thinkgem / **jeesite** Public

Code | Issues | Pull requests | Actions | Projects | Wiki | Security and quality | Insights

main | **19** Branches | **105** Tags | Go to file | Go to file | Code

think-gem 增强 i18n 初始化基础配置托底 b632926 · 2 weeks ago

.gitee	点击右上角【Star】收藏本软件 ^_^	4 years ago
.vscode	调整目录结构	last year
bin	update scripts	8 months ago
packages	增强 i18n 初始化基础配置托底	2 weeks ago
web	5.17.0	3 weeks ago
.editorconfig	update .editorconfig	2 years ago
.gitignore	update .gitignore	last month
.npmrc	同步更新	last year
.prettierignore	update monorepo	2 years ago
.prettierrc.mjs	点击右上角【Star】收藏本软件 ^_^	4 months ago
.stylelintignore	update monorepo	2 years ago
LICENSE	点击右上角【Star】收藏本软件 ^_^	4 months ago
README.md	update README.md	3 weeks ago
eslint.config.mjs	点击右上角【Star】收藏本软件 ^_^	4 months ago
package.json	更新 axios 1.15.0、vite 8.0.5	3 weeks ago
pnpm-lock.yaml	更新 axios 1.15.0、vite 8.0.5	3 weeks ago
pnpm-workspace.yaml	@jeesite/vite	11 months ago
stylelint.config.mjs	5.16.0	last month
terms.md	点击右上角【Star】收藏本软件 ^_^	last year
tsconfig.json	升级 vite 8.0.3、typescript 6.0.2	29 days ago
turbo.json	使用 @jeesite/build 引用	last year
uno.config.ts	更改 prettier printWidth to 120	last year

README | Apache-2.0 license

JeeSite Vue3 前端源码
使用 Turborepo、Monorepo、pnpm

快速构建、模块化、代码复用、高效管理

如果你喜欢 JeeSite，请给她一个 ⭐ Star，您的支持将是我们前行的动力。

---

1. 单仓多包 pnpm + Turborepo 涡轮增压，提升编译速度，方便统一管理脚本任务
2. 按功能模块进行拆分为不同的包，方便进行团队开发源码管理，可根据需要进行发包
3. 模块之间松耦合，单依赖，公共模块，公共组件，公共工具，方便代码复用
4. 可方便从传统架构版本，升级到 Monorepo 模块化、分包架构

技术交流

- 官方网站：https://jeesite.com
- 使用文档：https://jeesite.com/docs
- 问题反馈：http://jeesite.net | gitcode | 新手必读
- 需求收集：https://gitee.com/thinkgem/jeesite5/issues/new
- 联系我们：http://s.jeesite.com
- 关注微信公众号，了解最新动态：

- QQ 群：`127515876`、`209330483`、`223507718`、`709534275`、`730390092`、`1373527`、`183903863(外包)`
- 微信群：如果二维码过期，请尝试点击图片并F5刷新，或者添加客服微信 jeesitex 邀请您进群

- 后端源码仓库地址：Gitee、GitCode、GitHub
- 前端源码仓库地址：Gitee、GitCode、GitHub
- 源码合集仓库地址：GVP、G-Star、GitHub

平台介绍

- JeeSite 快速开发平台，低代码，轻量级，不仅仅是一个后台开发框架，它是一个企业级快速开发解决方案，后端基于经典组合 Spring Boot、Shiro、MyBatis，前端采用分离版 Vue3、Vite、Ant Design Vue、TypeScript、Vben Admin 最先进技术栈，或者 Beetl、Bootstrap、AdminLTE 经典开发模式。
- 提供在线数据源管理、数据表建模、代码生成等功能，可自动创建业务模块代码工程和微服务模块代码工程，自动生成前端代码和后端代码；包括核心功能模块如：组织机构、用户、角色、岗位、管理员、权限审计、菜单及按钮权限、数据权限、模块管理、系统参数、字典管理、系统监控、数据监控等；扩展功能如：工作流引擎、内容管理、消息推送、单点登录、第三方登录、在线作业调度、对象存储、可视化数据大屏、报表设计器、在线文件预览、国际化、全文检索、统一认证服务等。

- 本平台采用松耦合设计，真正的轻量级，微内核，快速部署，插件架构，模块增减便捷，支持扩展 SaaS 架构、集群部署、读写分离、分库分表、Spring Cloud 微服务架构；并内置了众多账号安全设置、密码策略、系统访问限制等安全解决方案，支持等保评测。
- 本平台专注于为初级研发人员提供强大的支持，使他们能够高效、快速地开发出复杂的业务功能，同时为中高级人员腾出宝贵的时间，专注于更具战略性和创新性的任务。我们致力于让开发者能够全心投入业务逻辑中，而将繁琐的技术细节交由平台来封装处理。这不仅降低了技术实现的难度，还确保了系统架构的稳定性和安全性，进而帮助企业节省人力成本、缩短项目周期，并提高整体软件的安全性和质量。
- 2013 年发布以来已被广大爱好者用到了企业、政府、医疗、金融、互联网等各个领域中，拥有：精良架构、易于扩展、大众思维的设计模式，工匠精神，用心打磨每一个细节，深入开发者的内心，并荣获开源中国《最受欢迎中国开源软件》多次奖项，期间也帮助了不少刚毕业的大学生，教师作为入门教材，快速的去实践。
- 2019 年换代升级，我们结合了多年总结和经验，以及各方面的应用案例，对架构完成了一次全部重构，也纳入很多新的思想。不管是从开发者模式、底层架构、逻辑处理还是到用户界面，用户交互体验上都有很大的进步，在不忘学习成本、提高开发效率的情况下，安全方面也做和很多工作，包括：身份认证、密码策略、安全审计、日志收集等众多安全选项供您选择。努力为大中小微企业打造全方位企业级快速开发解决方案。
- 2021 年终发布 Vue3 的前后分离版本，使得 JeeSite 拥有同一个后台服务 Web 来支撑分离版和全栈版两套前端技术栈。
- 对接 OpenAPI、Ollama、DeepSeek 等热门 AI 大模型，凭借检索增强生成 RAG 技术，为企业知识库打造专属智能对话。
- 提供大模型 Tool 本地工具调用及 MCP 服务端和客户端工具调用，助力大模型与您的业务深度融合，实现高效交互。
- 支持国产化软件和硬件环境，如国产芯片、操作系统、数据库、中间件、国密算法等。

核心优势

- JeeSite 非常易于二次开发，可控性强。整体架构清晰、技术栈稳定且先进，源代码规范严谨。所采用的均为业界通用、社区活跃的经典技术，经典技术会的人多、学习成本低、无论是维护还是扩展都十分便捷，系统安全性和稳定性也得到了充分保障。
- JeeSite 功能全面，知识点非常多，也非常少。这看似矛盾，实则源于其“大道至简”的设计理念：功能模块和组件的设计，使用的都是一些通用的技术，通俗直观的设计风格，绝大多数开发者都能轻松掌握，所以只要掌握这些组件用法，即可高效完成业务系统的开发。
- JeeSite 在架构设计、工具调用、操作体验、代码整洁、技术规范以及系统安全等方面投入了大量精力。这些往往属于“隐形投入”——虽然用户未必一眼可见，却对系统的稳定性、可维护性和长期发展至关重要。然而，许多产品更倾向于追求表面光鲜的界面和看似炫目的功能，不愿意在用户看不见的地方投入较多的研发经费，而忽视了这些深层次的基础建设。
- JeeSite 是一个低代码开发平台，具备高度的封装性与出色的扩展能力。这里的“封装”并非限制您的自由，而是在提供开箱即用便捷性的同时，保留了充分的灵活性。当平台暂未覆盖某些特定功能时，JeeSite 会通过清晰的扩展接口和原生调用方式，让您轻松实现自定义需求。
- 许多开发者都在使用 Spring 框架，并学习其优秀的设计理念——尤其是它强大的扩展机制。但试想一下：有多少人真正去修改过 Spring 的源码？即便有人这么做了，一旦框架升级，往往就会陷入兼容性困境，甚至导致系统难以维护。这样的例子屡见不鲜。
- 正因如此，JeeSite 在设计之初就高度重视这一点：我们坚持“不侵入、可扩展”的原则，确保您在享受高效开发的同时，无需担心未来升级带来的麻烦。JeeSite 的扩展能力，正是为了帮您彻底摆脱这类后顾之忧。
- 为什么说 JeeSite 比较易于学习？JeeSite 很好的把握了设计的 “度”，避免过度设计的情况。过度设计是在产品设计过程中忽略了产品和用户的实际需求，反而带来了不必要的复杂性，而忽略了系统的学习、开发和维护成本。
- JeeSite 商业版基于社区版扩展，我们维护一套代码库，有效避免资源浪费和重复造轮子，不仅加速了功能迭代与优化、保障版本稳定性输出，还能快速反哺社区，推动创新与生态共赢，确保项目健康发展；即便您使用社区版，也无需担忧版本停滞及相关衍生问题。

- 至今 JeeSite 平台架构已经非常稳定，我们持续升级，并不失架构的先进性。
- JeeSite 精益求精，用心打磨每一个细节，界面 UI 操作便捷，体验性好。
- JeeSite 是一个专业的平台，是一个可以让您，用着省心的平台。
- 社区版基于 Apache License 2.0 开源协议，永久免费使用。

架构特点及安全方面的优势：https://jeesite.com/docs/feature/

Vue 前端简介

基于 Vue3、Vite、Ant-Design-Vue、TypeScript 和 Vue Vben Admin 等前沿技术栈构建，本软件采用最先进的技术架构， 帮助初学者快速上手并融入团队开发。内置组织机构、角色用户、菜单授权、数据权限、系统参数等核心模块，结合强大的组件封装 与数据驱动视图设计，为微小、中大型项目提供开箱即用的解决方案和丰富的示例，助力高效开发。

用户界面专为信息化管理后台量身打造，在界面设计上精益求精，每一处细节都彰显着精致，为用户带来优雅且直观的操作体验。 它提供多样化的菜单布局、智能的页签管理、高效的树表操作体验、强大的表格组件、灵活的表单组件设计，具备强大的扩展能力， 同时支持黑暗布局风格，为用户提供高效、灵活且美观的操作体验，满足各类管理后台的复杂需求。

支持 Turborepo + Monorepo 快速构建、模块化、代码复用、支持分包开发， 详见：https://gitee.com/thinkgem/jeesite-vue

前端技术特点

定义众多组件，非常贴心的组件属性及小功能，符合 JeeSite 以往的设计思想，列表和表单以数据驱动视图， 极大简化了业务功能开发， 注释分解详见【源码解析】

为什么做数据驱动视图？前端向下兼容一直是最大的问题，有了一套相应的标准，会对框架升级帮助很大。 比如你可以非常小的成本，业务代码改动非常小的情况下，去升级前端；数据驱动视图可以为未来自定义拖拽表单做更好的铺垫， 数据存储结构更清晰化，更利于维护。

提示：请仔细阅读源码解析，表单视图和列表视图上的注释哦，支持复杂表单以及多表单联合使用。

学习准备

- VSCode - 推荐 IDE 集成开发工具
- Node.js 20 和 git - 开发环境
- Vite - 熟悉 Vite 特性
- Vue-v3 - 熟悉 Vue 基础语法
- TypeScript - 熟悉 TS 基本语法
- ES6+ - 熟悉 ES6 基本语法
- Vue-Router-v4 - 熟悉 vue-router 基本使用
- Vue-Vben-Admin - 熟悉 UI 及表单列表及常用组件使用
- Ant-Design-Vue - 熟悉 UI 基本使用

快速体验

在线演示

1. 地址：http://vue.jeesite.com/

快速运行

1. 环境准备：Docker
2. 根据您的操作系统，选择以下对应命令一键拉取 Docker 镜像并启动 JeeSite：

- Linux 或 macOS

```
docker pull crpi-u3zm0t8trv68xpyx.cn-qingdao.personal.cr.aliyuncs.com/thinkgem/jeesite:latest && docker run --name js
```

- Windows

```
cmd /c "docker pull crpi-u3zm0t8trv68xpyx.cn-qingdao.personal.cr.aliyuncs.com/thinkgem/jeesite:latest && docker run -
```

> 容器启动后，系统数据将持久化保存在本地 ~/jeesite-data（Linux/macOS）或 %USERPROFILE%\jeesite-data（Windows）目录中。

3. Vue分离版本地址：http://127.0.0.1:8980/vue/login
4. 全栈版本地址：http://127.0.0.1:8980/a/login
5. 初始登录账号：（管理员）`system`，密码：`admin`

本地编译运行

- 如果没有安装 Node.js 20+，下载地址：https://nodejs.org

```
# 验证
node -v
# 配置国内源
npm config set registry https://registry.npmmirror.com
```

- 如果没有安装 Pnpm 执行安装

```
npm i -g pnpm
# 验证
pnpm -v
# 配置国内源
pnpm config set registry https://registry.npmmirror.com
```

- 获取源代码

```
# 注意：不要在带有中文或空格的目录下执行。
git clone https://gitee.com/thinkgem/jeesite-vue.git
cd jeesite-vue
```

- 安装依赖

```
pnpm install
```

- 开发环境运行访问（方式一）

```
pnpm dev
```

开发环境会加载文件较多，便于调试，请耐心等待。

- 编译打包后运行访问（方式二）

```
pnpm preview
```

编译打包后，会整合这些文件，所以访问性能会大大提高，生产环境可以开启 gzip

- 打包发布程序

```
pnpm build
```

打包完成后，会在根目录生成 dist 文件夹，发布 nginx。

详见文档：https://jeesite.com/docs/vue-install-deploy/#部署到正式服务器

安装后端服务

- 安装后台服务 JeeSite v5.x
- 打开 .env.development 文件，修改后台接口：

```
# 代理设置，可配置多个，不能换行，格式：[访问接口的根路径, 代理地址, 是否保持Host头]
# VITE_PROXY = [["/js","https://vue.jeesite.com/js",true]]
VITE_PROXY = [["/js","http://127.0.0.1:8980/js",false]]

# 访问接口的根路径（例如：https://vue.jeesite.com）建议为空
VITE_GLOB_API_URL =
```

```
# 访问接口的前缀，在根路径之后
VITE_GLOB_API_URL_PREFIX = /js
```

如果您使用的 VSCode 的话，推荐安装以下插件：

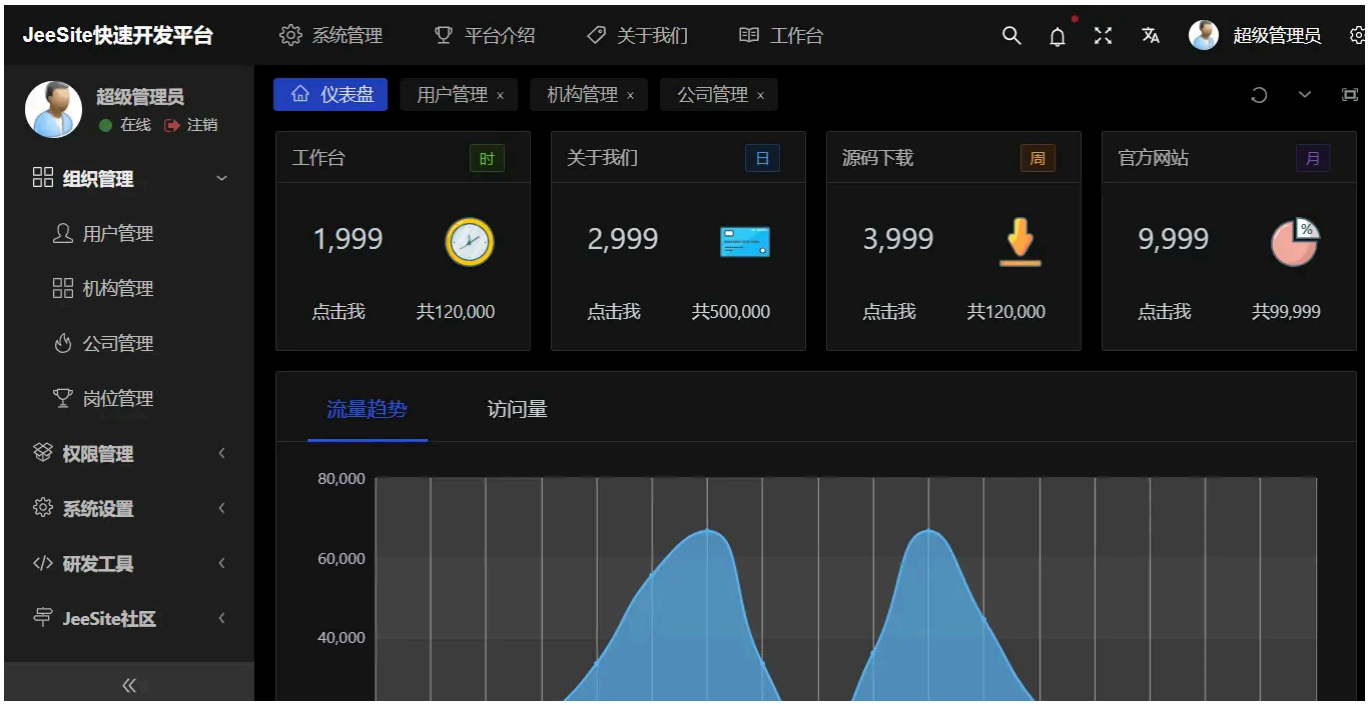
- UnoCSS - UnoCSS 提示插件
- Iconify - Iconify 图标插件
- I18n-ally - i18n 插件
- Volar - Vue3 开发必备（Vetur禁用）
- ESLint - 脚本代码检查
- Prettier - 代码格式化
- Stylelint - CSS 格式化
- DotENV - .env 文件高亮

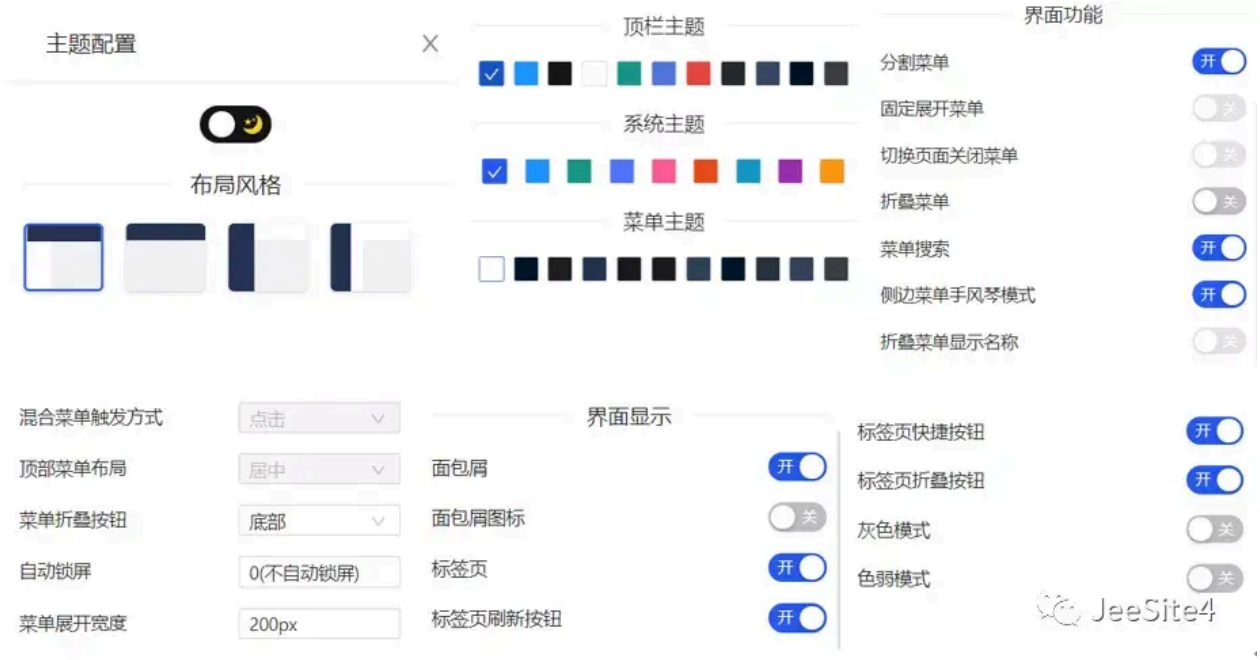
常见问题

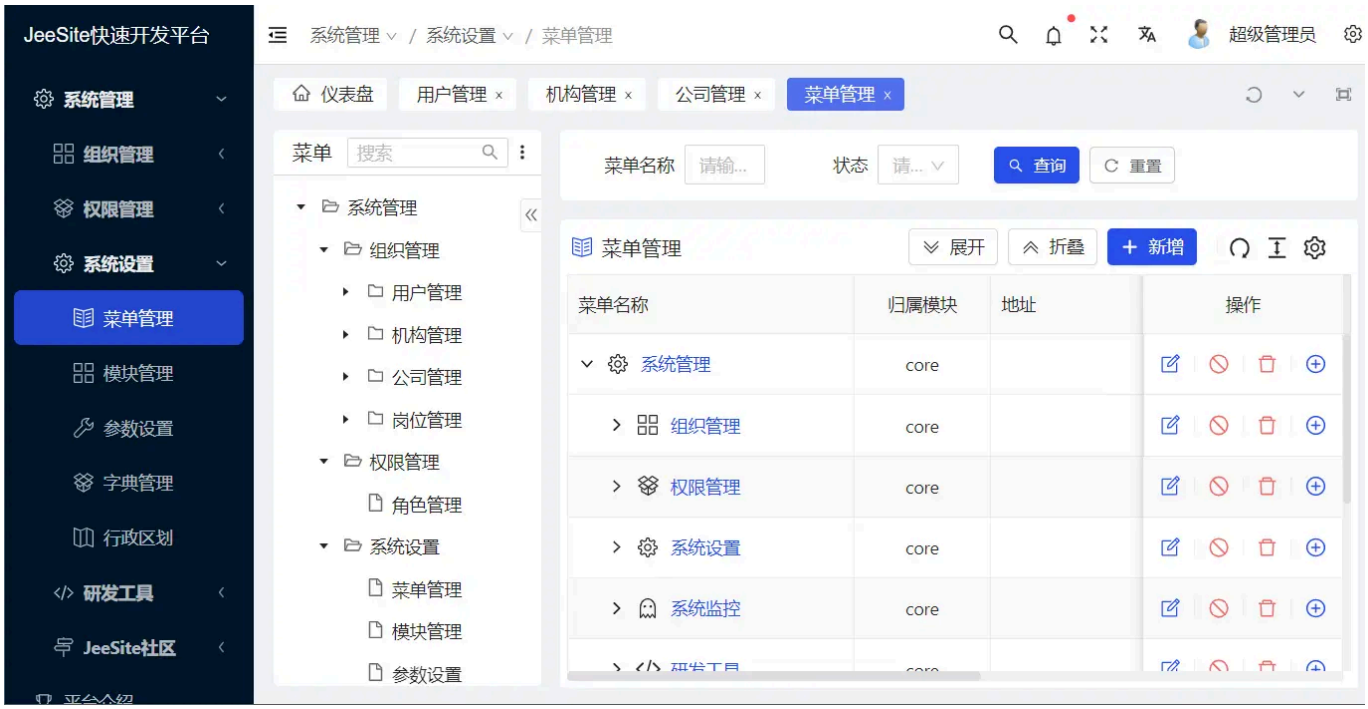
- Vue 版本的浏览器支持情况：支持所有现代浏览器，Vue3 已不再支持 IE 浏览器。
- 为什么使用抽屉作为表单组件，因为抽屉空间更大，可以展示更多内容，且操作更友好。
- 如何将表单抽屉改为弹窗，替换 list 和 form 页面的 Drawer 为 Modal 即可，V5.6增加了路由表单和弹窗表单的代码生成。
- 打不开代码生成工具怎么办？提示 404，请检查 .env.development 中的代理配置 VITE_PROXY 最后一个参数（是否保持Host头），本地服务 127.0.0.1 应设置为 false，远程服务设置为 true。
- 更多文档详见：https://jeesite.com/docs/vue-faq/#常见问题

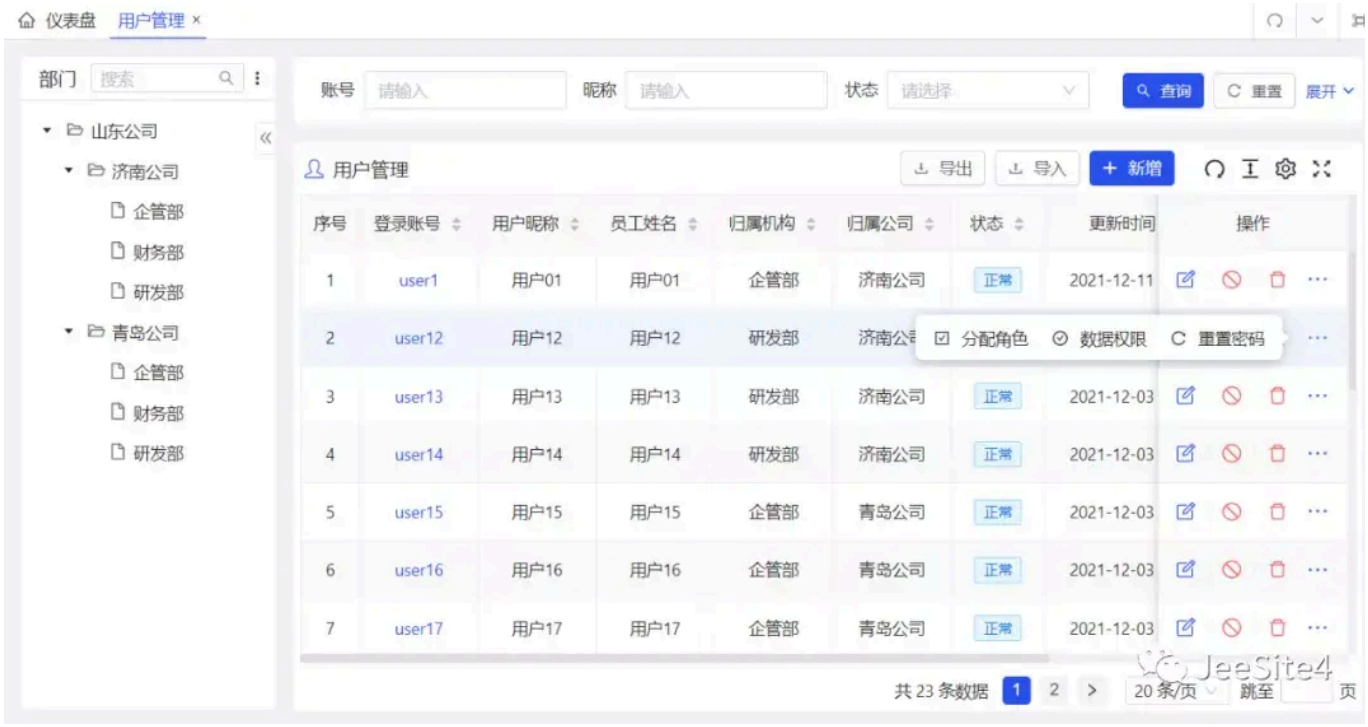
软件截图


JeeSite快速开发平台
系统管理
平台介绍
关于我们
工作台
超级管理员
超级管理员
在线
注销
组织管理
用户管理
机构管理
公司管理
岗位管理
权限管理
系统设置
研发工具
JeeSite社区
仪表盘
用户管理
机构管理
公司管理
工作台
时
1,999
点击我
共120,000
关于我们
日
2,999
点击我
共500,000
源码下载
周
3,999
点击我
共120,000
官方网站
月
9,999
点击我
共99,999
流量趋势
访问量
80,000
60,000
40,000
22:00


JeeSite快速开发平台
系统管理
平台介绍
关于我们
工作台
超级管理员
超级管理员
在线
注销
组织管理
用户管理
机构管理
公司管理
岗位管理
权限管理
系统设置
研发工具
JeeSite社区
仪表盘
用户管理
机构管理
公司管理
工作台
时
1,999
点击我
共120,000
关于我们
日
2,999
点击我
共500,000
源码下载
周
3,999
点击我
共120,000
官方网站
月
9,999
点击我
共99,999
流量趋势
访问量
80,000
60,000
40,000


主题配置
布局风格
顶栏主题
系统主题
菜单主题
界面功能
分割菜单
固定展开菜单
切换页面关闭菜单
折叠菜单
菜单搜索
侧边菜单手风琴模式
折叠菜单显示名称
混合菜单触发方式
点击
顶部菜单布局
居中
菜单折叠按钮
底部
自动锁屏
0(不自动锁屏)
菜单展开宽度
200px
界面显示
面包屑
面包屑图标
标签页
标签页刷新按钮
标签页快捷按钮
标签页折叠按钮
灰色模式
色弱模式
JeeSite4


JeeSite快速开发平台
系统管理
组织管理
权限管理
系统设置
菜单管理
模块管理
参数设置
字典管理
行政区划
研发工具
JeeSite社区
系统管理 / 系统设置 / 菜单管理
超级管理员
仪表盘
用户管理
机构管理
公司管理
菜单管理
菜单
搜索
菜单名称
状态
查询
重置
展开
折叠
新增
归属模块
地址
操作
core



附录

表单视图

```
<template>
  <!-- 弹出抽屉组件，如果想改为弹窗，Drawer 换为 Modal 即可快速替换 -->
  <BasicDrawer
    v-bind="$attrs"     -- 传递来自父组件的属性
    :showFooter="true" -- 显示弹窗底部按钮组
    :okAuth="'test:testData:edit'" -- 提交按钮权限，控制按钮是否显示
    @register="registerDrawer"      -- 弹窗后的回调方法
```

```
    @ok="handleSubmit" -- 提交按钮调用方法
    width="60%"        -- 弹窗宽度，支持按比例
  >
    <!-- 弹窗标题 -->
    <template #title>
      <Icon :icon="getTitle.icon" class="pr-1 m-1" /> -- 图标
      <span> {{ getTitle.value }} </span>  -- 标题名称
    </template>
    <!-- 表单组件 -->
    <BasicForm @register="registerForm">
      <!-- 定义表单控件插槽、个性化表单控件，如：这是一个表单子表插槽 -->
      <template #testDataChildList>
        <BasicTable
          @register="registerTestDataChildTable"
          @row-click="handleTestDataChildRowClick"
        />
        <!-- 子表新增按钮 -->
        <a-button class="mt-2" @click="handleTestDataChildAdd">
          <Icon icon="i-ant-design:plus-circle-outlined" /> {{ t('新增') }}
        </a-button>
      </template>
    </BasicForm>
  </BasicDrawer>
</template>
<!-- script name: 当前组件名称（与路由名一致，如果不一致会页面缓存失效）-->
<script lang="ts" setup name="ViewsTestTestDataForm">

  // 导入当前用到的对象，部分省略
  import { ref, unref, computed } from 'vue';
  import { officeTreeData } from '/@/api/sys/office';

  // 页面事件定义
  const emit = defineEmits(['success', 'register']);

  // 国际化方法调用，参数是国际化编码的根路径
  const { t } = useI18n('test.testData');

  // 消息弹窗方法
  const { showMessage } = useMessage();

  // 路由meta信息
  const { meta } = unref(router.currentRoute);

  // 当前页面数据记录
  const record = ref<Recordable>({});

  // 当前页面标题定义，来自菜单管理定义
  const getTitle = computed(() => ({
    icon: meta.icon || 'ant-design:book-outlined',
    value: record.value.isNewRecord ? t('新增数据') : t('编辑数据'),
  }));

  // 输入表单控件定义
  const inputFormSchemas: FormSchema[] = [
    {
      label: t('单行文本'), // 控件前面的页签
      field: 'testInput',  // 字段提交参数名
      component: 'Input',  // 控件类型（可自定义，更多查看 componentMap.ts ）
      componentProps: {    // 组件属性定义
        maxlength: 200,
      },
      required: true,      // 表单验证，是否必填（快速定义）
      rules: [             // 如果不只是必填，需要通过 rules 定义，举例：
        { required: true },
        { min: 4, max: 20, message: t('请输入长度在 4 到 20 个字符之间') },
        { pattern: /^[\u0391-\uFFE5\w]+$/, message: t('不能输入特殊字符') },
        {
          validator(_rule, value) {
             return new Promise((resolve, reject) => {
              if (!value || value === '') return resolve();
              // 远程验证，访问后台校验数据是否重复
              checkTestInput(record.value.testInput || '', value)
                .then((res) => (res ? resolve() : reject(t('数据已存在'))))
```

```
                .catch((err) => reject(err.message || t('验证失败')));
            });
          },
          trigger: 'blur', // 如果是远程验证，可以减少请求频率
        },
      ],
      colProps: { lg: 24, md: 24 }, // 栅格布局（遵循 Ant Design 风格）
    },
    {
      label: t('下拉框'),
      field: 'testSelect',
      component: 'Select',    // 选择框还有 RadioGroup、CheckboxGroup
      componentProps: {
        dictType: 'sys_menu_type', // 下拉框选项数据（支持直接指定字典类型）
        allowClear: true,          // 启用空选项，可清空选择
        mode: 'multiple',          // 下拉框模块，启用多选
      },
    },
    {
      label: t('日期选择'),
      field: 'testDate',
      component: 'DatePicker',
      componentProps: {
        format: 'YYYY-MM-DD',      // 日期选择
        showTime: false,           // 关闭时间选择
      },
    },
    {
      label: t('日期时间'),
      field: 'testDatetime',
      component: 'DatePicker',
      componentProps: {
        format: 'YYYY-MM-DD HH:mm',    // 日期时间选择
        showTime: { format: 'HH:mm' }, // 设置时间的格式
      },
    },
    {
      label: t('用户选择'),
      field: 'testUser.userCode',
      fieldLabel: 'testUser.userName', //【支持返回，如下拉框或树选择的节点名】
      component: 'TreeSelect',         // 树选择控件
      componentProps: {
        api: officeTreeData,           // 数据源 API 定义，支持 ztree 格式
        params: { isLoadUser: true, userIdPrefix: '' }, // API 参数
        canSelectParent: false,        // 是否允许选择父级
        allowClear: true,
      },
    },
    {
      label: t('子表数据'),
      field: 'testDataChildList',
      component: 'Input',
      colProps: { lg: 24, md: 24 },
      slot: 'testDataChildList',      // 指定插槽、个性化控件内容
    },
  ];

  // 当前表单的参数定义
  const [registerForm, { resetFields, setFieldsValue, validate }] = useForm({
    labelWidth: 120,                  // 控件前面的标签宽度
    schemas: inputFormSchemas,        // 控件定义列表
    baseColProps: { lg: 12, md: 24 }, // 控件默认栅格布局方式（响应式）
  });

  // 当前表单子表格定义
  const [registerTestDataChildTable, testDataChildTable] = useTable({
    actionColumn: {  // 子表的操作列定义
      width: 60,     // 操作列宽度
      actions: (record: Recordable) => [
        {
          icon: 'i-ant-design:delete-outlined',
          color: 'error',
          popConfirm: { // 是否需要启用确认框
```

```
            title: '是否确认删除',
            confirm: handleTestDataChildDelete.bind(this, record),
          },
          auth: 'sys:empUser:edit',  // 按钮权限（可控制按钮是否显示）
        },
      ],
    },
    rowKey: 'id',      // 子表主键名
    pagination: false,// 关闭分页
    bordered: true,    // 开启表格边框
    size: 'small',     // 单元格间距
    inset: true,       // 是否内嵌（去除一些边距）
  });

  // 当前表单子表自动定义
  async function setTestDataChildTableData(_res: Recordable) {
    testDataChildTable.setColumns([
      {
        title: t('单行文本'),
        dataIndex: 'testInput',
        width: 230,
        align: 'left',
        editRow: true,           // 是否启用编辑
        editComponent: 'Input', // 编辑控件（可自定义，更多查看 componentMap.ts ）
        editRule: true,          // 控件验证（是否必填）
      },
      {
        title: t('下拉框'),
        dataIndex: 'testSelect',
        width: 130,
        align: 'left',
        dictType: 'sys_menu_type',    // 指定字典类型，自动显示字典标签
        editRow: true,
        editComponent: 'Select',
        editComponentProps: {         // 控件属性
          dictType: 'sys_menu_type', // 下拉框的字段类型
          allowClear: true,
        },
        editRule: false,
      },
      // 更多组件控件不举例了，同表单控件 ...
    ]);
    // 设定子表数据
    testDataChildTable.setTableData(record.value.testDataChildList || []);
  }

  // 点击行，启用编辑
  function handleTestDataChildRowClick(record: Recordable) {
    record.onEdit?.(true, false);
  }

  // 添加编辑行，可指定初始数据
  function handleTestDataChildAdd() {
    testDataChildTable.insertTableDataRecord({
      id: new Date().getTime(),
      isNewRecord: true,
      editable: true,
    });
  }

  // 删除编辑行方法
  function handleTestDataChildDelete(record: Recordable) {
    testDataChildTable.deleteTableDataRecord(record);
  }

  // 获取子表数据（支持返回删除未提交的数据）
  async function getTestDataChildList() {
    let testDataChildListValid = true;
    let testDataChildList: Recordable[] = [];
    for (const record of testDataChildTable.getDataSource()) {
      // 验证控件内容，并取消行的编辑状态（如果验证失败返回false）
      if (!(await record.onEdit?.(false, true))) {
        testDataChildListValid = false;
```

```
      }
      testDataChildList.push({
        ...record,
        id: !!record.isNewRecord ? '' : record.id,
      });
    }
    for (const record of testDataChildTable.getDelDataSource()) {
      if (!!record.isNewRecord) continue;
      testDataChildList.push({
        ...record,
        status: '1',
      });
    }
    // 子表验证事件，抛出异常消息
    if (!testDataChildListValid) {
      throw { errorFields: [{ name: ['testDataChildList'] }] };
    }
    return testDataChildList;
  }

  // 弹窗后的回调事件，进行一些表单数据初始化等操作
  const [registerDrawer, { setDrawerProps, closeDrawer }] = useDrawerInner(async (data) => {
    resetFields(); // 重置表单数据
    setDrawerProps({ loading: true }); // 显示加载框
    const res = await testDataForm(data); // 查询表单数据
    record.value = (res.testData || {}) as Recordable;
    setFieldsValue(record.value);  // 设置字段值
    setTestDataChildTableData(res);  // 设置子表数据（没有子表可不写）
    setDrawerProps({ loading: false }); // 隐藏加载框
  });

  // 表单提交按钮方法
  async function handleSubmit() {
    try {
      const data = await validate(); // 验证表单，并返回数据
      setDrawerProps({ confirmLoading: true }); // 显示提交加载中
      // 设置提交的参数（QueryString，后台 Controller 的 get 接受）
      const params: any = {
        isNewRecord: record.value.isNewRecord,
        id: record.value.id,
      };
      // 获取并设置子表数据
      data.testDataChildList = await getTestDataChildList();
      // console.log('submit', params, data, record);
      // 将数据提交给后台（如果失败跳转到 catch）
      const res = await testDataSave(params, data);
      showMessage(res.message); // 显示提交结果
      setTimeout(closeDrawer);  // 隐藏抽屉弹窗
      emit('success', data);    // 触发事件，列表数据刷新
    } catch (error: any) {
      if (error && error.errorFields) {
        showMessage(t('您填写的信息有误，请根据提示修正。'));
      }
      console.log('error', error);
    } finally {
      setDrawerProps({ confirmLoading: false }); // 隐藏提交加载中
    }
  }
</script>
```

列表视图

```
<template>
  <div>
    <!-- 表格组件 -->
    <BasicTable @register="registerTable">
      <!-- 表格标题插槽 -->
      <template #tableTitle>
        <Icon :icon="getTitle.icon" class="m-1 pr-1" />
        <span> {{ getTitle.value }} </span>
      </template>
```

```
      <!-- 表格右侧按钮插槽，其中 v-auth 是按钮权限控制 -->
      <template #toolbar>
        <a-button type="primary" @click="handleForm({})" v-auth="'test:testData:edit'">
          <Icon icon="i-fluent:add-12-filled" /> {{ t('新增') }}
        </a-button>
      </template>
      <!-- 首列插槽 -->
      <template #firstColumn="{ record }">
        <a @click="handleForm({ id: record.id })">
          {{ record.testInput }}
        </a>
      </template>
    </BasicTable>
    <!-- 点击表格行进入的输入表单弹窗 -->
    <InputForm @register="registerDrawer" @success="handleSuccess" />
  </div>
</template>
<!-- script name: 当前组件名称（与路由名一致，如果不一致会页面缓存失效）-->
<script lang="ts" setup name="ViewsTestTestDataList">

  // 导入当前用到的对象，部分省略
  import InputForm from './form.vue';

  // 国际化方法调用，参数是国际化编码的根路径
  const { t } = useI18n('test.testData');

  // 消息弹窗方法
  const { showMessage } = useMessage();

  // 路由meta信息
  const { meta } = unref(router.currentRoute);

  // 当前页面标题定义，来自菜单管理定义
  const getTitle = {
    icon: meta.icon || 'ant-design:book-outlined',
    value: meta.title || t('数据管理'),
  };

  // 表格搜索表单控件定义
  const searchForm: FormProps = {
    baseColProps: { lg: 6, md: 8 }, // 表单栅格布局
    labelWidth: 90,                 // 表单标签宽度
    schemas: [
      {
        label: t('单行文本'),        // 表单标签
        field: 'testInput',         // 字段提交参数名
        component: 'Input',         // 表单控件
      },
      {
        label: t('下拉框'),
        field: 'testSelect',
        component: 'Select',    // 选择框还有 RadioGroup、CheckboxGroup
        componentProps: {
          dictType: 'sys_menu_type', // 下拉框选项数据（支持直接指定字典类型）
          allowClear: true,          // 启用空选项，可清空选择
          mode: 'multiple',          // 下拉框模块，启用多选
        },
      },
      // 更多控件，再次不展示了，和上一节表单视图一致
    ],
  };

  // 表格列定义
  const tableColumns: BasicColumn[] = [
    {
      title: t('单行文本'),    // 表头标题
      dataIndex: 'testInput', // 表列实体属性名
      key: 'a.test_input',    // 排序数据库字段名
      sorter: true,           // 点击表头是否可排序
      width: 230,             // 列宽
      align: 'left',          // 列的对齐方式
      // 个性化列，可定义插槽（如样式，增加控件等）
      slot: 'firstColumn',
```

```
    },
    {
      title: t('下拉框'),
      dataIndex: 'testSelect',
      key: 'a.test_select',
      sorter: true,
      width: 130,
      align: 'center',
      dictType: 'sys_menu_type', // 字典列，快速显示字典标签
    },
  ];

  // 表格操作列定义
  const actionColumn: BasicColumn = {
    width: 160, // 操作列宽
    actions: (record: Recordable) => [
      {
        icon: 'i-clarity:note-edit-line',
        title: t('编辑数据'),
        onClick: handleForm.bind(this, { id: record.id }),
        // 按钮权限控制，指定权限字符串
        auth: 'test:testData:edit',
      },
      {
        icon: 'i-ant-design:stop-outlined',
        color: 'error',
        title: t('停用数据'),
        // 是否需要启用确认框
        popConfirm: {
          title: t('是否确认停用数据'),
          confirm: handleDisable.bind(this, { id: record.id }),
        },
        // 按钮权限控制，指定权限字符串
        auth: 'test:testData:edit',
        // 控制按钮是否显示（区别：show 是显示或隐藏；ifShow 是显示或移除）
        show: () => record.status === '0',
        ifShow: () => record.status === '0',
      },
    ],
    // 操作列更多按钮定义
    dropDownActions: (record: Recordable) => [
      {
        icon: 'i-ant-design:reload-outlined',
        label: t('重置密码'),
        onClick: handleResetpwd.bind(this, { userCode: record.userCode }),
        auth: 'sys:empUser:resetpwd',
      },
    ],
  };

  // 点击首列或编辑按钮是的抽屉弹窗定义
  const [registerDrawer, { openDrawer }] = useDrawer();

  // 表格定义
  const [registerTable, { reload }] = useTable({
    api: testDataListData,     // 表格数据源 API
    beforeFetch: (params) => {
      return params;           // API 提交之前的参数修改
    },
    columns: tableColumns,     // 表格列
    actionColumn: actionColumn,// 操作列
    formConfig: searchForm,    // 搜索表单
    showTableSetting: true,    // 是否显示右上角的设置按钮
    useSearchForm: true,       // 是否显示搜索表单
    canResize: true,           // 是否自适应表单高度
  });

  // 弹窗操作方法
  function handleForm(record: Recordable) {
    openDrawer(true, record);
  }

  // 操作列停用按钮方法
```

```
  async function handleDisable(record: Recordable) {
    const res = await testDataDisable(record);
    showMessage(res.message);
    handleSuccess();
  }

  // 刷新表格数据（含表单回调）
  function handleSuccess() {
    reload();
  }
</script>
```

更多介绍

- 架构特点：https://jeesite.com/docs/feature/
- 内置功能：https://jeesite.com/docs/function/
- 目录结构：https://jeesite.com/docs/catalog/
- 参数配置：https://jeesite.com/docs/config/
- 开发规范：https://jeesite.com/docs/standard/
- 数表设计：https://jeesite.com/docs/treetable/

产品列表

- JeeSite 源码仓库：https://gitee.com/thinkgem/jeesite5
- JeeSite Vue 前端源码：https://gitee.com/thinkgem/jeesite-vue
- JeeSite Cloud 微服务：https://gitee.com/thinkgem/jeesite-cloud
- JeeSite 跨平台手机端：https://gitee.com/thinkgem/jeesite-uniapp
- JeeSite 客户端安装程序：https://gitee.com/thinkgem/jeesite-client
- 内外网中间件：https://my.oschina.net/thinkgem/blog/4624519
- 统一认证平台：https://jeesite.com/docs/oauth2-server

学习文档

- 库表生成、代码生成：https://jeesite.com/docs/code-gen/
- 菜单权限、按钮权限：https://jeesite.com/docs/permi-shiro/
- 数据权限、库事务：https://jeesite.com/docs/service-datascope/#数据权限
- 表结构、数据字典：https://jeesite.com/docs/code-gen/#表结构数据字典
- 持久层框架、SQL：https://jeesite.com/docs/dao-mybatis/
- 后端常用工具：https://jeesite.com/docs/sys-utils/

分离版

- 版本介绍：https://jeesite.com/docs/jeesite-vue/
- 源码解析：https://jeesite.com/docs/vue-crud-view/
- 表单组件：https://jeesite.com/docs/vue-basic-form/
- 表格组件：https://jeesite.com/docs/vue-basic-table/
- 参数配置：https://jeesite.com/docs/vue-settings/
- 常用组件：https://jeesite.com/docs/vue-comp/
- 前端权限：https://jeesite.com/docs/vue-auth/
- 图标组件：https://jeesite.com/docs/vue-icon/
- 前端样式库：https://jeesite.com/docs/vue-style/
- 多语言国际化：https://jeesite.com/docs/vue-i18n/

经典版

- 表单组件：https://jeesite.com/docs/views-beetl/
- 表格组件：https://jeesite.com/docs/datagrid/

- 常用工具：https://jeesite.com/docs/jeesite-js/
- 自定义主题：https://jeesite.com/docs/custom-views/

更多文档

- AI、CMS、RAG、Tool、MCP 人工智能助手：https://jeesite.com/docs/ai-cms
- BPM 业务流程引擎（Flowable）：https://jeesite.com/docs/bpm/
- CMS 多站点内容管理模块：https://jeesite.com/docs/cms/
- 消息推送消息提醒：https://jeesite.com/docs/msg-push-use/
- 对象存储模块：https://jeesite.com/docs/oss-client
- 单点登录模块：https://jeesite.com/docs/sso-cas
- 在线任务调度：https://jeesite.com/docs/job/
- 大屏设计器：https://jeesite.com/docs/visual/
- 报表设计器：https://jeesite.com/docs/ureport/
- 文件管理分享：https://jeesite.com/docs/filemanager/
- 文件在线预览：https://jeesite.com/docs/filepreview/
- 集群、高可用架构：https://jeesite.com/docs/cluster/
- SaaS 多租户架构：https://jeesite.com/docs/saas-corp-use/
- 读写分离分片分表：https://jeesite.com/docs/sharding/
- Spring监控系统：https://jeesite.com/docs/webadmin/
- 分布式跨应用事务：https://jeesite.com/docs/seata/
- 追踪系统集成：https://jeesite.com/docs/skywalking/

Releases 12

JeeSite Vue v5.17.0 发布，快速开发 Latest
2 weeks ago

+ 11 releases

Packages

No packages published

Contributors 5

Languages

Vue 58.3% **TypeScript** 37.9% **Less** 2.5% **CSS** 0.8% **JavaScript** 0.4% **HTML** 0.1%