑，调用`startPage()`方法即可自动完成服务端分页。

```
@PostMapping("/list")
@ResponseBody
public TableDataInfo list(User user)
{
```

```
    startPage();  // 此方法配合前端完成自动分页
    List<User> list = userService.selectUserList(user);
    return getDataTable(list);
}
```

常见坑点1：selectPostById莫名其妙的分页。例如下面这段代码

```
startPage();
List<User> list;
if(user != null){
    list = userService.selectUserList(user);
} else {
    list = new ArrayList<User>();
}
Post post = postService.selectPostById(1L);
return getDataTable(list);
```

原因分析：这种情况下由于user存在null的情况，就会导致 `PageHelper` 生产了一个分页参数，但是没有被消费，这个参数就会一直保留在这个线程上。当这个线程再次被使用时，就可能导致不该分页的方法去消费这个分页参数，这就产生了莫名其妙的分页。上面这个代码，应该写成下面这个样子才能保证安全。

```
List<User> list;
if(user != null){
    startPage();
    list = userService.selectUserList(user);
} else {
    list = new ArrayList<User>();
}
Post post = postService.selectPostById(1L);
return getDataTable(list);
```

常见坑点2：添加了startPage方法。也没有正常分页。例如下面这段代码

```
startPage();
Post post = postService.selectPostById(1L);
List<User> list = userService.selectUserList(user);
return getDataTable(list);
```

原因分析：只对该语句以后的第一个查询（Select）语句得到的数据进行分页。上面这个代码，应该写成下面这个样子才能正常分页。

```
Post post = postService.selectPostById(1L);
startPage();
List<User> list = userService.selectUserList(user);
return getDataTable(list);
```

注意：如果改为其他数据库需修改配置 `application.yml` `helperDialect=你的数据库`

导入导出

在实际开发中经常需要使用导入导出功能来加快数据的操作。在项目中可以使用注解来完成此项功能。在需要被导入导出的实体类属性添加 `@Excel` 注解，目前支持参数如下：

参数	类型	默认值	描述
name	String	空	导出到Excel中的名字
dateFormat	String	空	日期格式, 如: yyyy-MM-dd
readConverterExp	String	空	读取内容转表达式 (如: 0=男,1=女,2=未知)
columnType	Enum	Type.STRING	导出类型（0数字 1字符串）
height	String	14	导出时在excel中每个列的高度 单位为字符
width	String	16	导出时在excel中每个列的宽 单位为字符
suffix	String	空	文字后缀,如% 90 变成90%
defaultValue	String	空	当值为空时,字段的默认值
prompt	String	空	提示信息
combo	String	Null	设置只能选择不能输入的列内容
targetAttr	String	空	另一个类中的属性名称,支持多级获取,以小数点隔开
type	Enum	Type.ALL	字段类型（0: 导出导入; 1: 仅导出; 2: 仅导入）

> 导出实现流程

1、前端调用封装好的方法$.table.init，传入后台 `exportUrl`

```
var options = {
    exportUrl: prefix + "/export",
    columns: [{
        field: 'id',
        title: '主键'
    },
    {
        field: 'name',
        title: '名称'
    }]
};
$.table.init(options);
```

2、添加导出按钮事件

```
<a class="btn btn-warning" onclick="$.table.exportExcel()">
    <i class="fa fa-download"></i> 导出
</a>
```

3、在实体变量上添加@Excel注解

```
@Excel(name = "用户序号", prompt = "用户编号")
private Long userId;

@Excel(name = "用户名称")
private String userName;

@Excel(name = "用户性别", readConverterExp = "0=男,1=女,2=未知")
private String sex;

@Excel(name = "最后登陆时间", width = 30, dateFormat = "yyyy-MM-dd HH:mm:ss")
private Date loginDate;
```

4、在Controller添加导出方法

```
@PostMapping("/export")
@ResponseBody
public AjaxResult export(User user)
{
    List<User> list = userService.selectUserList(user);
    ExcelUtil<User> util = new ExcelUtil<User>(User.class);
    return util.exportExcel(list, "用户数据");
}
```

> 导入实现流程

1、前端调用封装好的方法$.table.init，传入后台importUrl。

```
var options = {
    importUrl: prefix + "/importData",
    columns: [{
        field: 'id',
        title: '主键'
    },
    {
        field: 'name',
        title: '名称'
    }]
};
$.table.init(options);
```

2、添加导入按钮事件

```
<a class="btn btn-info" onclick="$.table.importExcel()">
    <i class="fa fa-upload"></i> 导入
</a>
```

3、添加导入前端代码，form默认id为importForm，也可指定importExcel(id)

```
<form id="importForm" enctype="multipart/form-data" class="mt20 mb10" style="display: none;">
```

```
    <div class="col-xs-offset-1">
        <input type="file" id="file" name="file"/>
        <div class="mt10 pt5">
            <input type="checkbox" id="updateSupport" name="updateSupport" title="如果登录账户已经存在，更新这条数据。"> 是否更新已经存在的用户数据
                 <a onclick="$.table.importTemplate()" class="btn btn-default btn-xs"><i class="fa fa-file-excel-o"></i> 下载模板</a>
        </div>
        <font color="red" class="pull-left mt10">
            提示：仅允许导入“xls”或“xlsx”格式文件！
        </font>
    </div>
</form>
```

4、在实体变量上添加@Excel注解，默认为导出导入，也可以单独设置仅导入Type.IMPORT

```
@Excel(name = "用户序号")
private Long id;

@Excel(name = "部门编号", type = Type.IMPORT)
private Long deptId;

@Excel(name = "用户名称")
private String userName;

/** 导出部门多个对象 */
@Excels({
    @Excel(name = "部门名称", targetAttr = "deptName", type = Type.EXPORT),
    @Excel(name = "部门负责人", targetAttr = "leader", type = Type.EXPORT)
})
private SysDept dept;

/** 导出部门单个对象 */
@Excel(name = "部门名称", targetAttr = "deptName", type = Type.EXPORT)
private SysDept dept;
```

5、在Controller添加导入方法，updateSupport属性为是否存在则覆盖（可选）

```
@PostMapping("/importData")
@ResponseBody
public AjaxResult importData(MultipartFile file, boolean updateSupport) throws Exception
{
    ExcelUtil<SysUser> util = new ExcelUtil<SysUser>(SysUser.class);
    List<SysUser> userList = util.importExcel(file.getInputStream());
    String operName = ShiroUtils.getSysUser().getLoginName();
    String message = userService.importUser(userList, updateSupport, operName);
    return AjaxResult.success(message);
}
```

上传下载

首先创建一张上传文件的表，例如：

```
drop table if exists sys_file_info;
create table sys_file_info (
  file_id           int(11)          not null auto_increment       comment '文件id',
  file_name         varchar(50)      default ''                    comment '文件名称',
  file_path         varchar(255)     default ''                    comment '文件路径',
  primary key (file_id)
) engine=innodb auto_increment=1 default charset=utf8 comment = '文件信息表';
```

> 上传实现流程

1、代码生成sys_file_info表相关代码并覆盖到对应目录。

2、参考示例修改代码。

```
<input id="filePath" name="filePath" class="form-control" type="file">
```

```
function submitHandler() {
    if ($.validate.form()) {
        uploadFile();
    }
}

function uploadFile() {
    var formData = new FormData();
    if ($('#filePath')[0].files[0] == null) {
        $.modal.alertWarning("请先选择文件路径");
        return false;
    }
    formData.append('fileName', $("#fileName").val());
    formData.append('file', $('#filePath')[0].files[0]);
    $.ajax({
        url: prefix + "/add",
        type: 'post',
        cache: false,
        data: formData,
        processData: false,
        contentType: false,
        dataType: "json",
        success: function(result) {
            $.operate.successCallback(result);
        }
    });
}
```

3、在FileInfoController添加对应上传方法

```
1.
2. @PostMapping("/add")
3. @ResponseBody
4. public AjaxResult addSave(@RequestParam("file") MultipartFile file, FileInfo fileInfo) throws IOException
5. {
6.     // 上传文件路径
7.     String filePath = RuoYiConfig.getUploadPath();
8.     // 上传并返回新文件名称
9.     String fileName = FileUploadUtils.upload(filePath, file);
10.    fileInfo.setFilePath(fileName);
11.    return toAjax(fileInfoService.insertFileInfo(fileInfo));
12. }
```

4、上传成功后需要预览可以对该属性格式化处理

```
1. {
2.     field : 'filePath',
3.     title: '文件预览',
4.     formatter: function(value, row, index) {
           return '<a href="javascript:downloadFile(' + row.fileId + ')"><img style="width:30;height:30px;"
5. src="/profile/upload/' + row.filePath + '"/></a>';
6.     }
7. },
```

如需对文件格式控制，设置 `application.yml` 中的 `multipart` 属性

```
1. # 文件上传
2. servlet:
3.    multipart:
4.      # 单个文件大小
5.      max-file-size:  10MB
6.      # 设置总上传的文件大小
7.      max-request-size:  20MB
```

注意：如果只是单纯的上传一张图片没有其他参数可以使用通用方法 `/common/upload` 请求处理方法 `com.ruoyi.web.controller.common.CommonController`

> 下载实现流程

1、参考示例代码。

```
1. function downloadFile(fileId){
2.     window.location.href = ctx + "system/fileInfo/downloadFile/" + fileId;
3. }
```

2、在Controller添加对应上传方法

```
1. @GetMapping("/downloadFile/{fileId}")
```

```
public void downloadFile(@PathVariable("fileId") Integer fileId, HttpServletResponse response,
        HttpServletRequest request) throws Exception
{
    FileInfo sysFile = fileInfoService.selectFileInfoById(fileId);
    String filePath = sysFile.getFilePath();
    String realFileName = sysFile.getFileName() + filePath.substring(filePath.indexOf("."));
    String path = RuoYiConfig.getUploadPath() + sysFile.getFilePath();
    response.setCharacterEncoding("utf-8");
    response.setContentType("multipart/form-data");
    response.setHeader("Content-Disposition",
            "attachment;fileName=" + FileUtils.setFileDownloadHeader(request, realFileName));
    FileUtils.writeBytes(path, response.getOutputStream());
}
```

事务管理

新建的Spring Boot项目中，一般都会引用spring-boot-starter或者spring-boot-starter-web，而这两个起步依赖中都已经包含了对于spring-boot-starter-jdbc或spring-boot-starter-data-jpa的依赖。当我们使用了这两个依赖的时候，框架会自动默认分别注入DataSourceTransactionManager或JpaTransactionManager。所以我们不需要任何额外配置就可以用 `@Transactional` 注解进行事务的使用。

> @Transactional注解只能应用到public可见度的方法上，可以被应用于接口定义和接口方法，方法会覆盖类上面声明的事务。

例如用户新增需要插入用户表、用户与岗位关联表、用户与角色关联表，如果插入成功，那么一起成功，如果中间有一条出现异常，那么回滚之前的所有操作，这样可以防止出现脏数据，就可以使用事务让它实现回退。做法非常简单，我们只需要在方法或类添加 `@Transactional` 注解即可。

```
@Transactional
public int insertUser(User user)
{
    // 新增用户信息
    int rows = userMapper.insertUser(user);
    // 新增用户岗位关联
    insertUserPost(user);
    // 新增用户与角色管理
    insertUserRole(user);
    return rows;
}
```

常见坑点1：遇到非检测异常时，事务开启，也无法回滚。例如下面这段代码，用户依旧增加成功，并没有因为后面遇到检测异常而回滚！！

```
@Transactional
public int insertUser(User user) throws Exception
{
    // 新增用户信息
    int rows = userMapper.insertUser(user);
    // 新增用户岗位关联
    insertUserPost(user);
    // 新增用户与角色管理
    insertUserRole(user);
    // 模拟抛出SQLException异常
    boolean flag = true;
    if (flag)
    {
        throw new SQLException("发生异常了..");
    }
    return rows;
}
```

原因分析：因为Spring的默认的事务规则是遇到运行异常（RuntimeException）和程序错误（Error）才会回滚。如果想针对非检测异常进行事务回滚，可以在@Transactional 注解里使用rollbackFor 属性明确指定异

常。例如下面这样，就可以正常回滚：

```
@Transactional(rollbackFor = Exception.class)
public int insertUser(User user) throws Exception
{
    // 新增用户信息
    int rows = userMapper.insertUser(user);
    // 新增用户岗位关联
    insertUserPost(user);
    // 新增用户与角色管理
    insertUserRole(user);
    // 模拟抛出SQLException异常
    boolean flag = true;
    if (flag)
    {
        throw new SQLException("发生异常了..");
    }
    return rows;
}
```

常见坑点2： 在业务层捕捉异常后，发现事务不生效。这是许多新手都会犯的一个错误，在业务层手工捕捉并处理了异常，你都把异常“吃”掉了，Spring自然不知道这里有错，更不会主动去回滚数据。例如：下面这段代码直接导致用户新增的事务回滚没有生效。

```
@Transactional
public int insertUser(User user) throws Exception
{
    // 新增用户信息
    int rows = userMapper.insertUser(user);
    // 新增用户岗位关联
    insertUserPost(user);
    // 新增用户与角色管理
    insertUserRole(user);
    // 模拟抛出SQLException异常
    boolean flag = true;
    if (flag)
    {
        try
        {
            // 谨慎：尽量不要在业务层捕捉异常并处理
            throw new SQLException("发生异常了..");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
    return rows;
}
```

推荐做法：在业务层统一抛出异常，然后在控制层统一处理。

```java
@Transactional
public int insertUser(User user) throws Exception
{
    // 新增用户信息
    int rows = userMapper.insertUser(user);
    // 新增用户岗位关联
    insertUserPost(user);
    // 新增用户与角色管理
    insertUserRole(user);
    // 模拟抛出SQLException异常
    boolean flag = true;
    if (flag)
    {
        throw new RuntimeException("发生异常了..");
    }
    return rows;
}
```

Transactional注解的常用属性表：

属性	说明
propagation	事务的传播行为，默认值为 REQUIRED。
isolation	事务的隔离度，默认值采用 DEFAULT
timeout	事务的超时时间，默认值为-1，不超时。如果设置了超时时间(单位秒)，那么如果超过该时间限制了但事务还没有完成，则自动回滚事务。
read-only	指定事务是否为只读事务，默认值为 false；为了忽略那些不需要事务的方法，比如读取数据，可以设置 read-only 为 true。
rollbackFor	用于指定能够触发事务回滚的异常类型，如果有多个异常类型需要指定，各类型之间可以通过逗号分隔。{xxx1.class, xxx2.class,……}
noRollbackFor	抛出 no-rollback-for 指定的异常类型，不回滚事务。{xxx1.class, xxx2.class,……}
….	

> 事务的传播机制是指如果在开始当前事务之前，一个事务上下文已经存在，此时有若干选项可以指定一个事务性方法的执行行为。即:在执行一个@Transactinal注解标注的方法时，开启了事务；当该方法还在执行中时，另一个人也触发了该方法；那么此时怎么算事务呢，这时就可以通过事务的传播机制来指定处理方式。

TransactionDefinition传播行为的常量：

常量	含义
TransactionDefinition.PROPAGATION_REQUIRED	如果当前存在事务，则加入该事务；如果当前没创建一个新的事务。这是默认值。
TransactionDefinition.PROPAGATION_REQUIRES_NEW	创建一个新的事务，如果当前存在事务，则把当起。
TransactionDefinition.PROPAGATION_SUPPORTS	如果当前存在事务，则加入该事务；如果当前没以非事务的方式继续运行。
TransactionDefinition.PROPAGATION_NOT_SUPPORTED	以非事务方式运行，如果当前存在事务，则把当起。

TransactionDefinition.PROPAGATION_NEVER	以非事务方式运行，如果当前存在事务，则抛出
TransactionDefinition.PROPAGATION_MANDATORY	如果当前存在事务，则加入该事务；如果当前没抛出异常。
TransactionDefinition.PROPAGATION_NESTED	如果当前存在事务，则创建一个事务作为当前事务来运行；如果当前没有事务，则该取值等价于TransactionDefinition.PROPAGATION_R

异常处理

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

所谓全局异常处理器就是使用 `@ControllerAdvice` 注解。示例如下：

1、统一返回实体定义

```
1. package com.ruoyi.common.core.domain;
2.
3. import java.util.HashMap;
4.
5. /**
6.  * 操作消息提醒
7.  *
8.  * @author ruoyi
9.  */
10. public class AjaxResult extends HashMap<String, Object>
11. {
12.     private static final long serialVersionUID = 1L;
13.
14.     /**
15.      * 返回错误消息
16.      *
17.      * @param code 错误码
18.      * @param msg 内容
19.      * @return 错误消息
20.      */
21.     public static AjaxResult error(String msg)
22.     {
23.         AjaxResult json = new AjaxResult();
24.         json.put("msg", msg);
25.         json.put("code", 500);
26.         return json;
27.     }
28.
29.     /**
30.      * 返回成功消息
31.      *
32.      * @param msg 内容
33.      * @return 成功消息
34.      */
35.     public static AjaxResult success(String msg)
36.     {
37.         AjaxResult json = new AjaxResult();
38.         json.put("msg", msg);
```

```
        json.put("code", 0);
        return json;
    }
}
```

2、定义登录异常定义

```
package com.ruoyi.common.exception;

/**
 * 登录异常
 *
 * @author ruoyi
 */
public class LoginException extends RuntimeException
{
    private static final long serialVersionUID = 1L;

    protected final String message;

    public LoginException(String message)
    {
        this.message = message;
    }

    @Override
    public String getMessage()
    {
        return message;
    }
}
```

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

```
package com.ruoyi.framework.web.exception;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import com.ruoyi.common.core.domain.AjaxResult;
import com.ruoyi.common.exception.LoginException;

/**
 * 全局异常处理器
 *
 * @author ruoyi
 */
@RestControllerAdvice
public class GlobalExceptionHandler
{
    private static final Logger log = LoggerFactory.getLogger(GlobalExceptionHandler.class);
```

```
19.
20.     /**
21.      * 登录异常
22.      */
23.     @ExceptionHandler(LoginException.class)
24.     public AjaxResult loginException(LoginException e)
25.     {
26.         log.error(e.getMessage(), e);
27.         return AjaxResult.error(e.getMessage());
28.     }
29. }
```

4、测试访问请求

```
1. @Controller
2. public class SysIndexController
3. {
4.     /**
5.      * 首页方法
6.      */
7.     @GetMapping("/index")
8.     public String index(ModelMap mmap)
9.     {
10.         /**
11.          * 模拟用户未登录，抛出业务逻辑异常
12.          */
13.         SysUser user = ShiroUtils.getSysUser();
14.         if (StringUtils.isNull(user))
15.         {
16.             throw new LoginException("用户未登录，无法访问请求。");
17.         }
18.         mmap.put("user", user);
19.         return "index";
20.     }
21. }
```

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

```
1. {
2.     "msg": "用户未登录，无法访问请求。",
3.     "code": 500
4. }
```

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

```
1. @ExceptionHandler(LoginException.class)
2. public Object loginException(HttpServletRequest request, LoginException e)
3. {
```

```
    log.error(e.getMessage(), e);

    if (ServletUtils.isAjaxRequest(request))
    {
        return AjaxResult.error(e.getMessage());
    }
    else
    {
        return new ModelAndView("/error/500");
    }
}
```

若依系统的全局异常处理器 `GlobalExceptionHandler` 注意：如果全部异常处理返回 `json` ，那么可以使用 `@RestControllerAdvice` 代替 `@ControllerAdvice` ，这样在方法上就可以不需要添加 `@ResponseBody` 。

系统日志

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

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

```
@Log(title = "用户管理", businessType = BusinessType.INSERT)
```

支持参数如下：

参数	类型	默认值	描述
title	String	空	操作模块
businessType	BusinessType	OTHER	操作功能（OTHER其他 INSERT新增 UPDATE修改 DELETE删除 GRANT授权 EXPORT导出 IMPORT导入 FORCE强退 GENCODE生成代码 CLEAN清空数据）
operatorType	OperatorType	MANAGE	操作人类别（OTHER其他 MANAGE后台用户 MOBILE手机端用户）
isSaveRequestData	boolean	true	是否保存请求的参数

> 关于自定义操作功能使用流程

1、在 `BusinessType` 中新增业务操作类型如：

```
/**
 * 测试
 */
TEST,
```

2、在 `sys_dict_data` 字典数据表中初始化操作业务类型

```
insert into sys_dict_data values(25, 10, '测试',     '10', 'sys_oper_type',       '',   'primary', 'N', '0', 'admin', '2018-03-16 11-33-00', 'ry', '2018-03-16 11-33-00', '测试操作');
```

3、在 `Controller` 中使用注解

```
@Log(title = "测试标题", businessType = BusinessType.TEST)
```

逻辑实现代码 `com.ruoyi.framework.aspectj.LogAspect` 查询操作详细记录可以登录系统（系统管理-操作日志）

数据权限

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

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

> 关于数据权限使用流程

支持参数如下：

参数	类型	默认值	描述
deptAlias	String	空	部门表的别名
userAlias	String	空	用户表的别名

1、在（系统管理-角色管理）设置需要数据权限的角色目前支持以下几种权限

- 全部数据权限
- 自定数据权限
- 部门数据权限
- 部门及以下数据权限
- 仅本人数据权限2、在需要数据权限控制方法上添加 `@DataScope` 注解，其中 `d` 和 `u` 用来表示表的别名

```
// 部门数据权限注解
@DataScope(deptAlias = "u")
// 部门及用户权限注解
@DataScope(deptAlias = "d", userAlias = "u")
```

3、在 `mybatis` 查询底部标签添加数据范围过滤

```
<!-- 数据范围过滤 -->
${params.dataScope}
```

用户管理（未过滤数据权限的情况）：

```
select u.user_id, u.dept_id, u.login_name, u.user_name, u.email
    , u.phonenumber, u.password, u.sex, u.avatar, u.salt
    , u.status, u.del_flag, u.login_ip, u.login_date, u.create_by
    , u.create_time, u.remark, d.dept_name
from sys_user u
    left join sys_dept d on u.dept_id = d.dept_id
where u.del_flag = '0'
```

用户管理（已过滤数据权限的情况）：

```
select u.user_id, u.dept_id, u.login_name, u.user_name, u.email
    , u.phonenumber, u.password, u.sex, u.avatar, u.salt
    , u.status, u.del_flag, u.login_ip, u.login_date, u.create_by
    , u.create_time, u.remark, d.dept_name
from sys_user u
    left join sys_dept d on u.dept_id = d.dept_id
where u.del_flag = '0'
    and u.dept_id in (
        select dept_id
        from sys_role_dept
        where role_id = 2
    )
```

结果很明显，我们多了如下语句。通过角色部门表（sys_role_dept）完成了数据权限过滤

```
and u.dept_id in (
    select dept_id
    from sys_role_dept
    where role_id = 2
)
```

逻辑实现代码 `com.ruoyi.framework.aspectj.DataScopeAspect`

多数据源

在实际开发中，经常可能遇到在一个应用中可能需要访问多个数据库的情况在需要切换数据源 `Service` 或 `Mapper` 方法上添加 `@DataSource` 注解@DataSource(value = DataSourceType.MASTER)，其中 `value` 用来表示数据源名称

> 关于多数据源使用流程（如果有多个，可以参考slave添加）

支持参数如下：

参数	类型	默认值	描述
value	DataSourceType	DataSourceType.MASTER	主库

1、在 `application-druid.yml` 配置从库数据源

```
# 从库数据源
slave:
    # 从数据源开关/默认关闭
    enabled: true
    url: jdbc:mysql://localhost:3306/test?
useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&useSSL=true&serverTimezone=GMT%2B8
    username: root
    password: password
```

2、在 `DataSourceType` 类添加数据源枚举

```
/**
 * 从库
 */
SLAVE
```

3、在 `DruidConfig` 配置读取数据源

```
@Bean
@ConfigurationProperties("spring.datasource.druid.slave")
@ConditionalOnProperty(prefix = "spring.datasource.druid.slave", name = "enabled", havingValue = "true")
public DataSource slaveDataSource(DruidProperties druidProperties)
{
    DruidDataSource dataSource = DruidDataSourceBuilder.create().build();
    return druidProperties.dataSource(dataSource);
}
```

4、在 `DruidConfig` 类 `dataSource` 方法添加数据源

```
targetDataSources.put(DataSourceType.SLAVE.name(), slaveDataSource);
```

5、在需要使用多数据源方法或类上添加 `@DataSource` 注解，其中 `value` 用来表示数据源

```
@DataSource(value = DataSourceType.SLAVE)
public List<SysUser> selectUserList(SysUser user)
{
    return userMapper.selectUserList(user);
}
```

```
@Service
@DataSource(value = DataSourceType.SLAVE)
public class SysUserServiceImpl
```

对于特殊情况可以通过 `DynamicDataSourceContextHolder` 手动实现数据源切换

```
public List<SysUser> selectUserList(SysUser user)
{
    DynamicDataSourceContextHolder.setDataSourceType(DataSourceType.SLAVE.name());
    List<SysUser> userList = userMapper.selectUserList(user);
    DynamicDataSourceContextHolder.clearDataSourceType();
    return userList;
}
```

逻辑实现代码 `com.ruoyi.framework.aspectj.DataSourceAspect`

注意：目前配置了一个从库，默认关闭状态。如果不需要多数据源不用做任何配置。另外可新增多个从库。支持不同数据源（Mysql、Oracle、SQLServer）

代码生成

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

> 关于代码生成使用流程

1、修改代码生成配置 单应用编辑 `resources目录下的application.yml` 多模块编辑 `ruoyi-generator` 中的 `resources` 目录下的 `generator.yml` `author` ： # 开发者姓名，生成到类注释上 `packageName` ： # 默认生成包路径 `autoRemovePre` ： # 是否自动去除表前缀 `tablePrefix` ： # 表前缀

2、新建数据库表结构（单表）

```
drop table if exists sys_student;
create table sys_student (
  student_id           int(11)         auto_increment    comment '编号',
  student_name         varchar(30)     default ''        comment '学生名称',
  student_age          int(3)          default null      comment '年龄',
  student_sex          char(1)         default '0'       comment '性别（0男 1女 2未知）',
  student_status       char(1)         default '0'       comment '状态（0正常 1停用）',
  student_birthday     datetime                          comment '生日',
  remark               varchar(500)    default null      comment '备注',
  primary key (student_id)
) engine=innodb auto_increment=1 comment = '学生信息表';
```

2、新建数据库表结构（树表）

```
drop table if exists sys_product;
create table sys_product (
  product_id        bigint(20)      not null auto_increment    comment '产品id',
  parent_id         bigint(20)      default 0                  comment '父产品id',
  product_name      varchar(30)     default ''                 comment '产品名称',
  order_num         int(4)          default 0                  comment '显示顺序',
  status            char(1)         default '0'                comment '产品状态（0正常 1停用）',
  primary key (product_id)
) engine=innodb auto_increment=1 comment = '产品表';
```

3、登录系统（系统工具 -> 代码生成 -> 导入对应表）

4、代码生成列表中找到需要表（可预览、修改、删除生成配置）

5、点击生成代码会得到一个ruoyi.zip 执行 `sql` 文件，覆盖文件到对应目录即可

多模块所有代码生成的相关业务逻辑代码在 `ruoyi-generator` 模块，可以自行调整或剔除

定时任务

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

> 关于定时任务使用流程

1、后台添加定时任务处理类（支持Bean调用、Class类调用） `Bean调用示例` ：需要添加对应Bean注解@Component或@Service。调用目标字符串：ryTask.ryParams('ry') `Class类调用示例` ：添加类和方法指定包即可。调用目标字符串：com.ruoyi.quartz.task.RyTask.ryParams('ry')

```
/**
 * 定时任务调度测试
 *
 * @author ruoyi
 */
@Component("ryTask")
public class RyTask
{
    public void ryMultipleParams(String s, Boolean b, Long l, Double d, Integer i)
    {
        System.out.println(StringUtils.format("执行多参方法： 字符串类型{}，布尔类型{}，长整型{}，浮点型{}，整形{}", s, b, l, d, i));
    }

    public void ryParams(String params)
    {
        System.out.println("执行有参方法：" + params);
    }

    public void ryNoParams()
    {
        System.out.println("执行无参方法");
    }
}
```

2、前端新建定时任务信息（系统监控 -> 定时任务） 任务名称：自定义，如：定时查询任务状态 任务分组：根据字典sys_job_group配置 调用目标字符串：设置后台任务方法名称参数 执行表达式：可查询官方cron表达式介绍 执行策略：定时任务自定义执行策略 并发执行：是否需要多个任务间同时执行 状态：是否启动定时任务 备注：定时任务描述信息

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

执行策略详解： `立即执行` （所有misfire的任务会马上执行）打个比方，如果9点misfire了，在10：15系统恢复之后，9点，10点的misfire会马上执行 `执行一次` （会合并部分的misfire，正常执行下一个周期的任务）假设9，10的任务都misfire了，系统在10：15分起来了。只会执行一次misfire，下次正点执行。 `放弃执行` （所有的

misfire不管，执行下一个周期的任务）

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

cron表达式语法:[秒] [分] [小时] [日] [月] [周] [年]

说明	必填	允许填写的值	允许的通配符
秒	是	0-59	, - /
分	是	0-59	, - /
时	是	0-23	, - /
日	是	1-31	, - /
月	是	1-12 / JAN-DEC	, - ? / L W
周	是	1-7 or SUN-SAT	, - ? / L #
年	是	1970-2099	, - * /

通配符说明： `*` 表示所有值。 例如:在分的字段上设置 *,表示每一分钟都会触发 `?` 表示不指定值。使用的场景为不需要关心当前设置这个字段的值。例如:要在每月的10号触发一个操作，但不关心是周几，所以需要周位置的那个字段设置为"?" 具体设置为 0 0 0 10 * ? `-` 表示区间。例如 在小时上设置 "10-12",表示 10,11,12点都会触发 `,` 表示指定多个值，例如在周字段上设置 "MON,WED,FRI" 表示周一，周三和周五触发 `/` 用于递增触发。如在秒上面设置"5/15" 表示从5秒开始，每增15秒触发(5,20,35,50)。 在月字段上设置'1/3'所示每月1号开始，每隔三天触发一次 `L` 表示最后的意思。在日字段设置上，表示当月的最后一天(依据当前月份，如果是二月还会依据是否是润年[leap]), 在周字段上表示星期六，相当于"7"或"SAT"。如果在"L"前加上数字，则表示该数据的最后一个。例如在周字段上设置"6L"这样的格式,则表示"本月最后一个星期五" `W` 表示离指定日期的最近那个工作日(周一至周五). 例如在日字段上置"15W"，表示离每月15号最近的那个工作日触发。如果15号正好是周六，则找最近的周五(14号)触发， 如果15号是周未，则找最近的下周一(16号)触发.如果15号正好在工作日(周一至周五)，则就在该天触发。如果指定格式为 "1W",它则表示每月1号往后最近的工作日触发。如果1号正是周六，则将在3号下周一触发。(注，"W"前只能设置具体的数字,不允许区间"-") `#` 序号(表示每月的第几个周几)，例如在周字段上设置"6#3"表示在每月的第三个周六.注意如果指定"#5",正好第五周没有周六，则不会触发该配置(用在母亲节和父亲节再合适不过了) ；小提示：'L'和 'W'可以一组合使用。如果在日字段上设置"LW",则表示在本月的最后一个工作日触发；周字段的设置，若使用英文字母是不区分大小写的，即MON与mon相同

常用表达式例子：

表达式	说明
0 0 2 1 ?	表示在每月的1日的凌晨2点调整任务
0 15 10 ? *MON-FRI*	表示周一到周五每天上午10:15执行作业
0 15 10 ? 6L 2002-2006	表示2002-2006年的每个月的最后一个星期五上午10:15执行作
0 0 10,14,16 ?	每天上午10点，下午2点，4点
0 0/30 9-17 ?	朝九晚五工作时间内每半小时
0 0 12 ? WED	表示每个星期三中午12点
0 0 12 ?	每天中午12点触发
0 15 10 ?	每天上午10:15触发

0 15 10 ?	每天上午10:15触发
0 15 10 ?	每天上午10:15触发
0 15 10 ? *2005*	2005年的每天上午10:15触发
0 14 ?	在每天下午2点到下午2:59期间的每1分钟触发
0 0/5 14 ?	在每天下午2点到下午2:55期间的每5分钟触发
0 0/5 14,18 ?	在每天下午2点到2:55期间和下午6点到6:55期间的每5分钟触发
0 0-5 14 ?	在每天下午2点到下午2:05期间的每1分钟触发
0 10,44 14 ? 3 WED	每年三月的星期三的下午2:10和2:44触发
0 15 10 ? *MON-FRI*	周一至周五的上午10:15触发
0 15 10 15 ?	每月15日上午10:15触发
0 15 10 L ?	每月最后一日的上午10:15触发
0 15 10 ? 6L	每月的最后一个星期五上午10:15触发
0 15 10 ? *6L 2002-2005*	2002年至2005年的每月的最后一个星期五上午10:15触发
0 15 10 ? 6#3	每月的第三个星期五上午10:15触发

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

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

系统接口

在现在的开发过程中还有很大一部分公司都是以口口相传的方式来进行前后端的联调，而接口文档很大一部分都只停留在了说说而已的地步，或者写了代码再写文档。还有一点就是文档的修改，定义好的接口并不是一成不变的，可能在开发过程中文档修改不止一次的变化，这个时候就会很难受了。只要不是强制性要求，没人会愿意写这东西，而且在写的过程中，一个字母的错误就会导致联调时候的很大麻烦，但是通过Swagger，我们可以省略了这一步，而且文档出错率近乎于零，只要你在写代码的时候，稍加几个注解，文档自动生成。

1、在控制层 `Controller` 中添加注解来描述接口信息如：

```
@Api("参数配置")
@Controller
@RequestMapping("/system/config")
public class ConfigController
```

2、在方法中配置接口的标题信息

```
@ApiOperation("查询参数列表")
@ResponseBody
public TableDataInfo list(Config config)
{
    startPage();
    List<Config> list = configService.selectConfigList(config);
    return getDataTable(list);
}
```

3、在 `系统工具-系统接口` 测试相关接口

`注意：SwaggerConfig可以指定根据注解或者包名扫描具体的API`

API详细说明

作用范围	API	使用位置
协议集描述	@Api	用于controller类上
对象属性	@ApiModelProperty	用在出入参数对象的字段上
协议描述	@ApiOperation	用在controller的方法上
Response集	@ApiResponses	用在controller的方法上
Response	@ApiResponse	用在 @ApiResponses里边
非对象参数集	@ApiImplicitParams	用在controller的方法上
非对象参数描述	@ApiImplicitParam	用在@ApiImplicitParams的方法里边
描述返回对象的意义	@ApiModel	用在返回对象类上

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

```
@Api(value = "/user", description = "用户管理")
```

与Controller注解并列使用。 属性配置：

属性名称	备注
value	url的路径值
tags	如果设置这个值、value的值会被覆盖
description	对api资源的描述
basePath	基本路径可以不配置
position	如果配置多个Api 想改变显示的顺序位置
produces	For example, "application/json, application/xml"
consumes	For example, "application/json, application/xml"
protocols	Possible values: http, https, ws, wss.
authorizations	高级特性认证时配置
hidden	配置为true 将在文档中隐藏

`ApiOperation` 标记，用在方法上，说明方法的作用，每一个url资源的定义,使用方式：

```
1. @ApiOperation("获取用户信息")
```

与Controller中的方法并列使用，属性配置：

属性名称	备注
value	url的路径值
tags	如果设置这个值、value的值会被覆盖
description	对api资源的描述
basePath	基本路径可以不配置
position	如果配置多个Api 想改变显示的顺序位置
produces	For example, "application/json, application/xml"
consumes	For example, "application/json, application/xml"
protocols	Possible values: http, https, ws, wss.
authorizations	高级特性认证时配置
hidden	配置为true将在文档中隐藏
response	返回的对象
responseContainer	这些对象是有效的 "List", "Set" or "Map"., 其他无效
httpMethod	"GET", "HEAD", "POST", "PUT", "DELETE", "OPTIONS" and "PATCH"
code	http的状态码 默认 200
extensions	扩展属性

`ApiParam` 标记，请求属性，使用方式：

```
1. public TableDataInfo list(@ApiParam(value = "查询用户列表", required = true)User user)
```

与Controller中的方法并列使用，属性配置：

属性名称	备注
name	属性名称
value	属性值
defaultValue	默认属性值
allowableValues	可以不配置
required	是否属性必填
access	不过多描述
allowMultiple	默认为false
hidden	隐藏该属性
example	举例子

`ApiResponse` 标记，响应配置，使用方式：

```
1. @ApiResponse(code = 400, message = "查询用户失败")
```

与Controller中的方法并列使用，属性配置：

属性名称	备注
code	http的状态码
message	描述
response	默认响应类 Void
reference	参考ApiOperation中配置
responseHeaders	参考 ResponseHeader 属性配置说明
responseContainer	参考ApiOperation中配置

`ApiResponses` 标记，响应集配置，使用方式：

```
1. @ApiResponses({ @ApiResponse(code = 400, message = "无效的用户") })
```

与Controller中的方法并列使用，属性配置：

属性名称	备注
value	多个ApiResponse配置

`ResponseHeader` 标记，响应头设置，使用方法

```
1. @ResponseHeader(name="head",description="响应头设计")
```

与Controller中的方法并列使用，属性配置：

属性名称	备注
name	响应头名称
description	描述
response	默认响应类 void
responseContainer	参考ApiOperation中配置

国际化支持

在我们开发WEB项目的时候，项目可能涉及到在国外部署或者应用，也有可能会有国外的用户对项目进行访问，那么在这种项目中，为客户展现的页面或者操作的信息就需要使用不同的语言，这就是我们所说的项目国际化。目前项目已经支持多语言国际化，接下来我们介绍如何使用。

> 关于国际化使用流程

1、修改 `I18nConfig` 设置默认语言，如默认 `中文` ：

```
// 默认语言，英文可以设置Locale.US
slr.setDefaultLocale(Locale.SIMPLIFIED_CHINESE);
```

2、修改配置 `application.yml` 中的 `basename` 国际化文件，默认是i18n路径下messages文件（比如现在国际化文件是xx_zh_CN.properties、xx_en_US.properties，那么 `basename` 配置应为是 `i18n/xx`

```
spring:
  # 资源信息
  messages:
    # 国际化资源文件路径
    basename: static/i18n/messages
```

3、 `i18n` 目录文件下定义资源文件美式英语 messages_en_US.properties

```
user.login.username=User name
user.login.password=Password
user.login.code=Security code
user.login.remember=Remember me
user.login.submit=Sign In
```

中文简体 messages_zh_CN.properties

```
user.login.username=用户名
user.login.password=密码
user.login.code=验证码
user.login.remember=记住我
user.login.submit=登录
```

4、html使用国际化#{资源文件key}

```
<form id="signupForm">
    <h4 class="no-margins">登录：</h4>
    <p class="m-t-md">你若不离不弃，我必生死相依</p>
    <input type="text"     name="username" class="form-control uname"  th:placeholder="#{user.login.username}"
/>
    <input type="password" name="password" class="form-control pword"  th:placeholder="#{user.login.password}"
/>
```

```
    <div class="row m-t" th:if="${captchaEnabled==true}">
        <div class="col-xs-6">
            <input type="text" name="validateCode" class="form-control code" th:placeholder="#{user.login.code}" maxlength="5" autocomplete="off">
        </div>
        <div class="col-xs-6">
            <a href="javascript:void(0);" title="点击更换验证码">
                <img th:src="@{captcha/captchaImage(type=${captchaType})}" class="imgcode" width="85%"/>
            </a>
        </div>
    </div>
    <div class="checkbox-custom" th:classappend="${captchaEnabled==false} ? 'm-t'">
        <input type="checkbox" id="rememberme" name="rememberme"> <label for="rememberme" th:text="#{user.login.remember}">记住我</label>
    </div>
    <button class="btn btn-success btn-block" id="btnSubmit" data-loading="正在验证登录，请稍后..." th:text="#{user.login.submit}">登录</button>
</form>
```

5、java代码使用MessageUtils获取国际化

```
MessageUtils.message("user.login.username")
MessageUtils.message("user.login.password")
MessageUtils.message("user.login.code")
MessageUtils.message("user.login.remember")
MessageUtils.message("user.login.submit")
```

6、js使用国际化首先在文件引入jquery-i18n-properties依赖，然后在初始化后即可通过JS函数获取对应国际化文件的内容。

```
<!--jQuery国际化插件-->
<script src="../static/js/jquery.i18n.properties.min.js" th:src="@{/js/jquery.i18n.properties.min.js}"></script>

<script th:inline="javascript">
    //获取应用路径
    var ROOT = [[${#servletContext.contextPath}]];

    //获取默认语言
    var LANG_COUNTRY = [[${#locale.language+'_'+#locale.country}]];

    //初始化i18n插件
    $.i18n.properties({
        path: ROOT + '/i18n/',//这里表示访问路径
        name: 'messages',//文件名开头
        language: LANG_COUNTRY,//文件名语言 例如en_US
        mode: 'map'//默认值
    });

    //初始化i18n函数
    function i18n(msgKey) {
        try {
            return $.i18n.prop(msgKey);
        } catch (e) {
            return msgKey;
        }
    }

    //获取国际化翻译值
    console.log(i18n('user.login.username'));
    console.log(i18n('user.login.password'));
    console.log(i18n('user.login.code'));
    console.log(i18n('user.login.remember'));
    console.log(i18n('user.login.submit'));
</script>
```

7、界面定义切换语言

```
<a href="?lang=en_US"> 英语 </a>
<a href="?lang=zh_CN"> 中文 </a>
```

前端组件

若依封装了一些常用的JS组件方法。

名称	代码	介绍
表格	$.table	表格封装处理
表格树	$.treeTable	表格树封装处理
表单	$.form	表单封装处理
弹出层	$.modal	弹出层封装处理
操作	$.operate	操作封装处理
校验	$.validate	校验封装处理
树插件	$.tree	树插件封装处理
通用方法	$.common	通用方法封装处理

通用方法

支持属性

方法	参数	介绍
$.table.init();	options（选项参数）	初始化表格参数
$.table.search();	formId（表单ID）	搜索-默认第一个form
$.table.exportExcel();	formId（表单ID）	导出-默认第一个form
$.table.importExcel();	formId（表单ID）	导入-默认importForm
$.table.importTemplate();	formId（表单ID）	模板下载
$.table.refresh();	无	刷新表格
$.table.selectColumns();	column（查询列值）	查询表格指定列值
$.table.selectFirstColumns();	无	查询表格首列值
$.table.destroy();	tableId（表格ID）	销毁表格-默认options.id
$.table.serialNumber();	index（序号）	序列号生成
$.table.dropdownToggle();	value（内容）	下拉按钮切换
$.table.imageView();	value（内容），path（路径），target（打开方式）	图片预览
$.table.showColumn();	column（列值）	显示表格特定的列
$.table.hideColumn();	column（列值）	隐藏表格特定的列
$.table.tooltip();	value（内容），length（截取长度）	超出指定长度浮动提示
$.table.selectDictLabel();	datas（字典列表），value（当前值）	回显数据字典
$.treeTable.init();	options（选项参数）	初始化表格树参数
$.treeTable.search();	formId（表单ID）	搜索-默认第一个form
$.treeTable.refresh();	无	刷新表格树
$.form.reset();	formId（表单ID）	表单重置
$.form.selectCheckeds();	name（name名称）	获取选中复选框项
$.form.selectSelects();	name（id名称）	获取选中下拉框项
$.modal.icon	type（图标类型）	显示图标
$.modal.msg	content（内容），type（图标类型）	消息提示
$.modal.msgError();	content（内容）	错误消息
$.modal.msgSuccess();	content（内容）	成功消息
$.modal.msgWarning();	content（内容）	警告消息
$.modal.alert	content（内容），type（图标类型）	消息提示

$.modal.msgReload	msg（消息）, type（图标类型）	消息提示并刷新父窗体
$.modal.alertError();	content（内容）	错误提示
$.modal.alertSuccess();	content（内容）	成功提示
$.modal.alertWarning();	content（内容）	警告提示
$.modal.close();	无	关闭窗体
$.modal.closeAll	无	关闭全部窗体
$.modal.confirm();	content（内容）, callBack（回调函数）	确认窗体
$.modal.open();	title, url, width, height, callBack（回调函数）	弹出层指定宽度
$.modal.openOptions();	options（选项参数）	弹出层指定参数选项
$.modal.openFull();	title, url, width, height	弹出层全屏
$.modal.openTab	title（标题）, url（地址）	选卡页方式打开
$.modal.disable	无	禁用按钮
$.modal.enable	无	启用按钮
$.modal.loading();	message（提示消息）	打开遮罩层
$.modal.closeLoading();	无	关闭遮罩层
$.modal.reload();	无	重新加载
$.operate.submit();	url, type, dataType, data, callback（回调函数）	提交数据
$.operate.post();	url（地址）, data（数据）, callback（回调函数）	post方式请求提交数据
$.operate.get();	url（地址）, callback（回调函数）	get请求传输数据
$.operate.detail();	id（数据ID）	详细信息
$.operate.remove();	id（数据ID）	删除信息
$.operate.removeAll();	无	批量删除信息
$.operate.clean();	无	清空信息
$.operate.add();	id（数据ID）	添加信息
$.operate.addTab();	id（数据ID）	添加信息（选项卡方式）
$.operate.addFull();	id（数据ID）	添加信息 全屏
$.operate.addUrl();	id（数据ID）	添加访问地址
$.operate.edit();	id（数据ID）	修改信息
$.operate.editTab();	id（数据ID）	修改信息（选项卡方式）
$.operate.editFull();	id（数据ID）	修改信息 全屏
$.operate.editUrl();	id（数据ID）	修改访问地址
$.operate.save();	url（地址）, data（数据）, callback（回调函数）	保存信息

$.operate.saveModal();	url（地址）, data（数据）, callback（回调函数）	保存信息 弹出提示框
$.operate.saveTab();	url（地址）, data（数据）, callback（回调函数）	保存选项卡信息
$.operate.ajaxSuccess();	result（返回结果）	保存结果弹出msg刷新table表格
$.operate.saveSuccess();	result（返回结果）	保存结果提示msg
$.operate.successCallback();	result（返回结果）	成功回调执行事件（静默更新）
$.operate.successTabCallback();	result（返回结果）	选项卡成功回调执行事件（静默更新）
$.validate.unique();	value（返回标识）	判断返回标识是否唯一
$.validate.form();	formId（表单ID）	表单验证-默认第一个form
$.tree.init();	options（选项参数）	初始化树结构
$.tree.searchNode();	无	搜索节点
$.tree.selectByIdName();	treeId, treeName, node	根据Id和Name选中指定节点
$.tree.showAllNode();	nodes（全部节点数据）	显示所有节点
$.tree.hideAllNode();	nodes（全部节点数据）	隐藏所有节点
$.tree.showParent();	treeNode（节点数据）	显示所有父节点
$.tree.showChildren();	treeNode（节点数据）	显示所有孩子节点
$.tree.updateNodes();	nodeList（全部节点数据）	更新节点状态
$.tree.getCheckedNodes();	column（列值）	获取当前被勾选集合
$.tree.notAllowParents();	_tree（树对象）	不允许根父节点选择
$.tree.toggleSearch();	无	隐藏/显示搜索栏
$.tree.collapse();	无	树折叠
$.tree.expand();	无	树展开
$.common.isEmpty();	value（值）	判断字符串是否为空
$.common.isNotEmpty();	value（值）	判断一个字符串是否为非空串
$.common.nullToStr();	value（值）	空对象转字符串
$.common.visible();	value（值）	是否显示数据 为空默认为显示
$.common.trim();	value（值）	空格截取
$.common.random();	min（最小）, max（最大）	指定随机数返回
$.common.startWith();	value（值）, start（开始值）	判断字符串是否是以start开头
$.common.endWith();	value（值）, end（结束值）	判断字符串是否是以end结尾

表格使用

表格组件基于 `bootstrap table` 组件进行封装，轻松实现数据表格。

- 表格初始化 `$.table.init表的各项(Table options )`

参数	类型	默认值	描述
url	String	Null	请求后台的URL
uniqueId	String	Null	指定唯一列属性 配合删除/修改使用 未指定则使用表格行首列
createUrl	String	Null	新增URL 配合使用 $.operate.add(), $.operate.addTab()
updateUrl	String	Null	修改URL 配合使用 $.operate.edit(), $.operate.editTab()
removeUrl	String	Null	删除URL 配合使用 $.operate.remove()
exportUrl	String	Null	导出URL 配合使用 $.table.exportExcel()
importUrl	String	Null	导入URL 配合使用 $.table.importExcel()
detailUrl	String	Null	详细URL 配合使用 $.operate.detail()
cleanUrl	String	Null	清空URL 配合使用 $.operate.clean()
importTemplateUrl	String	Null	模板URL 配合使用 $.table.importTemplate()
height	String	undefined	表格的高度
striped	String	false	是否显示行间隔色
sortName	String	Null	排序列名称
sortOrder	String	Null	排序方式 asc 或者 desc
pagination	Boolean	true	默认为true表格的底部工具栏会显示分页条，设为false不显示
pageSize	int	10	每页的记录行数（*）
pageList	Array	[10, 25, 50]	可供选择的每页的行数
id	String	bootstrap-table	表格ID属性
toolbar	String	toolbar	表格工具栏ID属性
escape	Boolean	false	是否转义HTML字符串
firstLoad	Boolean	true	是否首次请求加载数据，对于数据较大可以配置false
showFooter	Boolean	false	默认为false隐藏表尾，设为true显示

sidePagination	String	server	server启用服务端分页client客户端分页
search	Boolean	true	默认为true显示搜索框功能，设为false隐藏
searchText	String	''	搜索框初始显示的内容，需要启用search设为true
showSearch	Boolean	true	默认为true显示检索信息，设为false隐藏
showPageGo	Boolean	false	默认为false不显示跳转页，设为true显示
showRefresh	Boolean	true	默认为true显示刷新按钮，设为false隐藏
showColumns	Boolean	true	默认为true显示某列下拉菜单，设为false隐藏
showToggle	Boolean	true	默认为true显示视图切换按钮，设为false隐藏
showExport	Boolean	true	默认为true显示导出文件按钮，设为false隐藏
clickToSelect	Boolean	false	默认为false不启用点击选中行，设为true启用
detailView	Boolean	false	是否启用显示细节视图
onClickRow	Function	onClickRow(row, $element)	点击某行触发的事件
onDblClickRow	Function	onDblClickRow(row, $element)	双击某行触发的事件
onClickCell	Function	onClickCell(field, value, row, $element)	单击某格触发的事件
onDblClickCell	Boolean	onDblClickCell(field, value, row, $element)	双击某格触发的事件
rememberSelected	Boolean	false	默认为false不启用翻页记住前面的选择，设为true启用
fixedColumns	Boolean	false	默认为false禁用冻结列，设为true启用冻结列（左侧）
fixedNumber	int	0	冻结列的个数，当fixedColumns设为true有效（左侧）
rightFixedColumns	Boolean	false	默认为false禁用冻结列，设为true启用冻结列（右侧）
rightFixedNumber	int	0	冻结列的个数，当fixedColumns设为true有效（右侧）
onReorderRow	Function	onReorderRow: function (data)	当拖拽结束后处理函数
rowStyle	Function	rowStyle(row, index)	改变某行的格式，需要两个参数：row行的数据index行的索引
params	Array	Null	当请求数据时，你可以通过修改queryParams向服务器发送参数
columns	Array	Null	默认空数组，在JS里面定义 参考列的各项(Column options)

responseHandler	object	responseHandler(res)	在加载服务器发送来的数据之前，处理数据的格式
onLoadSuccess	object	onLoadSuccess(data)	当所有数据被加载时触发处理函数
exportOptions	Array	[0]	前端导出忽略列索引如[0,5,10]
detailFormatter	Function	(index, row, element)	detailView设为true，启用了显示detail view。用于格式化细节视图

列的各项(Column options)

参数	类型	默认值	描述
radio	Boolean	false	默认false不显示radio（单选按钮），设为true则显示，radio宽度是固定的
checkbox	Boolean	false	默认false不显示checkbox（复选框），设为true则显示，checkbox的每列宽度已固定
field	String	Null	是每列的字段名，不是表头所显示的名字，通过这个字段名可以给其赋值，相当于key，表内唯一
title	String	Null	这个是表头所显示的名字，不唯一，如果你喜欢，可以把所有表头都设为相同的名字
titleTooltip	String	true	当悬浮在某控件上，出现提示 - 参考 Bootstrap 提示工具（Tooltip）插件
class	boolean	false	表格列样式
rowspan	Number	true	每格所占的行数
colspan	Number	true	每格所占的列数
align	String	true	每格内数据的对齐方式，有：left（靠左）、right（靠右）、center（居中）
halign	String	true	table header（表头）的对齐方式，有：left（靠左）、right（靠右）、center（居中）
falign	String	true	table footer（表脚，的对齐方式，有：left（靠左）、right（靠右）、center（居中）
valign	String	true	每格数据的对齐方式，有：top（靠上）、middle（居中）、bottom（靠下）
width	Number	Null	每列的宽度。如果没有自定义宽度自适应
sortable	Boolean	false	默认false就默认显示，设为true则会被排序
order	String	asc	默认的排序方式为"asc（升序）"，也可以设为"desc（降序）"。
visible	Boolean	true	默认为true显示该列，设为false则隐藏该列
cardVisible	Boolean	true	默认为true显示该列，设为false则隐藏。
switchable	Boolean	true	默认为true显示该列，设为false则禁用列项目的选项卡。
clickToSelect	Boolean	true	默认true不响应，设为false则当点击此行的某处时，不会自动选中此行的checkbox（复选框）或radiobox（单选按钮）
formatter	Function	Null	某格的数据转换函数，需要三个参数： -value: field（字段名） -row: 行的数据 -index: 行的（索引）index
			某格的数据转换函数，需要一个参数： -data: 所有行数据

footerFormatter	Function	Null	的数组 函数需要返回（return）footer某格内所要显示的字符串的格式
events	Object	Null	当某格使用formatter函数时，事件监听会响应，需要四个参数： -event: -value: 字段名 -row: 行数据 -index: 此行的index
sorter	Function	Null	自定义的排序函数，实现本地排序，需要两个参数： - a: 第一个字段名 - b: 第二个字段名
sortName	String	Null	排序列名称
cellStyle	Function	Null	对某列中显示样式改变
searchable	Boolean	true	默认true，表示此列数据可被查询
searchFormatter	Boolean	true	默认true，可使用格式化的数据查询
escape	Boolean	false	是否转义HTML字符串

- 表单搜索 $.table.search

```
<a onclick="$.table.search();">搜索</a>
```

- 表格数据导出 $.table.exportExcel

```
<a onclick="$.table.exportExcel();">导出</a>
```

- 数据模板下载 $.table.importTemplate

```
<a onclick="$.table.importTemplate();">下载模板</a>
```

- 表格数据导入 $.table.importExcel

```
<a onclick="$.table.importExcel();">导入</a>
<form id="importForm" enctype="multipart/form-data" class="mt20 mb10" style="display: none;">
  <div class="col-xs-offset-1">
      <input type="file" id="file" name="file"/>
      <div class="mt10 pt5">
          <input type="checkbox" id="updateSupport" name="updateSupport" title="如果登录账户已经存在，更新这条数据。"> 是否更新已经存在的用户数据
                <a onclick="$.table.importTemplate()" class="btn btn-default btn-xs"><i class="fa fa-file-excel-o"></i> 下载模板</a>
      </div>
      <font color="red" class="pull-left mt10">
          提示：仅允许导入“xls”或“xlsx”格式文件！
      </font>
  </div>
</form>
```

- 表格销毁 $.table.destroy

```
<a onclick="$.table.destroy();">销毁</a>
```

- 表格数据刷新 $.table.refresh

```
<a onclick="$.table.refresh();">刷新</a>
```

- 选择表格行具体列 $.table.selectColumns

```
var loginName = $.table.selectColumns("loginName");
```

- 选择表格行首列 $.table.selectFirstColumns

```
var firstColumn = $.table.selectFirstColumns();
```

- 显示表格特定的列 $.table.showColumn

```
$.table.showColumn("userName");
```

- 隐藏表格特定的列 $.table.hideColumn

```
$.table.hideColumn("userName");
```

- 序列号生成 $.table.serialNumber

```
{
  title: "序号",
  formatter: function (value, row, index) {
      return $.table.serialNumber(index);
  }
},
```

- 超出指定长度浮动提示（单击文本可复制） $.table.tooltip

```
{
  field: 'remark',
  title: '备注',
  align: 'center',
  formatter: function(value, row, index) {
      return $.table.tooltip(value);
  }
},
```

- 回显数据字典 $.table.selectDictLabel

```
var datas = [[${@dict.getType('sys_common_status')}]];
{
  field: 'status',
  title: '用户状态',
  align: 'center',
  formatter: function(value, row, index) {
      return $.table.selectDictLabel(datas, value);
  }
},
```

- 下拉按钮切换 $.table.dropdownToggle

```
formatter: function(value, row, index) {
  var actions = [];
  actions.push('<a class="' + editFlag + '" href="#" onclick="$.operate.edit(\'' + row.deptId + '\')"><i class="fa fa-edit"></i>编辑</a>');
  actions.push('<a class="' + removeFlag + '" href="#" onclick="$.operate.remove(\'' + row.deptId + '\')"><i class="fa fa-trash"></i>删除</a>');
  actions.push('<a class="' + addFlag + '" href="#" onclick="$.operate.add(\'' + row.deptId + '\')"><i class="fa fa-plus"></i>添加下级部门</a>');
  return $.table.dropdownToggle(actions.join(''));
}
```

- 图片预览 $.table.imageView

```
{
  field: 'avatar',
  title: '用户头像',
  formatter: function(value, row, index) {
      return $.table.imageView(value);
  }
},
```

弹层使用

弹层组件目前基于 `layer` 组件进行封装，提供了弹出、消息、提示、确认、遮罩处理等功能。

- 提供成功、警告和错误等反馈信息

```
$.modal.msg("默认反馈");
$.modal.msgError("错误反馈");
$.modal.msgSuccess("成功反馈");
$.modal.msgWarning("警告反馈");
```

- 提供成功、警告和错误等提示信息

```
$.modal.alert("默认提示");
$.modal.alertError("错误提示");
$.modal.alertSuccess("成功提示");
$.modal.alertWarning("警告提示");
$.modal.confirm("确认信息", function() {});
```

- 提供弹出层信息

```
// title 标题 url 请求链接 width 宽度 height 高度 options 选项
$.modal.open(title, url, width, height);
$.modal.openTab(title, url);
$.modal.openOptions(options);
$.modal.openFull(title, url, width, height);
$.modal.close();
$.modal.closeAll();
$.modal.reload();
```

- 提供遮罩层信息

```
$.modal.loading("正在导出数据，请稍后...");
$.modal.closeLoading();
```

权限使用

使用thymeleaf模板整合了shiro标签 - 界面可以直接使用。（此处简单介绍两个，更多请参考官方文档）

```
<a href="#" shiro:hasPermission="system:user:add">包含权限字符串才能看到</a>
<a href="#" shiro:hasRole="admin">管理员才能看到</a>
```

如果需要在JS中使用权限，使用封装方法

```
var addFlag = [[${@permission.hasPermi('system:user:add')}]];
var removeFlag = [[${@permission.hasRole('admin')}]];
<a class="btn btn-success btn-xs ' + editFlag + '">包含权限字符串才能看到</a>
<a class="btn btn-danger btn-xs ' + removeFlag + '">管理员才能看到</a>
```

字典使用

配置好相关的数据字典信息即可正常使用（系统管理-字典管理）

```
<select name="status" th:with="type=${@dict.getType('sys_normal_disable')}">
  <option value="">所有</option>
  <option th:each="dict : ${type}" th:text="${dict.dictLabel}" th:value="${dict.dictValue}">
  </option>
</select>
```

如果在想Table表格数据使用字典，使用formatter格式化

```
// 获取数据字典数据
var datas = [[${@dict.getType('sys_normal_disable')}]];

// 格式化数据字典
formatter: function(value, row, index) {
    return $.table.selectDictLabel(datas, value);
}
```

参数使用

配置好相关的参数信息即可正常使用（系统管理-参数管理）

```
<body th:classappend="${@config.getKey('sys.index.skinName')}">
```

如果需要在JS中使用参数，使用封装方法

```
var skinName = [[${@config.getKey('sys.index.skinName')}]];
$("#id").val(skinName);
```

- bootstrap-datetimepicker
- laydate
- layer
- bootstrap-select
- select2
- jasny-bootstrap
- bootstrap-fileinput
- bootstrap-duallistbox
- jquery-validate
- bootstrap-suggest
- bootstrap-typeahead

bootstrap-datetimepicker

bootstrap-datetimepicker是一款时间插件（已经汉化）

代码演示参考 若依系统 → 实例演示 → 表单元素 → 日期与时间 `form/datetime.html` （项目使用需要引入css和js）

1、<th:block th:include="include :: datetimepicker-css" />2、<th:block th:include="include :: datetimepicker-js" />

属性	默认值	描述	备注
format	mm/dd/yyyy	日期格式	任意时间日期格式组合搭配，满足不同需求 yyyy mm dd hh ii ss
weekStart	0	一周从哪一天开始	0（星期日）到6（星期六）
startDate	无	开始时间	可以选择的最早日期，将禁用所有较早日期
endDate	无	结束时间	可以选择的最晚日期，所有较迟的日期都将被禁用
daysOfWeekDisabled	[]	每周禁用一天	
autoclose	false	当选择一个日期之后是否立即关闭此日期时间选择器。	
startView	2	日期时间选择器打开之后首先显示的视图	0 小时 1 天 2 月 3 年 4 十年
minView	0	日期时间选择器所能够提供的最精确的时间选择视图	0 小时 1 天 2 月 3 年 4 十年
maxView	4	日期时间选择器最高能展示的选择范围视图	0 小时 1 天 2 月 3 年 4 十年
todayBtn	false	是否显示当前日期（今天）按钮	
todayHighlight	false	是否高亮当前日期	
keyboardNavigation	true	是否启用键盘方向键选择改变日期	
language	en	语言	zh-cn 中文 en 英文
forceParse	true	强制解析	当选择器关闭的时候，是否强制解析输入框中的值
minuteStep	5	分钟选择视图，每5分钟一个间隔选择	只有minView设置 支持分钟，才能看到
pickerReferer	default		没有特殊要求，无序设置
pickerPosition	bottom-right	时间选择器窗口的位置	bottom-left左下 bottom-right右下 top-left左上 top-right左下
viewSelect	取minView的值	视图选择	decade year month day hour

showMeridian	false	是否为日视图和小时视图启用子午线视图	
initialDate	new Date()	初始日期。	默认是现在，您可以指定昨天或今天……

laydate

laydate是一款时间插件

代码演示参考 若依系统 → 实例演示 → 表单元素 → 日期与时间 `form/datetime.html`

属性	默认值	描述	备注
elem	无	绑定元素	必填项，用于绑定执行日期渲染的元素，值一般为选择器，或DOM对象
type	date	控件选择类型	用于单独提供不同的选择器类型 year年 month年月 date年月日 time时分秒 datetime年月日时分秒
range	false	开启左右面板范围选择	如果设置 true，将默认采用 “ - ” 分割。 你也可以直接设置 分割字符 range: '~'
format	yyyy-MM-dd	自定义格式	通过日期时间各自的格式符和长度，来设定一个你所需要的日期格式
value	new Date()	初始值	支持传入符合format参数设定的日期格式字符，或者 new Date()
isInitValue	true	初始值填充	用于控制是否自动向元素填充初始值（需配合 value 参数使用）
min	1900-1-1	最小范围内的日期时间值	设定最小日期或时间值，不在范围内的将不可选中
max	2099-12-31	最大范围内的日期时间值	设定最大日期或时间值，不在范围内的将不可选中
trigger	focus	自定义弹出控件的事件	如果绑定的元素非输入框，则默认事件为: click
show	false	默认显示	则控件默认显示在绑定元素的区域。通常用于外部事件调用控件
position	absolute	定位方式	用于设定控件的定位方式，有以下三种可选值 abolute绝对定位 fixed固定定位 static静态定位
zIndex	66666666	层叠顺序	一般用于解决与其它元素的互相被遮掩的问题。如果 position 参数设为 static 时，该参数无效
showBottom	true	是否显示底部栏	如果设置 false，将不会显示控件的底部栏区域
btns	['clear', 'now', 'confirm']	工具按钮	右下角显示的按钮，会按照数组顺序排列，内置可识别的值有: clear、now、confirm
lang	cn	语言	两种语言版本: cn（中文版）、en（国际版，即英文版）
theme	default	主题	内置了多种主题，theme的可选值有: default（默认简约）、molv（墨绿背景）、#颜色值（自定义颜色背景）、grid（格子主题）
calendar	false	是否显示公历节日	内置了一些我国通用的公历重要节日，通过设置 true 来开启。国际版不会显示。
		标注重要	calendar 参数所代表的公历节日更多情况下是一个摆设。

mark	无	日子	因此，我们还需要自定义标注重要日子，比如结婚纪念日 日程等

layer

layer是一款web弹层组

代码演示参考 若依系统 → 实例演示 → 表单元素 → 日期与时间 `modal/layer.html`

属性	默认值	描述	备注
type	0	基本层类型	layer提供了5种层类型。可传入的值有：0（信息框，默认）1（页面层）2（iframe层）3（加载层）4（tips层）。 若你采用layer.open({type: 1})方式调用，则type为必填项（信息框除外）
title	'信息'	标题	title支持三种类型的值，若你传入的是普通的字符串，那么只会改变标题文本；若你还需要自定义标题区域样式，那么你可以title: ['文本', 'font-size:18px;']，如果你不想显示标题栏，你可以title: false
content	''	内容	content可传入的值是灵活多变的，不仅可以传入普通的html内容，还可以指定DOM，更可以随着type的不同而不同。
skin	''	样式类名	skin不仅允许你传入layer内置的样式class名，还可以传入您自定义的class名。意味着你可以借助skin轻松完成不同的风格定制。目前layer内置的skin有：layui-layer-lan layui-layer-molv
area	'auto'	宽高	在默认状态下，layer是宽高都自适应的，但当你只想定义宽度时，你可以area: '500px'，高度仍然是自适应的。当你宽高都要定义时，你可以area: ['500px', '300px']
offset	垂直水平居中	坐标	offset默认情况下不用设置。但如果你不想垂直水平居中 offset: ['100px', '50px']同时定义top、left坐标
icon	-1（信息框）	图标	息框默认不显示图标。当你想显示图标时，默认皮肤可以传入0-6如果是加载层，可以传入0-2
btn	'确认'	按钮	信息框模式时，btn默认是一个确认按钮，其它层类型则默认不显示，加载层和tips层则无效。当您只想自定义一个按钮时，你可以btn: '我知道了'，当你要定义两个按钮时，你可以btn: ['yes', 'no']。
btnAlign	r	按钮排列	你可以快捷定义按钮的排列位置，btnAlign的默认值为r，即右对齐。btnAlign: 'l' 按钮左对齐 btnAlign: 'c' 按钮居中对齐 btnAlign: 'r' 按钮右对齐。默认值，不用设置
closeBtn	1	关闭按钮	layer提供了两种风格的关闭按钮，可通过配置1和2来展示，如果不显示，则closeBtn: 0
shade	0.3	遮罩	即弹层外区域。默认是0.3透明度的黑色背景（'#000'）。如果你想定义别的颜色，可以shade: [0.8, '#393D49']；如果你不想显示遮罩，可以shade: 0
shadeClose	false	是否点击遮罩关闭	如果你的shade是存在的，那么你可以设定shadeClose来控制点击弹层外区域关闭。
time	0	自动关闭所需毫秒	默认不会自动关闭。当你想自动关闭时，可以time: 5000，即代表5秒后自动关闭，注意单位是毫秒（1秒=1000毫秒）
id	''	用于控制弹层唯一	设置该值后，不管是什么类型的层，都只允许同时弹出一个。一般用于页面层和iframe层模式

		标识	
anim	0	弹出动画	目前anim可支持的动画类型有0-6 如果不想显示动画，设置 anim: -1 即可。（0平滑放大）（1从上掉落） （2从最底部往上滑入）（3从左滑入）（4从左翻滚） （5渐显）（6抖动）
isOutAnim	true	关闭动画	默认情况下，关闭层时会有一个过度动画。如果你不想开启，设置 isOutAnim: false 即可
maxmin	false	最大最小化	该参数值对type:1和type:2有效。默认不显示最大小化按钮。需要显示配置maxmin: true即可
fixed	true	固定	即鼠标滚动时，层是否固定在可视区域。如果不想，设置fixed: false即可
resize	true	是否允许拉伸	默认情况下，你可以在弹层右下角拖动来拉伸尺寸。如果对指定的弹层屏蔽该功能，设置 false即可。该参数对loading、tips层无效
resizing	null	监听窗口拉伸动作	当你拖拽弹层右下角对窗体进行尺寸调整时，如果你设定了该回调，则会执行。回调返回一个参数：当前层的DOM对象
scrollbar	true	是否允许浏览器出现滚动条	默认允许浏览器滚动，如果设定scrollbar: false，则屏蔽
maxWidth	360	最大宽度	请注意：只有当area: 'auto'时，maxWidth的设定才有效。
maxHeight	无	最大高度	请注意：只有当高度自适应时，maxHeight的设定才有效。
zIndex	19891014	层叠顺序	一般用于解决和其它组件的层叠冲突。
move	'.layui-layer-title'	触发拖动的元素	默认是触发标题区域拖拽。如果你想单独定义，指向元素的选择器或者DOM即可。如move: '.mine-move'。你还配置设定move: false来禁止拖拽
moveOut	false	是否允许拖拽到窗口外	默认只能在窗口内拖拽，如果你想让拖到窗外，那么设定moveOut: true即可
moveEnd	null	拖动完毕后的回调方法	默认不会触发moveEnd，如果你需要，设定moveEnd: function(layero){}即可。其中layero为当前层的DOM对象
tips	2	方向和颜色	tips层的私有参数。支持上右下左四个方向，通过1-4进行方向设定。如tips: 3则表示在元素的下面出现。有时你还可能会定义一些颜色，可以设定tips: [1, '#c00']
tipsMore	false	是否允许多个tips	允许多个意味着不会销毁之前的tips层。通过tipsMore: true开启
		层弹出后	

success	null	的成功回调方法	当你需要在层创建完毕时即执行一些语句，可以通过该回调。success会携带两个参数，分别是当前层DOM当前层索引。
yes	null	确定按钮回调方法	该回调携带两个参数，分别为当前层索引、当前层DOM对象。
cancel	null	右上角关闭按钮触发的回调	该回调携带两个参数，分别为：当前层索引参数（index）、当前层的DOM对象（layero），默认会自动触发关闭。如果不想关闭，return false即可
end	null	层销毁后触发的回调	无论是确认还是取消，只要层被销毁了，end都会执行，不携带任何参数。
full	null	最大化	最大化触发的回调
min	null	最小化	最小化触发的回调
restore	null	还原后	还原后触发的回调

内置方法。

名称	描述	备注
layer.ready(callback)	初始化就绪	由于我们的layer内置了轻量级加载器，所以你根本不需要单独引入css等文件。但是加载总是需要过程的。当你在页面一打开就要执行弹层时，你最好是将弹层放入ready方法中
layer.open(options)	原始核心方法	基本上是露脸率最高的方法，不管是使用哪种方式创建层，都是走layer.open()，创建任何类型的弹层都会返回一个当前层索引，上述的options即是基础参数
layer.alert(content, options, yes)	普通信息框	它的弹出似乎显得有些高调，一般用于对用户造成比较强烈的关注，类似系统alert，但却比alert更灵便。它的参数是自动向左补齐的。通过第二个参数，可以设定各种你所需要的基础参数，但如果你不需要的话，直接写回调即可。
layer.confirm(content, options, yes, cancel)	询问框	请求远程校验。url 类似系统confirm，但却远胜confirm，另外它不是和系统的confirm一样阻塞你需要把交互的语句放在回调体中。同样的，它的参数也是自动补齐的。
layer.msg(content, options, end)	提示框	消息提示
layer.load(icon, options)	加载层	type:3的深度定制。load并不需要你传太多的参数，但如果你不喜欢默认的加载风格，你还有选择空间。icon支持传入0-2如果是0，无需传。另外特别注意一点：load默认是不会自动关闭的，因为你一般会在ajax回调体中关闭它。
layer.tips(content, follow,	tips层	type:4的深度定制。也是我本人比较喜欢的一个层类型，因为它拥有和msg一样的低调和自觉，而且会智

options)	tips层	能定位，即灵活地判断它应该出现在哪边。默认是在元素右边弹出
layer.close(index)	关闭特定层	关于它似乎没有太多介绍的必要，唯一让你疑惑的，可能就是这个index了吧。事实上它非常容易得到。
layer.closeAll(type)	关闭所有层	如果你很懒，你不想去获取index你只想关闭。那么closeAll真的可以帮上你。如果你不指向层类型的话，它会销毁掉当前页所有的layer层。
layer.style(index, cssStyle)	重新定义层的样式	该方法对loading层和tips层无效。参数index为层的索引，cssStyle允许你传入任意的css属性
layer.title(title, index)	改变层的标题	使用方式：layer.title('标题变了', index)
layer.getChildFrame(selector, index)	获取iframe页的DOM	当你试图在当前页获取iframe页的DOM元素时，你可以用此方法。selector即iframe页的选择器
layer.getFrameIndex(windowName)	获取特定iframe层的索引	此方法一般用于在iframe页关闭自身时用到。
layer.iframeAuto(index)	指定iframe层自适应	调用该方法时，iframe层的高度会重新进行适应
layer.iframeSrc(index, url)	重置特定iframe url	似乎不怎么常用的样子。使用方式：layer.iframeSrc(index, 'http://ruoyi.vip')
layer.setTop(layero)	置顶当前窗口	非常强大的一个方法，虽然一般很少用。但是当你的页面有很多很多layer窗口，你需要像Window窗体那样，点击某个窗口，该窗体就置顶在上面，那么setTop可以来轻松实现。它采用巧妙的逻辑，以使这种置顶的性能达到最优
layer.full()	最大化	一般用于在自定义元素上触发最大化。
layer.min()	最小化	一般用于在自定义元素上触发最大小。
layer.restore()	还原后	一般用于在自定义元素上触发还原后。
layer.prompt(options, yes)	输入层	prompt的参数也是向前补齐的。options不仅可支持传入基础参数，还可以传入prompt专用的属性。当然，也可以不传。yes携带value 表单值index 索引elem 表单元素
layer.tab(options)	tab层	tab层只单独定制了一个成员，即tab: [],
layer.photos(options)	相册层	相册层，也可以称之为图片查看器。它的出场动画从layer内置的动画类型中随机展现。photos支持传入json和直接读取页面图片两种方式。

bootstrap-select

bootstrap.select.js是一款下拉框插件

代码演示参考 若依系统 → 实例演示 → 表单元素 → 下拉框 `form/select.html` （项目使用需要引入css和js）

1、<th:block th:include="include :: bootstrap-select-css" />2、<th:block th:include="include :: bootstrap-select-js" />

属性	类型	默认值	描述
actionsBox	bool	false	当设置为true，增加了两个按钮，下拉菜单的顶部（全选和取消全选）
container	string	false	当设置为一个字符string，追加选择一个特定的元素或选择器，例如 container: 'body'
countSelectedText	string	function	设置当selectedTextFormat是显示文本的格式count或count > #。{0} 是所选择的量。{1}是用于选择的总可用。当设定为一个函数，第一个参数是所选择的选项的数目，并且第二个是选项的总数。该函数必须返回一个字符string
deselectAllText	string	'Deselect All'	当取消选择所有选项按钮上的文本actionsBox被启用
dropdownAlignRight	bool 'auto'	false	对齐菜单，而不是左右。如果设置为'auto'，如果在左对齐没有余地菜单的全宽度的菜单会自动右对齐
dropupAuto	bool	true	进行检查以查看其具有更多的空间，上方或下方。如果dropup有足够的空间完全打开正常，但上面有更大的空间，在dropup仍能正常打开。否则，就变成了dropup。如果dropupAuto设置为false，dropups必须手动调用
header	string	false	增加了菜单的顶部的头部；默认包含关闭按钮
hideDisabled	bool	false	从菜单中删除禁用的选项和optgroups data-hide-disabled: true
iconBase	string	'glyphicon'	将基地使用不同的图标字体代替Glyphicons。如果改变iconBase，你也可能要更改tickIcon，以防新的图标字体使用了不同的命名方案
liveSearch	bool	false	当设置为true，增加了一个搜索框的下拉selectpicker的顶部
liveSearchNormalize	bool	false	设置liveSearchNormalize以true允许不区分重音的搜索
liveSearchPlaceholder	string	null	当设置为一个字符string，一个占位符属性等于该字符string将被添加到实况搜索输入
liveSearchStyle	string	'contains'	当设置为'contains'，搜索将显示包含搜索到的文本选项。例如，搜索，返回鸭都为PL PL E，PL嗯，和PL antain。当设置为'startsWith'，寻找PL只会返回PL UM和PL antain

maxOptions	integer	false	当设置为一个integer ，并在多选择，所选选项的数量不能超过给定值。该选项还可以存在作为数据属性为optgroup，在这种情况下，它仅适用于optgroup
maxOptionsText	string	function	启用maxOptions时所显示的文本，并为给定的方案选项的最大数量已被选定。如果使用的功能，它必须返回一个数组。阵列[0]是当maxOptions被施加到整个选择元件使用的文本。阵列[1]是当maxOptions上的OPTGROUP用于使用的文本。如果使用字符string，相同的文字用于元素和OPTGROUP两者
mobile	bool	false	当设置为true，使能选择菜单中的设备的本机菜单
multipleSeparator	string	', '	坐落在分隔所选选项的按钮显示的字符
noneSelectedText	string	'Nothing selected'	当多个选择时所显示的文本没有选择的选择
selectAllText	string	'Select All'	当选择了所有选项，按钮上的文本actionsBox被启用
selectedTextFormat	'values' 'static' 'count'	'values'	指定选择如何显示有多个选择。'values'显示所选择的选项（由分隔的列表multipleSeparator。'static'简单地显示所述选择元件的标题。'count'显示所选选项的总数量。'count > x'行为类似于'values'直到所选选项的数目大于x;在此之后，它的行为象'count'
selectOnTab	bool	false	当设置为true，对待像selectpicker下拉列表中输入或空格字符制表符
showContent	bool	true	当设置为true，显示与该按钮选择的选项（一个或多个）相关联的自定义的HTML。当设置为false，期权价值将被显示
showIcon	bool	true	当设置为true，与在按钮选择的选项（一个或多个）相关联的显示的图标（一个或多个）
showSubtext	bool	false	当设置为true与所述按钮选择的选项相关联，显示潜台词
showTick	bool	false	show（没有的项目上选择的选项勾选multiple属性）
size	'auto' integer false	'auto'	当设置为'auto'，菜单始终打开，以显示尽可能多的项目窗口将允许在没有被切断。当设置为integer 时，菜单将显示项目的给定数量，即使下拉被切断。当设置为false，菜单会一直显示所有项目
style	string null	null	当设置为一个字符string，添加值到该按钮的风格
tickIcon	string null	'glyphicon-ok'	设置要使用的图标旁边显示的“滴答”来选择的选项
title	string	null	null
width	'auto' 'fit' css-width false	false	当设置为auto，所述selectpicker的宽度被自动调节，以适应最宽的选项。当设置为一个css-宽度，所述selectpicker的宽度内联强制为给定值。当设置为false，所有宽度信息被删除

windowPadding	integer array	0	这是在该窗口中有一个下拉菜单中不应该涉及的领域情况下非常有用-例如一个固定的头。当设置为一个integer ，同样填充将被添加到四面八方。可替代地，一个integer 数组可以在格式来使用[top, right, bottom, left]

```
// selectpicker 常用方法
$('#id').selectpicker('val');                       // 取值
$('#id').selectpicker('val', 'RuoYi');              // 单个赋值
$('#id').selectpicker('val', ['Admin','RuoYi']);    // 多选赋值
$('#id').selectpicker('selectAll');                 // 全选
$('#id').selectpicker('deselectAll');               // 反选
$('#id').selectpicker('render');                    // 渲染
$('#id').selectpicker('refresh');                   // 刷新
$('#id').prop('disabled', true);                    // 禁用
$('#id').prop('disabled', false);                   // 启用
$('#id').selectpicker('toggle');                    // 切换
$('#id').selectpicker('hide');                      // 隐藏
$('#id').selectpicker('show');                      // 显示
$('#id').selectpicker('destroy');                   // 销毁
$('#id').selectpicker('mobile');                    // 适应手机模式
<!--  事件监听
show.bs.select
shown.bs.select
hide.bs.select
hidden.bs.select
loaded.bs.select
rendered.bs.select
refreshed.bs.select
changed.bs.select
-->
$('#id').on('changed.bs.select', function (e, clickedIndex, isSelected, previousValue) {
  // 处理自己的业务
});
```

select2

select2.js是一款下拉框插件

代码演示参考 若依系统 → 实例演示 → 表单元素 → 下拉框 `form/select.html` （项目使用需要引入css和js）

1、<th:block th:include="include :: select2-css" />2、<th:block th:include="include :: select2-js" />

属性	类型	默认值	描述
data	Array	Null	数据集合，基础数据格式{id:"", text:"", selected: true, disabled: true}
width	string	空	宽度
style	string	空	样式
ajax	object	null	Ajax请求数据
minimumResultsForSearch	Integer	null	设置支持搜索的最小集合，设置为负数，隐藏搜索框
minimumInputLength	Integer	空	输入指定长度字符后开始搜索
multiple	boolean	false	是否多选，默认单选
maximumSelectionLength	Integer	空	支持最大的选择数量，int/function
maximumInputLength	Integer	空	支持搜索的最大字符数
placeholder	String	空	选择提示
allowClear	Boolean	false	是否显示清除按钮，只有设置了placeholder才有效
closeOnSelect	Boolean	true	是否选中后关闭选择框，默认true
templateSelection	callback	空	选中项样式
templateResult	callback	空	选项样式
matcher	callback	空	过滤选项集合
sorter	callback	空	选项结果集排序
theme	String	空	主题，可以设置bootstrap主题
tags	Boolean	空	是否可动态创建选项
tokenSeparators	Array	空	输入时使用分隔符创建新选项
createTag	callback	空	创建新标签
insertTag	callback	空	在选项集合后插入标签
disabled	boolean	false	是否失效
debug	boolean	false	是否开启debug

```
// select2 常用方法
$('#id').select2('val');                    // 取值
$("#id").select2("val", ["RuoYi"]);         // 单个赋值
$("#id").val(["RuoYi"]).trigger("change");  // 单个赋值
```

```
$('#id').val(['Admin', 'RuoYi']).trigger("change"); // 多个赋值
$("#id").select2("open");                           // 打开下拉框
$("#id").select2("close");                          // 关闭下拉框
$('#id').prop('disabled', true);                    // 禁用
$('#id').prop('disabled', false);                   // 启用
$('#id').select2('destroy');                        // 销毁
<!--  事件监听
change
change.select2
select2:closing
select2:close
select2:opening
select2:open
select2:selecting
select2:select
select2:unselecting
select2:unselect
-->
$('#id').on('select2:select', function (e) {
  // 处理自己的业务
});
```

jasny-bootstrap

jasny.js是一个功能扩展插件（含文件上传）

代码演示参考 若依系统 → 实例演示 → 表单元素 → 功能扩展 `form/jasny.html` （项目使用需要引入css和js）

1、<th:block th:include="include :: jasny-bootstrap-css" />2、<th:block th:include="include :: jasny-bootstrap-js" />

属性	类型	默认值	描述
name	string	当前元素	使用指定元素设置

```
$('#id').fileinput(options)     // 初始
$('#id').fileinput('clear')     // 清除
$('#id').fileinput('reset')     // 重置

<!--  事件监听
change.bs.fileinput
clear.bs.fileinput
reset.bs.fileinput
-->
$('#id').on('change.bs.fileinput ', function (e) {
  // 处理自己的业务
});
```

bootstrap-fileinput

bootstrap-fileinput.js是一款文件上传插件

代码演示参考 若依系统 → 实例演示 → 表单元素 → 文件上传 `form/upload.html` （项目使用需要引入css和js）

1、<th:block th:include="include :: bootstrap-fileinput-css" />2、<th:block th:include="include :: bootstrap-fileinput-js" />

属性	类型	默认值	描述
language	String	'en'	多语言设置，使用时需入locales文件夹下对言文件，中文zh，引入件必须放在fileinpu后
showCaption	Boolean	true	是否显示被选文件的简
showBrowse	Boolean	true	是否显示浏览按钮
showPreview	Boolean	true	是否显示预览区域
showRemove	Boolean	true	是否显示移除按钮
showUpload	Boolean	true	是否显示上传按钮
showCancel	Boolean	true	是否显示取消按钮
showClose	Boolean	true	是否显示关闭按钮
showUploadedThumbs	Boolean	true	这个属性是基于这样一方法的：选择若干个文击右下角上传按钮批量待全部完成后再选择一件，此时之前上传成功是否要保存。就是这个制的。注意，点击文件下面的上传按钮不会导的成功上传的文件消失这里设置了true
browseOnZoneClick	Boolean	false	
autoReplace	Boolean	false	是否自动替换当前图片为true时，再次选择文会将当前的文件替换掉
generateFileId	Object	null	
previewClass	String	空	添加预览按钮的类属性
captionClass	String	空	添加标题类属性
frameClass	String	'krajee-default'	针对每个缩略图的框架
mainClass	String	'file-caption-main'	针对元素类属性
mainTemplate	Object	null	
purifyHtml	Boolean	true	
fileSizeGetter	Object	null	
initialCaption	String	空	

initialPreview	Array/Object	[]	通过这个参数，我们可 件区设置一些默认的缩
initialPreviewDelimiter	String	'$$'	
initialPreviewAsData	Boolean	false	
initialPreviewFileType	String	'image'	
initialPreviewConfig	Array/Object	[]	
initialPreviewThumbTags	Array/Object	[]	
previewThumbTags	Object	{}	
initialPreviewShowDelete	Boolean	true	
removeFromPreviewOnError	Boolean	false	
deleteUrl	String	空	删除图片时的请求路径
deleteExtraData	Object	{}	删除图片时额外传入的
overwriteInitial	Boolean	true	
previewZoomButtonIcons	Object	{prev: '',next: '',toggleheader: '',fullscreen: '',borderless: '',close: ''},	
previewZoomButtonClasses	Object	{prev: 'btn btn-navigate',next: 'btn btn-navigate',toggleheader: 'btn btn-default btn-header-toggle',fullscreen: 'btn btn-default',borderless: 'btn btn-default',close: 'btn btn-default'},	
preferIconicPreview	Boolean	false	
preferIconicZoomPreview	Boolean	false	
allowedPreviewTypes	undefined	undefined	
allowedPreviewMimeTypes	Object	null	
allowedFileTypes	Object	null	接收的文件后缀，如['gif', 'png'],不 制上传文件后缀类型
allowedFileExtensions	Object	null	指出带有哪些后缀名的 是被接受上传的，设置 msgInvalidFileEx 可以个性化出现此错误 误信息
defaultPreviewContent	Object	null	
customLayoutTags	Object	{}	
customPreviewTags	Object	{}	
previewFileIcon	String	空	当文件无法被预览时出 略图中的图标，默认是

previewFileIconClass	String	'file-other-icon'	
previewFileIconSettings	Object	{}	可以将不同的后缀的文同的缩略图图标
previewFileExtSettings	Object	{}	
buttonLabelClass	String	'hidden-xs'	
browseIcon	String	空	
browseClass	String	'btn btn-primary'	指出了右下角选择按钮式。一般尽量不要用b btn-lg，会和左边的 协调
removeIcon	String	空	删除按钮相关的属性
removeClass	String	'btn btn-default'	
cancelIcon	String	空	
cancelClass	String	'btn btn-default'	
uploadIcon	String	空	上传按钮相关的属性
uploadClass	String	'btn btn-default'	
uploadUrl	String	null	上传文件路径
uploadAsync	boolean	true	是否为异步上传
uploadExtraData	Object	{}	上传文件时额外传递的置
zoomModalHeight	number	480	
minImageWidth	String	null	图片的最小宽度
minImageHeight	String	null	图片的最小高度
maxImageWidth	String	null	图片的最大宽度
maxImageHeight	String	null	图片的最大高度
resizeImage	Boolean	false	
resizePreference	String	'width'	
resizeQuality	number	0.92	
resizeDefaultImageType	String	'image/jpeg'	
minFileSize	number	0	单位为kb，上传文件的小值
maxFileSize	number	0	单位为kb，如果为0表制文件大小
resizeDefaultImageType	number	25600(25MB)	
minFileCount	number	0	表示同时最小上传的文
maxFileCount	number	0	表示允许同时上传的最个数
validateInitialCount	Boolean	false	
msgValidationErrorClass	String	'text-danger'	
msgValidationErrorIcon	String	空	

msgErrorClass	String	'file-error-message'	
progressThumbClass	String	"progress-bar progress-bar-success progress-bar-striped active"	
rogressClass	String	"progress-bar progress-bar-success progress-bar-striped active"	
progressCompleteClass	String	"progress-bar progress-bar-success"	
progressErrorClass	String	"progress-bar progress-bar-danger"	
progressUploadThreshold	number	99	
previewFileType	String	'image'	预览文件类型,内置['image', 'html' 'text', 'video', 'audio', 'flash' 'object',‘other‘
elCaptionContainer	String	null	
elCaptionText	String	null	设置标题栏提示信息
elPreviewContainer	String	null	
elPreviewImage	String	null	
elPreviewStatus	String	null	
elErrorContainer	String	null	
errorCloseButton	String	×'	
slugCallback	function	null	选择后未上传前 回调
dropZoneEnabled	Boolean	true	是否显示拖拽区域
dropZoneTitleClass	String	'file-drop-zone-title'	拖拽区域类属性设置
fileActionSettings	Object	{}	设置预览图片的显示样
otherActionButtons	String	空	编码设置
textEncoding	String	'UTF-8'	
ajaxSettings	Object	{}	
ajaxDeleteSettings	Object	{}	
showAjaxErrorDetails	Boolean	true	

Method说明。

方法名	描述
fileerror	异步上传错误结果处理$('#uploadfile').on('fileerror', function(event, data, msg) {});
fileuploaded	异步上传成功结果处理$("#uploadfile").on("fileuploaded", function (event, data, previewId, index) {})
filebatchuploaderror	批量上传错误结果处理 $('#uploadfile').on('filebatchuploaderror', function(event, data, msg) {});

filebatchuploadsuccess	批量上传成功结果处理$('#uploadfile').on('filepreupload', function(event, data, previewId, index) {});
filebatchselected	选择文件后处理事件$("#fileinput").on("filebatchselected", function(event, files) {});
upload	文件上传方法$("#fileinput").fileinput("upload");
fileuploaded	上传成功后处理方法$("#fileinput").on("fileuploaded", function(event, data, previewId, index) {});
filereset	Possible values: http, https, ws, wss.
fileclear	点击浏览框右上角X 清空文件前响应事件 $("#fileinput").on("fileclear",function(event, data, msg) {});
filecleared	点击浏览框右上角X 清空文件后响应事件 $("#fileinput").on("filecleared",function(event, data, msg){});
fileimageuploaded	在预览框中图片已经完全加载完毕后回调的事件

bootstrap-duallistbox

bootstrap-duallistbox是一款双重列表框插件

代码演示参考 若依系统 → 实例演示 → 表单元素 → 左右互选组件 `form/duallistbox.html` （项目使用需要引入css和js）

1、<th:block th:include="include :: bootstrap-duallistbox-css" />2、<th:block th:include="include :: bootstrap-duallistbox-js" />

属性	默认值	描述
bootstrap2Compatible	false	兼容bootstrap2
filterTextClear	'show all'	清空过滤条件按钮的文本
filterPlaceHolder	'Filter'	过滤条件的input框的placeholder
moveSelectedLabel	'Move selected'	添加选中option按钮的label
moveAllLabel	'Move all'	添加全部option按钮的label
removeSelectedLabel	'Remove selected'	移除选中option按钮的label
removeAllLabel	'Remove all'	移除全部option按钮的label
moveOnSelect	true	是否移动选中的option:为false时,moveSelected和removeSelected的按钮显示生效:默认为true;为true只能光标连续选取,松开鼠标,选中的项会移动;为false则可配合键盘的Ctrl和Shift使用,点击moveSelectedLabel和removeSelectedLabel的按钮, option才会移动
preserveSelectionOnMove	false	'moved'或'all'时，展示移动到target列表中的元素（背景色显示）。配合moveOnSelect为false使用
selectedListLabel	false	已选中list的label
nonSelectedListLabel	false	未选中list的label
helperSelectNamePostfix	'_helper'	为selector的name的后缀为_helper,朱选中的list后面拼接1,已选中的拼接2:也可通过setHelperSelectNamePostfix(value, refresh)方法修改
selectorMinimalHeight	100	selector的最小高度,小于元素的height()则高度的值为height()
showFilterInputs	true	是否显示过滤的nput输入框,默认tue显示;为fase则过滤相关的内容不起作用、不显示
nonSelectedFilter	''	未选中option的过滤条件,默认为空字符串,也可用正则方式,例如:ion([7-9][1]0-2)过滤7、8、9、10、11、12
selectedFilter	''	已选中option的过滤条件,默认为空字符串;参考:nonSelectedFilter;-般不设置已选中的过滤条件,会导致某些选中的项不在已选中option的过滤条件范围内,无法显示
	'Showing	

	all {0}'	
infoTextFiltered	'Filtered {0} from {1}'	过滤信息,默认< span class="label label-Warning">Filtered{0} from {1}。从m项中筛选n项
infoTextEmpty	'Empty list'	当筛选条件为'',且选中/未选中列表无option时显示的内容
filterOnValues	false	过滤选项的值
eventMoveoverride	false	设置为true,可自定义moveSelected的事件
eventMoveAlloverride	false	设置为true,可自定义moveAll的事件
eventRemoveOverride	false	设置为true,可自定义removeSelectecd的事件
eventRemoveAlloverride	false	设置为true,可自定义removeAll的事件

Method说明。

方法名	描述
refresh()	更新UI插件元素
destroy()	恢复原始元素
getContainer()	获取容器元素
setBootstrap2Compatible(value, refresh)	更改 bootstrap2Compatible 参数
setFilterTextClear(value, refresh)	更改 filterTextClear 参数
setFilterPlaceHolder(value, refresh)	更改 filterPlaceHolder 参数
setMoveSelectedLabel(value, refresh)	更改 moveSelectedLabel 参数
setMoveAllLabel(value, refresh)	更改 moveAllLabel 参数
setRemoveSelectedLabel(value, refresh)	更改 removeSelectedLabel 参数
setRemoveAllLabel(value, refresh)	更改 removeAllLabel 参数
setMoveOnSelect(value, refresh)	更改 moveOnSelect 参数
setPreserveSelectionOnMove(value, refresh)	更改 preserveSelectionOnMove 参数
setSelectedListLabel(value, refresh)	更改 selectedListLabel 参数
setNonSelectedListLabel(value, refresh)	更改 nonSelectedListLabel 参数
setHelperSelectNamePostfix(value, refresh)	更改 helperSelectNamePostfix 参数
setSelectOrMinimalHeight(value, refresh)	更改 selectorMinimalHeight 参数
setShowFilterInputs(value, refresh)	更改 showFilterInputs 参数
setNonSelectedFilter(value, refresh)	更改 nonSelectedFilter 参数
setSelectedFilter(value, refresh)	更改 selectedFilter 参数
setInfoText(value, refresh)	更改 infoText 参数
setInfoTextFiltered(value, refresh)	更改 infoTextFiltered 参数
setInfoTextEmpty(value, refresh)	更改 infoTextEmpty 参数
setFilterOnValues(value, refresh)	更改 filterOnValues 参数

jquery-validate

jQuery Validate 插件为表单提供了强大的验证功能，让客户端表单验证变得更简单，同时提供了大量的定制选项，满足应用程序各种需求。该插件捆绑了一套有用的验证方法，包括 URL 和电子邮件验证，同时提供了一个用来编写用户自定义方法的 API。

代码演示参考 若依系统 → 实例演示 → 表单元素 → 表单验证 `form/validate.html`

默认校验规则。

属性	描述
required:true	必须输入的字段
remote:"/action"	使用ajax方法调用action验证输入值
email:true	必须输入正确格式的电子邮件
url:true	必须输入正确格式的网址
date:true	必须输入正确格式的日期。日期校验 ie6 出错，慎用
dateISO:true	必须输入正确格式的日期（ISO），例如：2009-06-23，1998/01/22。只验证格式，不验证有效性
number:true	必须输入合法的数字（负数，小数）
digits:true	必须输入整数
creditcard:	必须输入合法的信用卡号
equalTo:"#field"	输入值必须和 #field 相同
accept:	输入拥有合法后缀名的字符串（上传文件的后缀）
maxlength:5	输入长度最多是 5 的字符串（汉字算一个字符）
minlength:10	输入长度最小是 10 的字符串（汉字算一个字符）
rangelength:[5,10]	输入长度必须介于 5 和 10 之间的字符串（汉字算一个字符）
range:[5,10]	输入值必须介于 5 和 10 之间
max:5	输入值不能大于 5
min:10	输入值不能小于 10
isIp:true	IP地址验证
isPhone:true	手机号码验证
isTel:true	电话号码验证
isName:true	姓名验证
isUserName:true	用户名验证
isIdentity:true	身份证验证
isBirth:true	出生日期验证

内置验证方式。

名称	类型	描述
required()	Boolean	必填验证元素
required(dependency-expression)	Boolean	必填元素依赖于表达式的结果
required(dependency-callback)	Boolean	必填元素依赖于回调函数的结果
remote(url)	Boolean	请求远程校验。url 通常是一个远程调用方法
minlength(length)	Boolean	设置最小长度
maxlength(length)	Boolean	设置最大长度
rangelength(range)	Boolean	设置一个长度范围 [min,max]
min(value)	Boolean	设置最小值
max(value)	Boolean	设置最大值
email()	Boolean	验证电子邮箱格式
range(range)	Boolean	设置值的范围
url()	Boolean	验证 URL 格式
date()	Boolean	验证日期格式（类似 30/30/2008 的格式，不验证日期准确性只验证格式）
dateISO()	Boolean	验证 ISO 类型的日期格式
dateDE()	Boolean	验证德式的日期格式（29.04.1994 或 1.1.2006）
number()	Boolean	验证十进制数字（包括小数的）
digits()	Boolean	验证整数
creditcard()	Boolean	验证信用卡号
accept(extension)	Boolean	验证相同后缀名的字符串
equalTo(other)	Boolean	验证两个输入框的内容是否相同
phoneUS()	Boolean	验证美式的电话号码

验证的触发方式修改。

触发方式	类型	默认值	描述
onsubmit	Boolean	true	提交时验证。设置为 false 就用其他方法去验证
onfocusout	Boolean	true	失去焦点时验证（不包括复选框/单选按钮）
onkeyup	Boolean	true	在 keyup 时验证
onclick	Boolean	true	在点击复选框和单选按钮时验证
focusInvalid	Boolean	true	提交表单后，未通过验证的表单（第一个或提交之前获得焦点的未通过验证的表单）会获得焦点
focusCleanup	Boolean	false	如果是 true 那么当未通过验证的元素获得焦点时，移除错误提示。避免和 focusInvalid 一起用

bootstrap-suggest

bootstrap-suggest这是一个基于bootstrap按钮式下拉菜单组件的搜索建议插件

代码演示参考 若依系统 → 实例演示 → 表单元素 → 搜索自动补全 `form/autocomplete.html` （项目使用需要引入js）

```
<th:block th:include="include :: bootstrap-suggest-js" />
```

属性	默认值	描述
url	null	请求数据的 URL 地址
jsonp	null	设置此参数名，将开启jsonp功能，否则使用json数据结构
data	[]	提示所用的数据，注意格式
indexId	0	每组数据的第几个数据，作为input输入框的data-id，设为 -1 且 idField 为空则不设置此值
indexKey	0	每组数据的第几个数据，作为input输入框的内容
idField	''	每组数据的哪个字段作为 data-id，优先级高于indexId 设置（推荐）
keyField	''	每组数据的哪个字段作为输入框内容，优先级高于indexKey 设置（推荐）
autoSelect	true	键盘向上/下方向键时，是否自动选择值
allowNoKeyword	TRUE	是否允许无关键字时请求数据
getDataMethod	'firstByUrl'	获取数据的方式，url: 一直从url请求；data: 从options.data 获取；firstByUrl: 第一次从Url获取全部数据，之后从options.data获取
delayUntilKeyup	false	获取数据的方式 为 firstByUrl 时，是否延迟到有输入时才请求数据
ignorecase	false	前端搜索匹配时，是否忽略大小写
effectiveFields	[]	有效显示于列表中的字段，非有效字段都会过滤，默认全部有效
effectiveFieldsAlias	{}	有效字段的别名对象，用于 header 的显示
searchFields	[]	有效搜索字段，从前端搜索过滤数据时使用，但不一定显示在列表中。effectiveFields 配置字段也会用于搜索过滤
twoWayMatch	true	是否双向匹配搜索。为 true 即输入关键字包含或包含于匹配字段均认为匹配成功，为 false 则输入关键字包含于匹配字段认为匹配成功
multiWord	false	以分隔符号分割的多关键字支持
separator	','	多关键字支持时的分隔符，默认为半角逗号
delay	300	搜索触发的延时时间间隔，单位毫秒
emptyTip	''	查询为空时显示的内容，可为 html
		ajax 搜索时显示的提示内容，当搜索时间较长时给

		出正在搜索的提示
hideOnSelect	false	鼠标从列表单击选择了值时，是否隐藏选择列表
autoDropup	false	选择菜单是否自动判断向上展开。设为 true，则当下拉菜单高度超过窗体，且向上方向不会被窗体覆盖，则选择菜单向上弹出
autoMinWidth	false	是否自动最小宽度，设为 false 则最小宽度不小于输入框宽度
showHeader	false	是否显示选择列表的 header。为 true 时，有效字段大于一列则显示表头
showBtn	true	是否显示下拉按钮
inputBgColor	''	输入框背景色，当与容器背景色不同时，可能需要该项的配置
inputWarnColor	'rgba(255,0,0,.1)'	输入框内容不是下拉列表选择时的警告色
listStyle	默认参考描述	列表的样式控制 { 'padding-top': 0, 'max-height': '375px', 'max-width': '800px', 'overflow': 'auto', 'width': 'auto', 'transition': '0.3s', '-webkit-transition': '0.3s', '-moz-transition': '0.3s', '-o-transition': '0.3s' }
listAlign	'left'	提示列表对齐位置，left/right/auto
listHoverStyle	'background: #07d; color:#fff'	提示框列表鼠标悬浮的样式
listHoverCSS	'jhover'	提示框列表鼠标悬浮的样式名称
clearable	false	是否可清除已输入的内容
keyLeft	37	向左方向键，不同的操作系统可能会有差别，则自行定义
keyUp	38	向上方向键
keyRight	39	向右方向键
keyDown	40	向下方向键
keyEnter	13	回车键
fnProcessData	processData	格式化数据的方法，返回数据格式参考 data 参数
fnGetData	getData	获取数据的方法，无特殊需求一般不作设置
fnAdjustAjaxParam	null	调整 ajax 请求参数方法，用于更多的请求配置需求。如对请求关键字作进一步处理、修改超时时间等
fnPreprocessKeyword	null	搜索过滤数据前，对输入关键字作进一步处理方法。注意，应返回字符串

```
1. // suggest 常用方法
2. $("input#test").bsSuggest("disable");    // 禁用提示
3. $("input#test").bsSuggest("enable");     // 启用提示
4. $("input#test").bsSuggest("destroy");    // 销毁插件
5. $("input#test").bsSuggest("version");    // 查看版本
6.
7. <!--  事件监听
8. onDataRequestSuccess  // 当 AJAX 请求数据成功时触发，并传回结果到第二个参数
```

```
onSetSelectValue      // 当从下拉菜单选取值时触发，并传回设置的数据到第二个参数
onUnsetSelectValue    // 当设置了 idField，且自由输入内容时触发（与背景警告色显示同步）
onShowDropdown        // 下拉菜单显示时触发
onHideDropdown        // 拉菜单隐藏式触发
-->
$("#test").bsSuggest('init', {
    url: "/rest/sys/getuserlist?keyword=",
    effectiveFields: ["userName", "email"],
    searchFields: [ "shortAccount"],
    effectiveFieldsAlias:{userName: "姓名"},
    clearable: true,
    idField: "userId",
    keyField: "userName"
}).on('onDataRequestSuccess', function (e, result) {
    console.log('onDataRequestSuccess: ', result);
}).on('onSetSelectValue', function (e, selectedData, selectedRawData) {
    console.log('onSetSelectValue: ', e.target.value, selectedData, selectedRawData);
}).on('onUnsetSelectValue', function () {
    console.log('onUnsetSelectValue');
}).on('onShowDropdown', function (e, data) {
    console.log('onShowDropdown', e.target.value, data);
}).on('onHideDropdown', function (e, data) {
    console.log('onHideDropdown', e.target.value, data);
});
```

bootstrap-typeahead

bootstrap-typeahead是一款搜索自动补全插件

代码演示参考 若依系统 → 实例演示 → 表单元素 → 搜索自动补全 `form/autocomplete.html` （项目使用需要引入js）

```
<th:block th:include="include :: bootstrap-typeahead-js" />
```

属性	类型	默认值	描述
source	array, function	[ ]	要查询的数据源。可能是一个字符串数组，一个具有name属性或函数的JSON对象数组。该函数接受两个参数， query 即输入字段中的值和 process 回调。该函数可以通过 process 回调的单个参数直接或异步返回数据源来同步使用。
items	number	8	在下拉列表中显示的最大项目数。也可以设置为“全部”
minLength	number	1	触发自动填充建议之前所需的最小字符长度。您可以将其设置为0，因此即使在调用查找功能时没有文本时也会显示建议。
showHintOnFocus	boolean	false	一旦输入得到焦点，就可以在适用时显示提示。
scrollHeight	number, function	0	可滚动父容器向下滚动的像素数（滚出视口）。
matcher	function	case insensitive	用于确定查询是否匹配项的方法。接受单个参数， item 用于测试查询。访问当前查询 this.query。true如果查询是匹配，则返回一个布尔值。
sorter	function	exact match,case sensitive,case insensitive	用于对自动完成结果进行排序的方法。接受单个参数， items 并具有类型头实例的范围。引用当前查询 this.query.
updater	function	returns selected item	用于返回所选项目的方法。接受单个参数， item 并具有类型头实例的范围。
highlighter	function	highlights all default matches	用于突出显示自动完成结果的方法。接受单个参数， item 并具有类型头实例的范围。应该返回html
displayText	function	item.name item	用于获取源的项目的文本表示的方法。接受单个参数， item 并具有类型头实例的范围。应该返回一个String。
autoSelect	boolean	true	允许您指定是否自动选择第一个建议。关闭自动选择也意味着如果没有选择 enter 或被 tab 击中，输入将不会被清除。
afterSelect	function	$.noop()	调用功能在选择一个项目后执行。它将当前活动项目置于参数中（如果有）。
delay	integer	0	在查找之间增加延迟。
addItem	JSONobject	false	将项目添加到列表的末尾，例如“新建条目”。例如，当在数据列表中找不到某个项目时，可以使用该对话框。

后台扩展

若依收集了一些其他小伙伴的扩展应用，欢迎反馈及分享。

名称	说明	地址
RuoYi-racsu	RuoYi多模块 Oracle版本	https://gitee.com/racsu/RuoYi-Oracle
RuoYi-gzizi	RuoYi多模块 SQLServer版本	https://gitee.com/gzizi/RuoYi-Sqlserver
RuoYi-zhangmrit	集成通用Mapper OSS模块 JWT 多数据源切面	https://gitee.com/zhangmrit/RuoYi
RuoYi-cloud	集成 SpringCloud, ant-design-vue、token、redis	https://gitee.com/zhangmrit/ruoyi-cloud
RuoYi-plus	集成 SpringCloud, config配置中心，使用 tk.mybatis、lombok	https://gitee.com/aimeng2017/RuoYi-plus/tree/master
RuoYi-yangzhengze	集成activiti 工作流	https://gitee.com/yangzhengze/RuoYi/tree/dev2/
RuoYi-lwslws	增加Mina 用户和部门导入 定时备份mysql数据库 邮件发送	https://gitee.com/lwslws/ry_New
RuoYi-duzunwu512	Redis实现Session共享多模块（支持Cacheable缓存）	https://gitee.com/duzunwu512/RuoYi
RuoYi-panda	集成redis-shiro 百度对象存储 JWT openApi 无xml注解	https://gitee.com/happy-panda/RuoYi
RuoYi-sushengbuyu	Mybatis-Plus多模块 Lombok插件（支持代码生成）	https://gitee.com/sushengbuyu/RuoYi
RuoYi-fast-mybatis-plus	Mybatis-Plus单应用（支持代码生成）	https://gitee.com/easy__/RuoYi-fast.git
RuoYi-depending-mp	Mybatis-Plus多模块（支持代码生成）	https://gitee.com/dotstable/depending_on_the_framework
RuoYi-file	新增文件上传修改预览（提取码：rvxs）	https://pan.baidu.com/s/1gtW0X0-8w6topQFNmfyHtw

RuoYi-yuejiu	动态数据源(从数据中查询并动态创建数据源)	https://gitee.com/yuejiu/RuoYi
RuoYi-zouyi	Redis实现Session共享单应用（提取码:erft）	https://pan.baidu.com/s/13QFqhcLRIpQhnhRivAgomg
RuoYi-3.1-activiti	RuoYi多模块工作流版本	https://pan.baidu.com/s/1Omj6GSB7j4xY6H_nfiNHPA
RuoYi-mybatis-plus	Mybatis-Plus多模块（支持代码生成）	https://pan.baidu.com/s/17ZrAuqJMuGkkSwjkNvVHsA
RuoYi-3.2-redis	RuoYi多模块Redis实现Session的共享（提取码:rz7c）	https://pan.baidu.com/s/1tPRlL3dLy82qWDYFps4cwg
RuoYi-qiqiim	RuoYi-fast 与 qiqiim layim 的整合项目	https://gitee.com/wenhaofan/RuoYi-qiqiim
RuoYi-layui	集成layui主题、MybatisPlus、代码生成改造	https://github.com/kongshanxuelin/ruoyiplus
RuoYi-fanling	Layui版本的RuoYi管理系统	https://gitee.com/ifanling/fanl-galaxy-venus
RuoYi-chenzz	Layui版本的RuoYi管理系统	https://gitee.com/chenzz/RuoYi-fast/tree/ver-layui/
RuoYi-zhangmrit	归属地整合纯真、百度、离线文件ip2region	https://gitee.com/zhangmrit/ruoyi-ip2region

前台扩展

名称	说明	地址
inspinia	inspinia（2.7.1后台主题UI框架汉化版）	https://pan.baidu.com/s/1Hx6xBu-B8TP78V_YoG3avw
inspinia	inspinia（2.8后台主题bootstrap4.1）	https://pan.baidu.com/s/1wUR7GmjEfe8NsQJ5geaQbw
Hplus	Hplus（4.1.0后台主题UI框架）	https://pan.baidu.com/s/1cpDPD39OjF7IVSmPrmE_oA
Distpicker	Distpicker（v2.0.4省市联动三级下拉框）	https://pan.baidu.com/s/1kGCWkUx7nsikcKt8oXj4gQ

如何不登录直接访问

在 ShiroConfig 中设置 `filterChainDefinitionMap` 配置url=anon

```
    /admins/**=anon              # 表示该 uri 可以匿名访问
    /admins/**=auth              # 表示该 uri 需要认证才能访问
    /admins/**=authcBasic        # 表示该 uri 需要 httpBasic 认证
    /admins/**=perms[user:add:*] # 表示该 uri 需要认证用户拥有 user:add:* 权限才能访问
    /admins/**=port[8080]        # 表示该 uri 需要使用 8080 端口
    /admins/**=roles[admin]      # 表示该 uri 需要认证用户拥有 admin 角色才能访问
    /admins/**=ssl               # 表示该 uri 需要使用 https 协议
    /admins/**=user              # 表示该 uri 需要认证或通过记住我认证才能访问
    /logout=logout               # 表示注销,可以当作固定配置

    注意：
    anon, authcBasic, authc, user 是认证过滤器。
    perms, roles, ssl, rest, port 是授权过滤器。
```

如何使用多数据源

- 在 `resources` 目录下修改 `application-druid.yml`

```
# 从库数据源
slave:
 # 开启从库
 enabled: true
 url: 数据源
 username: 用户名
 password: 密码
```

- 在Service实现中添加DataSource注解

```
@DataSource(value = DataSourceType.SLAVE)
public List<User> selectUserList()
{
 return mapper.selectUserList();
}
```

如何更换主题皮肤

1、修改主框架页-默认皮肤，在菜单 `参数设置` 修改参数键名 `sys.index.skinName` 支持如下几种皮肤

- 蓝色 skin-blue
- 绿色 skin-green
- 紫色 skin-purple

- 红色 skin-red
- 黄色 skin-yellow2、修改主框架页-侧边栏主题，在菜单 `参数设置` 修改参数键名 `sys.index.sideTheme` 支持如下几种主题
- 深色主题theme-dark
- 浅色主题theme-light注：如需新增修改皮肤主题可以在 `skins.css` 中调整

如何获取用户登录信息

- 第一种方法

```
// 获取当前的用户信息
User currentUser = ShiroUtils.getSysUser();
// 获取当前的用户名称
String userName = currentUser.getUserName();
```

- 第二种方法（子模块可使用）

```
// 获取当前的用户名称
String userName = (String) PermissionUtils.getPrincipalProperty("userName");
```

3、界面获取当前用户信息（支持任意th标签）

```
<input th:value="${@permission.getPrincipalProperty('userName')}">
```

4、js中获取当前用户信息

```
var userName = [[${@permission.getPrincipalProperty('userName')}]];
```

如何防止请求重复提交

- 前端通过 `js` 控制

```
// 禁用按钮
$.modal.disable();
// 启用按钮
$.modal.enable();
```

- 后端通过 `@RepeatSubmit` 注解控制

```
/**
* 在对应方法添加注解 @RepeatSubmit
*/
@RepeatSubmit
```

```
5. public AjaxResult editSave()
```

如何配置允许跨域访问

现在开发的项目一般都是前后端分离的项目，所以跨域访问会经常使用。

1、单个控制器方法CORS注解

```
1. @RestController
2. @RequestMapping("/system/test")
3. public class TestController {
4.
5.     @CrossOrigin
6.     @GetMapping("/{id}")
7.     public AjaxResult getUser(@PathVariable Integer userId) {
8.         // ...
9.     }
10.
11.    @DeleteMapping("/{userId}")
12.    public AjaxResult delete(@PathVariable Integer userId) {
13.        // ...
14.    }
15. }
```

2、整个控制器启用CORS注解

```
1. @CrossOrigin(origins = "http://ruoyi.vip", maxAge = 3600)
2. @RestController
3. @RequestMapping("/system/test")
4. public class TestController {
5.
6.     @GetMapping("/{id}")
7.     public AjaxResult getUser(@PathVariable Integer userId) {
8.         // ...
9.     }
10.
11.    @DeleteMapping("/{userId}")
12.    public AjaxResult delete(@PathVariable Integer userId) {
13.        // ...
14.    }
15. }
```

3、全局CORS配置（在 `ResourcesConfig` 重写addCorsMappings方法）

```
1. /**
2.  * web跨域访问配置
3.  */
4. @Override
5. public void addCorsMappings(CorsRegistry registry)
```

```
{
    // 设置允许跨域的路径
    registry.addMapping("/**")
            // 设置允许跨域请求的域名
            .allowedOrigins("*")
            // 是否允许证书
            .allowCredentials(true)
            // 设置允许的方法
            .allowedMethods("GET", "POST", "DELETE", "PUT")
            // 设置允许的header属性
            .allowedHeaders("*")
            // 跨域允许时间
            .maxAge(3600);
}
```

日期插件精确到时分秒

1、界面设置时间格式 `data-format`

```
<li class="select-time">
    <label>创建时间： </label>
    <input type="text" class="time-input" id="startTime" placeholder="开始时间" name="params[beginTime]" data-format="yyyy-MM"/>
    <span>-</span>
    <input type="text" class="time-input" id="endTime" placeholder="结束时间" name="params[endTime]" data-format="yyyy-MM"/>
</li>
```

2、通过js函数设置 `datetimepicker` 日期控件可以设置 `format`

```
$('.input-group.date').datetimepicker({
    format: 'yyyy-mm-dd hh:ii:ss',
    autoclose: true,
    minView: 0,
    minuteStep:1
});
```

`laydate` 日期控件可以设置 `common.js` 配置type=datetime

```
layui.use('laydate', function() {
    var laydate = layui.laydate;
    var startDate = laydate.render({
        elem: '#startTime',
        max: $('#endTime').val(),
        theme: 'molv',
        trigger: 'click',
        type : 'datetime',
        done: function(value, date) {
            // 结束时间大于开始时间
            if (value !== '') {
                endDate.config.min.year = date.year;
                endDate.config.min.month = date.month - 1;
                endDate.config.min.date = date.date;
            } else {
                endDate.config.min.year = '';
                endDate.config.min.month = '';
                endDate.config.min.date = '';
            }
        }
    });
    var endDate = laydate.render({
        elem: '#endTime',
        min: $('#startTime').val(),
        theme: 'molv',
        trigger: 'click',
        type : 'datetime',
        done: function(value, date) {
            // 开始时间小于结束时间
            if (value !== '') {
                startDate.config.max.year = date.year;
                startDate.config.max.month = date.month - 1;
                startDate.config.max.date = date.date;
            } else {
                startDate.config.max.year = '';
                startDate.config.max.month = '';
                startDate.config.max.date = '';
            }
        }
    });
});
```

代码生成不显示新建表

默认条件需要表注释，特殊情况可在 `GenMapper.xml` 去除table_comment条件

```
<select id="selectTableByName" parameterType="String" resultMap="TableInfoResult">
    <include refid="selectGenVo"/>
    where table_comment <> '' and table_schema = (select database())
</select>
```

提示您没有数据的权限

这种情况都属于权限标识配置不对在 `菜单管理` 配置好权限标识（菜单&按钮）

- 确认此用户是否已经配置角色
- 确认此角色是否已经配置菜单权限

- 确认此菜单权限标识是否和后台代码一致如参数管理后台配置 `@RequiresPermissions("system:config:view")` 对应参数管理权限标识为 `system:config:view`

注：如需要角色权限，配置角色权限字符 使用 `@RequiresRoles("admin")`

富文本编辑器文件上传

富文本控件采用的summernote，图片上传处理需要设置callbacks函数

```
$('.summernote').summernote({
    height : '220px',
    lang : 'zh-CN',
    callbacks: {
        onImageUpload: function(files, editor, $editable) {
            var formData = new FormData();
            formData.append("file", files[0]);
            $.ajax({
                type: "POST",
                url: ctx + "common/upload",
                data: data,
                cache: false,
                contentType: false,
                processData: false,
                dataType: 'json',
                success: function(result) {
                    if (result.code == web_status.SUCCESS) {
                        $(obj).summernote('editor.insertImage', result.url, result.fileName);
                    } else {
                        $.modal.alertError(result.msg);
                    }
                },
                error: function(error) {
                    $.modal.alertWarning("图片上传失败。");
                }
            });
        }
    }
});
```

富文本编辑器底部回弹

富文本控件采用的summernote，如果不需要底部回弹设置 `followingToolbar: false`

```
$('.summernote').summernote({
    placeholder: '请输入公告内容',
    height : 192,
    lang : 'zh-CN',
    followingToolbar: false,
    callbacks: {
        onImageUpload: function (files) {
            sendFile(files[0], this);
        }
    }
});
```

如何创建新的菜单页签

建新新的页签有以下两种方式（js&html）

```
// 方式1 打开新的选项卡
function dept() {
    var url = ctx + "system/dept";
    $.modal.openTab("部门管理", url);
}
```javascript
// 方式2 选卡页同一页签打开
function dept() {
    var url = ctx + "system/dept";
    $.modal.parentTab("部门管理", url);
}
// 方式3 html创建
<a class="menuItem" href="/system/dept">部门管理</a>
```

表格数据进行汇总统计

对于某些数据需要对金额，数量等进行汇总，可以配置showFooter `true` 表示尾部统计

```
// options 选项中添加尾部统计
showFooter: true,
// columns 中添加
{
    field : 'balance',
    title : '余额',
    sortable: true,
    footerFormatter:function (value) {
        var sumBalance = 0;
        for (var i in value) {
            sumBalance += parseFloat(value[i].balance);
        }
        return "总金额：" + sumBalance;
    }
},
```

表格设置行列单元格样式

1、 `options` 参数中配置属性

```
rowStyle: rowStyle,
```

2、对应js添加响应方法（根据 `row` 或 `index` 定义规则）即可

```
function rowStyle(row, index) {
    var style = { css: { 'color': '#ed5565' } };
    return style;
}
```

如何去除数据监控广告
```

服务监控中使用的Driud，默认底部有阿里的广告。如果是一个商业项目这个是很不雅也是不允许的

- 找到本地maven库中的对应的druid-1.1.xx.jar文件，用压缩包软件打开
- 找到support/http/resource/js/common.js，打开找到 buildFooter 方法

```
this.buildFooter();
buildFooter : function() {
 var html ='此处省略一些相关JS代码';
 $(document.body).append(html);
},
```

- 删除此函数和及初始方法后覆盖文件
- 重启项目后，广告就会消失了

如何支持多类型数据库

对于某些特殊需要支持不同数据库，参考以下支持 `oracle` `mysql` 配置

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

```
# 数据源配置
spring:
    datasource:
        type: com.alibaba.druid.pool.DruidDataSource
        druid:
            # 主库数据源
            master:
                url: jdbc:mysql://127.0.0.1:3306/ry?
useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&useSSL=true&serverTimezone=GMT%2B8
                username: root
                password: password
            # 从库数据源
            slave:
                # 从数据源开关/默认关闭
                enabled: true
                url: jdbc:oracle:thin:@127.0.0.1:1521:oracle
                username: root
                password: password
```

`对于不同数据源造成的驱动问题，可以删除driverClassName。会自动识别驱动` `如需要对不同数据源分页需要操作application.yml中的pagehelper配置 删除helperDialect: mysql 会自动识别数据源 新增autoRuntimeDialect=true 表示运行时获取数据源`

如何实现翻页保留选择

- 配置checkbox选项field属性为state

```
{
 field: 'state',
 checkbox: true
},
```

- 表格选项options添加rememberSelected

```
rememberSelected: true,
```

如何实现跳转至指定页

- 表格选项options添加showPageGo

```
showPageGo: true,
```

如何自定义查询条件参数

1、在 `options` 中添加 `queryParams` 参数

```
var options = {
    url: prefix + "/list",
    queryParams: queryParams,
    columns: [{
        field: 'id',
        title: '主键'
    },
    {
        field: 'name',
        title: '名称'
    }]
};
$.table.init(options);
```

2、在当前页添加 `queryParams` 方法设置自定义查询条件如 `userName`

```
function queryParams(params) {
    var search = $.table.queryParams(params);
    search.userName = $("#userName").val();
    return search;
}
```

请求后台参数为：pageSize、pageNum、searchValue、orderByColumn、isAsc、 `userName`

3、如果是表格树，添加参数 `ajaxParams` 参数

```
var options = {
    code: "deptId",
    parentCode: "parentId",
    uniqueId: "deptId",
    url: prefix + "/list",
    ajaxParams: {
        "userId": "1",
        "userName": "ruoyi"
    },
    columns: [{
        field: 'id',
        title: '主键'
    },
    {
        field: 'name',
        title: '名称'
    }]
};
$.treeTable.init(options);
```

如何降低mysql驱动版本

1、在pom.xml中properties新增节点如:

```
<mysql.version>6.0.6</mysql.version>
```

2、单应用可以不添加，多模块需要在dependencyManagement声明依赖

```
<!-- Mysql驱动包 -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>${mysql.version}</version>
</dependency>
```

注意：如果是6以下的版本需要修改 `application-druid.yml` 中 `driverClassName` com.mysql.jdbc.Driver 是 mysql-connector-java 5中的com.mysql.cj.jdbc.Driver 是 mysql-connector-java 6中的

如何配置tomcat访问日志

1、修改 `application.yml` 中的 `server` 开发环境配置

```
# 开发环境配置
server:
  # 服务器的HTTP端口，默认为80
  port: 80
  servlet:
    # 应用的访问路径
    context-path: /
  tomcat:
    # 存放Tomcat的日志目录
    basedir: D:/tomcat
    accesslog:
        # 开启日志记录
        enabled: true
        # 访问日志存放目录
        directory: logs
    # tomcat的URI编码
    uri-encoding: UTF-8
    # tomcat最大线程数，默认为200
    max-threads: 800
    # Tomcat启动初始化的线程数，默认值25
    min-spare-threads: 30
```

2、重启项目后，在D:/tomcat/logs目录就可以看到服务器访问日志了

如何汉化系统接口Swagger

想必很多小伙伴都曾经使用过Swagger，但是打开UI界面是纯英文的界面并不太友好，作为国人还是习惯中文界面。

- 找到m2/repository/io/springfox/springfox-swagger-ui/x.x.x/springfox-swagger-ui-x.x.x.jar
- 修改对应springfox-swagger-ui-x.x.x.jar包内 `resources` 目录下 `swagger-ui.html` ，添加如下JS代码

```
<!-- 选择中文版 -->
<script src='webjars/springfox-swagger-ui/lang/translator.js' type='text/javascript'></script>
<script src='webjars/springfox-swagger-ui/lang/zh-cn.js' type='text/javascript'></script>
```

- 本地修改结束后，在覆盖压缩包文件重启就实现汉化了

如何在html页面格式化日期

Thymeleaf主要使用org.thymeleaf.expression.Dates 这个类来处理日期，在模板中使用"#dates"来表示这个对象。

1、格式化日期 `[[${#dates.format(date)}]]` 或 `th:text="${#dates.format(date)}` `[[${#dates.formatISO(date)}]]` 或 `th:text="${#dates.formatISO(date)}` `[[${#dates.format(date, 'yyyy-MM-dd HH:mm:ss')}]]` 或 `th:text="${#dates.format(date, 'yyyy-MM-dd HH:mm:ss')}`

2、获取日期字段获取当前的年份： `[[${#dates.year(date)}]]` 获取当前的月份： `[[${#dates.month(date)}]]` 获取当月的天数： `[[${#dates.day(date)}]]` 获取当前的小时： `[[${#dates.hour(date)}]]` 获取当前的分钟： `[[${#dates.minute(date)}]]` 获取当前的秒数： `[[${#dates.second(date)}]]` 获取当前的毫秒： `[[${#dates.millisecond(date)}]]` 获取当前的月份名称： `[[${#dates.monthName(date)}]]` 获取当前是星期几： `[[${#dates.dayOfWeek(date)-1}]]`

如何在表格中实现图片预览

对于某些图片需要在表格中显示，可以使用 `imageView` 方法

```
// 在columns中格式化对应相关的列属性
{
    field: 'avatar',
    title: '用户头像',
    formatter: function(value, row, index) {
        return $.table.imageView(value, '/profile/avatar');
    }
},
```

如何去掉页脚及左侧菜单栏

1、去除页脚 `修改style.css`

```
#content-main {
    height: calc(100%);
    overflow: hidden;
}
```

2、去左侧菜单栏（收起时隐藏左侧菜单） `修改style.css`

```
body.fixed-sidebar.mini-navbar #page-wrapper {
    margin: 0 0 0 0px;
}

body.body-small.fixed-sidebar.mini-navbar #page-wrapper {
    margin: 0 0 0 0px;
}
```

3、去左侧菜单栏（收起时隐藏左侧菜单） `修改index.js`

```
function() {
    if ($(this).width() < 769) {
        $('body').addClass('mini-navbar');
        $('.navbar-static-side').fadeIn(); // 换成 $('.navbar-static-side').hide();
        $(".sidebar-collapse .logo").addClass("hide");
    }
});

function SmoothlyMenu() {
    if (!$('body').hasClass('mini-navbar')) {
        $(".navbar-static-side").show();  // 添加显示这一行
        $('#side-menu').hide();
        $(".sidebar-collapse .logo").removeClass("hide");
        setTimeout(function() {
            $('#side-menu').fadeIn(500);
        },
        100);
    } else if ($('body').hasClass('fixed-sidebar')) {
        $(".navbar-static-side").hide();  // 添加隐藏这一行
        $('#side-menu').hide();
        $(".sidebar-collapse .logo").addClass("hide");
        setTimeout(function() {
            $('#side-menu').fadeIn(500);
        },
        300);
    } else {
        $('#side-menu').removeAttr('style');
    }
}
```

4、隐藏左侧菜单，需要添加.canvas-menu到body元素

```
<body class = "canvas-menu">
```

如何Excel导出子对象多个字段

```
// 单个字段导出
@Excel(name = "部门名称", targetAttr = "deptName", type = Type.EXPORT)
private Dept dept;

// 多个字段导出
@Excels({
    @Excel(name = "部门名称", targetAttr = "deptName", type = Type.EXPORT),
    @Excel(name = "部门负责人", targetAttr = "leader", type = Type.EXPORT)
})
private Dept dept;
```

单元格内容过长显示处理方法

1、使用系统自带的方法格式化处理

```
{
    field: 'remark',
    title: '备注',
    align: 'center',
    formatter: function(value, row, index) {
        return $.table.tooltip(value);
    }
},
```

2、添加css控制

```
.select-table table {
    table-layout:fixed;
}

.select-table .table td {
    /* 超出部分隐藏 */
    overflow:hidden;
    /* 超出部分显示省略号 */
    text-overflow:ellipsis;
    /*规定段落中的文本不进行换行 */
    white-space:nowrap;
    /* 配合宽度来使用 */
    height:40px;
}
```

视频教程

若依全套付费视频教程。支付宝扫码付费268元即可获得。

1、视频教程（共59节）2、开发文档（离线版，包含安装部署视频）3、视频配套演示代码4、新版本优先体验，新版本更新视频介绍5、凡是购买的用户，我会单独拉入到一个付费群里，优先解答问题。

付款后联系群主QQ：346039442。


```
教程内容
├── 1、项目概述
├── 2、环境部署
├── 3、项目介绍
|       └── 3.1、文件结构介绍
|       └── 3.2、配置文件介绍
├── 4、SpringBoot相关
|       └── 4.1、SpringBoot简介
|       └── 4.2、服务器配置
|       └── 4.3、多环境使用
|       └── 4.4、配置文件获取
|       └── 4.5、自定义资源映射
|       └── 4.6、缓存使用
|       └── 4.7、异步调用
|       └── 4.8、定时任务
|       └── 4.9、热部署
├── 5、Shiro安全控制
|       └── 5.1、Shiro简介
|       └── 5.2、登录实现
|       └── 5.3、退出实现
|       └── 5.4、验证码实现
|       └── 5.5、记住我实现
|       └── 5.6、权限控制实现
|       └── 5.7、会话实现
├── 6、Thymeleaf模板
|       └── 6.1、Thymeleaf简介
|       └── 6.2、对象使用
```

```
28. |       └── 6.3、常用语法
29. |       └── 6.4、模板片段
30. |       └── 6.5、内联语法
31. |       └── 6.6、调用后台
32. |       └── 6.7、自定义标签
33. |       └── 6.8、自定义内置对象
34. ├── 7、前端封装组件
35. |       └── 7.1、简介
36. |       └── 7.2、表格
37. |       └── 7.3、表格树
38. |       └── 7.4、表单
39. |       └── 7.5、弹出层
40. |       └── 7.6、操作处理
41. |       └── 7.7、校验
42. |       └── 7.8、树插件
43. |       └── 7.9、通用方法
44. ├── 8、导入导出
45. ├── 9、分页实现
46. ├── 10、上传下载
47. ├── 11、事务管理
48. ├── 12、异常处理
49. ├── 13、系统日志
50. |       └── 13.1、操作日志
51. |       └── 13.2、登录日志
52. |       └── 13.3、日志配置
53. ├── 14、数据权限
54. ├── 15、多数据源
55. ├── 16、代码生成
56. ├── 17、定时任务
57. ├── 18、国际化支持
58. ├── 19、字典参数
59. |       └── 19.1、字典介绍
60. |       └── 19.2、字典实现
61. |       └── 19.3、系统参数
62. ├── 20、Swagger2
63. ├── 21、其他
64. |       └── 21.1、XSS脚本过滤
65. |       └── 21.2、开发文档部署
66. |       └── 21.3、4.0更新介绍
67. ├── 22、开发实战
68. |       └── 22.1、新建自己的模块
69. |       └── 22.2、开发自己的业务
```