



大汉软件

JPaaS3 平台

前端开发规范

大汉软件股份有限公司

修订历史记录

日期	版本	说明	作者
2019-05-07	0.1	初始版本	陈健
2020-03-18	0.2	指南补充、git 补充	陈健
2020-05-09	2.0	JPaas2 开发平台指南，增加应用入口、工作台部件	陈健
2021-05-15	3.0	JPaas3 前端开发指南，微前端架构	陈健
2022-05-13	4.0	合并前端开发指南和微应用开发指南	陈健
2023-02-06	4.1	修改表单提交和弹窗提交规范 添加组件库文档地址 添加 URL 命名规范 更新插件推荐 新增路由菜单配置 新增按照服务状态控制路由	陈健
2023-06-20	5.0	添加脚手架 3.3.0 新增功能 添加代码审查、部署说明 添加全局变量和工具 添加 vscode 插件 添加常见问题	陈健

目录

修订历史记录	1
目录	2
1 简介	11
1.1 定义、首字母缩写词和缩略语	11
1.2 参考资料	11
2 前端开发平台组成	11
3 前端架构	12
4 前端技术栈	15
5 开发环境	15
5.1 NODE 安装	15
5.2 包管理器安装	16
5.3 VUE CLI 安装	16
5.4 JPAAS CLI 安装	16
5.5 代码编辑器安装	17
5.5.1 安装 Visual Studio Code	17
5.5.2 安装插件	17
5.5.2.1 ANT DESIGN VUE HELPER (必须)	18
5.5.2.2 PRETTIER (必须)	18
5.5.2.3 ESLINT (必装)	18

5.5.2.4 VUE LANGUAGE FEATURES (VOLAR) (必装)	19
5.5.2.5 VSC-COMMITIZEN (必装)	19
5.5.2.6 MICROFONTENDS SIMULATOR (必装)	19
5.5.2.7 COMPONENT HELPER (必装)	20
5.5.2.8 LOW CODE (必装)	20
5.5.2.9 APPLICATION CREATOR (必装)	20
5.5.2.10 WHATCHANGED	20
5.5.2.11 GIT HISTORY	20
5.5.2.12 VUE-STYLE-BEAUTIFY	20
5.5.2.13 PROJECT MANAGER	20
5.6 代码仓库配置	21
5.6.1 安装及配置 Git	21
5.6.2 SSH 密钥设置	21
5.7 调试工具安装	22
5.7.1 Vue Devtools (必装)	22
6 开发指南	23
6.1 基础	23
6.1.1 新建项目	23
6.1.2 安装	24
6.1.3 项目配置	24
6.1.4 配置本地开发服务器	25
6.1.5 启动服务	26

6.1.5.1 VS CODE 执行脚本（推荐）	27
6.1.5.2 终端执行命令	27
6.1.6 本地开发模式	28
6.1.6.1 登录	28
6.1.6.2 导航	29
6.1.6.3 模拟器	29
6.1.7 路由和菜单	30
6.1.7.1 配置	30
6.1.7.2 实例	31
6.1.8 页面布局	33
6.1.8.1 打开全屏页面	35
6.1.8.2 进入另一个页面视图	35
6.1.8.3 返回上一个页面	36
6.1.8.4 自适应页面高度	36
6.1.8.5 定制布局（BASICLAYOUT）	36
6.1.8.6 全屏布局（FULLLAYOUT）	38
6.1.9 编写页面	39
6.1.9.1 新增 VUE 文件	39
6.1.9.2 编写业务代码	39
6.1.10 组件库使用	41
6.1.10.1 组件引入	41

6.1.10.2 组件库升级	41
6.1.11 和服务端交互	42
6.1.11.1 API 接口文档管理	43
6.1.11.2 编写 API 方法	44
6.1.11.3 调用 API 方法	46
6.1.11.4 错误处理	46
6.1.11.5 VUE 组件内请求 API	47
6.1.12 权限控制	48
6.1.12.1 应用菜单权限配置	48
6.1.12.2 路由权限控制	48
6.1.12.3 操作权限控制	48
6.1.13 路径	49
6.1.13.1 URL	49
6.1.13.2 文件/模块路径	50
6.1.14 应用入口	50
6.1.14.1 添加应用入口	51
6.1.14.2 自定义角色设置	51
6.1.14.3 设置导航菜单	52
6.1.15 构建和发布	52
6.1.15.1 ESLINT 检查	52
6.1.15.2 修改应用信息	53

6.1.15.3 修改发布配置文件	53
6.1.15.4 构建	54
6.1.15.4.1 自动构建	54
6.1.15.4.2 本地编译	54
6.1.16 部署	55
6.1.16.1 NGINX 部署	55
6.1.16.2 RANCHER 部署	55
6.1.17 升级微应用	56
6.2 进阶	57
6.2.1 工具库/函数	57
6.2.1.1 动态加载脚本、样式	57
6.2.1.2 通讯加密	58
6.2.1.3 BASE64 编码/解码	59
6.2.1.4 生成唯一标识	59
6.2.1.5 数据校验	59
6.2.1.6 LOCALSTORAGE 管理	60
6.2.1.7 COOKIE 管理	61
6.2.2 编写工作台部件	61
6.2.2.1 目录结构	61
6.2.2.2 部件配置	61
6.2.2.3 部件编写	62

6.2.3 基础应用扩展	64
6.2.3.1 应用的角色授权	64
6.2.3.2 审批中心办件详情	65
6.2.3.3 日志管理	66
6.2.3.4 应用评论管理	67
6.2.3.5 工具箱	68
6.2.4 获取全局/应用内参数配置	69
6.2.5 动态控制路由	70
6.2.6 查看前端应用版本号	70
6.2.7 常用脚本	70
6.2.8 全局变量/方法	71
6.3 开发辅助	72
6.3.1 浏览器开发者工具	72
6.3.1.1 元素 (ELEMENT) 面板	72
6.3.1.2 控制台 (CONSOLE) 面板	73
6.3.1.3 源代码 (SOURCES) 面板	73
6.3.1.4 网络 (NETWORK) 面板	74
6.3.1.5 应用 (APPLICATION) 面板	75
6.3.2 代码调试	76
6.3.2.1 VUE DEVTOOLS (必装)	76
6.3.2.2 VS CODE 中断点调试 (推荐)	76

6.3.2.3 DEBUGGER 语句断点调试	78
6.3.2.4 线上调试	78
6.3.3 Mock 测试	80
6.3.3.1 MOCK 数据配置	80
6.3.3.2 本地连接 MOCKSERVER 测试	81
6.3.3.3 浏览器 MOCK 插件	82
6.3.4 VSCode 开发辅助插件	83
6.3.4.1 应用创建	83
6.3.4.2 组件开发辅助	85
6.3.4.3 微前端环境模拟器	85
6.3.4.4 低代码开发辅助	86
6.3.4.5 代码片段	88
6.4 代码审查	89
6.4.1 SonarQube	89
6.4.2 SonarLint (vscode 插件)	92
6.5 合规检查	94
7 开发规范	95
7.1 目录结构	95
7.2 命名规范	96
7.2.1 目录名	96
7.2.2 文件命名	96

7.2.3 变量命名	97
7.2.4 组件名	97
7.2.5 Prop 属性名	98
7.3 组件/实例	98
7.3.1 选项顺序	98
7.3.2 单文件组件的顶级元素的顺序	100
7.4 样式编写规范	100
7.5 API 调用编写规范	100
7.6 图标	102
7.6.1 内置图标引入	102
7.6.2 图标文件引入	103
7.6.3 图标使用	104
7.7 代码管理	104
7.7.1 分支	104
7.7.2 版本	104
7.7.3 工作流	105
7.7.4 Commit Message	105
7.7.5 CHANGELOG	107
7.8 路由规范	108
7.9 URL 命名规范	108
8 界面规范	108

8.1 页面布局	109
8.2 主题色	109
8.3 文字	109
8.4 图片	110
8.5 列表	110
8.6 对话框	110
8.7 抽屉	110
8.8 表单	111
8.9 提示和通知	111
9 常见问题	112
附录	113
1 组件文档	113
2 其他参考文档	114

1 简介

前端开发环境的基本搭建、平台使用、编码指导、开发流程、代码管理及其他统一规范

1.1 定义、首字母缩写词和缩略语

kebab-case 风格：全小写，多个单词之间以“-”分隔，如：notice-detail

UpperCamelCase 风格：大骆驼拼写，每个词的首字母都大写，如：UserMenu

lowerCamelCase 风格：小骆驼拼写，第一个词的首字母小写，后面每个词的首字母大写，
如：infoList

1.2 参考资料

《页面标准化规范》

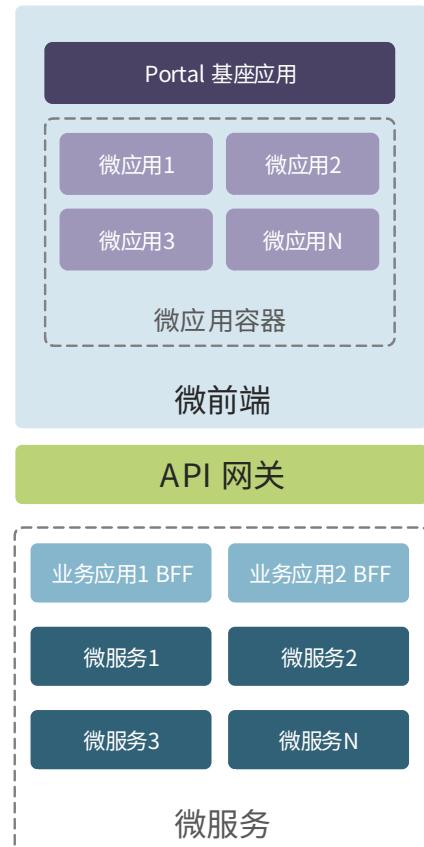
2 前端开发平台组成

名称	定义	描述
jpaas-portal	主应用（基座）	生产环境运行时，用于将各个微应用聚合在一个框架内运行。 包括基础依赖、框架布局、工作台、微组件管理、微应用管理、全局状态管理
jpaas-common	基础应用	基础业务及通用功能支撑
jpaas-mcommon	移动基础应用	移动相关基础业务及功能支撑

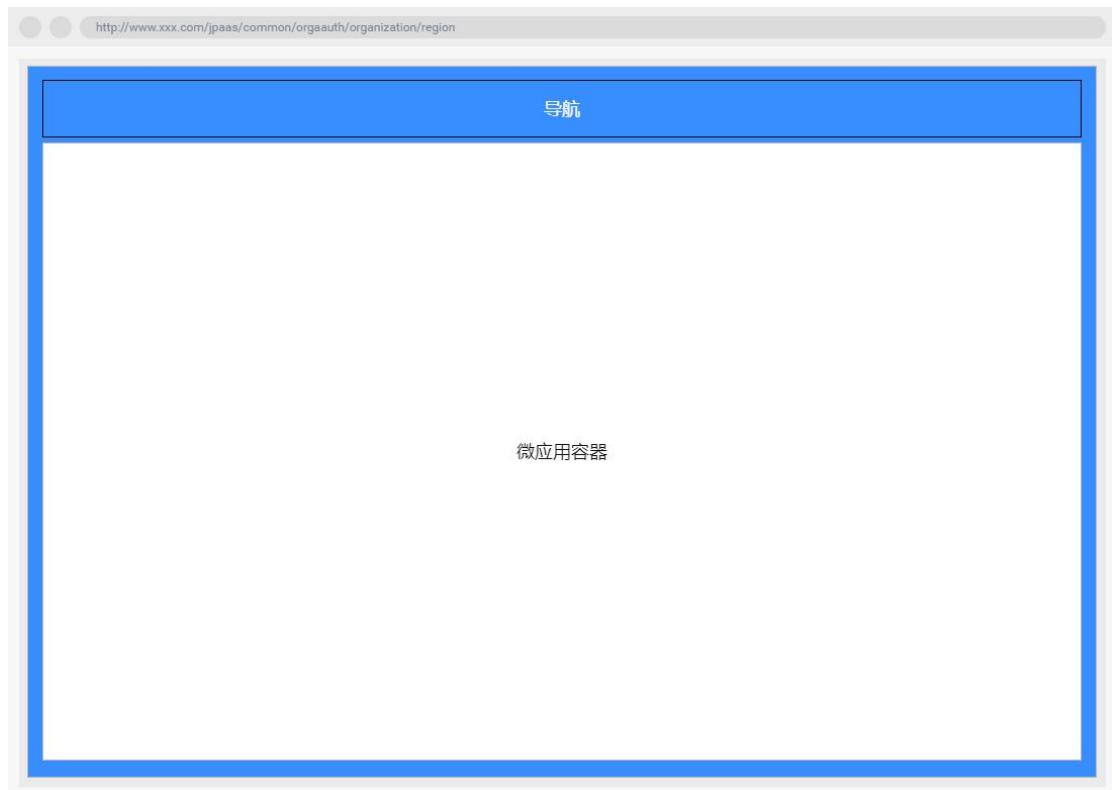
jpaas-micro-core	微应用开发内核	微应用基础模块及本地开发模拟环境。作为 NPM 依赖使用
jpaas-micro-starter	微应用开发脚手架	微前端应用开发模板
jpaas-component	组件库	基本组件、通用业务组件。作为 NPM 依赖使用
jpaas-mcomponent	移动组件库	移动基本组件、通用业务组件。作为 NPM 依赖使用
jpaas-cli	命令行开发工具	应用开发辅助工具，提供多种类型的应用脚手架
jpaas-ide	开发辅助工具	vscode、chrome 等开发辅助插件

3 前端架构

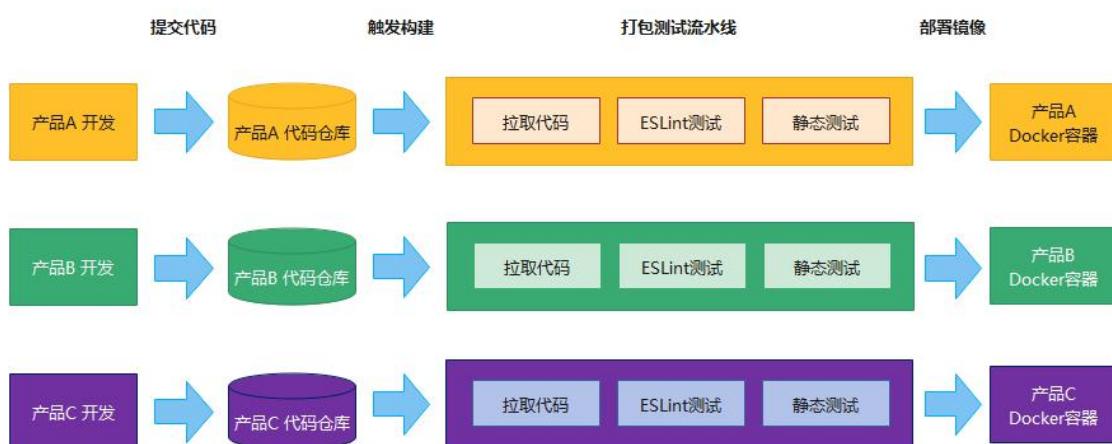
jpaas3 前端采用微前端架构，基础应用、产品应用通过基座框架聚合到一起，组成完整的 jpaas 前端应用。各个前端应用可以独立的开发、测试、部署、运行



jpaas 微前端架构图



微前端主框架结构示意图



微应用开发构建流程

4 前端技术栈

Vue	用于构建用户界面的渐进式框架。通过尽可能简单的 api 实现响应的数据绑定和组合的视图集合
Vuex	状态管理模式。驱动数据源、映射视图、响应变化
Vue Router	单页应用路由管理器。控制页面切换、按需加载等
Ant Design	UI 组件库
Axios	HTTP 客户端，用于异步请求
Vue CLI	构建、管理工具，提供 vue 项目脚手架、插件管理、图形化项目管理
Webpack	打包工具
Babel	javascript 转义器，用于兼容新版 ES 语法规则
qiankun	一个基于 single-spa 的微前端实现库

5 开发环境

5.1 Node 安装

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。

Node 官网查看最新版本，<https://nodejs.org/zh-cn/>

下载 Linux 二进制文件 (x64) 版本

1. 下载后解压，执行命令：tar -xf node-v18.16.1-linux-x64.tar.xz
2. 修改环境变量，执行命令：sudo vi /etc/profile
3. 输入密码

4. 添加配置，NODE_HOME 指向解压缩后的目录

```
NODE_HOME=/data/home/eric/program/node/node-v18.16.1-linux-x64  
PATH=$NODE_HOME/bin:$PATH  
export NODE_HOME PATH  
export NODE_OPTIONS="--openssl-legacy-provider"
```

5. 输入:wq 保存

6. 使用内网私服，修改 npm 仓库：(必须)

```
npm config set registry http://192.168.5.32:9111/repository/hanweb-ui-npm/
```

5.2 包管理器安装

公司统一使用 PNPM 作为软件包管理器，安装依赖及打包勿使用 NPM

终端中安装 pnpm (全局)：(官网：<https://pnpm.io/>)

```
npm install -g pnpm
```

使用内网私服，修改 npm 仓库：(必须)

```
pnpm config set registry http://192.168.5.32:9111/repository/hanweb-ui-npm/
```

5.3 Vue CLI 安装

安装 CLI，终端执行：

```
npm install -g @vue/cli
```

5.4 JPaaS CLI 安装

JPaaS 前端命令行工具

安装 CLI , 终端执行 :

```
npm install -g jpaas-cli
```

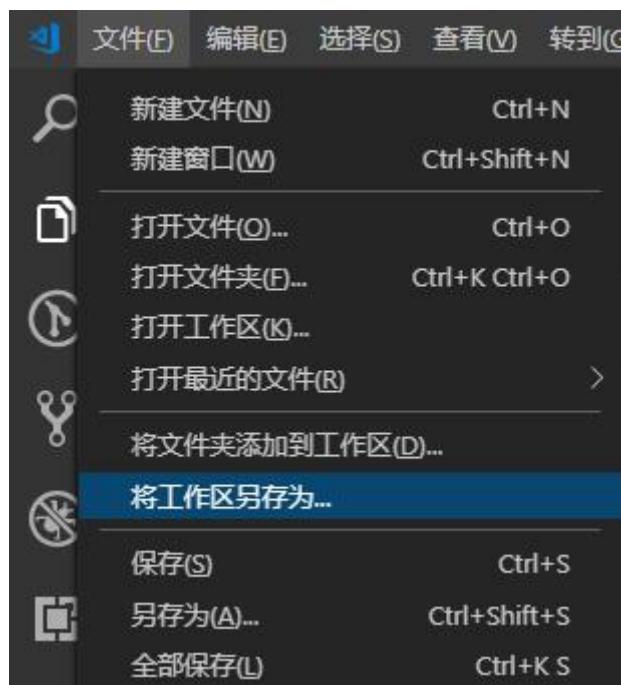
5.5 代码编辑器安装

5.5.1 安装 Visual Studio Code

1. 下载安装

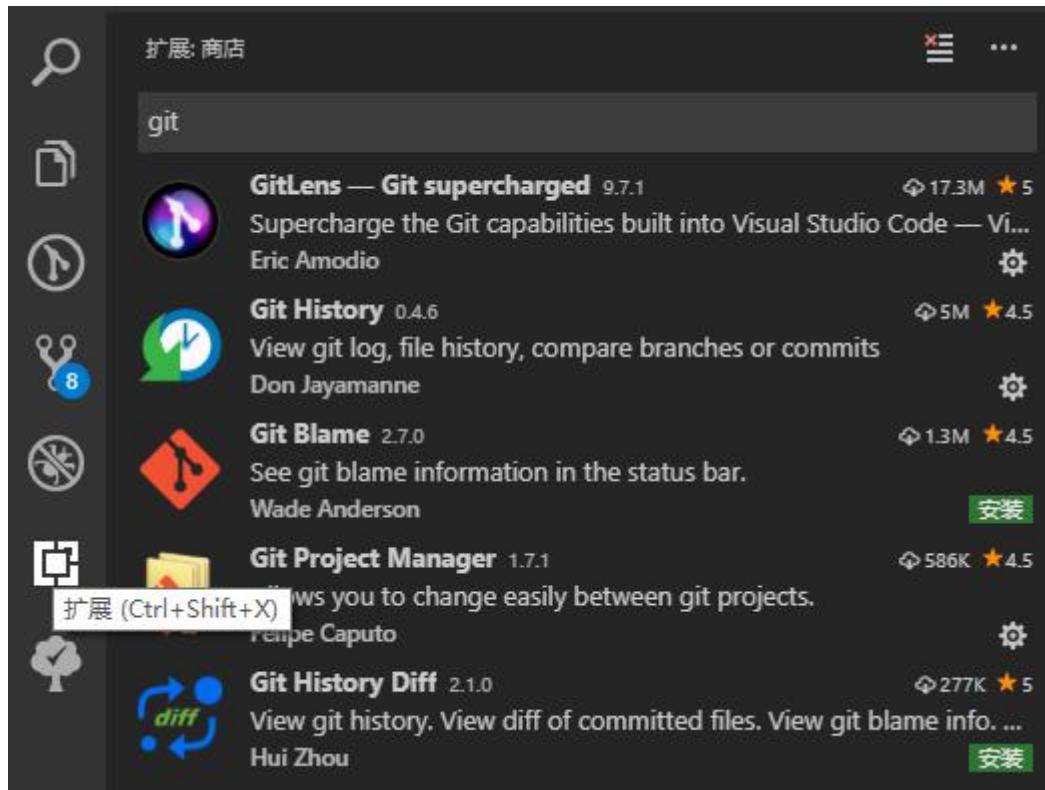
<https://code.visualstudio.com/>

2. 资源管理中创建工作空间目录 , 如 : workspace。以后代码都存放到该目录下。
3. 打开 vscode , 快捷键 alt , 显示菜单栏
4. 点击“将工作区另存为” , 选择“workspace”目录 , 并为工作区起一个名字



5.5.2 安装插件

点击扩展 , 搜索。选择需要的扩展进行安装。



5.5.2.1 Ant Design Vue helper (必须)

Ant Design Vue 组件代码提示、文档查询

5.5.2.2 Prettier (必须)

代码格式化

5.5.2.3 ESLint (必装)

代码检测工具，此为 vscode 插件，需要安装 ESLint。

该插件用来校验规范和格式代码，必须安装

1. 安装 ESLint，终端中运行 `npm install -g eslint`

2. 打开 设置 -> 用户设置 -> 扩展 -> ESLint

3. 编辑 settings.json , 添加配置

```
//为了符合 eslint 的两个空格间隔原则
"editor.tabSize": 2,
//配置 eslint
"eslint.autoFixOnSave": true,
"eslint.validate": [
  "javascript",
  "html",
  "vue"
],
"eslint.options": {
  "plugins": ["html"]
}
```

5.5.2.4 Vue Language Features (Volar) (必装)

vue 开发辅助插件。支持.vue 文件语法高亮、代码提示、Emmet、错误检测、格式设置等

5.5.2.5 vsc-commitizen (必装)

格式化 Commit message 的插件。为保证 commit message 格式正确，以便生成 changelog，推荐使用此插件。

5.5.2.6 Microfontends Simulator (必装)

微前端环境模拟器。用于在本地模拟微前端环境，进行微应用调试。

5.5.2.7 Component Helper (必装)

组件开发辅助。用于在编码时提供组件代码辅助编写。

5.5.2.8 Low Code (必装)

低代码开发辅助。可视化配置，快速生成代码。

5.5.2.9 Application Creator (必装)

应用创建。选择开发模板快速创建应用。

5.5.2.10 whatchanged

changelog 生成插件

发布时，必须包含当前版本的 CHANGELOG 文件

5.5.2.11 Git History

git 历史记录查看、对比

5.5.2.12 vue-style-beautify

vue 文件中样式格式化、优化 css 顺序

5.5.2.13 Project Manager

多项目切换

5.6 代码仓库配置

5.6.1 安装及配置 Git

1. 下载并安装：<https://git-scm.com/>
2. 登录 GitLab：<http://192.168.5.15/>
3. 点击右上角头像 -> 设置，查看 Commit email，如果没有就设一个



4. 打开终端，配置：

```
git config --global user.name "用户名"
```

```
git config --global user.email "邮箱"
```

用户名为 GitLab 的登录名，邮箱为你的 Commit email

5.6.2 SSH 密钥设置

用于使用 SSH 协议连接仓库

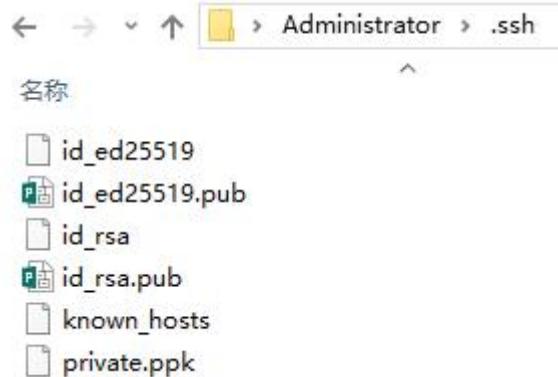
1. 打开“Git Bash”或“命令提示符”，使用命令生成密钥：

```
ssh-keygen -t ed25519 -C "email@example.com"
```

此处邮箱为前端开发人员 gitlab 账号的 Commit email

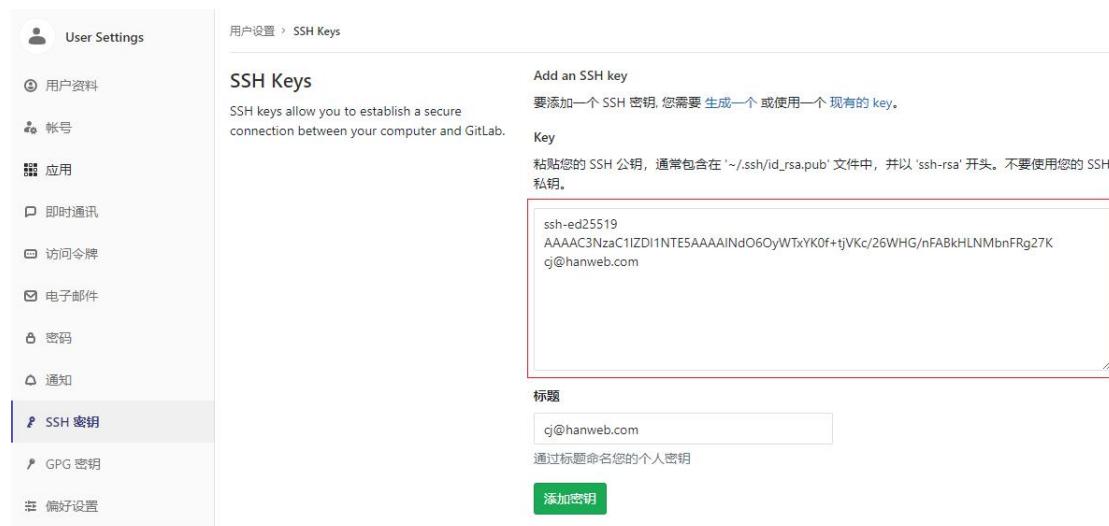
一路回车

2. 默认密钥被生成到当前用户目录下的.ssh 目录内



3. 打开公钥文件 id_ed25519.pub 文件，复制内容

4. 重新回到 GitLab 的设置，点击左侧“SSH 密钥”，将内容粘贴到 Key 中，点击“添加密钥”



User Settings > SSH Keys

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

要添加一个 SSH 密钥, 您需要 [生成一个](#) 或使用一个 [现有的 key](#).

Key

粘贴您的 SSH 公钥, 通常包含在 `~/.ssh/id_rsa.pub` 文件中, 并以 'ssh-rsa' 开头。不要使用您的 SSH 私钥。

```
ssh-ed25519
AAAAC3NzaC1I2D1NTESAAAAInD06OyWTxYK0f+tjVKc/26WHG/nFABkHLNMbnFRg27K
cj@hanweb.com
```

标题

cj@hanweb.com

通过标题命名您的个人密钥

添加密钥

5.7 调试工具安装

5.7.1 Vue Devtools (必装)

chrome 中安装：

<https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnaanhbledajbpd>

6 开发指南

6.1 基础

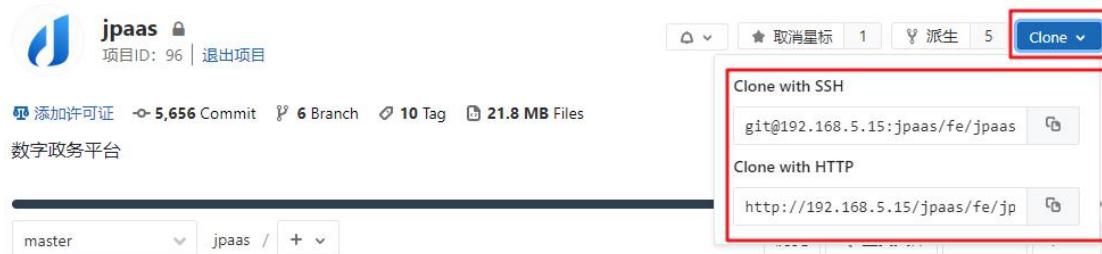
6.1.1 新建项目

1. 获取仓库的 gitlab 地址

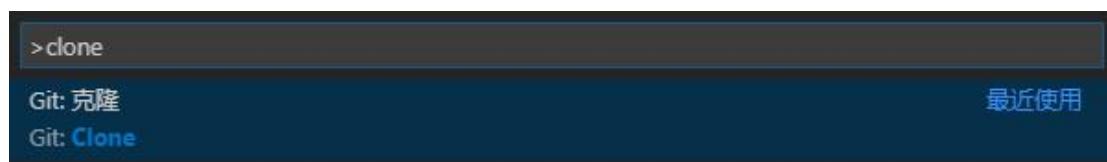
向配置工程师（CM）申请创建 gitlab 仓库或获得已有仓库地址

2. 浏览器进入项目的 gitlab 首页

3. 点击右上角“clone”，复制 SSH 地址（需要先配置 SSH 秘钥，参考 [SSH 密钥](#)）



4. vscode 中快捷键 ctrl+shift+p 打开命令输入框，输入 clone，找到 Git:克隆



5. 选择存储库位置，选择之前建立的[工作空间目录](#)，如“workspace”

6. 将 SSH 地址粘贴进去，开始下载代码，并添加到当前工作区

7. 如果是空项目，使用 [JPaaS CLI 工具](#) 或 [应用创建插件 Application Creator](#) 对项目初始化。

CLI 工具：终端进入项目所在目录，执行：jpaas init，选择“JPaaS 微应用”

应用创建插件：点击 vscode 下方“创建应用”，选择需要的应用模板

6.1.2 安装

终端进入项目根路径下执行：

```
pnpm install
```

(统一使用 pnpm，不要使用其他包管理工具，可能会跟自动构建安装的依赖版本不一致)

6.1.3 项目配置

打开 vue.config.js，使用 defineConfig 创建配置，没有特殊需求不需要配置。

```
const { defineConfig } = require('jpaas-micro-core/tools/defineConfig')

module.exports = defineConfig({
  pages: {
    'microcomponents/contact': 'src/microcomponents/contact',
    'microcomponents/notice': 'src/microcomponents/notice'
  }
})
```

devServer: 本地开发服务配置

pages: multi-page 模式配置，可用于配置微组件入口

configureWebpack: webpack 配置，目前支持如下配置

configureWebpack.resolve.alias: 别名

configureWebpack.externals: 外部扩展。通过 script 从外部获取这些扩展依赖

configureWebpack.plugins: webpack 插件

publicPath: 部署应用包时的基本 URL

transpileDependencies: 需要编译的依赖

6.1.4 配置本地开发服务器

- 默认接口请求被代理到公司内开发环境 `http://jpaasdev2.hanwbe.com`，可根据项目调试环境自行配置

修改 `/devserver.config.js` 文件，配置 `devServer`：

```
module.exports = {
  port: 8003,
  compress: true,
  // host: 'www.test.com',
  https: false,
  // 配置允许跨域
  headers: {
    'Access-Control-Allow-Origin': '*',
    'Access-Control-Allow-Methods': 'GET,POST,PUT,OPTIONS'
  },
  client: {
    overlay: false
  },
  proxy: {
    '/api-gateway': {
      target: 'https://jpaasdev2.hanweb.com',
      ws: false,
      changeOrigin: true
    },
    '/ucenter_files': {
      target: 'https://jpaasdev2.hanweb.com',
      ws: false,
      changeOrigin: true
    },
    '/workflow_files': {
      target: 'https://jpaasdev2.hanweb.com',
      ws: false,
      changeOrigin: true
    }
  }
}
```

portal: 服务端口

compress: gzip 压缩

https: https 访问模式

headers: 请求头信息

client.overlay: 编译错误或警告时全屏覆盖

proxy: 代理配置

option.target: 代理目标基础路径

option.ws: 是否代理 websocket

option.changeOrigin: false 时 , 请求头中 host 仍然是浏览器发送过来的 host ; true 时 , 发送请求头中 host 会设置成 target 的值

option.pathRewrite: 路径重写

更多详见 : <https://cli.vuejs.org/zh/config/#devserver-proxy>

配置成功后启动本地开发服务 , 将按配置代理到后端 API 服务

注意 :

- 此处代理配置与线上部署无关 , 线上部署时需要运维人员进行代理配置
 - 若直接代理某个后端开发 pc 上的服务 , 没有使用接口网关 , 则接口的 URL 不会以 /api-gateway 开头 , 需要配置 pathRewrite: {'^/api-gateway': ''} 进行去除
 - 开发过程中可代理到公司内 MockServer 服务器 , 参见 : [Mock 测试](#)
- 联调阶段 , 改回开发环境的服务地址

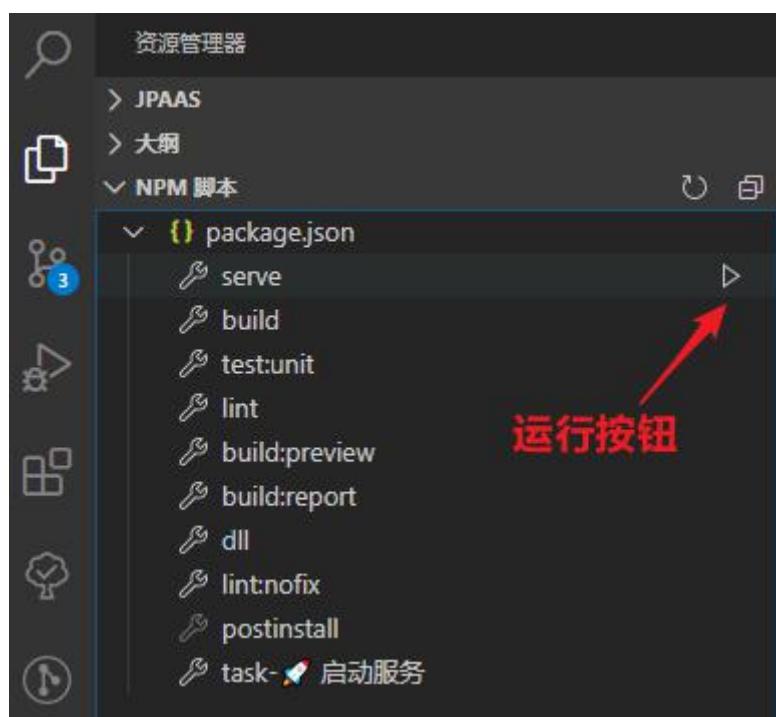
6.1.5 启动服务

启动本地开发服务有两种方式 :

1. VSCode 运行 NPM 脚本 (推荐)
2. 命令行工具 , 适合开发过程中 , 快速启动

6.1.5.1 VSCode 执行脚本 (推荐)

1. 左侧资源管理器 , 展开 NPM 脚本面板
2. 点击“serve”后的运行按钮



6.1.5.2 终端执行命令

1. 打开终端面板 , 当前项目所在路径下 , 输入 pnpm run serve , 即可运行
2. 运行成功后 , 按住 ctrl 点击链接 , 即可在浏览器中浏览

```
You may use special comments to disable some warnings.  
Use // eslint-disable-next-line to ignore the next line.  
Use /* eslint-disable */ to ignore all warnings in a file.  
App running at:  
- Local: http://localhost:8001/complat4/  
- Network: http://192.168.217.1:8001/complat4/  
  
Note that the development build is not optimized.  
To create a production build, run npm run build.
```

6.1.6 本地开发模式

每个产品即一个微应用，本地开发时以开发模式运行。可单独访问，**不需要依赖 jpaas 基座应用（jpaas-portal）及基础应用（jpaas-common）**，发布到测试或生产环境中（已编译）则无法单独访问。



6.1.6.1 登录

开发模式包含一个基本的登录界面，编译到生产环境时会去除。使用开发环境，可以点击“创建账号”，使用管理员账号自行创建开发账号

本地开发登录

默认使用开发环境dev2账号登录

用户名

密码

登 录 创建账号

6.1.6.2 导航

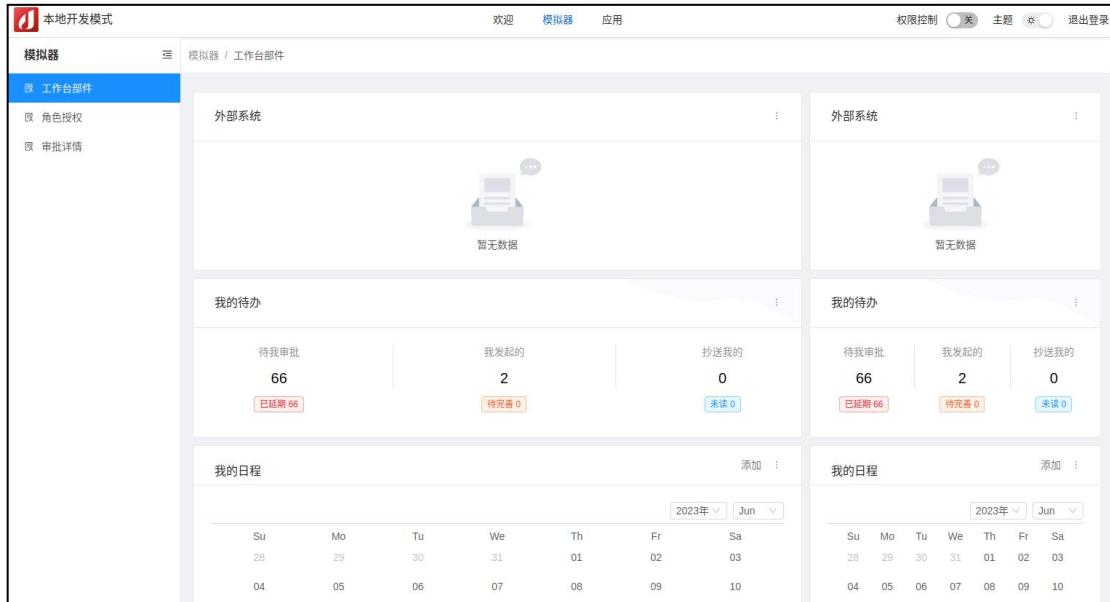


本地开发时使用开发模式的头部导航：

- 导航菜单自动加载本地一级路由及工作台组件开发预览页，方便进入页面进行开发调试；线上部署时，该头部将被自动替换成平台统一导航。
- 权限控制开关用于控制是否加载路由及菜单权限，参考：[路由和菜单](#)
- 主题开关用于切换明暗两种主题，方便开发适配

6.1.6.3 模拟器

本地开发基础应用扩展的模拟环境。包括：工作台部件、角色授权、审批详情等



6.1.7 路由和菜单

路由和菜单是组织起一个应用的关键骨架。

路由配置存放在/src/config/router 下，各应用按照应用名称（appCode）编写路由配置，**每个应用只有一套菜单**

6.1.7.1 配置

- path 路由地址，须填写从应用第一级开始的完整路由地址
- name 路由名称，应用内唯一，按照 path 编写，分隔符为‘-’
- component 路由匹配的组件，
- redirect 重定向，可设为“default”，即默认路径，路由重定向到第一个可见的子路由地址
- hideChildrenInMenu 用于隐藏不需要在菜单中展示的子路由。
- hidden 可以在菜单中不展示这个路由，包括子路由。
- meta.title 和 meta.icon 分别代表生成菜单项的文本和图标。

- meta.target 同 a 标签的 target 属性，指定页面在浏览器中的打开方式
- meta.permission 用来配置这个路由的权限，如果配置了将会验证当前用户的权限，并决定是否展示，具体配置参考：[路由规范](#)
- meta.hiddenHeaderContent 可以强制当前页面不显示 PageHeader 组件中的页面带的面包屑和页面标题栏
- meta.overlay 打开详情页的视图模式设置（自动 keepAlive 上一页），page：页面模式，full：全屏模式，用于从详情页后退回列表页时，能回到进入前的状态
- meta.pageReturn 页面返回按钮，true：显示返回按钮，点击返回上一页，false：不显示返回按钮，function：点击事件，object：兼容旧版本，object.click 点击事件，string：目标路由 path
- meta.showMenu 强制显示左侧导航栏，当只有一个可见菜单时，默认不显示左侧导航栏

6.1.7.2 实例

1. 菜单跳转到外部地址

```
{  
  path: 'https://pro.loacg.com/docs/getting-started',  
  name: 'docs',  
  meta: {  
    title: '文档',  
    target: '_blank' // 打开到新窗口  
  }  
}
```

2. 新增布局

在脚手架中我们通过嵌套路由来实现布局模板。配置中第一级数据就是我们的布局，如果

你需要新增布局可以再增加一个新的第一级配置

```
{  
  path: '/new-router',  
  name: 'newRouter',  
  redirect: '/new-router/ahaha',  
  component: RouteView, // 使用 RouteView 布局组件  
  meta: { title: '仪表盘', keepAlive: true, permission: [ 'dashboard' ] },  
  children: [  
    {  
      path: '/new-router/ahaha',  
      name: 'ahaha',  
      component: () => import('@/views/dashboard/Analysis'),  
      meta: { title: '分析页', keepAlive: false, permission: [ 'ahaha' ] }  
    }  
  ]  
}
```

3. 其他配置

```
{  
  hidden: true,  
  hideChildrenInMenu: true,  
  meta: {  
    icon: 'dashboard',  
    title: '菜单标题',  
    keepAlive: true,  
    permission: [ 'admin' ]  
  }  
}
```

- hidden: 当前路由从菜单中隐藏，但是路由然后可以访问到。
- hideChildrenInMenu: 当前路由显示为没有子路由的菜单 Menu.Item。并且从菜单上隐藏该路由下的所有子菜单。
- meta.icon: 当前路由在菜单下的图标名。
- meta.title: 当前路由在菜单和面包屑中的名称

- meta.keepAlive: 缓存页面
- meta.permission: 允许展示的权限，不设则都可见

4. 默认路径配置

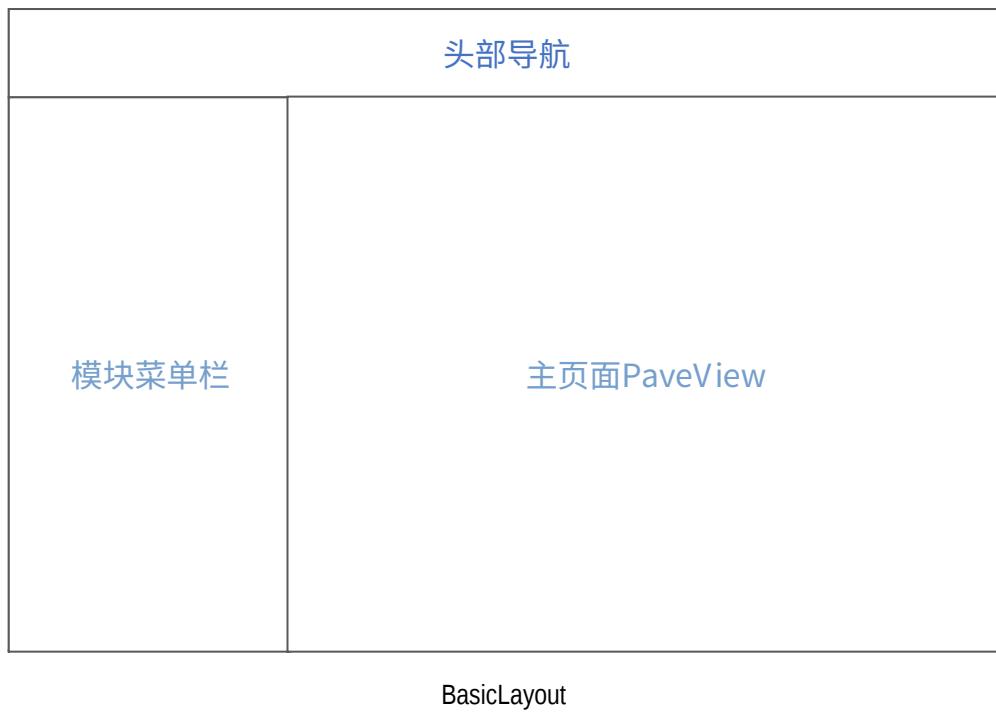
```
{  
  path: '/website',  
  name: 'website',  
  component: BasicLayout,  
  redirect: '/website/default', // default 为当前应用的默认页（第一个有权限的页面）  
  meta: {  
    title: '站群管理'  
  },  
  children: [...]  
}
```

6.1.8 页面布局

页面整体布局是一个产品最外层的框架结构，往往会包含导航、页脚、侧边栏、通知栏以及内容等。在页面之中，也有很多区块的布局结构。在真实项目中，页面布局通常统领整个应用的界面，有非常重要的作用。

通用布局，都放在 jpaas-component/layouts 目录中：

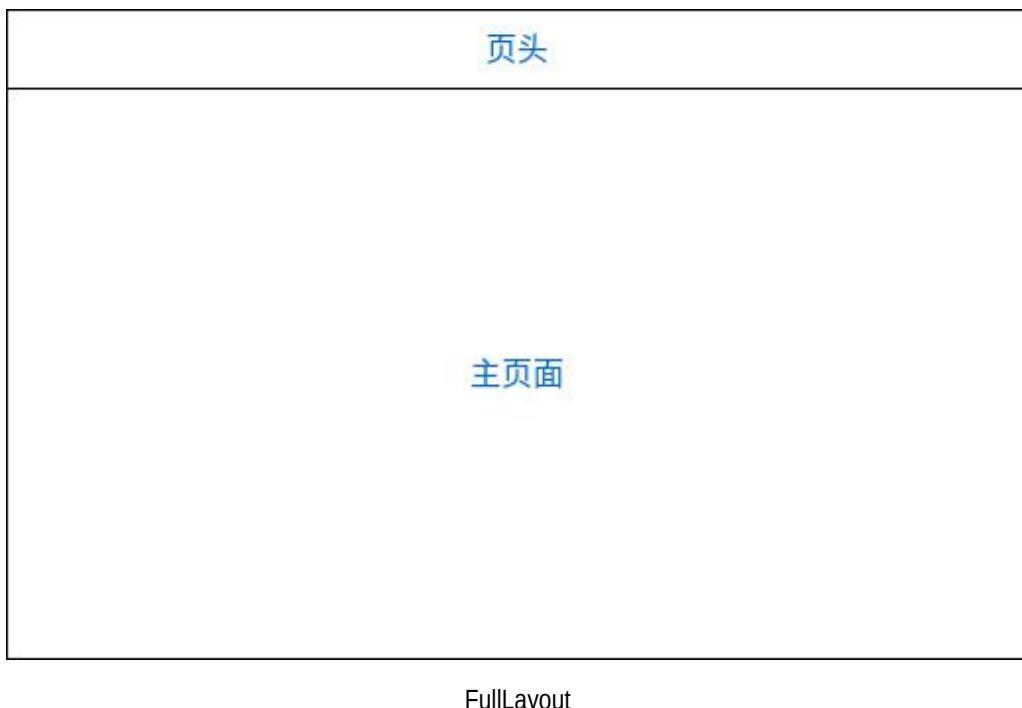
- BasicLayout：标准布局，包含了头部导航，侧边菜单和主页面（PageView）
- PageView：基础页面布局，包含了当前位置，和 HRouterView
- HRouterView：路由视图，可根据配置开启 KeepAlive，用于自定义布局中嵌套路由
- FullLayout：全屏布局，包括页头和主页面



BasicLayout



PageView



FullLayout

6.1.8.1 打开全屏页面

如：在列表页打开一个详情页，当从详情页返回列表页，仍保持之前的状态

- 全屏页面的路由应设为一级路由，建议采用全屏布局组件 FullLayout
- 无需多添加一级路由来设置布局，除非需要路由嵌套
- 全屏页面的 meta 下增加 overlay: 'full'

6.1.8.2 进入另一个页面视图

- **当前路由不要设置 keepAlive**
- 目标页面的 meta 下增加 overlay: 'page'，用于缓存其上一页
- 目标页面的返回按钮配置 pageReturn，pageReturn 类型：boolean、function、string、object

如果配置了 overlay: 'page'，pageReturn 默认值为 true

6.1.8.3 返回上一个页面

- 在配置了 overlay 的页面，其上一页被自动添加了 keepAlive (**不要自行配置 meta.keepAlive 参数，如已存在请删除**)
- 使用当前页面的 pageReturn 配置或调用 this.\$router.back() 返回上一页
- 若返回后，需要更新数据，则在上一个页面的 activated 钩子中，实现数据刷新

6.1.8.4 自适应页面高度

在页面组件的根元素中，可添加 class="fitheight"，将自动适配页面高度

6.1.8.5 定制布局 (BasicLayout)

可创建一个基于 BasicLayout 的定制布局组件。BasicLayout 提供了扩展能力。

1. 自定义侧边菜单

```
<template>
<basic-layout ref="BasicLayout" :menu="menu" :isLoading="isLoading"></basic-layout>
</template>

<script>
export default {
  data () {
    return {
      menu: {
        title: '评论管理',
        select: key => this.selectMenu(key),
        menus: [],
        defaultSelectedKey: null
      },
      isLoading: false
    }
  }
}
</script>
```

menu :

- title : 左侧菜单顶部标题
- select (事件) : 被选中时调用
- menus : 菜单项配置 (包含 title、key、meta)
- defaultSelectedKey : 默认选中的菜单 key

2. 扩展菜单插槽

```
<basic-layout ref="BasicLayout" mode="route">
<template #menuAction>
  action
</template>
<template #menuBottom>
  bottom
</template>
</basic-layout>
```

mode 默认为 component 模式，菜单项通过 menu 属性传入。若要扩展路由模式的菜单，则 mode 设为 route

3. API

参数	说明	类型	默认值
isLoading	是否加载中，若为 true，则不会加载右侧主页面组件	boolean	false
menu	菜单项，component 模式时有效	object	{ title: null, menus: [], defaultSelectedKey: null }
showMenu	强制显示菜单，若为 false，则按	boolean	false

	照 menu 或路由配置显示，可见 菜单只有一个则不显示		
mode	菜单模式 component：组件模式，route：路 由模式	string	component
menuAction	标题和菜单之间的位置插槽	slot	-
menuBottom	菜单下方位置插槽	slot	-

6.1.8.6 全屏布局 (FullLayout)

创建一个标准全屏布局的页面，包含页头和主页面部分。

由于全屏布局时，大部分都需要自定义页头部分，所以 FullLayout 一般在单文件组件中使用

```
<template>
  <full-layout>
    <template #headerLeft>
      <h-button>返回</h-button>
    </template>
    <template #title>
      这是一个标题
    </template>
    <template #headerRight>
      <h-button>查看</h-button>
    </template>
  </full-layout>
</template>
```

参数	说明	类型	默认值
headerHeight	页头高度	number	50
showHeader	是否显示页头	boolean	true
header	页头插槽，当整个页头都需要自定义时使	slot	-

	用，此时 headerLeft、title、headerRight 无效		
headerLeft	页头左侧插槽	slot	-
title	页头标题插槽	slot	-
headerRight	页头右侧插槽	slot	-
content	主页面插槽，也是默认插槽，若未设置则为路由视图	slot	-

6.1.9 编写页面

6.1.9.1 新增 vue 文件

采用文件扩展名为 .vue 的 single-file components(单文件组件)的形式编写业务页面及业务组件。

1. 在 src/views 下新建页面的 vue 文件，如果相关页面有多个，可以新建一个文件夹来放置相关文件
2. 加入菜单和路由的方式请参照 路由和菜单 中的介绍完成。加好后，访问路由中配置的路径或点击菜单就可以看到新增的页面了。
3. 布局及路由都配置好之后，回到之前新建的 vue 文件，可以开始写业务代码了

6.1.9.2 编写业务代码

.vue 文件是一个自定义的文件类型，用类 HTML 语法描述一个 Vue 组件。每个 .vue 文件包含三种类型的顶级语言块 <template>、<script> 和 <style>：

```
<template>
<div class="example">{{ msg }}</div>
</template>

<script>
```

```
export default {
  data () {
    return {
      msg: 'Hello world!'
    }
  }
}
</script>

<style>
.example {
  color: red;
}
</style>
```

1. 模版

每个 .vue 文件最多包含一个 <template> 块

2. 脚本

每个 .vue 文件最多包含一个 <script> 块

3. 样式

- 默认匹配 :`^.css$/` , 可以使用 lang 这个 attribute 来声明预处理器语言 , **公司内可使用**

less , 不要使用其他预处理语言

- 一个 .vue 文件可以包含多个 <style> 标签。

- <style> 标签可以有 scoped 或者 module 属性 (查看 scoped CSS 和 CSS Modules) 以帮助

你将样式封装到当前组件。具有不同封装模式的多个 <style> 标签可以在同一个组

件中混合使用。

- 任何匹配 .css 文件 (或通过它的 lang 特性指定的扩展名) 的 webpack 规则都将会运用

到这个 <style> 块的内容中

4. 自定义块

可以在 .vue 文件中添加额外的自定义块来实现项目的特定需求，例如 <docs> 块。 vue-loader 将会使用标签名来查找对应的 webpack loader 来应用在对应的块上。 webpack loader 需要在 vue-loader 的选项 loaders 中指定。

更多细节，查看[自定义块](#)

5. Src 导入

如果喜欢把 .vue 文件分隔到多个文件中，你可以通过 src 属性导入外部文件：

```
<template src="./template.html"></template>
<style src="./style.css"></style>
<script src"./script.js"></script>
```

更多内容参考：<https://vue-loader.vuejs.org/zh/spec.html>

6.1.10 组件库使用

前端开发平台基于 Ant Design Vue，可参考官方文档：[组件文档](#)

平台组件库 jpaas-component 增加了部分组件，或对一些组件进行了二次封装，脚手架中已默认安装，文档见：开发环境 dev2\前端组件，[前端组件文档](#)

6.1.10.1 组件引入

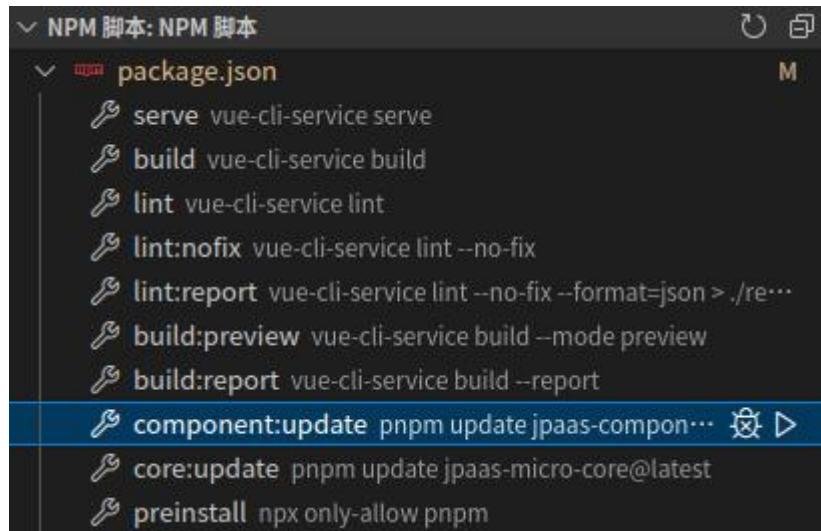
jpaas-component 组件按需引入方法，如：

```
import { HInput, HModal } from 'jpaas-component'
```

6.1.10.2 组件库升级

脚本方式：

1. 执行 component:update



2. 命令行方式

终端执行：pnpm update jpaas-component@latest

latest : 最新正式版

alpha : 内部测试版

beta : 公开测试版

注意：

- 不要手动修改 package.json 里的版本号
- 产品正式发布时，请检查组件库是否为正式版，测试版本在组件库正式版本发布后会删除，有可能造成安装失败

查看当前安装版本：pnpm ls jpaas-component

查看最新版本：pnpm view jpaas-component version

6.1.11 和服务端交互

为了方便管理维护，统一的请求处理都放在 @/src/api 文件夹中，并且一般按照 model 维度

进行拆分文件，如：

```
api/  
  user.js  
  permission.js  
  goods.js  
  ...
```

6.11.1 API 接口文档管理

API 接口文档由后端开发在设计阶段提供，设计评审时需评审接口文档，若未提供，不要同意通过评审

公司内部署了在线文档管理系统 YAPI，可实现项目文档管理、接口测试、Mock 测试：

<http://192.168.10.55:3000/>

禁止直接访问后端开发提供的 swagger，设计阶段要求后端开发将文档导入 YAPI
请使用真实姓名和公司邮箱进行注册使用

一般情况下由后端开发进行 swagger 导入和更新



The screenshot shows the 'Data Management' tab selected in a software interface. On the left, there's a 'Data Import' section with dropdown menus for 'Swagger' and 'Public Category', and a 'Sync Data' section with a 'Fully Covered' dropdown and a toggle switch for 'Enable URL Import'. Below these are instructions to click or drag files into a dashed upload area, and links for Swagger import (V2.0+) and command-line import. On the right, there's a 'Data Export' section with a dropdown for 'Export Method', a radio button for 'All Interfaces' (selected), and a 'Export' button.

6.1.11.2 编写 API 方法

- 导入 axios 实例。其中 `jpaas-component/utils/request.js` 是基于 axios 的封装，便于统一处理 POST, GET 等请求参数，请求头，以及错误提示信息等。它封装了全局 request 拦截器、response 拦截器、统一的错误处理、baseURL 设置等。
- 编写 API 方法，使用 axios 与服务端交互。`get` 请求时，参数使用 `params` 传递；`post` 请求时，参数使用 `data` 传递
- 使用名称 `xxxAPI`，包装 `model` 的所有 API 方法，`export` 导出

```
import { axios } from 'jpaas-component/utils/request'
```

```
const api = {
  tree: '/manager/district/find-by-pid-keyword',
  add: '/manager/district/add'
}

export const regionAPI = {
  // 获取树菜单数据
  tree (parameter) {
    return axios({
      url: api.tree,
      method: 'get',
      params: parameter
    })
  },
  // 新增行政区划
  add (parameter) {
    return axios({
      url: api.add,
      method: 'post',
      data: parameter
    })
  }
}
```

指定 Content-Type :

- application/x-www-form-urlencoded (默认)
- multipart/form-data
- application/json

修改示例 :

```
export function getInfo () {
  return axios({
    url: api.UserInfo,
    method: 'get',
    headers: {
      'Content-Type': 'application/json;charset=UTF-8'
    }
  })
}
```

6.1.11.3 调用 API 方法

导入 API 使用

```
<template>
<tree :data="loadData"></tree>
</template>

<script>
import Tree from '@/components'
import { regionAPI } from '@/api/orgauth/region'

export default {
  // 引入组件
  components: {
    Tree
  },
  data () {
    return {
      ...
    },
    methods: {
      loadTableData: parameter => {
        return regionAPI.tree(parameter)
          .then(res => {
            return res.data
          })
        ...
      }
    }
  }
}
</script>
```

6.1.11.4 错误处理

平台统一对 500、403、401 异常和接口返回的操作失败做了处理

若需要对正常接口返回的操作失败自行处理，使用 `ignoreError` 参数

```
uploadCustomPperms (parameter) {
```

```
return axios({
  url: api.uploadCustomPperms,
  method: 'post',
  data: parameter,
  ignoreError: true,
  headers: { 'Content-Type': 'multipart/form-data' }
})
```

then 中处理

```
roleAPI.uploadCustomPperms(form).then(res => {
  this.current = 1
  if (res.success) {
    this.status = true
    this.result = res.data.result
    file.onSuccess(res, file.file)
    file.status = 'done'
  } else {
    file.onError()
    this.status = false
    file.status = 'error'
    this.message = res.message
  }
})
```

6.1.11.5 vue 组件内请求 API

vue 组件可使用 `vm.$axios()`方法发起请求

只有当 API 接口地址不确定，如动态拼接，或在封装通用组件时接口地址为外部传入时，

才能使用该方法

```
this.$axios({
  url: '/common-worktop-server/manager/extendapp/find-worktop-app',
  method: 'get',
  params: { appType: 'tool' }
}).then(res => {
  if (res.success) {
```

```
this.appList = res.data
}
})
```

6.1.12 权限控制

6.1.12.1 应用菜单权限配置

默认由基础服务提供用户的应用权限查询，当权限需求复杂时，可由应用服务自行实现。

配置 app.config.js 文件

```
{
  "menuPermissionAPI": "/common-ucenter-server/manager/menu/find-menu-
permissions"
}
```

6.1.12.2 路由权限控制

通过路由配置，判断当前用户是否能够看到菜单或有权限访问页面。参考 [路由和菜单](#)

6.1.12.3 操作权限控制

当需要对页面上的展现和操作进行更精细的权限控制，可使用函数的方式进行灵活判断

```
▼ {roles: null,...}
  code: "200"
  ▶ data: {pageNo: 1, pageSize: 10, totalCount: 0, totalPage: 0, searchText: "", sortField: null,...}
    message: null
  ▶ permissions: ["cms:info:revoke", "cms:info:batchmodify", "cms:info:create", "cms:info:relative", "cms:info:archive",...]
    0: "cms:info:revoke"
    1: "cms:info:batchmodify"
    2: "cms:info:create"
    3: "cms:info:relative"
    4: "cms:info:archive"
    5: "cms:info:sort"
    6: "cms:info:move"
    7: "cms:info:showtime"
    8: "cms:info:infohistory"
    9: "cms:info:top"
    10: "cms:info:import"
    11: "cms:info:export"
    12: "cms:info:preview"
    13: "cms:info:remove"
    14: "cms:info:jirdpush"
    15: "cms:info:publish"
    16: "cms:info:applyaudit"
    17: "cms:info:checkcomment"
    18: "cms:info:modify"
    19: "cms:info:referencenews"
    20: "cms:info:timedpublish"
    21: "cms:info:operatorrecord"
    22: "cms:info:conditionalsend"
    23: "cms:info:send"
    24: "cms:info:check"
  roles: null
  success: true
  traceId: "831459e48418421d8418043a3def4c05"
```

通过 API 接口获得权限（页面的接口里的 permissions 属性），通过实例方法\$setPermissions

将权限注入组件，例如：

```
this.$setPermissions(permissions)
```

通过实例方法\$auth 进行权限判断，例如：

```
<span v-if="$auth('cms:info:modify')" class="text">新增</span>
```

6.1.13 路径

6.1.13.1 URL

通过浏览器访问的路径

\$BASE_URL : 应用根路径，即 publicPath，如访问 public 目录下的静态资源

```

```

@ : 指向 <projectRoot>/src , 如访问 assets 下的静态资源

(仅作用于模版中) ``

```
<style lang="less">
  .intro {
    background: url(@/assets/a.png) no-repeat;
  }
</style>
/
相对于当前文件的路径
..

```

6.1.13.2 文件/模块路径

在文件中 import 导入的路径

@ : 指向 <projectRoot>/src , 如 :

```
import { HSiderPanel } from '@/components'
```

/ 相对于当前模块的文件目录 , 如 :

```
import RouteView from './RouteView'
```

6.1.14 应用入口

应用入口主要位于头部的应用导航菜单中。

基础服务

审批中心
文件中心
工具箱
组织&授权
通知公告
消息中心
工作流
应用中心
文本分析
元数据

基础应用

工作流
审批中心
信息资源库
站群管理
区块链
评论管理
元数据

6.1.14.1 添加应用入口

1. 需要应用的后端服务在初始化时注册到平台中
2. 为用户设置该应用的权限
3. 系统管理员登录，进入系统设置中，设置导航菜单，将应用添加到菜单中

6.1.14.2 自定义角色设置

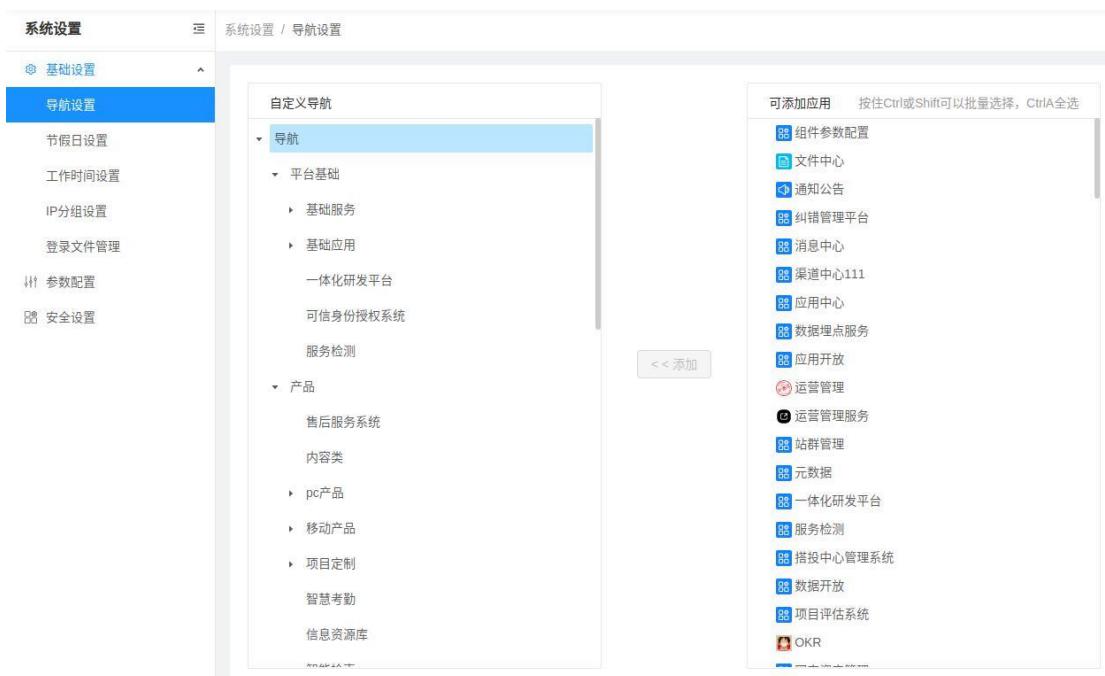
平台为业务应用提供了默认的角色设置页面。

但有时应用内的权限比较复杂，跟业务关系比较紧密，业务应用可以根据自己的需要开发

自己的角色设置页，参考：[基础应用扩展](#)

6.1.14.3 设置导航菜单

须使用系统管理员账号登录。进入“系统设置”》“基础设置”》“导航设置”，从右侧将系统中已注册的应用添加到左侧导航菜单中。也可在菜单中添加外部链接。添加的应用菜单受用户权限控制，外部链接则不受控制。



The screenshot shows the 'System Configuration' interface with the path 'System Configuration / Navigation Settings'. On the left, there's a sidebar with 'Basic Settings' and 'Navigation Settings' selected. The main area has a title 'Custom Navigation' and a tree view under 'Navigation' containing 'Platform Foundation' (with 'Basic Services' and 'Basic Applications'), 'Integrated Development Platform', 'Identity Authorization System', 'Service Detection', and 'Products' (with 'After-sales Service System', 'Content Categories', 'PC Products', 'Mobile Products', 'Customization Projects', 'Smart Attendance', and 'Information Resource Library'). To the right, there's a list of 'Addable Applications' with checkboxes next to them, including 'Component Parameter Configuration', 'File Center', 'Announcement Center', 'Error Correction Management Platform', 'Message Center', 'Channel Center 111', 'Application Center', 'Data Node Service', 'Application Opening', 'Operations Management', 'Operations Management Service', 'Cluster Management', 'Big Data', 'Integrated Research Platform', 'Service Detection', 'Investment Center Management System', 'Data Opening', 'Project Evaluation System', and 'OKR'. A button labeled '<< Add' is located between the navigation tree and the application list.

6.1.15 构建和发布

6.1.15.1 eslint 检查

开始构建前，先在本地进行 eslint 检查，并修复所有问题，自动化构建中如有 eslint 问题，则构建失败

自动修复命令：pnpm run lint

6.1.15.2 修改应用信息

修改 package.json , 应用英文名 name , 版本号 version , 描述 description

6.1.15.3 修改发布配置文件

修改 /docker/sample-fe.yaml 文件名为应用名 appCode-fe.yaml , 如 : jvms-fe.yaml

编辑文件 , 替换所有的 sample 为应用名 appCode , 如 :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jpaas-jvms-fe
  namespace: product
spec:
  replicas: <REPLICA_NUM>
  selector:
    matchLabels:
      app: jpaas-jvms-fe
  template:
    metadata:
      labels:
        app: jpaas-jvms-fe
    spec:
      containers:
        - name: jpaas-jvms-fe
          image: docker.hanweb.com/jpaas/jvms-fe:<BUILD_TAG>
          volumeMounts:
            - mountPath: /etc/nginx/nginx.conf
              name: nginx-config-volume
              subPath: path/to/nginx.conf
      volumes:
        - name: nginx-config-volume
      configMap:
        name: nginx-config-jpaas-jvms-fe
        items:
          - key: nginx.conf
            path: path/to/nginx.conf
```

6.1.15.4 构建

6.1.15.4.1 自动构建

在一体化研发平台中进行构建：<https://work.hanweb.com/jpaas/jdevops>

按照[使用指南](#)创建服务版本，并进行自动构建

6.1.15.4.2 本地编译

当项目开发完毕，只需要运行一行命令就可以打包你的应用：

```
pnpm run build
```

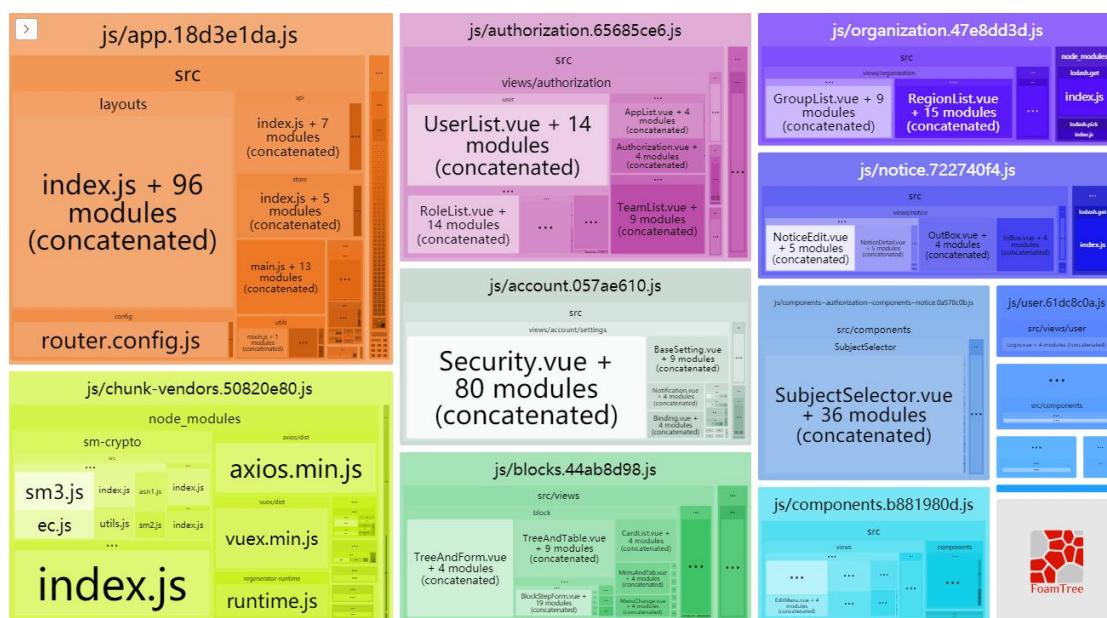
一般用于本地测试和优化，或定制开发，通常情况下应采用自动构建

构建打包成功之后，会在根目录生成 dist 文件夹，里面就是构建打包好的文件，通常是

.js、.css、index.html 等静态文件，也包括了项目根的 public/ 下的所有文件。

另外，使用 pnpm run build:report，可以在打包完成后，在 dist 目录下生成 report.html 打包报

告，可根据打包报告进行优化



6.1.16 部署

通常情况下，生产环境中应用由运维部署，测试环境中应用由测试部署，开发环境中应用由开发自行部署。以下两种方式提供给开发人员在本地或开发环境中部署应用参考，请根据具体环境选择。

6.1.16.1 Nginx 部署

此处以 nginx 举例：

```
gzip on;
gzip_min_length 50k;
gzip_types text/plain application/javascript application/x-javascript text/css application/xml text/javascript
application/x-httpd-php image/jpeg image/gif image/png application/vnd.ms-fontobject font/ttf font/opentype
font/x-woff image/svg+xml;
gzip_comp_level 5;

server {
    listen 80;
    server_name www.test.com;

    location ^~ /common {
        alias D:/git-workspace/common/dist;
        index index.html index.htm;
        try_files $uri $uri/ /common/index.html;
        add_header Pragma no-cache;
    }

    location /api-gateway {
        proxy_pass http://127.0.0.1:8080/;
    }
}
```

注意：必须添加 `add_header Pragma no-cache;` 来采用协商缓存，防止浏览器使用磁盘缓存，

导致页面无法更新

6.1.16.2 Rancher 部署

1. 部署服务

进入“工作负载”，点击“部署服务”



填写负载名称、从一体化研发平台构建好的镜像、命名空间

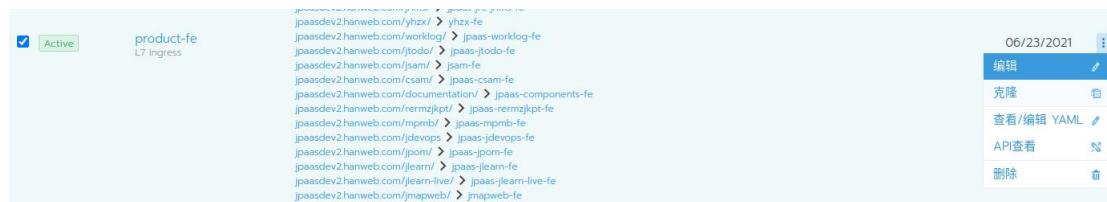
部署工作负载

名称 *	kbs-fe	类型	Deployment 部署无状态应用 1 个Pod	更多选项
Docker镜像 *	docker.hanweb.com/product/kbs-fe:vdev.108	命名空间 *	product	创建新的命名空间

启动配置好的工作负载

2. 添加负载均衡规则

进入“负载均衡”，编辑“product-fe”的负载均衡规则



新增“工作负载”，填写访问路径，并选择之前部署好的工作负载

目标后端

+ 服务	+ 工作负载
访问路径(如需使用后端重写功能,请查看下方标签/注释):	服务/工作负载
/jpaas	jpaas-portal-fe
/common	jpaas-common-fe
	容器端口 *
	80

6.1.17 升级微应用

JPaaS 微应用包括 jpaas-micro-core、jpaas-component 两个核心依赖。

通常情况下，只要按照需要更新安装即可：

```
pnpm update jpaas-micro-core@latest
```

pnpm update jpaas-component@latest

在某些大版本升级时，这两个核心依赖的版本需要匹配，并且可能对项目的目录结构、文件名、引用路径、使用方法等做出了改动，需要使用命令行工具进行升级操作：

确保安装了最新版本的命令行工具：npm install -g jpaas-cli

如已经安装，则升级到最新版本：npm update -g jpaas-cli

项目目录下执行升级命令：jpaas update

6.2 进阶

6.2.1 工具库/函数

6.2.1.1 动态加载脚本、样式

当依赖未提供 NPM 方式安装时，采用 vue 示例上的全局方法\$GlobalLoader 动态加载脚本文件：

将文件放到 public 目录下，**禁止直接加载 jpaas 外部文件（如某些常见库的 CDN 地址）**

```
this.$GlobalLoader.load([
  this.$BASE_URL + 'plug/highlight.pack.js',
  this.$BASE_URL + 'plug/dark.css'
]).then(() => {
  // 加载完成回调
  ...
})
```

离开页面，不再使用，须自行在 destroyed 钩子函数中自行移除脚本创建的 DOM 及全局变量，防止与系统其他模块冲突

语法：

```
this.$GlobalLoader.load(urls, [document], [async])
window.$GlobalLoader.load(urls, [document], [async])
```

参数：

urls	String Array<String>	静态资源地址
document	Document	要加载静态资源的 HTML 文档的根节点
async	Boolean	是否异步加载，默认：false

返回值：

Promise	Promise 对象
---------	------------

6.2.1.2 通讯加密

部分敏感内容，如个人密码之类的信息，在提交过程中需要进行加密（获取时脱敏）。目前采用国密 sm4、sm2

```
import { accountAPI } from '@/api/account/account'
import CE from 'jpaas-component/utils/ce.js'

const ce = new CE(pubKey)
const phone = ce.encrypt(values.phone)
const result = accountAPI.editInfo({ phone }, ce.sKey)
```

在接口请求时，将 sKey 放到 headers 中

```
...
```

```
editInfo (parameter, sKey) {  
    return axios({  
        url: api.editInfo,  
        method: 'post',  
        headers: { sKey },  
        data: parameter  
    })  
}
```

6.2.1.3 Base64 编码/解码

```
import { Base64 } from 'js-base64'  
  
...  
Base64.encode('dankogai')  
Base64.decode('ZGFua29nYWk=')
```

参考：<https://github.com/dankogai/js-base64>

6.2.1.4 生成唯一标识

前端生成唯一标识

安装：pnpm add nanoid

```
import { nanoid } from 'nanoid'  
  
const id = nanoid()
```

参考：<https://github.com/ai/nanoid/blob/HEAD/README.zh-CN.md>

6.2.1.5 数据校验

常见的校验规则，Pattern 包含一些正则校验规则，此外还有一些校验函数

导入/src/utils/validator.js

```
<template>
  <a-input placeholder="请输入您的姓名" v-decorator="['name', {rules: [{required: true, min: 2, message: '请输入账户姓名，姓名至少 2 字符'}, {Pattern.username2]}]]">

  <a-input v-decorator="['ipFrom${item.key}', {rules: [{required: true, message: '请填写起始 ip!'}, {max: 15, message: 'ip 地址最多输入 15 位'}, {validator: validateIP, message: 'ip 格式不正确'}], validateTrigger: 'blur'}]]">
</template>

<script>
import { Pattern, validateIP } from 'jpaas-component/utils/validator'

...
methods: {
  Pattern
}
...
</script>
```

6.2.1.6 LocalStorage 管理

```
new Vue({
  el: '#app',
  mounted: function() {
    this.$ls.set('foo', 'boo');
    //Set expire for item
    this.$ls.set('foo', 'boo', 60 * 60 * 1000); //expiry 1 hour
    this.$ls.get('foo');
    this.$ls.get('boo', 10); //if not set boo returned default 10
    let callback = (val, oldVal, uri) => {
      console.log('localStorage change', val);
    }
    this.$ls.on('foo', callback) //watch change foo key and triggered callback
    this.$ls.off('foo', callback) //unwatch
```

```
this.$ls.remove('foo');  
}  
});
```

更多参考 [vue-ls](#)

6.2.1.7 Cookie 管理

```
import Cookies from 'js-cookie'  
  
Cookies.set('login', true)  
let accessToken = Cookies.get('access-token')  
Cookies.remove('access-token')
```

更多参考：[js-cookie](#)

6.2.2 编写工作台部件

6.2.2.1 目录结构

- src/microcomponents/workbench-widgets 下创建部件目录
- 目录名以 UpperCamelCase 风格命名
- 必要的文件：manifest.json、index.js、icon.svg
- 必要的目录：screenshots，图片为 png 格式，多个文件以索引号
- 截图命名：如：0.png、1.png、2.png...

6.2.2.2 部件配置

- name：部件名称（唯一标识），UpperCamelCase 风格命名
- desc：功能描述，展示于订阅列表和部件详情页

- version : 版本号 , [主版本].[次版本].[修订版本]
- updated : 更新时间
- imageList : 截图数量

6.2.2.3 部件编写

1. vue 部件

```
<template>
<workbench-widget
  mode="box"
  title="天气预报"
  :isEdit="isEdit"
  appCode="weather"
>
  <iframe
    width="100%"
    :scrolling="'no'"
    height="60"
    :frameborder="0"
    :allowtransparency="true"
    :src="`/i.tianqi.com/index.php?c=code&id=12&color=%23&icon=3&num=${num}&site=14`"
    title=""
  />
</workbench-widget>

</template>

<script>
import WorkbenchWidget from 'jpaas-component/business/WorkbenchWidget'

export default {
  name: 'Weather',
  components: {
    WorkbenchWidget
  },
  props: {
    position: {
      type: String,
      required: true
    },
    isEdit: {

```

```
        type: Boolean,  
        default: false  
    }  
},  
computed: {  
    num () {  
        return this.position === 'left' ? 5 : 2  
    }  
}  
  
</script>
```

部件模板以 WorkbenchWidget 组件包裹

接收父组件传值：position（布局位置：‘left’|‘right’）、isEdit（工作台状态）

WorkbenchWidget 属性：

title*	标题	String	-
appCode*	名称（唯一标识）	String	-
mode	模式，normal-标准、inline-单行、box-盒子	String	'normal'
customOperationList	扩展菜单	Array	[]
linkOperation	扩展链接	Array	[]
isEdit	编辑状态（接收父组件传值）	Boolean	false

2. 非 vue 部件

工作台部件不限技术栈，可加载外部应用、其他技术实现的 web 部件

后端注册部件，需要配置 frameUrl，指定部件的入口地址

部件入口页面中需要编写部件的声明周期函数：

```
var render = (position) => {
  // 这里可以编写加载成功后的回调
  return Promise.resolve()
}

(global => {
  // MyWidget 为部件唯一标识，不能和其他部件同名
  global['MyWidget'] = {
    bootstrap: () => {
      return Promise.resolve()
    },
    mount: (props) => {
      var position = props.data.position
      return render(position)
    },
    unmount: () => {
      return Promise.resolve()
    }
  }
})(window)
```

6.2.3 基础应用扩展

为方便业务应用对基础应用中的一些功能进行定制，在基础应用中定义了一些扩展点，提供给业务应用的前端开发人员进行扩展开发。

扩展开发不需要修改基础应用，只需修改并部署在对应的业务应用中，基础应用会自动加载到扩展点并运行。

6.2.3.1 应用的角色授权

可定制应用对角色的授权页面，实现复杂的授权设置

组织授权 / 授权管理 / 用户管理

用户管理 封停管理 离退管理

内容管理平台

网站实施部案例库

大汉软件旧

Jrobot交互机器人旧

微联互动

微联互动旧

大汉软件

Jrobot交互机器人

中国政务服务掌办指数

网站集约化检测平台

大汉软件内网

管理角色

 网站管理员

网站管理员拥有对网站进行管理的所有权限

 栏目管理员 [设置栏目范围](#)

栏目管理员主要负责对网站下的栏目进行管理

 模板管理员

模板管理员拥有对网站中的模板库与展现样式套系进行管理的权限

信息角色 注：信息角色请选择网站具体的栏目范围，并编辑栏目维护高级权限，不设置则默认包含该网站内所有栏目下信息编辑权限。 信息维护员 [设置栏目信息权限](#)

信息维护员拥有对网站栏目中的信息进行编辑发布的权限，在给用户或群组设置该角色时，可设置详细的栏目信息维护权限

保存

保存并应用于其他网站

返回

1. /src 目录下创建 microcomponents/permission 目录
2. permission 目录下创建 index.js、PermissionSetting.vue

index.js 文件示例：

```
import PermissionSetting from './PermissionSetting.vue'

export default PermissionSetting
```

6.2.3.2 审批中心办件详情

可定制应用的办件详情页



编号: APPROVE-CSZDGL2020120800001

孙举提交的测试字段关联(2020-12-08 09:58)

类型: 测试字段关联

等待 孙举处理

| 办件详情 | 编辑 |

单选框:	字典项2
文本框:	111111

| 办理流程 |

- 发起节点 孙举 (提交) 2020-12-08 09:58
申请流程
- 流程节点 孙举 (审批中)

[拒绝] [通过]

1. /src 目录下创建 microcomponents/approval 目录
2. approval 目录下创建 index.js (参考角色授权)、HandlePieceSetting.vue

index.js 文件示例：

```
import HandlePieceSetting from './HandlePieceSetting.vue'  
  
export default HandlePieceSetting
```

6.2.3.3 日志管理

可定制日志检索区

所属服务与应用：	全部	公共服务	资源库	国资委征集			
所属模块：	站点服务	栏目服务	信息服务	文件服务	标准管理	目录管理	收起全部 ^
	标签管理	接口归属	数据库归属	清洗规则	数据共享	Web采集	

操作人	操作名	操作描述					
<input type="text" value="请输入操作人姓名"/>	<input type="text" value="请输入操作名关键字"/>	<input type="text" value="请输入操作描述关键字"/>					
操作时间:	<input type="button" value="近3日"/>	<input type="button" value="近1周"/>	<input type="button" value="近1月"/>	<input type="button" value="开始日期 -"/>	<input type="button" value="结束日期"/>	<input type="button" value="查询"/>	<input type="button" value="重置"/>

<input type="button" value="导出报表"/>							
TraceID	操作名	操作描述	操作结果	操作时间	操作人(登录名)	IP	操作
828882488b5e402b99e0161af68d90af	新增信息资源【文件导入】	文件导入，新增信息入库【威海市工商联信息导入2100】	成功	2021-04-13 16:46:58	无(元)		详情
ddec86d6abe7434e94f33ebf2053a82d	新增信息资源【文件导入】	文件导入，新增信息入库【威海市工商联信息导入2084】	成功	2021-04-13 16:46:57	无(元)		详情
461d067a08ba4cfb81a460bcfb3bcf5d	新增信息资源【文件导入】	文件导入，新增信息入库【威海市工商联信息导入2085】	成功	2021-04-13 16:46:57	无(元)		详情

1. /src 目录下创建 microcomponents/log 目录
2. log 目录下创建 index.js (参考角色授权)、LogSearchView.vue

index.js 文件示例：

```
import LogSearchView from './LogSearchView.vue'

export default LogSearchView
```

6.2.3.4 应用评论管理

可定制应用评论的菜单工具栏，如：增加切换网站组件



1. /src 目录下创建 microcomponents/comment 目录
2. comment 目录下创建 index.js (参考角色授权)、CommentView.vue

index.js 文件示例：

```
import CommentView from './CommentView.vue'

export default CommentView
```

6.2.3.5 工具箱

提供运维管理员对当前产品进行维护、升级、数据迁移等相关功能。

工具箱相关文件存放在/src/views/toolkit 目录下

路由配置：

```
import { ToolkitLayout } from 'jpaas-component/layouts'

export default {
  path: '/toolkit',
  name: 'toolkit',
  component: ToolkitLayout,
  redirect: '/toolkit/common',
  meta: {
    title: '工具箱'
  },
  children: [
    {
      path: 'common',
      name: 'common',
      component: () => import('@/views/toolkit/common')
    }
  ]
}
```

```

path: '/toolkit/common',
name: 'toolkit-common',
component: () => import('@/views/toolkit/Toolkit'),
meta: {
  title: '基础服务'
}
]
}

```

注：需使用 ToolkitLayout 布局

6.2.4 获取全局/应用内参数配置

用于获取全局/应用内参数配置。

系统管理员登录后，在“管理》参数配置”中进行设置，产品固定参数由产品初始化时自动

创建

系统设置		系统设置 / 参数配置			
		+ 新建分类			
		+ 新增			
参数配置	基础设置	参数名称	参数标识	所属分类	创建时间
全局配置	GLOBAL	地图配置	MAP	全局配置(GLOBAL)	2023-07-27 01:07:24
访问统计	JUBA	上传配置	UPLOADCONFIG	全局配置(GLOBAL)	2023-07-27 01:07:24
测试	test	访问路径	VISIT_URL	访问统计(JUBA)	2023-07-27 01:07:24
一体化研发平台	devops	预览配置	PREVIEWCONFIG	全局配置(GLOBAL)	2023-07-27 01:07:24
		手写签名地址配置	SIGNURATECONFIG	全局配置(GLOBAL)	2023-07-27 01:07:24
		主题配置	THEME	全局配置(GLOBAL)	2023-07-27 01:07:24
		测试	test	测试(test)	2023-07-27 01:07:24
		服务类别配置	SERVERTYPE	一体化研发平台(devops)	2023-07-27 01:07:24
		changelog	CHANGELOG	一体化研发平台(devops)	2023-07-27 01:07:24
		aaad	dddd	全局配置(GLOBAL)	2023-07-27 01:07:24

前端调用 /common-basesettings-server/manager/config/findConfig 接口，获取对应参数，接口参数

名为 params，参数值为 “[参数分类]_[参数标识]” , 多个参数以逗号分隔，如：

/common-basesettings-

server/manager/config/findConfig?params=GLOBAL_UPLOADCONFIG,GLOBAL_PREVIEWCONFIG,GLOB

AL_MAP

全局参数，可以直接从 vuex 的 global 模块获得

6.2.5 动态控制路由

通过内置函数\$addHiddenPath、\$removeHiddenPath 或状态控制 action 方法 addHiddenPath、removeHiddenPath 动态控制路由，以及路由菜单的显示、隐藏。

例如：通过查询服务的可用状态，设置是否显示相关菜单及路由

```
if (!store.state.global.serviceStates['common-mpush-server']) {  
    store.dispatch('addHiddenPath', '/account/notification')  
}  
if (!store.state.global.serviceStates['common-basesettings-server']) {  
    store.dispatch('addHiddenPath', ['/sysconf/Holiday', '/sysconf/Worktime', '/sysconf/IPList'])  
}
```

```
mounted() {  
    this.$addHiddenPath('/account/notification')  
    this.$addHiddenPath(['/sysconf/Holiday', '/sysconf/Worktime', '/sysconf/IPList'])  
}
```

6.2.6 查看前端应用版本号

在浏览器中查看线上各前端应用版本号：

- 基座版本：/jpaas/version.json
- 微应用版本：/[应用 appCode]/version.json

6.2.7 常用脚本

终端执行命令：pnpm run [脚本]

脚本	说明
serve	启动本地开发服务

start	启动本地开发服务
lint	eslint 检查并修复
lint:nofix	eslint 仅检查
lint:report	eslint 仅检查，输出报告（sonar 中使用）
build	编译
build:preview	编译为预览版
build:report	编译并输出 HTML 报告到 dist 目录下，可用于优化打包
update:component	升级组件库到最新版本
update:core	升级微前端内核到最新版

6.2.8 全局变量/方法

vm 为 vue 实例

变量 / 方法	说明
window.contextPath	系统环境路径，通常为“/jpaas/”
window._platform	系统变量，由系统管理员在“登录文件管理”中配置： <pre>var _platform = { 'title': 'dev2', 'logo': '/ucenter_files/tempfile/logo/logo.svg', 'favicon': '/ucenter_files/tempfile/logo/favicon.png', 'globalHeaderConfig': { 'showContact': true, 'showNotice': true } }</pre>
window.globalConfig	全局配置，包括：upload、preview、map、theme、signurate 可由系统管理员在系统设置=》参数配置=》全局配置中设置
vm.\$BASE_URL	应用根路径，通常为“/[应用唯一标志]”

vm.\$APP_CODE	应用唯一标志
window.\$GlobalLoader.load(urls, [document], [async]) vm.\$GlobalLoader.load(urls, [document], [async])	动态加载文件
vm.\$addHiddenPath([urls])	动态隐藏路由
vm.\$removeHiddenPath([urls])	动态取消隐藏路由
vm.\$ls	LocalStorage 管理
vm.\$axios	Axios 实例

6.3 开发辅助

6.3.1 浏览器开发者工具

F12 或 Ctrl+Shift+I 打开浏览器的开发者工具

6.3.1.1 元素 (Element) 面板

审查 DOM 元素和样式，包括查看 DOM 树、元素样式、布局、时间监听、属性等

可通过点击 DOM 树上的元素或通过左上角  箭头工具，选中想要检查的元素

右侧面板将显示当前元素的样式、计算样式、布局、时间监听、属性等信息

样式 计算样式 布局 事件监听器 DOM 断点 属性 无障碍功能

过滤 :hov .cls + □

```
element.style {  
}  
.htoolbar .left .ant-btn, .htoolbar .left .ant-calendar-picker, .htoolbar .left .ant-input, .htoolbar .left .ant-select, .htoolbar .left .h-search {  
    margin-right: 10px;  
}  
.ant-btn:not([disabled]):hover {  
    text-decoration: none;  
}  
.htoolbar .left .ant-btn, .htoolbar .left .ant-calendar-picker, .htoolbar .left .ant-input, .htoolbar .left .ant-select, .htoolbar .left .h-search {  
    margin-right: 10px;  
}  
.ant-btn:not([disabled]):hover {  
    text-decoration: none;  
}  
.htoolbar .left .ant-btn, .htoolbar .left .ant-calendar-picker, .htoolbar .left .ant-input, .htoolbar .left .ant-select, .htoolbar .left .h-search {  
    margin-right: 10px;  
}
```

6.3.1.2 控制台 (Console) 面板

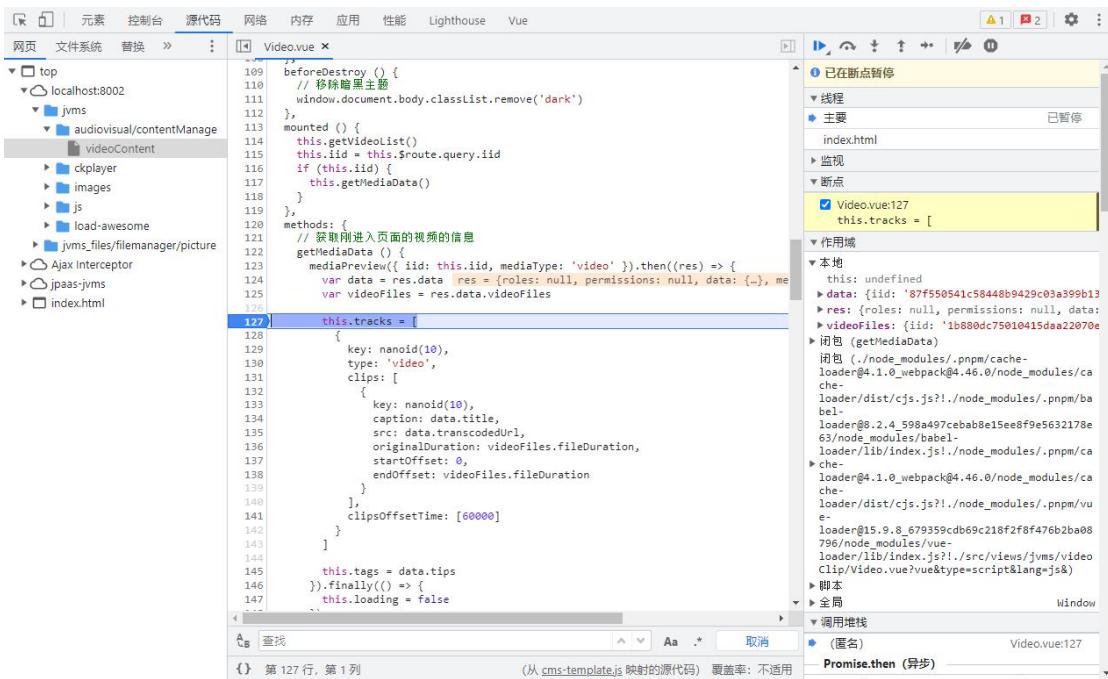
查看日志诊断信息帮助你调试 web 应用。

输入命令和页面或 DevTools 进行交互



6.3.1.3 源代码 (Sources) 面板

可进行代码查看和断点调试



```

109 beforeDestroy() {
110   // 移除暗黑主题
111   window.document.body.classList.remove('dark')
112 }
113 mounted() {
114   this.getVideoList()
115   this.iid = this.$route.query.iid
116   if (this.iid) {
117     this.getMediaData()
118   }
119 }
120 methods: {
121   // 新歌刚进入页面的视频的信息
122   getMediaData() {
123     mediaReview({ iid: this.iid, mediaType: 'video' }).then((res) => {
124       var data = res.data | res = {roles: null, permissions: null, data: {}}, me
var videoFiles = res.data.videoFiles
125       ...
126     })
127     this.tracks = [
128       {
129         key: nanoid(10),
130         type: 'video',
131         clips: [
132           {
133             key: nanoid(10),
134             caption: data.title,
135             src: data.transcodedUrl,
136             originalDuration: videoFiles.fileDuration,
137             startOffset: 0,
138             endOffset: videoFiles.fileDuration
139           }
140         ],
141         clipsOffsetTime: [60000]
142       }
143     ]
144     this.tags = data.tips
145   }).finally(() => {
146     this.loading = false
147   }
148 }
149 
```

Ctrl+P , 输入文件名可以快速定位到想要调试的文件

6.3.1.4 网络 (Network) 面板

Network面板即网络面板，用于记录页面上网络请求的详情信息，根据它可进行网络性能优化。

元素 控制台 源代码 网络 内存 应用 性能 Lighthouse Vue

过滤 反转 隐藏数据网址 全部 Fetch/XHR JS CSS 图片 媒体 字体 文档 WS Wasm 清单 其他 有已拦截的Cookie 被屏蔽的请求 第三方请求

2000毫秒 4000毫秒 6000毫秒 8000毫秒 10000毫秒 12000毫秒 14000毫秒 16000毫秒 18000毫秒 20000毫秒 22000毫秒 2

名称	标头	载荷	预览	响应	启动器	时间	Cookie
/api-gateway/common-ucenter-server/manager/onlineluser							
find							
/api-gateway/common-worktop-server/manager/favourite							
findConfig?params=GLOBAL_UPLOADCONFIG,GLOBAL_PRE							
/api-gateway/common-basesettings-server/manager/config							
info?t#=1652429594511							
192.168.83.110/sockjs-node							
media-list							
/api-gateway/jpaas-jvms-server/manager/media							
previewId#=87550541c58448b429c03a39b13ee&media							
/api-gateway/jpaas-jvms-server/manager/media							
find-tree?parentId=0&classType=video							
/api-gateway/jpaas-jvms-server/manager/media/class							
list							
/api-gateway/jpaas-jvms-server/manager/transcoding/para							
find-tree?parentId=0&classType=video							
/api-gateway/jpaas-jvms-server/manager/media/class							
list							
/api-gateway/jpaas-jvms-server/manager/transcoding/para							
find-tree?parentId=0&classType=video							
/api-gateway/jpaas-jvms-server/manager/media/class							
media-list?classId=&ordenum=4&searchTitle=&pageSiz...							
/api-gateway/jpaas-jvms-server/manager/media							

第 12 项请求，共 50 项 | 已传输 20.1 kB，共 1.7 MB | 所选资源

点击请求记录，可以查看请求的详细信息和返回信息

6.3.1.5 应用 (Application) 面板

可以查看和修改本地存储 LocalStorage、Cookie 等

元素 控制台 源代码 网络 内存 应用 性能 Lighthouse Vue

应用

- 清单
- Service Workers
- 存储

存储

- 本地存储空间
- http://localhost:8002
- chrome-extension://nhpjggchkhv
- 会话存储空间
- IndexedDB
- Web SQL
- Cookie
- http://localhost:8002
- 信任令牌
- 兴趣群体

缓存

- 缓存空间
- 往返缓存

后台服务

- 后台提取
- 后台同步
- 通知
- 付款处理程序
- 定期后台同步
- 推送消息
- 报告 API

帧

top

密钥	值
jpaas_jvms_DEFAULT_FIXED_...	{"value":true,"expire":null}
jpaas_jvms_DEFAULT_FIXED_S...	{"value":true,"expire":null}
jpaas_jvms_access_token	{"value":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJleHAiOiE2NTI0NDY3NDUsImIhdCl6MTY1MjQyODc0NSwidXNlcm5hbWUiOiJai9...
jpaas_jvms_DEFAULT_MULTI_...	{"value":false,"expire":null}
jpaas_jvms_DEFAULT_LAYOUT...	{"value":"sidemenu","expire":null}
jpaas_jvms_DEFAULT_COLOR...	{"value":false,"expire":null}
jpaas_jvms_DEFAULT_THEME	{"value":"dark","expire":null}
jpaas_jvms_SIDEBAR_TYPE	{"value":true,"expire":null}
jpaas_jvms_DEFAULT_COLOR	{"value": "#1B99FF", "expire": null}
jpaas_jvms_DEFAULT_CONTE...	{"value": "Fixed", "expire": null}
loglevelwebpack-dev-server	SILENT
jpaas_jvms_DEFAULT_FIXED_...	{"value":false,"expire":null}

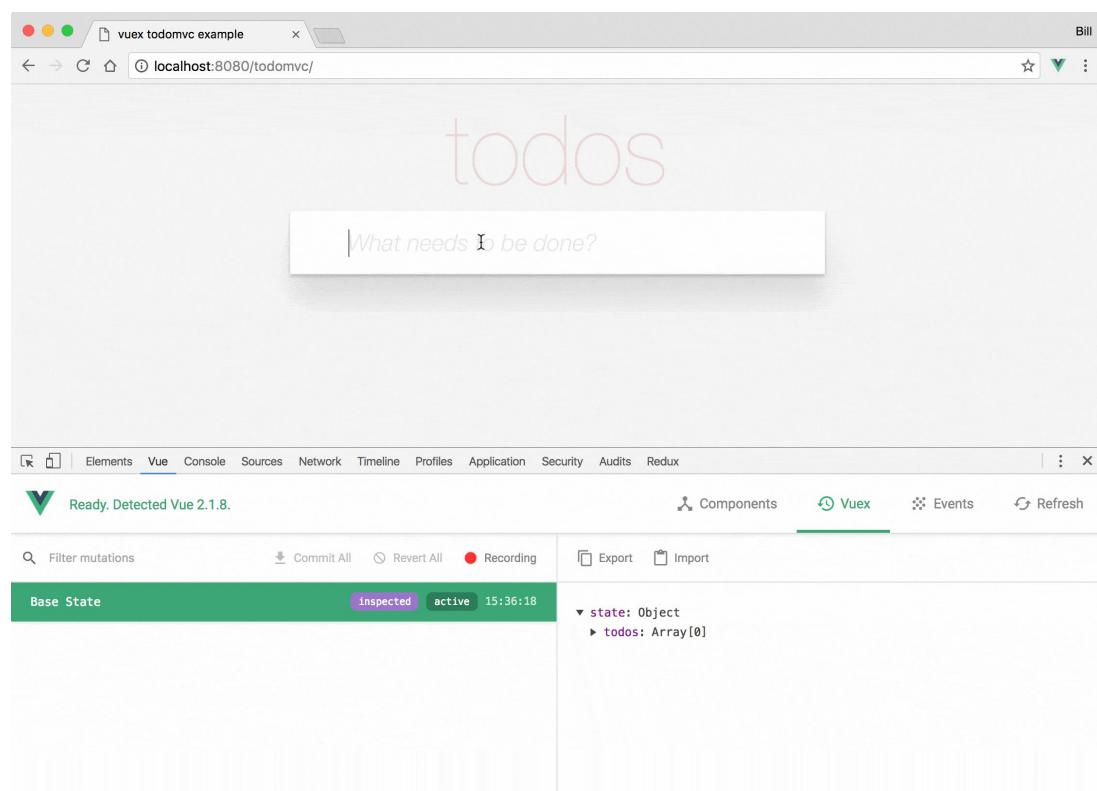
```
{
  value: true,
  expire: null
}
value: true
```

6.3.2 代码调试

6.3.2.1 Vue Devtools (必装)

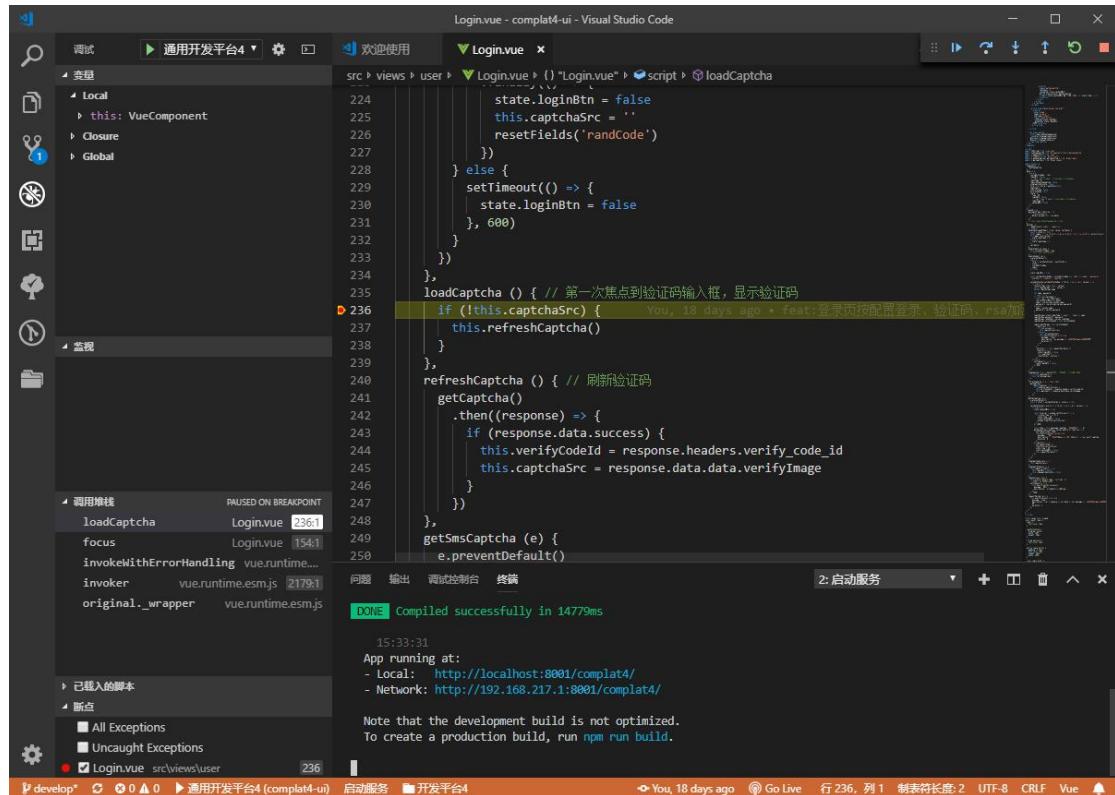
chrome 中安装：<https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnaanhbledajbpd>

使用 devtools 有很多好处，比如它能够让你能够实时编辑数据属性并立即看到其反映出来的变化。另一个主要的好处是能够为 Vuex 提供时间旅行式的调试体验



6.3.2.2 VSCode 中断点调试 (推荐)

1. 服务启动后，输入快捷键 F5 会打开一个 Chrome 运行调试
2. 可以在 vscode 中添加断点



插件配置文件/.vscode/launch.json 参考：请根据项目自行修改对应名称

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "chrome",
      "request": "launch",
      "name": "Chrome 调试",
      "url": "http://localhost:8002/common", // devserver 的访问地址
      "webRoot": "${workspaceFolder}/src", // 源代码目录
      "userDataDir": false,
      "sourceMaps": true,
      "sourceMapPathOverrides": {
        "webpack:///jpaas-common/src/*": "${webRoot}/*", // sourcemap 文件映射的源码
        "webpack:///./src/*": "${webRoot}/*"
      }
    }
  ]
}
```

注意：务必保证 sourceMap 路径与浏览器中一致

6.3.2.3 debugger 语句断点调试

可以直接在代码中使用原生的 debugger 语句，来增加断点。如果你选择了这种方式，请务必在调试完毕之后把 debugger 移除

```
<script>
  export default {
    data() {
      return {
        message: ""
      }
    },
    mounted() {
      const hello = 'Hello World!'
      debugger
      this.message = hello
    }
  };
</script>
```

6.3.2.4 线上调试

在生产环境中进行 sourcemap 调试

1. 在一体化研发平台中查看前端服务构建的 workspace 目录（流水线 -> 构建详情 -> Workspace）

[← 返回](#) 一体化研发平台 / 版本管理 / 产品 / 构建详情

最新运行 构建历史 [Workspace](#)

Dashboard > ba215ed27ff246c7ae4d549ff8798bff > #119 > Allocate node : Start > Workspace

↑ Up **Workspace**

Status / sourcemap / →

Console Output Workspace 打开 Blue Ocean

↳	1032.019ae408.js.map	2023-6-27 16:40:16	113.42 KB
↳	1053.7bc2530f.js.map	2023-6-27 16:40:16	43.96 KB
↳	128.e1da22db.js.map	2023-6-27 16:40:16	42.70 KB
↳	25.1dbf647c.js.map	2023-6-27 16:40:16	277.56 KB
↳	3171.4d866213.js.map	2023-6-27 16:40:16	51.95 KB
↳	3258.f71f0f32.js.map	2023-6-27 16:40:16	31.28 KB
↳	3563.b20d4c34.js.map	2023-6-27 16:40:16	37.12 KB
↳	3652.b0ac1b6c.js.map	2023-6-27 16:40:16	55.34 KB
↳	3712.8b195842.js.map	2023-6-27 16:40:16	2.94 MB
↳	418.d7290707.js.map	2023-6-27 16:40:16	140.16 KB
↳	420.7b840edd.js.map	2023-6-27 16:40:16	54.20 KB
↳	465.3ef55cc0.js.map	2023-6-27 16:40:16	542.23 KB
↳	4813.946f98dd.js.map	2023-6-27 16:40:16	57.47 KB
↳	4882.4bb1f39f.js.map	2023-6-27 16:40:16	37.11 KB
↳	5075.0986f407.js.map	2023-6-27 16:40:16	4.39 KB
↳	520.07e26384.js.map	2023-6-27 16:40:16	199.63 KB
↳	541.926b63e8.js.map	2023-6-27 16:40:16	31.59 KB
↳	5439.7ea9ef46.js.map	2023-6-27 16:40:16	20.45 KB

2. 进入 sourcemap 目录，拉到页面底部，点击“打包下载全部文件”

↳ ucenter-account.a084bba7.js.map	2023-6-27 16:40:16	308.10 KB	🔗	🔗
↳ ucenter-organization.caf3f104.js.map	2023-6-27 16:40:16	170.86 KB	🔗	🔗
↳ widgets.c3716af9.js.map	2023-6-27 16:40:16	170.88 KB	🔗	🔗
↳ workbench.c936c7b0.js.map	2023-6-27 16:40:16	44.82 KB	🔗	🔗

↳ (打包下载全部文件)

3. 解压缩 sourcemap 目录，并在其下创建 sourcemap/[appCode]目录，将所有文件移至其

下。如：sourcemap/sourcemap/common/

4. 使用 vscode 打开解压缩后的目录

5. 启动 Live Server 插件，端口 5500

6. 浏览器中即可进入源代码面板进行源码调试

6.3.3 Mock 测试

6.3.3.1 mock 数据配置

使用 YAPI 提供的 MockServer 进行 mock 测试

方式 1. mockjs

基于 [mockjs](#) , 跟 Mockjs 区别是 yapi 基于 json + 注释 定义 mock 数据 , 无法使用 mockjs 原有的函数功能

返回数据设置 [json](#)

JSON RAW

模板 预览

基于 mockjs 和 json5 使用注释方式写参数说明 ①, 具体使用方法请 [查看文档](#), “全局编辑”或“退出全屏”请按 F9

```
1~`{
2  "type": "object",
3  "properties": {
4    "code": {
5      "type": "string"
6    },
7    "data": {
8      "type": "object",
9      "properties": {}
10   },
11   "message": {
12     "type": "string"
13   },
14   "success": {
15     "type": "boolean"
16   }
17 },
18 "title": " JsonResult",
19 "description": "统一ajax返回值",
20 "$$ref": "#/definitions/ JsonResult"
21 }`
```

方式 2. json-schema

返回数据设置 [json-schema](#)

JSON RAW

模板 预览

导入 json

root object mock 统一ajax返回值 +
code string mock 备注 - x +
data object mock 备注 - x +
message string mock 备注 - x +
success boolean mock 备注 - x +

设置完成，通过设置反向代理使用 Mock

基本信息

接口名称:	获取菜单权限	创建人:
状态:	● 已完成	更新时间:
接口路径:	GET /common-ucenter-server/manager/menu/find-menu-permissions	
Mock地址:	http://192.168.10.55:3000/mock/37/common-ucenter-server/manager/menu/find-menu-permissions	

其他使用参考：<https://github.com/YMFE/yapi/blob/master/docs/documents/mock.md>

6.3.3.2 本地连接 mockserver 测试

本地通过配置 vue.config.js 中的 devServer 代理，在开发模式下访问 mock 接口

如：

```
'/api-gateway/common-ucenter-server': {  
  target: 'http://192.168.10.55:3000/mock/37',  
  ws: false,  
  pathRewrite: {  
    '^/api-gateway': ''  
  }  
}
```

例如 mock 地址为：

<http://192.168.10.55:3000/mock/37/common-ucenter-server/test>

将所有请求包含/api-gateway/common-ucenter-server 的请求代理到

<http://192.168.10.55:3000/mock/37>，即可

- 由于 mock 地址中没有“/api-gateway”部分，所以需要通过 pathRewrite 将请求中的“/api-gateway”去除
- 代理配置自上而下匹配，即若上个代理配置已经匹配，则不会再以后面的配置进行匹配。所以代理某个接口或服务，须将其配置放在其父级代理配置之上。

例如：

common-ucenter-server 下新增了一个接口 find-user-count，则：

- /api-gateway/common-ucenter-server/find-user-count 的配置必须在/api-gateway/common-ucenter-server 的配置（若存在）之上；
- /api-gateway/common-ucenter-server 的配置必须在/api-gateway 的配置之上

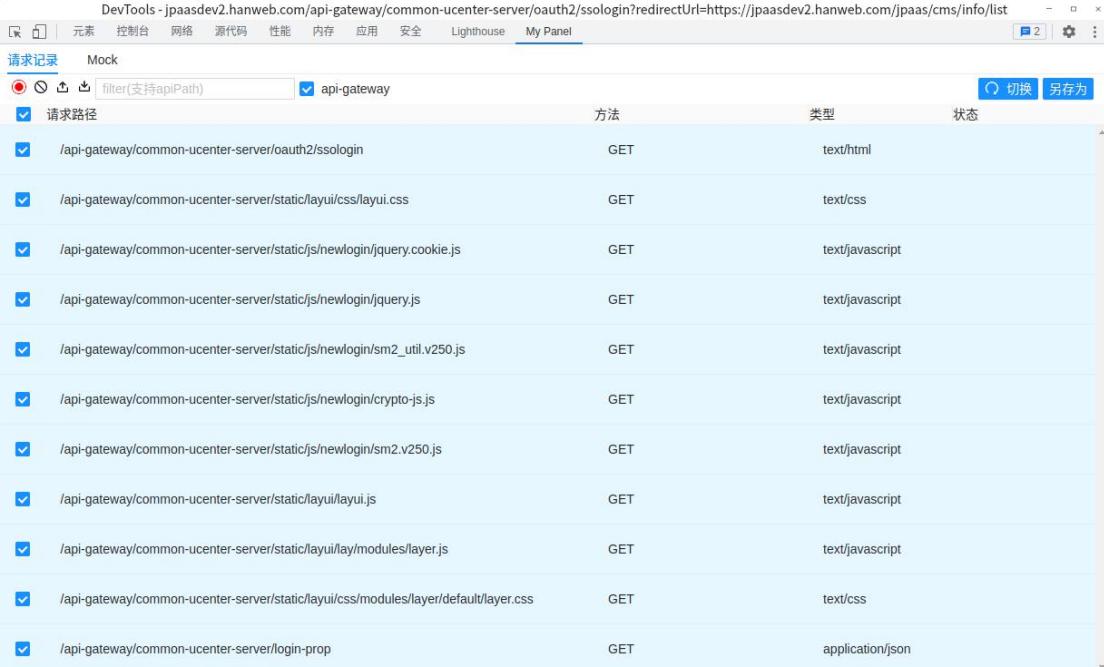
```
proxy: {
  '/api-gateway/common-hos-server': {
    target: 'http://192.168.5.133:30006',
    ws: false,
    changeOrigin: true,
    pathRewrite: {
      '^/api-gateway': ''
    }
  },
  '/api-gateway': {
    target: 'http://jpaasdev2.hanweb.com',
    ws: false,
    changeOrigin: true
  },
}
```

6.3.3.3 浏览器 mock 插件

LiveMock 提供接口请求录制及 mock 功能

打开开发者工具，切换到“请求记录”选项卡

刷新页面，会拦截到所有的请求



The screenshot shows the DevTools interface with the title "DevTools - jpaasdev2.hanweb.com/api-gateway/common-ucenter-server/oauth2/sslogin?redirectUrl=https://jpaasdev2.hanweb.com/jpaas/cms/info/list". The "Mock" tab is selected. A filter bar at the top has "filter(支持apiPath)" and "api-gateway" selected. Below is a table of requests:

请求路径	方法	类型	状态
/api-gateway/common-ucenter-server/oauth2/sslogin	GET	text/html	
/api-gateway/common-ucenter-server/static/layui/css/layui.css	GET	text/css	
/api-gateway/common-ucenter-server/static/js/newlogin/jquery.cookie.js	GET	text/javascript	
/api-gateway/common-ucenter-server/static/js/newlogin/jquery.js	GET	text/javascript	
/api-gateway/common-ucenter-server/static/js/newlogin/sm2_util.v250.js	GET	text/javascript	
/api-gateway/common-ucenter-server/static/js/newlogin/crypto-js.js	GET	text/javascript	
/api-gateway/common-ucenter-server/static/js/newlogin/sm2.v250.js	GET	text/javascript	
/api-gateway/common-ucenter-server/static/layui/layui.js	GET	text/javascript	
/api-gateway/common-ucenter-server/static/layui/lay/modules/layer.js	GET	text/javascript	
/api-gateway/common-ucenter-server/static/layui/css/modules/layer/default/layer.css	GET	text/css	
/api-gateway/common-ucenter-server/login-prop	GET	application/json	

另存为，将请求的返回值保存到新的记录集

点击“切换”，选择一个记录集

当页面请求对应接口时，将返回所选记录集中的数据

6.3.4 VSCode 开发辅助插件

以下插件，可通过安装套件 JPaaS Extension Pack 统一安装

6.3.4.1 应用创建

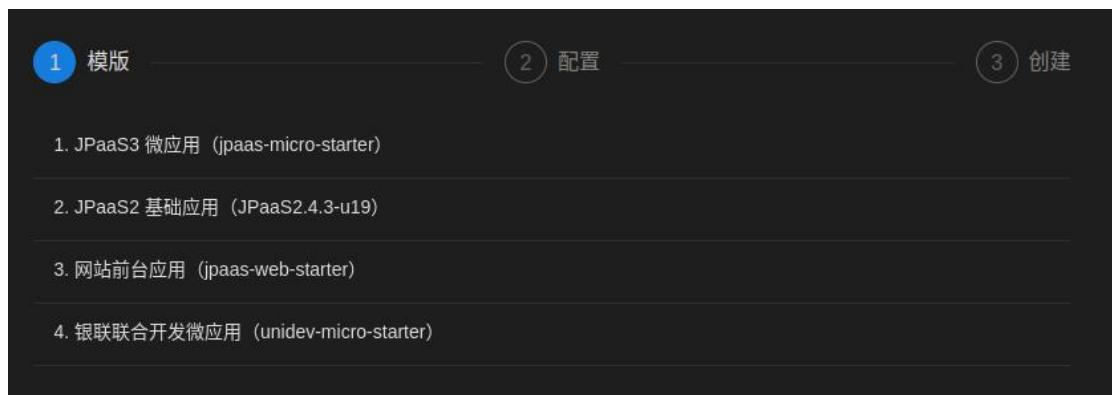
可视化脚手架工具，通过选择模板，创建不同类型的应用

扩展商店中安装 JPaaS Application Creator 插件

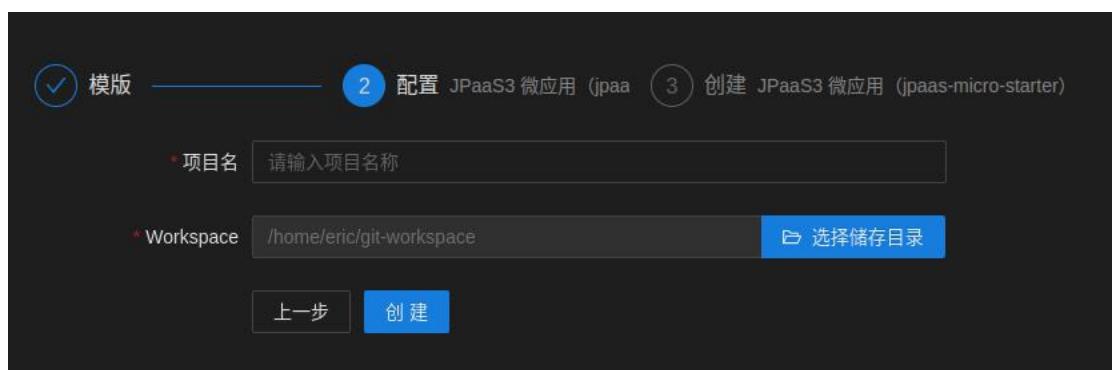
1. 点击下方状态栏中“创建应用”



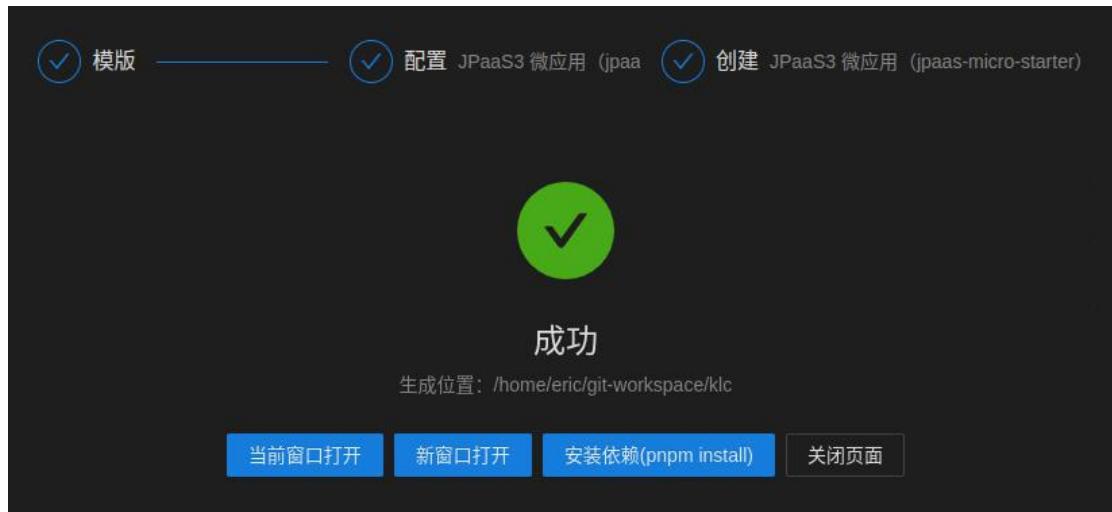
2. 选择需要的模板



3. 填写项目名称、选择 workspace 目录



4. 点击“创建”完成



6.3.4.2 组件开发辅助

提供 jpaas-component 组件代码提示，自动引入等功能

扩展商店中安装 JPaaS Component Helper 插件

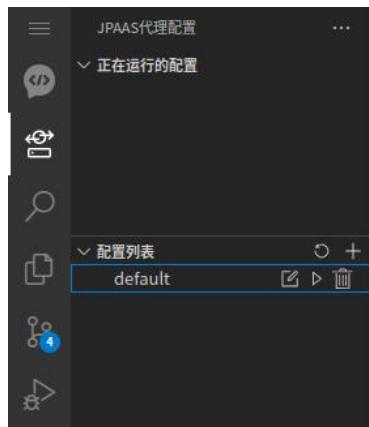


6.3.4.3 微前端环境模拟器

模拟微前端环境，对本地微应用进行集成开发和调试。

扩展商店中安装 JPaaS Microfontends Simulator 插件

1. 打开左侧代理配置面板，添加配置



2. 配置基座服务和本地微应用代理，保存

JPaas微前端环境模拟器

环境名称: default 基座版本: 3.7.5 启动端口: 8002

代理配置

匹配路径 (location)	目标路径 (target)	操作
/api-gateway	https://jpaasdev2.hanweb.com	
/common/	https://jpaasdev2.hanweb.com	
/microstarter/	http://localhost:8003	
location	target	

应用注册

应用名称 (name)	应用标识 (appCode)	操作
demo	microstarter	
name	appCode	

保存

3. 在配置列表中启动服务，打开浏览器

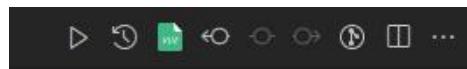
6.3.4.4 低代码开发辅助

可视化配置，生成前端代码

扩展商店中安装 JPaaS Low Code 插件

1. 新建 vue 文件，并打开

2. 点击右上角代码生成按钮



3. 选择创建的组件类型，包括：表格、表单、菜单、树、下拉树、树+表格、菜单+表格



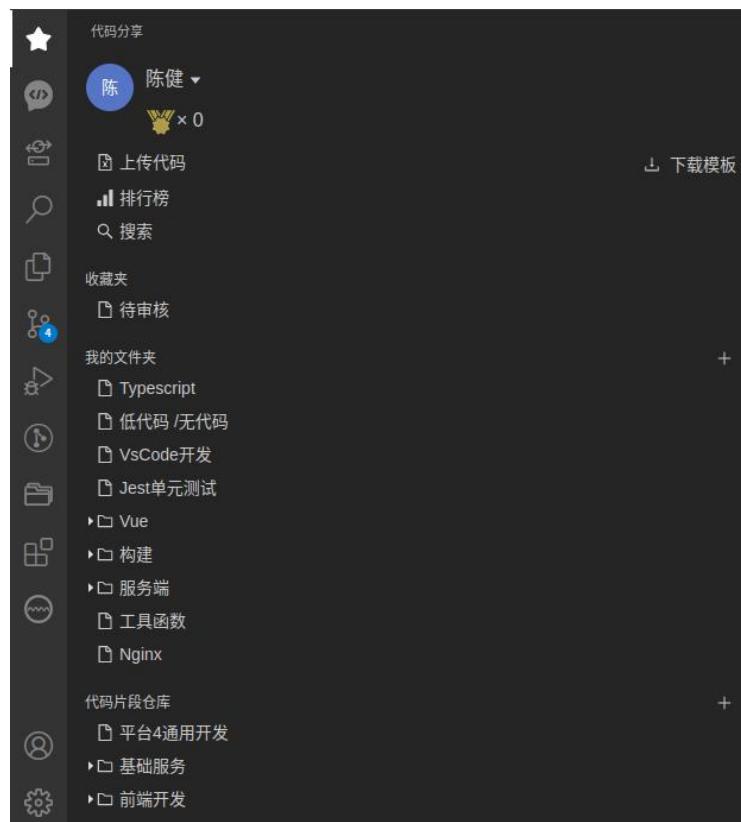
4. 配置组件及数据源

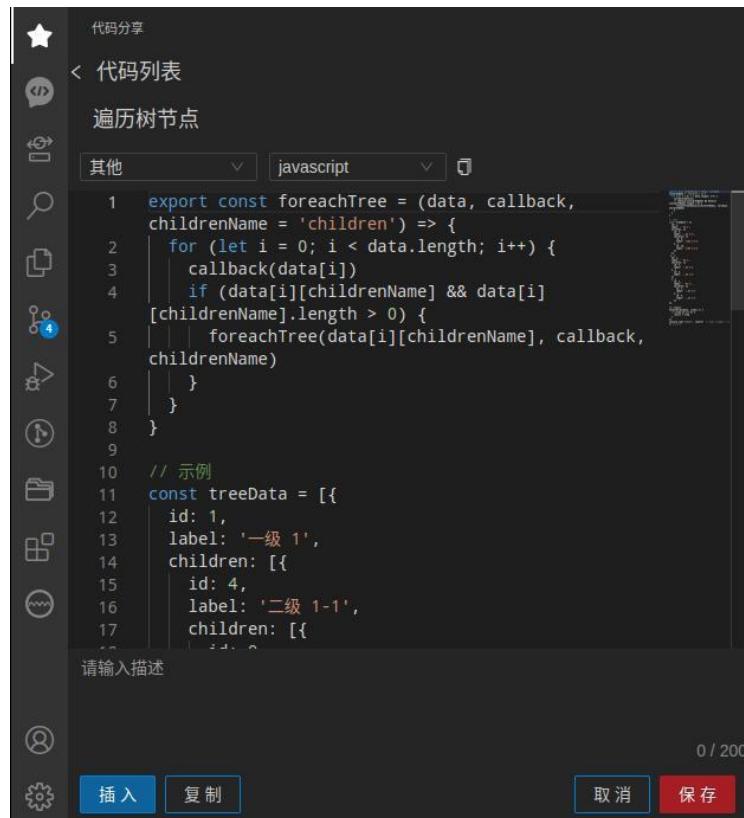
5. 点击“生成代码”，代码将保存到 vue 文件中

6.3.4.5 代码片段

将常用或优秀的代码片段收藏起来，重复使用或分享给其他开发使用

扩展商店中安装 JPaaS Code Snippets 插件





1. vscode 设置》扩展》代码片段，填写动态秘钥、登录名、密码
2. 左侧面板点击“代码片段”，对分类和代码片段进行管理
3. 进入代码片段详细页，可复制或将代码插入光标处
4. 代码编辑器中选择代码片段，右键选择“保存代码片段”，选择分类，保存

6.4 代码审查

6.4.1 SonarQube

1. 进入一体化研发平台，找到需要审查的服务版本，构建选择 sonar 审查的流水线



2. 使用部门账号登录 sonarqube , 进入要审查的项目 , 查看查看“全部代码”的审查结果

The screenshot shows the SonarQube interface for the 'approval-fe' project. At the top, it displays the project name, version (main), and a note from April 20, 2023. Below this are tabs for '总览' (Overview), '问题' (Issues), '安全热点' (Security Hotspots), '指标' (Metrics), '代码' (Code), and '活动' (Activities). The '总览' tab is selected. On the left, a red box highlights '错误' (Errors) with 1 condition failure. A note says '不推荐向“自己的干净代码”的质量圈添加其他条件。请检查 质量圈' (Do not recommend adding other conditions to the quality circle of your own clean code. Please check Quality Circle). Below this, it says '全部代码有 1 个条件失败' (1 condition failure in all code). In the center, there are several metrics: 10 Bug (可靠性 C), 0 漏洞 (安全性 A), 14 安全热点 (0.0% 审查, 安全审查 E), 1 天 1 小时 负债 (可维护性 A), 70 异味 (0.0% 单元测试), and 16.4% 重复率, 代码 18K 行 (199 重复块).

3. 点击左侧失败条件 , 进入修改对应问题

质量阙状态 ②

错误

6 个条件失败

项目使用的质量阙没有遵循“自己的干净代码”

修复 [此质量阙](#) 会帮助你达到干净代码的状态。

[了解原因](#)

全部代码有 6 个条件失败

C 可靠性等级 劣于 A

5 ⚙ Bug 大于 0

47 ⚡ 异味 大于 30

14.7% 注释 (%) 小于 15.0%

3 严重违规 大于 0

23 主要违规 大于 5

4. 查看问题代码位置，及问题原因，对源代码进行修改

Remove this conditional structure or edit its code blocks so that they're not all the same.

[获取永久链接](#)

All branches in a conditional structure should not have exactly the same implementation [javascript:S3923](#)

7个月前 L483

[Bug](#) [主要](#) [打开](#) [未分配](#) 15min 工作 0 评论

[无标签](#)

问题的位置	问题原因
<pre>jinspect-fe src/views/jinspect/inspectList/InspectDetail.vue</pre>	<p>Remove this conditional structure or edit its code blocks so that they're not all the same.</p> <pre>474 h1f2... seePic () { 475 this.showModel = true 476 }, 477 onSelectChange (selectedRowKeys, selectRows) { 478 this.selectedRowKeys = selectedRowKeys 479 if (this.selectedRowKeys.length > 0) { 480 // this.batchOps = true 481 } else { 482 // this.batchOps = false 483 } 484 this.selectRows = selectRows 485 }, 486 handleOk (e) { 487 this.showModel = false 488 }, 489 handleCancel (e) { 490 this.showModel = false 491 }, 492 handleOkWord (e) { 493 }</pre>

5. 修改完成，在一体化研发平台中重新提交审查

6. “全部代码”的审查结果，右侧等级全部要达到 A，左侧质量与状态为“通过”

Remove this conditional structure or edit its code blocks so that they're not all the same.
All branches in a conditional structure should not have exactly the same implementation javascript:S3923
Bug 主要 打开 未分配 15min 工作 0 评论 7个月前 L483 无标签

问题的位置 问题原因

jinspect-fe src/views/jinspect/inspectList/InspectDetail.vue

474 h1f2... seePic () {
475 this.showModel = true
476 },
477 onSelectChange (selectedRowKeys, selectRows) {
478 this.selectedRowKeys = selectedRowKeys
479 if (this.selectedRowKeys.length > 0) {
480 // this.batchOps = true
481 } else {
482 // this.batchOps = false
483 }

Remove this conditional structure or edit its code blocks so that they're not all the same.

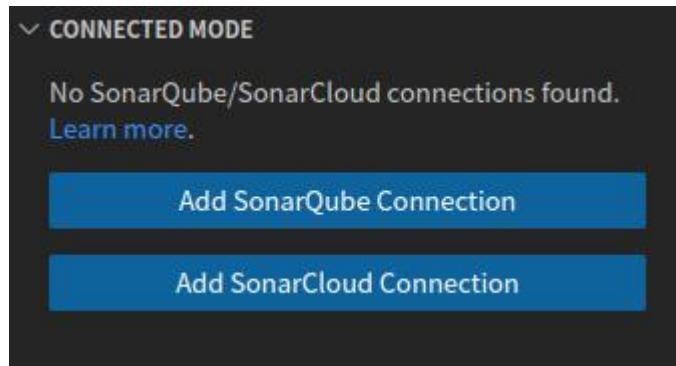
484 this.selectRows = selectRows
485 },
486 handleOk (e) {
487 this.showModel = false
488 },
489 handleCancel (e) {
490 this.showModel = false
491 },
492 handleOkWord (e) {

若存在误报或开源代码中无法修改等情况，在其他问题全部修改通过后，提交审查小组进

行人工审查

6.4.2 SonarLint (vscode 插件)

1. 添加 SonarQube 连接



2. 输入 SonarQube 服务地址，点击“Generate Token”生成令牌

New SonarQube Connection

Server URL

http://192.168.5.47:30002/

Generate Token

Incorrect URL or server is not available

3. 允许连接，创建令牌

允许 SonarLint 连接？



Visual Studio Code SonarLint 请求访问 SonarQube.

是否允许 SonarLint 连接？将创建一个令牌并分享给 SonarLint。

允许连接

4. 保存连接

User Token
..... Token Received!

Connection Name
http-192-168-5-47-30002-

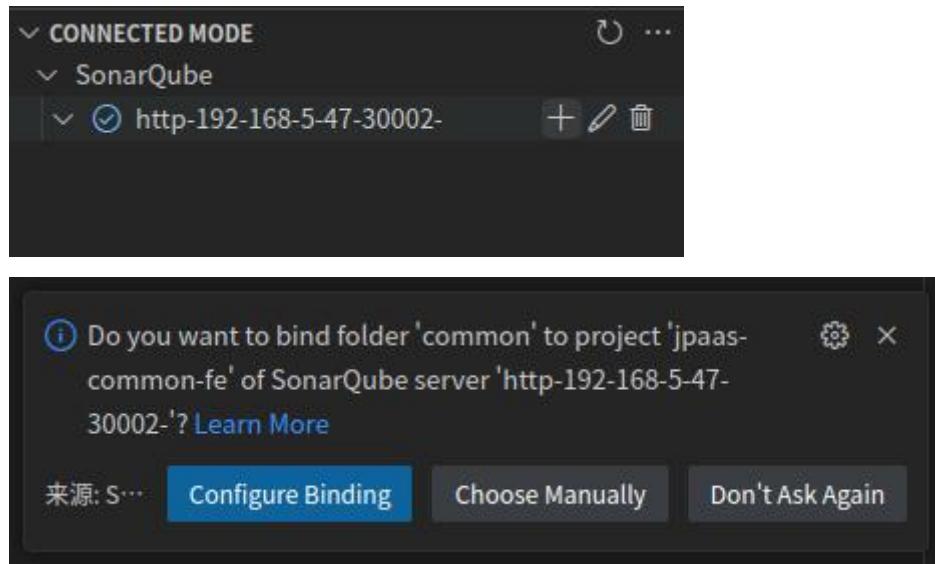
Receive notifications from SonarQube

You will receive **notifications** from SonarQube in situations like:

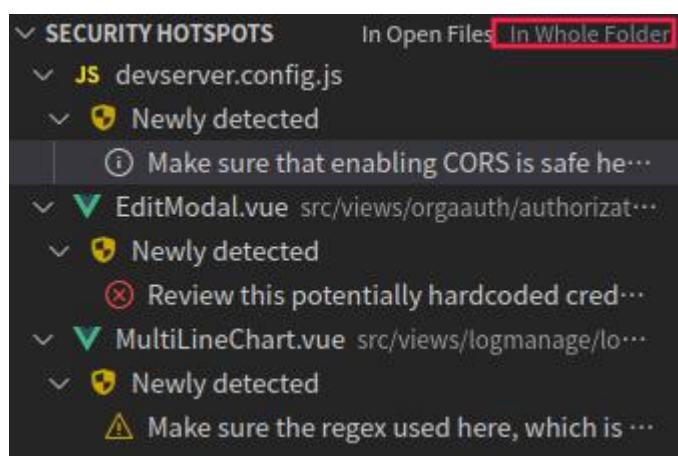
- the Quality Gate status of a bound project changes
- the latest analysis of a bound project on SonarQube raises new issues assigned to you

Save Connection

5. 绑定项目



6. 扫描整个工程



6.5 合规检查

参考内网 ->制度与流程->开发类->开发管理流程->[安全扫描工具使用规范](#)

7 开发规范

7.1 目录结构

dist/		编译输出目录
docker/		docker 相关配置
node_modules/		依赖模块
public/		静态资源
sourcemap/		源代码映射
src/		源代码
	api/	接口配置
	assets/	本地静态资源
	microcomponents/	微组件
	router/	路由配置
	store/	状态管理配置
	utils/	通用工具库
	views/	页面
	App.vue	应用根组件
	main.js	应用入口
template/		页面 HTML 模板
.gitignore		git 忽略管理配置
app.config.js		应用配置
CHANGELOG.md		更新日志

devserver.config.js		本地开发服务配置
package.json		包配置
README.md		项目说明
vue.config.js		vue 项目配置

7.2 命名规范

7.2.1 目录名

1. 组件目录使用 UpperCamelCase 风格，必须遵从驼峰形式

如：CountDown

2. 其他目录使用 kebab-case 风格

如：user-center

3. 禁止使用不存在的自创英文缩写或拼音

7.2.2 文件命名

1. .vue 文件使用 UpperCamelCase 风格，必须遵从驼峰形式

如：RegionList.vue

2. 其他文件使用 lowerCamelCase 风格，必须遵从驼峰形式

如：userSettings.js, logo.png

3. 禁止使用不存在的自创英文缩写或拼音

4. 不同尺寸的同一张图片，文件名后追加尺寸，格式：[名称]_[宽]_[高].[后缀名]

如：banner_400_300.jpg

正方形图片可简写为，格式：[名称]_[长].[后缀名]

如：logo_32.png

5. vue 文件名中的单词顺序，以高级别的(通常是一般化描述的)单词开头，以描述性的修饰词结尾。如：SearchButtonClear 而不是 ClearSearchButton.vue

6. 常用 vue 文件名：

列表页面的后缀为 List，如：UserList.vue

对话框页面的后缀为 Modal，如：UserModal.vue

其他页面不使用后缀，如：PermissionSet.vue

7.2.3 变量命名

1. 函数名、参数名、局部变量都统一使用 lowerCamelCase 风格，必须遵从驼峰形式
2. 常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长

如：MAX_STOCK_COUNT

3. 引用 this

this 代表当前对象，在程序中随着上下文随时变化。如在内部函数、回调函数中，this

已不是原来所指的对象，此时若需要调用，则在调用函数中，定义 that 变量指向

this，如：const that = this，不要使用 _this 来定义。之后即可在内部函数中使用 that 来调用

7.2.4 组件名

1. 文件名和 JavaScript 代码中使用 UpperCamelCase 风格

2. 组件名 name 属性必填，同文件名
3. 在模版中，统一使用 kebab-case 风格，全小写，多个单词之间以“-”分隔

7.2.5 Prop 属性名

1. 在声明 prop 的时候，其命名应该始终使用 lowerCamelCase
2. 在模板和 JSX 中应该始终使用 kebab-case

7.3 组件/实例

7.3.1 选项顺序

1. 副作用 (触发组件外的影响)
 - el
2. 全局感知 (要求组件以外的知识)
 - name
 - parent
3. 组件类型 (更改组件的类型)
 - functional
4. 模板修改器 (改变模板的编译方式)
 - delimiters
 - comments
5. 模板依赖 (模板内使用的资源)

- components

- directives

- filters

6. 组合 (向选项里合并属性)

- extends

- mixins

7. 接口 (组件的接口)

- inheritAttrs

- model

- props/propsData

8. 本地状态 (本地的响应式属性)

- data

- computed

9. 事件 (通过响应式事件触发的回调)

- watch

● 生命周期钩子 (按照它们被调用的顺序)

- beforeCreate

- created

- beforeMount

- mounted

- beforeUpdate

- updated

- activated

- deactivated
- beforeDestroy
- destroyed

10. 非响应式的属性 (不依赖响应系统的实例属性)

- methods

11. 渲染 (组件输出的声明式描述)

- template/render
- renderError

7.3.2 单文件组件的顶级元素的顺序

顺序 : <template>、<script>、<style>

7.4 样式编写规范

- 在多页面中共用的样式，写到单独的样式文件中，样式须添加模块前缀，如 channel-record，避免样式冲突
- 只在当前页面使用的样式时，可写在<style> 标签中，添加 scoped 属性
- 禁止全局定义 HTML 元素样式，如<p>、的样式

7.5 API 调用编写规范

- API 调用按照对象模型分文件编写，/src/api 的应用目录下，不要出现多层次目录
- 同一个对象模型相关的方法都在同一个文件中编写，不要分成多个文件
- 使用名称 xxxAPI，包装 model 的所有方法，export 导出

- 对数据的操作行为，请求方法使用 POST
- 对数据的查询，请求方法使用 GET

```
import { axios } from '@/utils/request'

const api = {
  tree: '/manager/district/find-by-pid-keyword',
  add: '/manager/district/add'
}

export const regionAPI = {
  // 获取树菜单数据
  tree (parameter) {
    return axios({
      url: api.tree,
      method: 'get',
      params: parameter
    })
  },
  // 新增行政区划
  add (parameter) {
    return axios({
      url: api.add,
      method: 'post',
      data: parameter
    })
  }
}
```

1. 在 index.js 文件中统一 export 导出，并添加注释

```
import { regionAPI } from './region'
import { teamAPI } from './team'
import { accountAPI } from './account'
import { roleAPI } from './role'
import { userAPI } from './user'
import { organizaAPI } from './organiza'
export {
  regionAPI, // 行政区划
  teamAPI, // 群组管理
}
```

```
accountAPI, // 个人中心及账户设置
roleAPI, // 角色管理
userAPI, // 用户管理
organizaAPI // 机构管理
}
```

7.6 图标

7.6.1 内置图标引入

- 尽量使用 AntDesign Vue 的图标，除非没有合适的或者设计师已提供
- AntDesign Vue 图标查询：<https://1x.antdv.com/components/icon-cn/>
- 在 node_modules/@ant-design/icons/svg 下自行查找图标文件
- 通过 import 方式引入，最后需要添加‘?inline’（方便 webpack loader 对 svg 图片进行处理）
- 名称须添加前缀 Icon，如：IconUser

```
import IconUser from '@ant-design/icons/svg/outline/user.svg?inline'

export default {
  name: 'AlreadySelect',
  components: {
    MultipleAdd,
    HAvatar
  },
  data () {
    return {
      IconUser,
      IconTeam,
      IconSolution,
      IconEnvironment,
      IconFileWord,
      IconFile,
      IconBars,
      IconDeploymentUnit,
    }
  }
}
```

```
    isActive: -1,
    multipleAddShow: false,
    multipleSelectedList: [],
    isCtrlA: false,
    shiftArr: [],
    selectIcon: process.env.BASE_URL + 'images/selectList.png'
}
},
...

```

7.6.2 图标文件引入

- 图标禁止使用位图，统一使用 svg 格式的矢量图
- 图标由产品经理协调设计部进行设计，放置到/src/assets/icons 对应的应用目录下
- 一般情况下尽量使用已存在的图标，以减少项目大小
- 前端开发人员不要私自使用自己搜集的图标，以免造成系统内不一致和版权问题
- 通过 import 方式引入，最后需要添加'`?inline`'（方便 webpack loader 对 svg 图片进行处理）
- 名称须添加前缀 Icon，如：`IconUser`

```
import IconMoreops from '@/assets/icons/moreops.svg?inline'
import IconReturn from '@/assets/icons/return.svg?inline'
export default {
  name: 'MicroDemo',
  components: {
    HTree,
    HSiderPanel,
  },
  data () {
    return {
      IconMoreops,
      IconReturn,
      hrefUrl: '',
      fileName: '',
    }
  }
}
```

7.6.3 图标使用

```
<a-icon :component="IconMoreops" />  
<h-button :icon="IconUpload">上传图片</h-button>
```

7.7 代码管理

7.7.1 分支

- master: 主分支(保护分支) , 不能直接在 master 上进行修改代码和提交
- dev: 开发分支 , 新需求开发
- feature-*: 新功能开发分支。可能改动影响较大 , 或周期较长时创建功能分支 , 避免对其他开发人员的代码产生影响。开发完成后合并到 dev 分支。合并完成可以删除
- hotfix-*: 紧急 bug 修复分支 , 根据实际情况对已发布的版本进行漏洞修复
- release-*: 预发布分支。编码完成进入测试阶段 , 所有的修改必须在此分支上进行 , 提交代码后要及时合并到 dev 分支

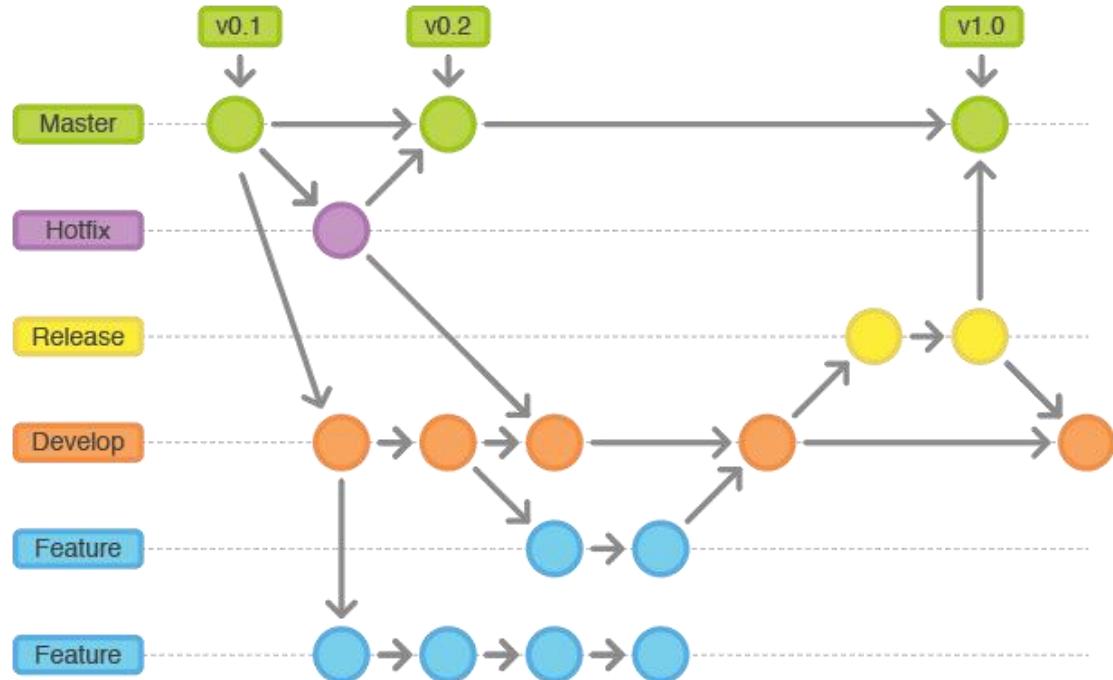
7.7.2 版本

版本发布或标记重要节点时可以使用 tag 标记

采用三段式 , 版本.里程碑.序号 , 如 1.2.3

- 架构升级或架构重大调整 , 修改第 1 位
- 新功能上线或者模块大的调整 , 修改第 2 位
- bug 修复上线 , 修改第 3 位
- 线上 bug 紧急修复 , 修改第 4 位 , 正常情况下版本号不应该出现 4 位

7.7.3 工作流



- 日常开发在 dev 分支上进行
- 进入测试阶段，则从 dev 上创建 Release 分支，所有 bug 修改在 Release 分支上进行
- 测试结束，将代码从 Release 合并到 Master 分支上，Tag 版本号。同时将代码合并到 dev 分支。之后删除 Release 分支
- 如果已发布版本有 bug 需要立即修复，则创建 Hotfix 分支，修复并测试通过后，合并到 Master 分支，Tag 版本。同时合并到 dev 分支

7.7.4 Commit Message

每次提交，Commit message 都包括三个部分：Header，Body 和 Footer。

注意：冒号为半角，之后有一个空格，务必正确，否则无法生成 changelog

推荐使用格式化插件，安装及设置请参考 [vsc-commitizen](#)

```
<type>(<scope>): <subject>
// 空一行
<body>
// 空一行
<footer>
```

1. Header

只有一行，包括三个字段：type（必需）、scope（可选）和 subject（必需）

type :

代表某次提交的类型，比如是修复一个 bug 还是增加一个新的 feature。

type 类型如下：

- feat : 新增功能
- fix : 修复 bug
- docs : 仅仅修改了文档，比如 README, CHANGELOG 等等
- style : 仅仅修改了空格、格式缩进等等，不改变代码逻辑
- refactor : 代码重构，没有加新功能或者修复 bug
- perf : 优化相关，比如提升性能、体验
- revert : 回滚到上一个版本
- chore : 不属于以上类型的其他类型

scope :

用于说明 commit 影响的范围，比如数据层、控制层、视图层等等，视项目不同而不同。

subject :

commit 目的的简短描述，不超过 50 个字符。

注意事项：

- 以动词开头，使用第一人称现在时，比如 change，而不是 changed 或 changes
- 第一个字母小写
- 结尾不加句号（.）

2. Body

对本次 commit 的详细描述，可以分成多行

需要描述的信息包括：

- 为什么这个变更是必须的？它可能是用来修复一个 bug，增加一个 feature，提升性能、可靠性、稳定性等等
- 他如何解决这个问题？具体描述解决问题的步骤
- 是否存在副作用、风险？

3. Footer

如果需要的话可以添加一个链接到 issue 地址或者其它文档，或者关闭某个 issue

7.7.5 CHANGELOG

版本正式发布时，需要产生 changelog 文档，便于后续问题追溯。

7.8 路由规范

- 路由匹配组件采用异步加载的方式进行配置，如：

```
component: () => import('@/views/website/web-apply/NewApply')
```

- 有权限控制需求时，注意添加 permission，配置权限，不配置则不受控制

permission 为权限代码数组，可以配置多个权限

权限代码组成：[appCode].[...moduleCode].[functionCode]

- appCode：应用名，唯一标识
- moduleCode：模块名，可配置多层级
- functionCode：不写则须拥有模块下所有权限

```
permission: ['common:orgaauth:rolemgt', 'common:orgaauth:usermgt', 'common:orgaauth:teammgt',
'common:orgaauth:teammgt:auth', 'common:orgaauth:usermgt:auth']
```

7.9 URL 命名规范

- URL 越短越好，便于记忆
- 避免太多参数，尽量使用静态 URL
- 尽量使用比较少的目录层级
- 文件及目录名要具有描述性
- **字母全部小写**
- 单词之间使用连接符“-”，或考虑分级，如：/user/role/add

8 界面规范

参考《页面标准化规范》

8.1 页面布局

最低适配 1280px 的屏幕宽度

页面内外边距 : 16px

按钮间距 : 8px

其他元素的间距应符合设计稿或上述标准

布局需自动适配。即修改分辨率或调整浏览器大小后，容器宽高应自动调整，内部元素需考虑换行时的展现效果。

内容溢出时应自动出现滚动条或隐藏溢出部分、可显示文字最后出现省略号等

8.2 主题色

系统定义了两种主题 : light、dark。并提供了状态 state.app.theme 或 window.globalConfig.theme
自定义的页面头部导航及侧边栏菜单，应自动响应 theme 状态变化

8.3 文字

- 文字默认使用全局字体、字号、颜色，非必要不要自行设置
- 如有设计稿，应充分还原设计稿中设计的相关样式，包括：字体、字号、颜色、间距、对齐方式等
- 注意检查文案有误错别字，是否逻辑清晰、语法正确、简明扼要

8.4 图片

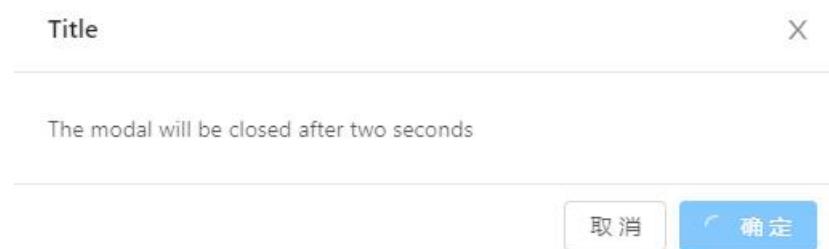
- 图片应保持原始宽高比例，避免拉伸。在容器中可使用 background-size:cover; 或 background-size:contain; 进行填充或使用 HImg 组件
- 图片文件无法访问，应显示缺省图片
- 页面避免直接嵌入超过 800X600 的原图，应加载压缩后的图片
- 加载的压缩图片尺寸应接近嵌入尺寸，否则会导致图片模糊或产生锯齿

8.5 列表

单行删除时，使用 Popconfirm 气泡确认框

8.6 对话框

- 推荐宽度尺寸：480px、520px、600px
- 窗体内最大高度：500px，内容过高会产生滚动条，可考虑多列布局
- 内边距：24px 或 0
- “确定”按钮点击需要请求接口时，须配置 confirm-loading 属性，防止重复提交



8.7 抽屉

弹出内容较多的时候采用抽屉方式

推荐宽度尺寸：480px、520px、600px

8.8表单

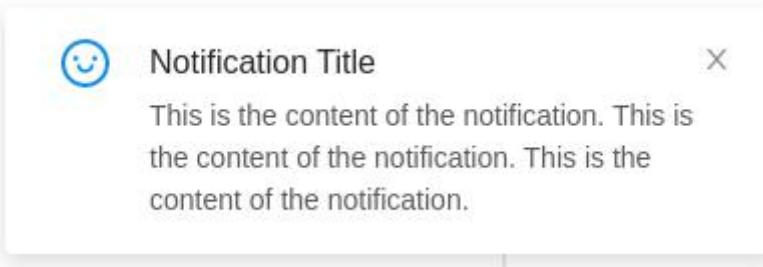
- 默认采用垂直布局，在较小的区域可以采用水平布局，**注意采用垂直布局时务必填写 layout 属性，垂直布局时上下间距与水平布局不同**
- 文本输入类组件，如：单行文本框、多行文本框、富文本编辑器等，必须控制输入长度。`input`、`textarea` 必须设置 `maxlength`
- 表单提交，提交须配置按钮 `loading` 属性，防止重复提交

8.9提示和通知

普通简短提示使用 `Message` 全局提示



后端返回的错误信息使用 `Notification` 通知提醒框



页面警告、提示使用 `Alert` 警告提示



9 常见问题

1. linux 运行时报错 : Error: error:0308010C:digital envelope routines::unsupported

在 vscode 的配置文件 settings.json 中添加配置

```
"terminal.integrated.env.linux": {  
    "NODE_OPTIONS": "--openssl-legacy-provider"  
}
```

2. 使用 HSiderPanel 布局的页面为什么高度没有自适应屏幕高度

平台对右侧页面的 HSiderPanel 布局做了自适应处理，能够自动适应页面高度。但 HSiderPanel 外部不能被其他容器包裹，如 div 或 ACard。有些场景下，页面上会存在弹窗组件和抽屉组件，但 template 中只能有一个根元素，HSiderPanel 提供了默认插槽，可以用于放置这类不可见组件。

3. 为什么无法框选文字，或无法使用浏览器右键菜单？

可能有些应用对 document 或 body 等 DOM 添加了样式或事件，阻止了原生事件。这类全局都可访问的对象，即使切换了页面或应用，其事件仍然存在。所以应禁止对此类元素绑定新的事件，如不可避免，需要在页面离开或组件销毁的时候，移除自定义事件。给 DOM 添加事件处理，不要使用 dom.onClick = function() {...} 等赋值方式，会将原本正常的事件处理覆盖。使用 addEventListener 方法添加，removeEventListener 方法移除。

4. 如何定制开发登录页面、修改系统名称、Logo？

jpaas 提供定制功能，由系统管理员自行设置，无需前端开发处理。

以系统管理员登录后，进入“管理》基础设置》登录文件管理”，可修改登录页模版。

platform.js 文件为系统信息配置，可修改标题和 logo 地址。

5. 为什么部署到集成环境中，进入子应用一片空白？

检查子应用是否能够访问到，在新窗口打开域名/应用标识，如：www.abc.com/cms。如果为空白页，右键查看源代码，其中的 css、js 等静态资源的 url 是否是应用标识开头的。无法访问、跳转到 404 或其他页面、静态资源路径非应用标识开头等，都说明环境或配置有问题。

检查子应用开发时使用的脚手架版本与基座应用版本是否对应。jpaas-portal 为 3.2.2.1 的版本，需要使用 2.2.0 以上的脚手架（jpaas-micro-starter）。如版本过低，建议使用最新脚手架，将业务代码重新加入。

6. 启动时提示 Multiple assets emit different content to the same filename index.html

可能配置了 multi-page 模式，原 HTML 模板文件在 public 目录下，而 page 的输出文件与模板文件同名，导致都输出到 dist 目录下时冲突。

将 public 下的模板文件移动到 template 目录下，并修改 vue.config.js 中 page 的 template 路径。参考 <https://cli.vuejs.org/zh/config/#pages>

附录

1 组件文档

前端开发平台基于 Ant Design Vue、Ant Design Pro of Vue

API 文档及相关示例参考：

Ant Design Vue : <https://1x.antdv.com/docs/vue/introduce-cn/>

Ant Design Pro of Vue : <https://pro.antdv.com/>

2 其他参考文档

ECMAScript 6 入门 : <http://es6.ruanyifeng.com/>

Vue.js : <https://cn.vuejs.org/>