link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=211 sentences=908, words=5739

1.准备工作

1.1 monorepo

- monoRepo： 是将所有的模块统一的放在一个主干分支之中管理。
- multiRepo： 将项目分化成为多个模块，并针对每一个模块单独的开辟一个Repo来进行管理。

1.2 Lerna

- Lerna是一个管理多个 npm 模块的工具,优化维护多包的工作流，解决多个包互相依赖，且发布需要手动维护多个包的问题

1.2.1 安装

```
npm i lerna -g
```

1.2.2 初始化

```
lerna init
```

命令 功能 lerna bootstrap 安装依赖 lerna clean 删除各个包下的node_modules lerna init 创建新的lerna库 lerna list 查看本地包列表 lerna changed 显示自上次release tag以来有修改的包， 选项通 list lerna diff 显示自上次release tag以来有修改的包的差异， 执行 git diff lerna exec 在每个包目录下执行任意命令 lerna run 执行每个包package.json中的脚本命令 lerna add 添加一个包的版本为各个包的依赖 lerna import 引入package lerna link 链接互相引用的库 lerna create 新建package lerna publish 发布

1.2.3 文件

1.2.3.1 package.json

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "lerna": "^4.0.0"
  }
}
```

1.2.3.2 lerna.json

```
{
  "packages": [
    "packages/*"
  ],
  "version": "0.0.0"
}
```

1.2.3.3 .gitignore

```
node_modules
.DS_Store
design
*.log
packages/test
dist
temp
.vuerc
.version
.versions
.changelog
```

1.2.4 yarn workspace

- yarn workspace允许我们使用 monorepo 的形式来管理项目
- 在安装 node_modules 的时候它不会安装到每个子项目的 node_modules 里面，而是直接安装到根目录下面，这样每个子项目都可以读取到根目录的 node_modules
- 整个项目只有根目录下面会有一份 yarn.lock 文件。子项目也会被 link 到 node_modules 里面，这样就允许我们就可以直接用 import 导入对应的项目
- yarn.lock文件是自动生成的,也完全Yarn来处理.yarn.lock锁定你安装的每个依赖项的版本，这可以确保你不会意外获得不良依赖

1.2.4.1 package.json

package.json

```
{
  "name": "root",
  "private": true,
+  "workspaces": [
+    "packages/*"
+  ],
  "devDependencies": {
    "lerna": "^4.0.0"
  }
}
```

1.2.4.2 lerna.json

lerna.json

```
{
  "packages": [
    "packages/*"
  ],
  "version": "1.0.0",
+ "useWorkspaces": true,
+ "npmClient": "yarn"
}
```

1.2.4.3 添加依赖

- yarnpkg (https://classic.yarnpkg.com/en/docs/cli)
- lerna (https://github.com/lerna/lerna#readme)

设置加速镜像

```
yarn config set registry http://registry.npm.taobao.org
 npm config set registry https://registry.npm.taobao.org
```

作用 命令 查看工作空间信息 yarn workspaces info 给根空间添加依赖 yarn add chalk cross-spawn fs-extra --ignore-workspace-root-check 给某个项目添加依赖 yarn workspace create-react-app3 add commander 删除所有的 node_modules lerna clean 等于 yarn workspaces run clean 安装和link yarn install 等于 lerna bootstrap --npm-client yarn --use-workspaces 重新获取所有的 node_modules yarn install --force 查看缓存目录 yarn cache dir 清除本地缓存 yarn cache clean

1.2.5 创建子项目

```
lerna create zhang-cli
lerna create  zhang-cli-shared-utils
```

1.2.5.1 zhang-cli

1.2.5.1.1 package.json

packages\zhang-cli\bin\vue.js

```
{
  "name": "zhang-cli",
  "version": "0.0.0",
  "description": "zhang-cli",
  "author": "zhangrenyang ",
  "homepage": "",
  "license": "MIT",
  "main": "bin/vue.js"
}
```

1.2.5.1.2 vue.js

packages\zhang-cli\bin\vue.js

```

console.log('vue cli');
```

1.2.5.2.1 package.json

packages\zhang-cli-shared-utils\package.json

```
{
  "name": "zhang-cli-shared-utils",
  "version": "0.0.0",
  "description": "zhang-cli-shared-utils",
  "author": "zhangrenyang ",
  "homepage": "",
  "license": "MIT",
  "main": "index.js"
}
```

1.2.5.2.2 index.js

packages\zhang-cli-shared-utils\index.js

1.2.6 创建软链接

```
yarn
cd packages/zhang-cli
npm link
npm root -g
zhang-cli
```

1.2.7 create命令

```
{
  "name": "root",
  "private": true,
  "workspaces": [
    "packages/*"
  ],
  "devDependencies": {
    "lerna": "^4.0.0"
  },
  "scripts": {
+   "create": "node ./packages/zhang-cli/bin/vue.js create hello1"
  }
}
```

1.2.8 调试命令

.vscode\launch.json

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "node",
            "request": "launch",
            "name": "vue-cli",
            "cwd":"${workspaceFolder}",
            "runtimeExecutable": "npm",
            "runtimeArgs": [
                "run",
                "create"
            ],
            "port":9229,
            "autoAttachChildProcesses": true,
            "stopOnEntry": true,
            "skipFiles": [
                "/**"
            ]
        }
    ]
}
```

1.3 安装依赖

```
npm config set registry=https://registry.npm.taobao.org
yarn config set registry https://registry.npm.taobao.org

cd packages/zhang-cli-shared-utils
yarn workspace zhang-cli-shared-utils add  chalk execa

cd packages/zhang-cli
yarn workspace zhang-cli add  zhang-cli-shared-utils commander inquirer execa chalk ejs globby  lodash.clonedeep fs-extra ora isbinaryfile
```

1.4 lerna vs yarn

- 两者很多功能是等价的
- yarn用来处理依赖，lerna用于初始化和发布

1.4 commander.js

- commander (https://github.com/tj/commander.js/blob/master/Readme_zh-CN.md) 是一款强大的命令行框架，提供了用户命令行输入和参数解析功能

```
const program = require('commander');
program
    .version(`zhang-cli 0.0.0}`)
    .usage(' [options]')

program
    .command('create ')
    .description('create a new project powered by vue-cli-service')
    .action((name) => {
        console.log(name);
    })

program.parse(process.argv)
```

```
node 1.2.commander.js
Usage: 1.2.commander  [options]

Options:
  -V, --version      output the version number
  -h, --help         display help for command

Commands:
  create   create a new project powered by vue-cli-service
  help [command]     display help for command

node 1.2.commander.js create hello
```

1.5 Inquirer.js

- Inquirer (https://github.com/SBoudrias/Inquirer.js)是一个交互式命令行工具

```
const inquirer = require('inquirer')
const isManualMode = answers => answers.preset === '__manual__';
let defaultPreset = {
    useConfigFiles: false,
    cssPreprocessor: undefined,
    plugins: {
        '@vue/cli-plugin-babel': {},
        '@vue/cli-plugin-eslint': {
            config: 'base',
            lintOn: ['save']
        }
    }
}
let presets = {
    'default': Object.assign({ vueVersion: '2' }, defaultPreset),
    '__default_vue_3__': Object.assign({ vueVersion: '3' }, defaultPreset)
}
const presetChoices = Object.entries(presets).map(([name, preset]) => {
    let displayName = name
    if (name === 'default') {
        displayName = 'Default'
    } else if (name === '__default_vue_3__') {
        displayName = 'Default (Vue 3)'
    }
    return {
        name: `${displayName}`,
        value: name
    }
})
const presetPrompt = {
    name: 'preset',
    type: 'list',
    message: `Please pick a preset:`,
    choices: [
        ...presetChoices,
        {
            name: 'Manually select features',
            value: '__manual__'
        }
    ]
}
let features = [
    'vueVersion',
    'babel',
    'typescript',
    'pwa',
    'router',
    'vuex',
    'cssPreprocessors',
    'linter',
    'unit',
    'e2e'
];
const featurePrompt = {
    name: 'features',
    when: isManualMode,
    type: 'checkbox',
    message: 'Check the features needed for your project:',
    choices: features,
    pageSize: 10
}
const prompts = [
    presetPrompt,
    featurePrompt
]

;(async function(){
 let result = await inquirer.prompt(prompts);
 console.log(result);
})();
```

1.6 execa

- execa 是可以调用 shell 和本地外部程序
- 它会启动子进程执行，是对 child_process.exec的封装

```
const execa = require('execa');

(async () => {
    const {stdout} = await execa('echo', ['hello']);
    console.log(stdout);
})();
```

1.7 chalk

- chalk (https://github.com/chalk/chalk)可以修改控制台字符串的样式，包括字体样式、颜色以及背景颜色等

```
const chalk = require('chalk');
console.log(chalk.blue('Hello world!'));
```

1.8 ejs

- ejs (https://ejs.bootcss.com)是高效的嵌入式 JavaScript 模板引擎
- slash (https://www.npmjs.com/package/slash)将Windows反斜杠路径转换为斜杠路径，如 `foo\\bar`➔ `foo/bar`
- globby (https://www.npmjs.com/package/globby)是用于模式匹配目录文件的

1.8.1 main.js

template\main.js

```
if (rootOptions.vueVersion === '3') { _%>
  import { createApp } from 'vue'
  import App from './App.vue'
  createApp(App).mount('#app')
else { _%>
  import Vue from 'vue'
  import App from './App.vue'
  Vue.config.productionTip = false
  new Vue({
    render: h => h(App),
  }).$mount('#app')
```

1.8.2 components

doc\template\components

```
  HelloWorld

export default {
  name: 'HelloWorld'
}
```

1.8.3 ejs.js

doc\1.7.ejs.js

```
const path = require('path');
const fs = require('fs');
const ejs = require('ejs');
const globby = require('globby')
const slash = require('slash')
let source = path.join(__dirname, 'template');
;(async function () {
    const _files = await globby(['**/*'], { cwd: source })
    let files = {};
    for (const rawPath of _files) {
        const sourcePath = slash(path.resolve(source, rawPath))
        const template = fs.readFileSync(sourcePath, 'utf8')
        const content = ejs.render(template, {
            rootOptions: { vueVersion: '2' }
        })
        files[sourcePath] = content;
    }
    console.log(files);
})();
```

1.9 isbinaryfile

- isbinaryfile (https://www.npmjs.com/package/isbinaryfile)可以检测一个文件是否是二进制文件

```
const path = require('path');
const { isBinaryFileSync } = require('isbinaryfile');
let logo = path.join(__dirname,'template/assets/logo.png');
let isBinary = isBinaryFileSync(logo);
console.log(isBinary);
let main = path.join(__dirname,'template/main.js');
isBinary = isBinaryFileSync(main);
console.log(isBinary);
```

1.10 ora

- ora (https://www.npmjs.com/package/ora)主要用来实现node.js命令行环境的loading效果，和显示各种状态的图标等

```
const ora = require('ora')
const spinner = ora()

exports.logWithSpinner = (msg) => {
    spinner.text = msg
    spinner.start();
}

exports.stopSpinner = () => {
    spinner.stop();
}

exports.logWithSpinner('npm install');
setTimeout(()=>{
    exports.stopSpinner();
},3000);
```

2.核心概念

- @vue/cli (https://cli.vuejs.org/zh/guide/)是一个基于 Vue.js 进行快速开发的完整系统

2.1 插件

- 插件 (https://cli.vuejs.org/zh/guide/plugins-and-presets.html)
- Vue CLI 使用了一套基于插件的架构。如果你查阅一个新创建项目的 package.json，就会发现依赖都是以 @vue/cli-plugin- 开头的。插件可以修改 webpack 的内部配置，也可以向 `vue-cli-service` 注入命令。在项目创建的过程中，绝大部分列出的特性都是通过插件来实现的
- 每个 CLI 插件都会包含一个 (用来创建文件的) 生成器和一个 (用来调整 webpack 核心配置和注入命令的) 运行时插件
- 官方插件格式 `@vue/cli-plugin-eslint`,社区插件 `vue-cli-plugin-apollo`,指定的 scope 使用第三方插件 `@foo/vue-cli-plugin-bar`

2.3 预设

- 一个 Vue CLI preset 是一个包含创建新项目所需预定义选项和插件的 JSON 对象，让用户无需在命令提示中选择它们
- 在 vue create 过程中保存的 preset 会被放在你的 home 目录下的一个配置文件中 (~/.vuerc)。你可以通过直接编辑这个文件来调整、添加、删除保存好的 preset
- Preset 的数据会被插件生成器用来生成相应的项目文件

```
exports.defaultPreset = {
  useConfigFiles: false,
  cssPreprocessor: undefined,
  plugins: {
    '@vue/cli-plugin-babel': {},
    '@vue/cli-plugin-eslint': {
      config: 'base',
      lintOn: ['save']
    }
  }
}
```

2.4 特性

- 在手工模式下，我们可以自由选择以下特性
    - vueVersion
    - babel
    - typescript
    - pwa
    - router
    - vuex
    - cssPreprocessors
    - linter
    - unit
    - e2e
- 选择不同的特性会添加不同的插件，不同的插件会生成不同的文件和修改项目的配置

2.5 create

3.参数解析

3.1 vue.js

packages\zhang-cli\bin\vue.js

```
const program = require('commander');
program
    .version(`@vue/zhang-cli ${require('../package').version}`)
    .usage(' [options]')

program
    .command('create ')
    .description('create a new project powered by vue-cli-service')
    .action((name) => {
        require('../lib/create')(name)
    })

program.parse(process.argv)
```

3.2 create.js

packages\zhang-\lib\create.js

```
const path = require('path');
async function create(projectName, options) {
  const cwd = process.cwd();
  const name = projectName;
  const targetDir = path.resolve(cwd, projectName);
  console.log(name);
  console.log(targetDir);
}

module.exports = (...args) => {
  return create(...args).catch(err => {
    console.log(err);
  })
}
```

4.获取预设

4.1 create.js

packages\zhang-cli\lib\create.js

```
const path = require('path');
+const Creator = require('./Creator');
+const { getPromptModules } = require('./util/createTools')
async function create(projectName) {
  const cwd = process.cwd();
  const name = projectName;
  const targetDir = path.resolve(cwd, projectName);
+ const promptModules = getPromptModules();
+ const creator = new Creator(name, targetDir,promptModules);
+ await creator.create();
}

module.exports = (...args) => {
  return create(...args).catch(err => {
    console.log(err);
  })
}
```

4.2 options.js

packages\zhang-cli\lib\options.js

```
exports.defaultPreset = {
  useConfigFiles: false,
  cssPreprocessor: undefined,
  plugins: {
    '@vue/cli-plugin-babel': {},
    '@vue/cli-plugin-eslint': {
      config: 'base',
      lintOn: ['save']
    }
  }
}

exports.defaults = {
  presets: {
    'default': Object.assign({ vueVersion: '2' }, exports.defaultPreset),
    '__default_vue_3__': Object.assign({ vueVersion: '3' }, exports.defaultPreset)
  }
}
```

4.3 PromptModuleAPI.js

packages\zhang-cli\lib\PromptModuleAPI.js

```
class PromptModuleAPI {
    constructor(creator) {
        this.creator = creator
    }

    injectFeature(feature) {
        this.creator.featurePrompt.choices.push(feature)
    }

    injectPrompt(prompt) {
        this.creator.injectedPrompts.push(prompt)
    }

    onPromptComplete(cb) {
        this.creator.promptCompleteCbs.push(cb)
    }
}
module.exports = PromptModuleAPI;
```

4.4 createTools.js

packages\zhang-cli\lib\util\createTools.js

```
exports.getPromptModules = () => {
    return [
        'vueVersion'
    ].map(file => require(`../promptModules/${file}`))
}
```

4.5 vueVersion.js

packages\zhang-cli\lib\promptModules\vueVersion.js

```
module.exports = cli => {

  cli.injectFeature({
    name: 'Choose Vue version',
    value: 'vueVersion',
    description: 'Choose a version of Vue.js that you want to start the project with',
    checked: true
  })

  cli.injectPrompt({
    name: 'vueVersion',
    when: answers => answers.features.includes('vueVersion'),
    message: 'Choose a version of Vue.js that you want to start the project with',
    type: 'list',
    choices: [
      {
        name: '2.x',
        value: '2'
      },
      {
        name: '3.x',
        value: '3'
      }
    ],
    default: '2'
  })

  cli.onPromptComplete((answers, options) => {
    if (answers.vueVersion) {
      options.vueVersion = answers.vueVersion
    }
  })
}
```

4.6 Creator.js

packages\zhang-cli\lib\Creator.js

```
const { defaults } = require('./options');
const PromptModuleAPI = require('./PromptModuleAPI');
const inquirer = require('inquirer')
const isManualMode = answers => answers.preset === '__manual__'
class Creator {
    constructor(name, context, promptModules) {
        this.name = name;
        this.context = process.env.VUE_CLI_CONTEXT = context;
        const { presetPrompt, featurePrompt } = this.resolveIntroPrompts();
        this.presetPrompt = presetPrompt;
        this.featurePrompt = featurePrompt;
        this.injectedPrompts = []
        this.promptCompleteCbs = []
        const promptAPI = new PromptModuleAPI(this)
        promptModules.forEach(m => m(promptAPI))
    }
    async create() {
        let preset = await this.promptAndResolvePreset()
        console.log('preset', preset);
    }
    resolveFinalPrompts() {
        this.injectedPrompts.forEach(prompt => {
            const originalWhen = prompt.when || (() => true)
            prompt.when = answers => {
                return isManualMode(answers) && originalWhen(answers)
            }
        })
        const prompts = [
            this.presetPrompt,
            this.featurePrompt,
            ...this.injectedPrompts,
        ]
        return prompts
    }
    async promptAndResolvePreset(answers = null) {
        if (!answers) {
            answers = await inquirer.prompt(this.resolveFinalPrompts())
        }
        let preset;
        if (answers.preset && answers.preset !== '__manual__') {
            preset = await this.resolvePreset(answers.preset)
        } else {
            preset = {
                plugins: {}
            }
            answers.features = answers.features || []
            this.promptCompleteCbs.forEach(cb => cb(answers, preset))
        }
        return preset
    }
    async resolvePreset (name) {
        const savedPresets = this.getPresets()
        return savedPresets[name];
    }
    getPresets() {
        return Object.assign({}, defaults.presets)
    }
    resolveIntroPrompts() {
        const presets = this.getPresets()
        const presetChoices = Object.entries(presets).map(([name]) => {
            let displayName = name
            if (name === 'default') {
                displayName = 'Default'
            } else if (name === '__default_vue_3__') {
                displayName = 'Default (Vue 3)'
            }
            return {
                name: `${displayName}`,
                value: name
            }
        })
        const presetPrompt = {
            name: 'preset',
            type: 'list',
            message: `Please pick a preset:`,
            choices: [
                ...presetChoices,
                {
                    name: 'Manually select features',
                    value: '__manual__'
                }
            ]
        }
        const featurePrompt = {
            name: 'features',
            when: isManualMode,
            type: 'checkbox',
            message: 'Check the features needed for your project:',
            choices: [],
            pageSize: 10
        }
        return {
            presetPrompt,
            featurePrompt
        }
    }
}
module.exports = Creator;
```

5.写入package.json

packages\zhang-cli-shared-utils\index.js

```
exports.chalk = require('chalk')
```

5.2 Creator.js

packages\zhang-cli\lib\Creator.js

```
const { defaults } = require('./options');
const PromptModuleAPI = require('./PromptModuleAPI');
const inquirer = require('inquirer')
+const cloneDeep = require('lodash.clonedeep')
+const writeFileTree = require('./util/writeFileTree')
+const { chalk } = require('zhang-cli-shared-utils')
const isManualMode = answers => answers.preset
class Creator {
    constructor(name, context, promptModules) {
        this.name = name;
        this.context = process.env.VUE_CLI_CONTEXT = context;
        const { presetPrompt, featurePrompt } = this.resolveIntroPrompts();
        this.presetPrompt = presetPrompt;
        this.featurePrompt = featurePrompt;
        this.injectedPrompts = []
        this.promptCompleteCbs = []
        const promptAPI = new PromptModuleAPI(this)
        promptModules.forEach(m => m(promptAPI))
    }
    async create() {
+       const {name,context} = this;
        let preset = await this.promptAndResolvePreset()
        console.log('preset', preset);
+       preset = cloneDeep(preset);
+       preset.plugins['@vue/cli-service'] = Object.assign({projectName: name}, preset);
+       console.log(`✨  Creating project in ${chalk.yellow(context)}.`)
+       const pkg = {
+           name,
+           version: '0.1.0',
+           private: true,
+           devDependencies: {}
+       }
+       const deps = Object.keys(preset.plugins)
+       deps.forEach(dep => {
+           pkg.devDependencies[dep] = 'latest';
+       })
+       await writeFileTree(context, {
+           'package.json': JSON.stringify(pkg, null, 2)
+       })
    }
    resolveFinalPrompts() {
        this.injectedPrompts.forEach(prompt => {
            const originalWhen = prompt.when || (() => true)
            prompt.when = answers => {
                return isManualMode(answers) && originalWhen(answers)
            }
        })
        const prompts = [
            this.presetPrompt,
            this.featurePrompt,
            ...this.injectedPrompts,
        ]
        return prompts
    }
    async promptAndResolvePreset(answers = null) {
        if (!answers) {
            answers = await inquirer.prompt(this.resolveFinalPrompts())
        }
        let preset;
        if (answers.preset && answers.preset !== '__manual__') {
            preset = await this.resolvePreset(answers.preset)
        } else {
            preset = {
                plugins: {}
            }
            answers.features = answers.features || []
            this.promptCompleteCbs.forEach(cb => cb(answers, preset))
        }
        return preset
    }
    async resolvePreset (name) {
        const savedPresets = this.getPresets()
        return savedPresets[name];
    }
    getPresets() {
        return Object.assign({}, defaults.presets)
    }
    resolveIntroPrompts() {
        const presets = this.getPresets()
        const presetChoices = Object.entries(presets).map(([name]) => {
            let displayName = name
            if (name
                displayName = 'Default'
            } else if (name
                displayName = 'Default (Vue 3)'
            }
            return {
                name: `${displayName}`,
                value: name
            }
        })
        const presetPrompt = {
            name: 'preset',
            type: 'list',
            message: `Please pick a preset:`,
            choices: [
                ...presetChoices,
                {
```

```
                    name: 'Manually select features',
                    value: '__manual__'
                }
            ]
        }
        const featurePrompt = {
            name: 'features',
            when: isManualMode,
            type: 'checkbox',
            message: 'Check the features needed for your project:',
            choices: [],
            pageSize: 10
        }
        return {
            presetPrompt,
            featurePrompt
        }
    }
}

module.exports = Creator;
```

5.3 writeFileTree.js

packages\zhang-cli\lib\util\writeFileTree.js

```
const fs = require('fs-extra')
const path = require('path')
module.exports = async function writeFileTree(dir, files) {
  Object.keys(files).forEach((name) => {
    const filePath = path.join(dir, name)
    fs.ensureDirSync(path.dirname(filePath))
    fs.writeFileSync(filePath, files[name])
  })
}
```

6.安装依赖

6.1 Creator.js

packages\zhang-cli\lib\Creator.js

```
const { defaults } = require('./options');
const PromptModuleAPI = require('./PromptModuleAPI');
const inquirer = require('inquirer')
const cloneDeep = require('lodash.clonedeep')
const writeFileTree = require('./util/writeFileTree')
+const { chalk, execa } = require('zhang-cli-shared-utils')
const isManualMode = answers => answers.preset
class Creator {
    constructor(name, context, promptModules) {
        this.name = name;
        this.context = process.env.VUE_CLI_CONTEXT = context;
        const { presetPrompt, featurePrompt } = this.resolveIntroPrompts();
        this.presetPrompt = presetPrompt;
        this.featurePrompt = featurePrompt;
        this.injectedPrompts = []
        this.promptCompleteCbs = []
+       this.run = this.run.bind(this)//运行函数
        const promptAPI = new PromptModuleAPI(this)
        promptModules.forEach(m => m(promptAPI))
    }
+   run(command, args) {
+       return execa(command, args, { cwd: this.context })
+   }
    async create() {
+       const {name,context,run} = this;
        let preset = await this.promptAndResolvePreset()
        console.log('preset', preset);
        preset = cloneDeep(preset);
        preset.plugins['@vue/cli-service'] = Object.assign({projectName: name}, preset);
        console.log(`✨  Creating project in ${chalk.yellow(context)}.`)
        const pkg = {
            name,
            version: '0.1.0',
            private: true,
            devDependencies: {}
        }
        const deps = Object.keys(preset.plugins)
        deps.forEach(dep => {
            pkg.devDependencies[dep] = 'latest';
        })
        await writeFileTree(context, {
            'package.json': JSON.stringify(pkg, null, 2)
        })
+       console.log(`📦  Initializing git repository...`)
+       await run('git init');
+       console.log(`⚙\u{fe0f} Installing CLI plugins. This might take a while...`)
+       await run('npm install');
    }
    resolveFinalPrompts() {
        this.injectedPrompts.forEach(prompt => {
            const originalWhen = prompt.when || (() => true)
            prompt.when = answers => {
                return isManualMode(answers) && originalWhen(answers)
            }
        })
        const prompts = [
            this.presetPrompt,
            this.featurePrompt,
            ...this.injectedPrompts,
        ]
        return prompts
    }
```

```
    async promptAndResolvePreset(answers = null) {
        if (!answers) {
            answers = await inquirer.prompt(this.resolveFinalPrompts())
        }
        let preset;
        if (answers.preset && answers.preset !== '__manual__') {
            preset = await this.resolvePreset(answers.preset)
        } else {
            preset = {
                plugins: {}
            }
            answers.features = answers.features || []
            this.promptCompleteCbs.forEach(cb => cb(answers, preset))
        }
        return preset
    }
    async resolvePreset (name) {
        const savedPresets = this.getPresets()
        return savedPresets[name];
    }
    getPresets() {
        return Object.assign({}, defaults.presets)
    }
    resolveIntroPrompts() {
        const presets = this.getPresets()
        const presetChoices = Object.entries(presets).map(([name]) => {
            let displayName = name
            if (name
                displayName = 'Default'
            } else if (name
                displayName = 'Default (Vue 3)'
            }
            return {
                name: `${displayName}`,
                value: name
            }
        })
        const presetPrompt = {
            name: 'preset',
            type: 'list',
            message: `Please pick a preset:`,
            choices: [
                ...presetChoices,
                {
                    name: 'Manually select features',
                    value: '__manual__'
                }
            ]
        }
        const featurePrompt = {
            name: 'features',
            when: isManualMode,
            type: 'checkbox',
            message: 'Check the features needed for your project:',
            choices: [],
            pageSize: 10
        }
        return {
            presetPrompt,
            featurePrompt
        }
    }
}

module.exports = Creator;
```

7.实现插件机制

7.1 Creator.js

packages\zhang-cli\lib\Creator.js

```
const { defaults } = require('./options');
const PromptModuleAPI = require('./PromptModuleAPI');
const inquirer = require('inquirer')
const cloneDeep = require('lodash.clonedeep')
const writeFileTree = require('./util/writeFileTree')
+const { chalk, execa,loadModule } = require('zhang-cli-shared-utils')
+const Generator = require('./Generator')
const isManualMode = answers => answers.preset
class Creator {
    constructor(name, context, promptModules) {
        this.name = name;
        this.context = process.env.VUE_CLI_CONTEXT = context;
        const { presetPrompt, featurePrompt } = this.resolveIntroPrompts();
        this.presetPrompt = presetPrompt;
        this.featurePrompt = featurePrompt;
        this.injectedPrompts = []
        this.promptCompleteCbs = []
        this.run = this.run.bind(this)//运行函数
        const promptAPI = new PromptModuleAPI(this)
        promptModules.forEach(m => m(promptAPI))
    }
    run(command, args) {
        return execa(command, args, { cwd: this.context })
    }
    async create() {
        const {name,context,run} = this;
        let preset = await this.promptAndResolvePreset()
        console.log('preset', preset);
        preset = cloneDeep(preset);
        preset.plugins['@vue/cli-service'] = Object.assign({projectName: name}, preset);
```

```
        console.log(`✨  Creating project in ${chalk.yellow(context)}.`)
        const pkg = {
            name,
            version: '0.1.0',
            private: true,
            devDependencies: {}
        }
        const deps = Object.keys(preset.plugins)
        deps.forEach(dep => {
            pkg.devDependencies[dep] = 'latest';
        })
        await writeFileTree(context, {
            'package.json': JSON.stringify(pkg, null, 2)
        })
        console.log(`🗃  Initializing git repository...`)
        await run('git init');
        console.log(`⚙\u{fe0f} Installing CLI plugins. This might take a while...`)
        await run('npm install');
+       console.log(`🚀  Invoking generators...`)
+       const plugins = await this.resolvePlugins(preset.plugins)
+       const generator = new Generator(context, {pkg,plugins})
+       await generator.generate();
    }
+   async resolvePlugins(rawPlugins) {
+       const plugins = []
+       for (const id of Object.keys(rawPlugins)) {
+           try{
+               const apply = loadModule(`${id}/generator`, this.context) || (() => {})
+               let options = rawPlugins[id] || {}
+               plugins.push({ id, apply, options })
+           }catch(error){
+               console.log(error);
+           }
+       }
+       return plugins
+   }
    resolveFinalPrompts() {
        this.injectedPrompts.forEach(prompt => {
            const originalWhen = prompt.when || (() => true)
            prompt.when = answers => {
                return isManualMode(answers) && originalWhen(answers)
            }
        })
        const prompts = [
            this.presetPrompt,
            this.featurePrompt,
            ...this.injectedPrompts,
        ]
        return prompts
    }
    async promptAndResolvePreset(answers = null) {
        if (!answers) {
            answers = await inquirer.prompt(this.resolveFinalPrompts())
        }
        let preset;
        if (answers.preset && answers.preset !== '__manual__') {
            preset = await this.resolvePreset(answers.preset)
        } else {
            preset = {
                plugins: {}
            }
            answers.features = answers.features || []
            this.promptCompleteCbs.forEach(cb => cb(answers, preset))
        }
        return preset
    }
    async resolvePreset (name) {
        const savedPresets = this.getPresets()
        return savedPresets[name];
    }
    getPresets() {
        return Object.assign({}, defaults.presets)
    }
    resolveIntroPrompts() {
        const presets = this.getPresets()
        const presetChoices = Object.entries(presets).map(([name]) => {
            let displayName = name
            if (name
                displayName = 'Default'
            } else if (name
                displayName = 'Default (Vue 3)'
            }
            return {
                name: `${displayName}`,
                value: name
            }
        })
        const presetPrompt = {
            name: 'preset',
            type: 'list',
            message: `Please pick a preset:`,
            choices: [
                ...presetChoices,
                {
                    name: 'Manually select features',
                    value: '__manual__'
                }
            ]
        }
        const featurePrompt = {
            name: 'features',
            when: isManualMode,
            type: 'checkbox',
            message: 'Check the features needed for your project:',
```

```
            choices: [],
            pageSize: 10
        }
        return {
            presetPrompt,
            featurePrompt
        }
    }
}
module.exports = Creator;
```

packages\zhang-cli-shared-utils\index.js

```
+['pluginResolution','module'].forEach(m => {
+    Object.assign(exports, require(`./lib/${m}`))
+})
exports.chalk = require('chalk')
exports.execa = require('execa')
```

7.3 module.js

packages\zhang-cli-shared-utils\lib\module.js

```
const Module = require('module')
const path = require('path')

exports.loadModule = function (request, context) {
    return Module.createRequire(path.resolve(context, 'package.json'))(request)
}
```

7.4 pluginResolution.js

packages\zhang-cli-shared-utils\lib\pluginResolution.js

```
const pluginRE = /^@vue\/cli-plugin-/
exports.toShortPluginId = id => id.replace(pluginRE, '');
exports.isPlugin = id => pluginRE.test(id)
exports.matchesPluginId = (input, full) => {
    return full === input;
}
```

7.5 mergeDeps.js

packages\zhang-cli\lib\util\mergeDeps.js

```
function mergeDeps(sourceDeps, depsToInject) {
  const result = Object.assign({}, sourceDeps)
  for (const depName in depsToInject) {
    result[depName] = depsToInject[depName];
  }
  return result
}

module.exports = mergeDeps;
```

7.6 normalizeFilePaths.js

packages\zhang-cli\lib\util\normalizeFilePaths.js

```
const slash = require('slash')
module.exports = function normalizeFilePaths (files) {
  Object.keys(files).forEach(file => {
    const normalized = slash(file)
    if (file !== normalized) {
      files[normalized] = files[file]
      delete files[file]
    }
  })
  return files
}
```

7.7 GeneratorAPI.js

packages\zhang-cli\lib\GeneratorAPI.js

```
const { toShortPluginId } = require('zhang-cli-shared-utils')
const mergeDeps = require('./util/mergeDeps')
const { isBinaryFileSync } = require('isbinaryfile')
const isString = val => typeof val === 'string'
const isObject = val => val && typeof val === 'object'
const path = require('path');
const fs = require('fs');
const ejs = require('ejs');
class GeneratorAPI {
    constructor(id, generator, options, rootOptions) {
        this.id = id
        this.generator = generator
        this.options = options
        this.rootOptions = rootOptions
        this.pluginsData = generator.plugins
            .filter(({ id }) => id !== `@vue/cli-service`)
            .map(({ id }) => ({ name: toShortPluginId(id) }))
    }
    hasPlugin(id) {
        return this.generator.hasPlugin(id)
    }
    extendPackage(fields) {
        const pkg = this.generator.pkg
        const toMerge = fields
        for (const key in toMerge) {
            const value = toMerge[key]
            const existing = pkg[key]
            if (isObject(value) && (key === 'dependencies' || key === 'devDependencies')) {
                pkg[key] = mergeDeps(existing || {}, value)
            } else {
                pkg[key] = value
            }
        }
    }
    _injectFileMiddleware(middleware) {
        this.generator.fileMiddlewares.push(middleware)
    }
    _resolveData(additionalData) {
        return Object.assign({
            options: this.options,
            rootOptions: this.rootOptions,
            plugins: this.pluginsData
          }, additionalData)
    }
    render(source,additionalData) {
        const baseDir = extractCallDir()
        if (isString(source)) {
            source = path.resolve(baseDir, source)
            this._injectFileMiddleware(async (files) => {
                const data = this._resolveData(additionalData)
                const globby = require('globby')
                const _files = await globby(['**/*'], { cwd: source })
                for (const rawPath of _files) {
                    const targetPath = rawPath.split('/').map(filename => {
                        if (filename.charAt(0) === '_' && filename.charAt(1) !== '_') {
                            return `.${filename.slice(1)}`
                        }
                        return filename
                    }).join('/')
                    const sourcePath = path.resolve(source, rawPath)
                    const content = renderFile(sourcePath, data)
                    files[targetPath] = content;
                }
            })
        }
    }
}
function extractCallDir() {
    const obj = {}
    Error.captureStackTrace(obj)
    const callSite = obj.stack.split('\n')[3]
    const namedStackRegExp = /\s\((.*):\d+:\d+\)$/
    let matchResult = callSite.match(namedStackRegExp)
    const fileName = matchResult[1]
    return path.dirname(fileName)
}
function renderFile(name, data) {
    if (isBinaryFileSync(name)) {
        return fs.readFileSync(name)
    }
    const template = fs.readFileSync(name, 'utf8')
    return ejs.render(template, data)
}
module.exports = GeneratorAPI
```

7.8 Generator.js

packages\zhang-cli\lib\Generator.js

```
const { isPlugin,matchesPluginId } = require('zhang-cli-shared-utils')
const GeneratorAPI = require('./GeneratorAPI')
const normalizeFilePaths = require('./util/normalizeFilePaths')
const writeFileTree = require('./util/writeFileTree')
const ejs = require('ejs')
class Generator {
    constructor(context, { pkg = {}, plugins = [] } = {}) {
        this.context = context
        this.plugins = plugins
        this.files = {}
        this.fileMiddlewares = []
        this.pkg = pkg;
        this.allPluginIds = Object.keys(this.pkg.dependencies || {})
            .concat(Object.keys(this.pkg.devDependencies || {}))
            .filter(isPlugin)
        const cliService = plugins.find(p => p.id === '@vue/cli-service')
        this.rootOptions = cliService.options;
    }
    async generate() {
        await this.initPlugins()

        this.extractConfigFiles()

        await this.resolveFiles()
        console.log(this.files);
        this.sortPkg()
        this.files['package.json'] = JSON.stringify(this.pkg, null, 2) + '\n'

        await writeFileTree(this.context, this.files)
    }
    sortPkg() {
        console.log('ensure package.json keys has readable order');
    }
    async resolveFiles() {
        const files = this.files
        for (const middleware of this.fileMiddlewares) {
            await middleware(files, ejs.render)
        }
        normalizeFilePaths(files)
    }
    extractConfigFiles() {
        console.log('extractConfigFiles');
    }
    async initPlugins() {
        const { rootOptions } = this
        for (const plugin of this.plugins) {
            const { id, apply, options } = plugin
            const api = new GeneratorAPI(id, this, options, rootOptions)
            await apply(api, options, rootOptions)
        }
    }
    hasPlugin(_id) {
        return [
          ...this.plugins.map(p => p.id),
          ...this.allPluginIds
        ].some(id => {
          return matchesPluginId(_id, id);
        })
      }
}
module.exports = Generator;
```

8.完成create命令

8.1 Creator.js

packages\zhang-cli\lib\Creator.js

```
const { defaults } = require('./options');
const PromptModuleAPI = require('./PromptModuleAPI');
const inquirer = require('inquirer')
const cloneDeep = require('lodash.clonedeep')
const writeFileTree = require('./util/writeFileTree')
const { chalk, execa,loadModule } = require('zhang-cli-shared-utils')
const Generator = require('./Generator')
const isManualMode = answers => answers.preset
class Creator {
    constructor(name, context, promptModules) {
        this.name = name;
        this.context = process.env.VUE_CLI_CONTEXT = context;
        const { presetPrompt, featurePrompt } = this.resolveIntroPrompts();
        this.presetPrompt = presetPrompt;
        this.featurePrompt = featurePrompt;
        this.injectedPrompts = []
        this.promptCompleteCbs = []
        this.run = this.run.bind(this)//运行函数
        const promptAPI = new PromptModuleAPI(this)
        promptModules.forEach(m => m(promptAPI))
    }
    run(command, args) {
        return execa(command, args, { cwd: this.context })
    }
    async create() {
        const {name,context,run} = this;
        let preset = await this.promptAndResolvePreset()
        console.log('preset', preset);
        preset = cloneDeep(preset);
        preset.plugins['@vue/cli-service'] = Object.assign({projectName: name}, preset);
        console.log(`✨  Creating project in ${chalk.yellow(context)}.`)
        const pkg = {
            name,
            version: '0.1.0',
            private: true,
```

```
            devDependencies: {}
        }
        const deps = Object.keys(preset.plugins)
        deps.forEach(dep => {
            pkg.devDependencies[dep] = 'latest';
        })
        await writeFileTree(context, {
            'package.json': JSON.stringify(pkg, null, 2)
        })
        console.log(`📦  Initializing git repository...`)
        await run('git init');
        console.log(`⚙\u{fe0f} Installing CLI plugins. This might take a while...`)
        await run('npm install');
        console.log(`🚀  Invoking generators...`)
        const plugins = await this.resolvePlugins(preset.plugins)
        const generator = new Generator(context, {pkg,plugins})
        await generator.generate();
+       console.log(`📦  Installing additional dependencies...`)
+       await run('npm install');
+       console.log('📄  Generating README.md...')
+       await writeFileTree(context, {
+           'README.md': `cd ${name}\n npm run serve`
+       })
+       await run('git', ['add', '-A'])
+       await run('git', ['commit', '-m', 'created', '--no-verify'])
+       console.log(`🎉  Successfully created project ${chalk.yellow(name)}.`)
+       console.log(
+           `👉  Get started with the following commands:\n\n` +
+           (chalk.cyan(`cd ${name}\n`)) +
+           (chalk.cyan(`npm run serve`))
+       )
+       generator.printExitLogs()
    }
    //遍历插件的generator,插件通过GeneratorAPI向package.json中加入依赖或字段，并通过render准备添加文件
    async resolvePlugins(rawPlugins) {
        const plugins = []
        for (const id of Object.keys(rawPlugins)) {
            try{
                const apply = loadModule(`${id}/generator`, this.context) || (() => {})
                let options = rawPlugins[id] || {}
                plugins.push({ id, apply, options })
            }catch(error){
                console.log(error);
            }
        }
        return plugins
    }
    resolveFinalPrompts() {
        this.injectedPrompts.forEach(prompt => {
            const originalWhen = prompt.when || (() => true)
            prompt.when = answers => {
                return isManualMode(answers) && originalWhen(answers)
            }
        })
        const prompts = [
            this.presetPrompt,
            this.featurePrompt,
            ...this.injectedPrompts,
        ]
        return prompts
    }
    async promptAndResolvePreset(answers = null) {
        if (!answers) {
            answers = await inquirer.prompt(this.resolveFinalPrompts())
        }
        let preset;
        if (answers.preset && answers.preset !== '__manual__') {
            preset = await this.resolvePreset(answers.preset)
        } else {
            preset = {
                plugins: {}
            }
            answers.features = answers.features || []
            this.promptCompleteCbs.forEach(cb => cb(answers, preset))
        }
        return preset
    }
    async resolvePreset (name) {
        const savedPresets = this.getPresets()
        return savedPresets[name];
    }
    getPresets() {
        return Object.assign({}, defaults.presets)
    }
    resolveIntroPrompts() {
        const presets = this.getPresets()
        const presetChoices = Object.entries(presets).map(([name]) => {
            let displayName = name
            if (name
                displayName = 'Default'
            } else if (name
                displayName = 'Default (Vue 3)'
            }
            return {
                name: `${displayName}`,
                value: name
            }
        })
        const presetPrompt = {
            name: 'preset',
            type: 'list',
            message: `Please pick a preset:`,
            choices: [
                ...presetChoices,
```

```
                {
                    name: 'Manually select features',
                    value: '__manual__'
                }
            ]
        }
        const featurePrompt = {
            name: 'features',
            when: isManualMode,
            type: 'checkbox',
            message: 'Check the features needed for your project:',
            choices: [],
            pageSize: 10
        }
        return {
            presetPrompt,
            featurePrompt
        }
    }
}

module.exports = Creator;
```

8.2 Generator.js

packages\zhang-cli\lib\Generator.js

```
const { isPlugin,matchesPluginId } = require('zhang-cli-shared-utils')
const GeneratorAPI = require('./GeneratorAPI')
const normalizeFilePaths = require('./util/normalizeFilePaths')
const writeFileTree = require('./util/writeFileTree')
const ejs = require('ejs')
class Generator {
    constructor(context, { pkg = {}, plugins = [] } = {}) {
        this.context = context
        this.plugins = plugins
        this.files = {}
        this.fileMiddlewares = []
        this.pkg = pkg;
        this.allPluginIds = Object.keys(this.pkg.dependencies || {})
            .concat(Object.keys(this.pkg.devDependencies || {}))
            .filter(isPlugin)
        const cliService = plugins.find(p => p.id
        this.rootOptions = cliService.options;
    }
    async generate() {
        await this.initPlugins()
        this.extractConfigFiles()
        await this.resolveFiles()
        this.sortPkg()
        this.files['package.json'] = JSON.stringify(this.pkg, null, 2) + '\n'
        await writeFileTree(this.context, this.files)
    }
    sortPkg() {
        console.log('ensure package.json keys has readable order');
    }
    async resolveFiles() {
        const files = this.files
        for (const middleware of this.fileMiddlewares) {
            await middleware(files, ejs.render)
        }
        normalizeFilePaths(files)
    }
    extractConfigFiles() {
        console.log('extractConfigFiles');
    }
    async initPlugins() {
        const { rootOptions } = this
        for (const plugin of this.plugins) {
            const { id, apply, options } = plugin
            const api = new GeneratorAPI(id, this, options, rootOptions)
            await apply(api, options, rootOptions)
        }
    }
    hasPlugin(_id) {
        return [
          ...this.plugins.map(p => p.id),
          ...this.allPluginIds
        ].some(id => {
          return matchesPluginId(_id, id);
        })
    }
+    printExitLogs(){
+        console.log('printExitLogs');
+    }
}

module.exports = Generator;
```