Admin工程化实战

> 崔老师 https://juejin.cn/post/6925668019884523534

一、目标

> 用工程化方法让祖传项目焕发青春

- 无需修改代码结构也能提炼通用组件库
- 快速剥离出新项目的脚手架
- 编写工程化CLI工具拒绝重复劳动

二、限制条件

- 主项目所有目录结构不能破坏

二、开发子模块

/moduleC

1. 粘贴一个chart视图
2. 创建路由/src/views/charts/index.js

```
/** When your routing table is too long, you can split it into small modules**/

import Layout from "@/layout";

const chartsRouter = {
  path: "/charts",
  component: Layout,
  redirect: "noRedirect",
  name: "Charts",
  meta: {
    title: "ChartsXX",
    icon: "chart",
  },
  children: [
    {
      path: "keyboard",
      component: () => import("./keyboard"),
      name: "KeyboardChartxxx",
```

```
      meta: { title: "Keyboard123 Chart", noCache: true },
    },
  ],
};

export default chartsRouter;
```

3. 修改路由依赖

   router/index

   ```
   import charts from '../../../moduleC/src/views/charts'
   ```

4. copy eslintignore .eslintrc.js

5.Dashboard

/src/views/dashboard/dashboard.vue

```
<template>
  <div class="dashboard-container">
    ModuleC
  </div>
</template>
<script>
export default {
};
</script>
```

添加路由

/src/views/dashboard/index.js

```
import Layout from '@/layout'
export default {

    path: "/",
    component: Layout,
    redirect: "/dashboard",
    children: [
      {
        path: "dashboard",
        component: () => import("./index.vue"),
        name: "Dashboard",
        meta: { title: "Dashboard", icon: "dashboard", affix: true },
      },
    ],
```

```
  };
```

6 Login

置换

7. 子模块路由

```
import { createRouter, createWebHashHistory } from 'vue-router'

/* Layout */
import Layout from '@/layout'

// import charts2 from '../views/charts2'
import charts2 from 'submodule/src/views/charts'
// import dashboard from '../../../sub-a/src/views/dashboard'

export const constantRoutes = [
  {
    path: '/redirect',
    component: Layout,
    hidden: true,
    children: [
      {
        path: '/redirect/:path(.*)',
        component: () => import('@/views/redirect/index'),
      },
    ],
  },
  {
    path: '/login',
    // component: () => import('@/views/login/index'),
    component: () => import('submodule/src/views/login/index'),
    hidden: true,
  },
  {
    path: '/',
    component: Layout,
    redirect: '/dashboard',
    children: [
      {
        path: 'dashboard',
        // component: () => import('@/views/dashboard/index'),
        component: () => import('submodule/src/views/dashboard/dashboard'),
        name: 'Dashboard',
```

```
        meta: { title: 'Dashboard', icon: 'dashboard', affix: true },
      },
    ],
  },
  // dashboard,
  charts2,

]

/**
 * asyncRoutes
 * the routes that need to be dynamically loaded based on user roles
 */
export const asyncRoutes = [
]

export default createRouter({
  // history: createWebHistory(), // require service support
  history: createWebHashHistory(),
  scrollBehavior: () => ({ top: 0 }),
  routes: constantRoutes,
})
```

8 使用别名解决问题

vue.config.js

```
  configureWebpack: {
    // provide the app's title in webpack's name field, so that
    // it can be accessed in index.html to inject the correct title.
    name: name,
    resolve: {
      alias: {
        '@': resolve('src'),
        submodule: submodule.path,
      },
    },
  },
```

submodule.js

```
const {resolve} = require('path')

module.exports = {
  path: resolve('../moduleA')
}
```

三、 提炼组件库

创建packages文件夹

packages.json

```
{
  "name": "admin-component",
  "version": "1.0.0",
  "description": "",
  "main": "dist/index.cjs.js",
  "module": "dist/index.bundle.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@rollup/plugin-babel": "^5.3.0",
    "@rollup/plugin-commonjs": "^17.1.0",
    "@rollup/plugin-json": "^4.1.0",
    "@rollup/plugin-node-resolve": "^11.2.0",
    "@rollup/plugin-replace": "^2.4.1",
    "@vue/compiler-sfc": "^3.0.7",
    "rollup": "^2.41.0",
    "rollup-plugin-peer-deps-external": "^2.2.4",
    "rollup-plugin-scss": "^2.6.1",
    "rollup-plugin-terser": "^7.0.2",
    "rollup-plugin-vue": "^6.0.0"
  }
}
```

/rollup.config.js

```
const vuePlugin = require("./rollup-plugin-vue/index");
```

```
import babel from "@rollup/plugin-babel";
// import vuePlugin from "rollup-plugin-vue";

import scss from 'rollup-plugin-scss'

const iife = {
  input: "entry.js",
  output: {
    file: "dist/index.js",
    name: "AdminCommon",
    format: "iife",
    globals: {
      vue: "Vue",
    },
  },
  external: ["vue"],
  plugins: [
    babel(),
    vuePlugin({
      css: true,
    }),
    scss(),
  ],
};

const es = {
  input: "entry.js",
  output: {
    file: "dist/index.bundle.js",
    name: "AdminCommon",
    format: "es",
    globals: {
      vue: "Vue",
    },
  },
  external: ["vue"],
  plugins: [
    babel(),
    vuePlugin({
      css: true,
    }),
    scss(),
  ],
};



import { terser } from "rollup-plugin-terser";
const minEs = {
```

```
  input: "entry.js",
  external: ["vue"],
  output: {
    file: "dist/index.min.js",
    name: "Element",
    format: "umd",
  },
  plugins: [
    babel(),
    vuePlugin({
      css: true,
    }),
    scss(),
    terser(),
  ],
};

const cjs = {
  input: "entry.js",
  external: ["vue"],
  output: {
    file: "dist/index.cjs.js",
    name: "Element",
    format: "cjs",
  },
  plugins: [
    babel(),
    vuePlugin({
      css: true,
    }),
    scss(),
  ],
};

export default [iife,es, minEs, cjs];
```

/entry.js 入口文件

```
import Keyboard from '../src/components/Charts/Keyboard.vue'

// 组件库
const AdminCommon = {
  Keyboard,
  install: app => {
    app.use(Keyboard)
  },
}
```

```
export { Keyboard }

export default AdminCommon
```

测试

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Document</title>
    <script src="../node_modules/vue/dist/vue.global.js"></script>
    <script src="../node_modules/echarts/dist/echarts.js"></script>
    <script src="dist/index.js"></script>
    <style></style>
  </head>

  <body>
    <div id="app"></div>
    <script>
      const { createApp, reactive, computed, watchEffect } = Vue;

      const MyComponent = {
        template: `
               <keyboard width="100%" height="600px"></keyboard>
            `,
      };

      createApp(MyComponent.default)
        .use(AdminCommon)
        .mount("#app");
    </script>
  </body>
</html>
```

配置模块定义

package.json

```
  "main": "dist/index.cjs.js",
  "module": "dist/index.bundle.js",
```

本地安装

```
# 在framework目录下
sudo npm link ./packages
```

组件测试

/moduleA/src/views/charts/keyboard.vue

```
<template>
  <div class="chart-container">
    <keyboard height="100%" width="100%" /> 加载
  </div>
</template>

<script>
import Chart from '@/components/Charts/Keyboard'
  // 一如
import { Keyboard } from "admin-component";
import Keyboard from
'../../../../framework/src/components/Charts/Keyboard.vue';

export default {
  name: 'KeyboardChart',
  components: { Keyboard }, // 注册
}
</script>
    Keyboard

<style scoped>
.chart-container{
  position: relative;
  width: 100%;
  height: calc(100vh - 84px);
}
</style>
```

四、CLI工具

1. 命令行界面

/bin/index.js

```
#!/usr/bin/env node

const { promisify } = require("util");
const figlet = promisify(require("figlet"));
const clear = require("clear");
const chalk = require("chalk");
const inquirer = require("inquirer");
const log = (content) => console.log(chalk.green(content));
const opt = {
  初始化脚手架: "init",
  启动项目: "start",
  创建子项目: "create",
  编译组件库: "compile",
  版本发布: "",
  Eslint格式校验: "",
  退出: "quit",
};

const question = [
  {
    type: "rawlist" /* 选择框 */,
    message: "请选择操作? ",
    name: "operation",
    choices: Object.keys(opt),
  },
];

// 打印欢迎画面
clear();
log(
  figlet.textSync("Element 3!", {
    // font: "Ghost",
    horizontalLayout: "default",
    verticalLayout: "default",
    width: 80,
    whitespaceBreak: true,
  })
);

query();

async function query() {
```

```
  const answer = await inquirer.prompt(question);

  console.log("answer", answer);
  if (answer.operation === "退出") return;

  await require(`../lib/operation/${opt[answer.operation]}`)();
  await query();
}
```

/package.json

```
  "bin": {
    "admin-cli": "./bin/index.js"
  },
```

2. 初始化工程

/operation/init.js

```
const { promisify } = require("util");
const download = promisify(require("git-pull-or-clone"));
const ora = require("ora");
const { resolve } = require("path");
const { spawn } = require("../api/process");
const chalk = require("chalk");
const log = (...args) => console.log(chalk.green(...args));
module.exports = async () => {
  console.log("path", resolve("."));
  // 项目名称
  const name = "admin";
  const repo = "git@gitee.com:josephxia/element3-admin-framework.git";
  // const repo = 'https://gitee.com/josephxia/element3-admin-framework.git'
  const desc = resolve(`./${name}`);
  console.log("desc", desc);
  const process = ora(`🚘下载.....${repo}`);
  process.start();
  try {
    await download(repo, desc);
    process.succeed();
  } catch (e) {
    console.log(e);
    process.fail();
  }

  // 安装依赖
```

```
  log("安装依赖");
  // 安装Framework依赖
  await spawn("npm", ["install"], { cwd: `${desc}/framework/` });

  log(`
👌安装完成:
To get Start:
==========================
    cd ${name}/framework
    npm run serve
==========================
            `);
};
```

3. 启动工程

start.js

```
const { spawn } = require("../api/process");
const fs = require("fs");
const inquirer = require("inquirer");
const compile = require("../api/compile");
const copy = require("../api/copy");
module.exports = async (basePath = "./admin") => {
  const choices = fs
    .readdirSync("./admin")
    .filter((v) => !["framework", "cli", ".git", "template"].includes(v));

  choices.unshift("主模块");

  const question = [
    {
      type: "list" /* 选择框 */,
      message: "请选择子项目? ",
      name: "moduleName",
      choices /* 选项*/,
    },
  ];

  const answer = await inquirer.prompt(question);

  // 修改入口文件
  console.log(`切换为: ${answer.moduleName} `);

  // 生成子模块配置文件
  compile(
```

```
    {
      name: answer.moduleName,
    },
    `${basePath}/framework/submodule.js`,
    `${basePath}/framework/template/submodule.js.hbs`
  );

  // Copy路由
  if (answer.moduleName === "主模块") {
    copy(
      `${basePath}/framework/src/router/index_origin.js`,
      `${basePath}/framework/src/router/index.js`
    );
  } else {
    copy(
      `${basePath}/${answer.moduleName}/src/router.js`,
      `${basePath}/framework/src/router/index.js`
    );
  }

  // 启动项目
  await spawn("npm", ["run", "serve"], { cwd: `${basePath}/framework` });
};
```

/api/compile.js

```
const fs = require("fs");
const chalk = require("chalk");
const handlebars = require("handlebars");
/**
 * 编译模板文件
 * @param meta 数据定义
 * @param filePath 目标文件路径
 * @param templatePath 模板文件路径
 */
function compile(meta, filePath, templatePath) {
  if (fs.existsSync(templatePath)) {
    const content = fs.readFileSync(templatePath).toString();
    const result = handlebars.compile(content)(meta);
    fs.writeFileSync(filePath, result);
    console.log(chalk.green(`🚀${filePath} 创建成功`));
  }
}

module.exports = compile;
```

/process.js

```
module.exports.spawn = async (...args) => {
    const { spawn } = require('child_process');
    return new Promise(resolve => {
        const proc = spawn(...args)
        proc.stdout.pipe(process.stdout)
        proc.stderr.pipe(process.stderr)
        proc.on('close', () => {
            resolve()
        })
    })
}
```

4. 创建子项目

/create.js

```
const inquirer = require("inquirer");
const chalk = require("chalk");
const copy = require('../api/copy')
module.exports = async (basePath = "./admin") => {
  const question = [
    {
      name: "name",
      message: "请输入子项目名称? ",
    },
  ];

  // 人机交互，详情见 question 文件
  const answer = await inquirer.prompt(question);
  if (answer.name) {

    copy(`${basePath}/template`,`${basePath}/${answer.name}`)
    console.log("🔥创建项目:", answer.name);
  }
};
```

/api/copy

```
const fs = require("fs");
const { constants } = require("os");
function copy(src, dst) {
  const paths = fs.readdirSync(src); //同步读取当前目录
  paths.forEach(function (path) {
    const _src = src + "/" + path;
    const _dst = dst + "/" + path;
    fs.stat(_src, function (err, stats) {
      //stats 该对象 包含文件属性
      if (err) throw err;
      if (stats.isFile()) {
        //如果是个文件则拷贝
        const readable = fs.createReadStream(_src); //创建读取流
        const writable = fs.createWriteStream(_dst); //创建写入流
        readable.pipe(writable);
      } else if (stats.isDirectory()) {
        //是目录则 递归
        checkDirectory(_src, _dst, copy);
      }
    });
  });
}
function checkDirectory(src, dst, callback) {
  // 文件
  fs.stat(src, function (err, stats) {
    if (stats.isFile()) {
      //如果是个文件则拷贝
      const readable = fs.createReadStream(src); //创建读取流
      const writable = fs.createWriteStream(dst); //创建写入流
      readable.pipe(writable);
    } else {
      // 目录递归
      fs.access(dst, fs.constants.F_OK, (err) => {
        if (err) {
          fs.mkdirSync(dst);
          callback(src, dst);
        } else {
          callback(src, dst);
        }
      });
    }
  });
}
// const SOURCES_DIRECTORY = "../operation"; //源目录
// checkDirectory(SOURCES_DIRECTORY, "./abc", copy);

module.exports = (src, desc) => checkDirectory(src, desc, copy);
```

5. 编译组件库

```
const { spawn } = require("../api/process");
module.exports = async (basePath = './admin/framework') => {
  console.log("编译组件....");
  // 安装Framework依赖
  await spawn("npm", ["install"], { cwd: `${basePath}/packages` });

  // Rollup打包
  await spawn("rollup", ["-c"], { cwd: `${basePath}/packages` });

  // 安装模块
  await spawn("sudo", ["npm", "link", "./packages"], {
    cwd: `./admin/framework`,
  });
};
```

技术不好 =》 工作累 =》 没时间 50% + 50% 效率管理

视频学习 =》 JS 🍆 黄书