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=154 sentences=388, words=2406

1. 创建项目

- 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

1.1 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

1.1.1 安装

```
npm i lerna -g
```

1.1.2 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

1.1.3 文件

1.1.3.1 package.json

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

1.1.3.2 lerna.json

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

1.1.3.3 .gitignore

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

1.1.4 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

1.1.4.1 package.json

package.json

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

1.1.4.2 lerna.json

lerna.json

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

1.1.4.3 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

1.1.5 创建子项目

```
lerna create vite-cli
lerna create  vite-project
```

1.1.5.1 vite-cli

1.1.5.1.1 package.json

```
{
  "name": "vite-cli",
  "version": "0.0.0",
  "bin":{
    "vite-cli":"./bin/vite.js"
  },
  "scripts": {}
}
```

1.1.5.1.2 vite.js

packages\vite-cli\bin\vite.js

```
function start() {
    require('../lib/cli')
}
start()
```

1.1.5.1.3 cli.js

packages\vite-cli\lib\cli.js

```
console.log('vite');
```

1.1.5.2 vite-project

1.1.5.2.1 package.json

```
{
  "name": "vite-project",
  "version": "0.0.0",
  "scripts": {}
}
```

1.1.6 创建软链接

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

1.2 安装依赖

```
cd packages/vite-project
yarn workspace vite-project add vite

cd packages/vite-cli
yarn workspace vite-cli add   es-module-lexer koa koa-static magic-string chalk dedent hash-sum
```

2. 启动并调试

2.1 package.json

packages\vite-project\package.json

```
{
  "name": "vite-project",
  "version": "0.0.0",
+ "scripts": {
+   "dev":"vite"
+ },
  "dependencies": {
    "vite": "^2.4.1"
  }
}
```

2.2 index.html

packages\vite-project\index.html

```

<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite Apptitle>
  head>
  <body>
    <div id="app">div>
    <script type="module" src="/src/main.js">script>
  body>
html>
```

2.3 src\main.js

packages\vite-project\src\main.js

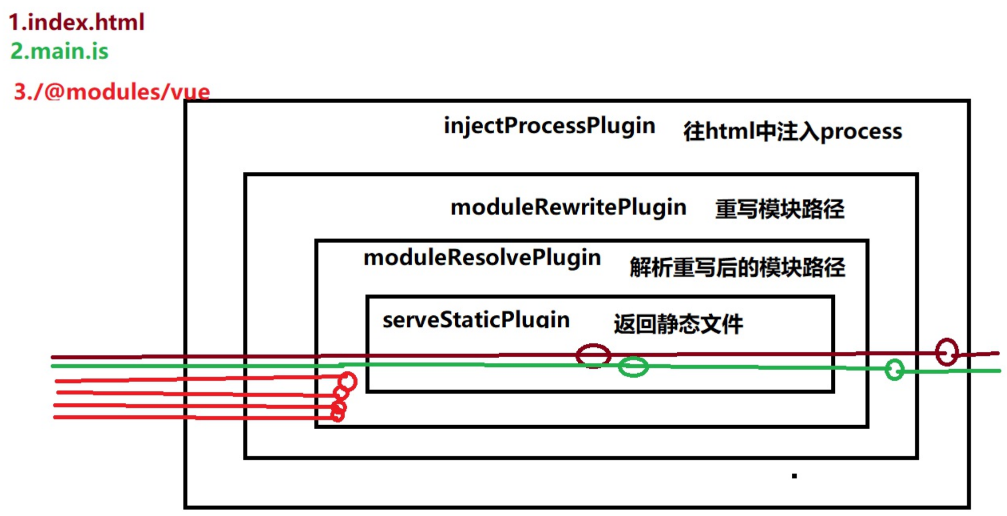
```
console.log('main.js');
```

2.4 launch.json

.vscode\launch.json

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

3. 实现静态服务



3.1 serverPluginServeStatic.js

packages\vite-cli\lib\serveStaticPlugin.js

```
const path = require('path');
const static = require('koa-static');

function serveStaticPlugin({app,projectRoot}){
   app.use(static(projectRoot));
}
module.exports = serveStaticPlugin;
```

3.2 cli.js

packages\vite-cli\lib\cli.js

```
const Koa = require('koa');
const serveStaticPlugin = require('./serveStaticPlugin');
function createServer() {
    const app = new Koa();
    const root = process.cwd();

    const context = {
        app,
        root
    }
    app.use((ctx, next) => {

        Object.assign(ctx, context);
        return next();
    });
    const resolvedPlugins = [
        serveStaticPlugin
    ];

    resolvedPlugins.forEach(plugin => plugin(context));
    return app;
}
createServer().listen(4000);
```

4.重写导入路径

- Vue单文件组件(SFC)规范 vue文件用于表示一个单一组件，其内使用类html语法，顶级标签有template,script,style和自定义的标签

4.1 安装

```
yarn workspace vite-project add  vue@3  @vitejs/plugin-vue @vue/compiler-sfc
node ./node_modules/esbuild/install.js
```

4.2 nodemon.json

packages\vite-project\nodemon.json

```
{
    "watch":["../vite-cli"]
}
```

启动服务

```
nodemon ../vite-cli/bin/vite.js
```

4.3 vite.config.js

packages\vite-project\vite.config.js

```
import { defineConfig } from "vite";
import vue from "@vitejs/plugin-vue";
export default defineConfig({
  plugins: [vue({})],
});
```

4.4 main.js

packages\vite-project\src\main.js

```
+import {createApp} from 'vue';
+console.log(createApp);
```

4.5 cli.js

packages\vite-cli\lib\cli.js

```
const Koa = require('koa');
const dedent = require('dedent');
const serveStaticPlugin = require('./serveStaticPlugin');
+const moduleRewritePlugin = require('./moduleRewritePlugin');
function createServer() {
  //koa的实例
  const app = new Koa();
  //当前命令所在的根目录
  const root = process.cwd();
  //上下文
  const context = {
    app,
    root
  }
  app.use((ctx, next) => {
    Object.assign(ctx, context);
    return next();
  });
  const resolvedPlugins = [
+   moduleRewritePlugin,
    serveStaticPlugin
  ]
  resolvedPlugins.forEach(plugin => plugin(context));
  return app;
}
createServer().listen(4000, async () => {
  const chalk = await import('chalk');
  console.log(
    dedent`${chalk.default.green(`vite-cli dev server running at:`)}
           > Local: http://localhost:4000/
    `
  );
});
```

4.6 serverPluginModuleRewrite.js

packages\vite-cli\lib\serverPluginModuleRewrite.js

```
let { readBody } = require('./utils');
let MagicString = require('magic-string');
let { parse } = require('es-module-lexer');
let path = require('path');
async function rewriteImports(content) {
  var magicString = new MagicString(content);
  let imports = await parse(content);
  if (imports && imports.length > 0) {
    for (let i = 0; i < imports[0].length; i++) {
      const { n, s, e } = imports[0][i];

      if (/^[^\/\.]/.test(n)) {
        const rewriteModuleId = `/node_modules/.vite/${n}.js`;
        magicString.overwrite(s, e, rewriteModuleId);
      }
    }
  }
  return magicString.toString();
}
function moduleRewritePlugin({ root, app }) {
  app.use(async (ctx, next) => {
    await next();

    if (ctx.body && ctx.response.is('js')) {
      const content = await readBody(ctx.body);
      const result = await rewriteImports(content);
      ctx.body = result;
    }
  });
}
module.exports = moduleRewritePlugin;
```

4.7 utils.js

packages\vite-cli\lib\utils.js

```
const { Readable } = require('stream')
async function readBody(stream) {
  if (stream instanceof Readable) {
    return new Promise((resolve) => {
      let buffers = [];
      stream
        .on('data', (chunk) => buffers.push(chunk))
        .on('end', () => resolve(Buffer.concat(buffers).toString()));
    })
  } else {
    return stream.toString()
  }
}
exports.readBody = readBody
```

5.解析vue文件

5.1 moduleResolvePlugin.js

packages\vite-cli\lib\moduleResolvePlugin.js

```
const fs = require('fs').promises;
const node_modulesRegexp = /^\/node_modules\/\.vite\/(.+?)\.js/
const { resolveVue } = require('./utils')
function moduleResolvePlugin({ app, root }) {
  const vueResolved = resolveVue(root)
  app.use(async (ctx, next) => {
    if (!node_modulesRegexp.test(ctx.path)) {
      return next();
    }
    const id = ctx.path.match(node_modulesRegexp)[1];
    const modulePath = vueResolved[moduleId];

    const content = await fs.readFile(modulePath, 'utf8');
    ctx.type = 'js';

    ctx.body = content
  });
}
module.exports = moduleResolvePlugin;
```

5.2 injectProcessPlugin.js

packages\vite-cli\lib\injectProcessPlugin.js

```
const { readBody } = require("./utils");
function injectProcessPlugin({ root, app }) {
  const devInjection = `

      window.process = {env:{NODE_ENV:'development'}}

    `
  app.use(async (ctx, next) => {
    await next();
    if (ctx.response.is('html')) {
      const html = await readBody(ctx.body);
      ctx.body = html.replace(//, `{{content}}amp;${devInjection}`)
    }
  })
}
module.exports = injectProcessPlugin
```

5.3 utils.js

packages\vite-cli\lib\utils.js

```
const { Readable } = require('stream');
const Module = require('module')
async function readBody(stream) {
    if(stream instanceof Readable){
        return new Promise((resolve) => {
            let buffers = [];
            //当我们从流中读取到数据后
            stream
            .on('data',chunk=>buffers.push(chunk))
            .on('end',()=>resolve(Buffer.concat(buffers).toString('utf8')))
        });
    }else{
        return Promise.resolve(stream.toString('utf8'));
    }
}
exports.readBody = readBody;

+function resolveVue(root) {
+  let require = Module.createRequire(root);
+  const resolvePath = (moduleName) => require.resolve(`@vue/${moduleName}/dist/${moduleName}.esm-bundler.+js`);
+  return {
+    '@vue/shared': resolvePath('shared'),
+    '@vue/reactivity': resolvePath('reactivity'),
+    '@vue/runtime-core': resolvePath('runtime-core'),
+    'vue': resolvePath('runtime-dom'),
+  }
+}
+exports.resolveVue = resolveVue;
```

5.4 cli.js

packages\vite-cli\lib\cli.js

```
const Koa = require('koa');
const dedent = require('dedent');
const serveStaticPlugin = require('./serveStaticPlugin');
const moduleRewritePlugin = require('./moduleRewritePlugin');
+const moduleResolvePlugin = require('./moduleResolvePlugin');
+const injectProcessPlugin = require('./injectProcessPlugin');
function createServer() {
  //koa的实例
  const app = new Koa();
  //当前命令所在的根目录
  const root = process.cwd();
  //上下文
  const context = {
    app,
    root
  }
  app.use((ctx, next) => {
    Object.assign(ctx, context);
    return next();
  });
  const resolvedPlugins = [
+   injectProcessPlugin,
    moduleRewritePlugin,
+   moduleResolvePlugin,
    serveStaticPlugin
  ]
  resolvedPlugins.forEach(plugin => plugin(context));
  return app;
}
createServer().listen(4000, async () => {
  const chalk = await import('chalk');
  console.log(
    dedent`${chalk.default.green(`vite-cli dev server running at:`)}
           > Local: http://localhost:4000/
    `
  );
});
```

6.编译vue模板

6.1 main.js

packages\vite-project\src\main.js

```
import {createApp} from 'vue';
+import App from './App.vue';
+createApp(App).mount("#app");
```

6.2 App.vue

packages\vite-project\src\App.vue

```

    App

export default {
    name:'App'
}

```

6.3 vuePlugin.js

packages\vite-cli\lib\vuePlugin.js

```
const fs = require('fs').promises;
const path = require('path');
const hash = require('hash-sum')
const { parse, compileScript, compileTemplate, rewriteDefault } = require('@vue/compiler-sfc');
var cache = new Map();
function vuePlugin({ root, app }) {
  app.use(async (ctx, next) => {
    if (!ctx.path.endsWith('.vue')) {
      return await next();
    }
    const filePath = path.join(root, ctx.path);
    const descriptor = await getDescriptor(filePath, root);
    let targetCode = ``;

    if (descriptor.script) {
      let script = compileScript(descriptor, { reactivityTransform: false });
      scriptCode = rewriteDefault(script.content, '_sfc_main')
      targetCode += scriptCode;
    }

    if (descriptor.template) {
      let templateContent = descriptor.template.content;
      const { code: templateCode } = compileTemplate({ source: templateContent });
      targetCode += templateCode;
    }
    targetCode += `\n_sfc_main.render=render`;
    targetCode += `\nexport default _sfc_main`;
    ctx.type = 'js';
    ctx.body = targetCode;
  });
}
async function getDescriptor(filePath) {
  if (descriptorCache.has(filePath)) {
    return descriptorCache.get(filePath);
  }
  const content = await fs.readFile(filePath, 'utf8');
  const { descriptor } = parse(content, { filename: filePath });
  descriptorCache.set(filePath, descriptor);
  return descriptor;
}
module.exports = vuePlugin;
```

```
const { parse, compileScript, compileTemplate, rewriteDefault } = require('@vue/compiler-sfc');
const dedent = require('dedent');
const App = `

  App

export default {
  name: 'App'
}

h1 {
  color: red;
}

h1 {
  background-color: green;
}

`;
let { descriptor } = parse(App, { filename: 'App.vue' });
let targetCode = '';

if (descriptor.styles.length > 0) {
  let styleCodes = '';
  descriptor.styles.forEach((style, index) => {
    const query = `?t=${Date.now()}&vue&type=style&index=${index}&lang.css`;
    const id = '/src/App.vue';
    const styleRequest = id + query;
    styleCodes += `\nimport ${JSON.stringify(styleRequest)}`
  });
  targetCode += styleCodes;
}

if (descriptor.script) {
  let scriptCode = compileScript(descriptor, {
    reactivityTransform
      : false
  });
  scriptCode = rewriteDefault(scriptCode.content, '_sfc_main');
  targetCode += scriptCode;
}
if (descriptor.template) {
  const templateContent = descriptor.template.content;
  let { code } = compileTemplate({
    source: templateContent
  });
  code = code.replace(/export function render/, 'function _sfc_render');
  targetCode += code;
}
targetCode += `
\n_sfc_main.render = _sfc_render;
\nexport default _sfc_main;
`;
console.log(targetCode);
```

6.4 cli.js

packages\vite-cli\lib\cli.js

```
const Koa = require('koa');
const dedent = require('dedent');
const serveStaticPlugin = require('./serveStaticPlugin');
const moduleRewritePlugin = require('./moduleRewritePlugin');
const moduleResolvePlugin = require('./moduleResolvePlugin');
const injectProcessPlugin = require('./injectProcessPlugin');
+const vuePlugin = require('./vuePlugin')
function createServer() {
  //koa的实例
  const app = new Koa();
  //当前命令所在的根目录
  const root = process.cwd();
  //上下文
  const context = {
    app,
    root
  }
  app.use((ctx, next) => {
    Object.assign(ctx, context);
    return next();
  });
  const resolvedPlugins = [
    injectProcessPlugin,
    moduleRewritePlugin,
    moduleResolvePlugin,
+   vuePlugin,
    serveStaticPlugin
  ]
  resolvedPlugins.forEach(plugin => plugin(context));
  return app;
}
createServer().listen(4000, async () => {
  const chalk = await import('chalk');
  console.log(
    dedent`${chalk.default.green(`vite-cli dev server running at:`)}
           > Local: http://localhost:4000/
    `
  );
});
```

7.支持样式

7.1 vuePlugin.js

packages\vite-cli\lib\vuePlugin.js

```javascript
const fs = require('fs').promises;
const path = require('path');
const hash = require('hash-sum')
const { parse, compileScript, compileTemplate, rewriteDefault, compileStyleAsync } = require('@vue/compiler-sfc');
var descriptorCache = new Map();
function vuePlugin({ root, app }) {
  app.use(async (ctx, next) => {
    if (!ctx.path.endsWith('.vue')) {
      return await next();
    }
    const filePath = path.join(root, ctx.path);
    const descriptor = await getDescriptor(filePath, root);
    if (ctx.query.type === 'style') {
      const block = descriptor.styles[Number(ctx.query.index)];
      let result = await transformStyle(block.content, descriptor, ctx.query.index);
      ctx.type = 'js';
      ctx.body = `
        let style = document.createElement('style');
        style.innerHTML = ${JSON.stringify(result.code)};
        document.head.appendChild(style);
      `;
    } else {
      let targetCode = ``;
      if (descriptor.styles.length) {
        let stylesCode = '';
        descriptor.styles.forEach((style, index) => {
          const query = `?vue&type=style&index=${index}&lang.css`
          const id =ctx.path;
          const styleRequest = (id + query).replace(/\\/g, '/');
          stylesCode += `\nimport ${JSON.stringify(styleRequest)}`
        });
        targetCode += stylesCode;
      }

      if (descriptor.script) {
        let script = compileScript(descriptor, { id: filePath, reactivityTransform: false });
        scriptCode = rewriteDefault(script.content, '_sfc_main')
        targetCode += scriptCode;
      }

      if (descriptor.template) {
        let templateContent = descriptor.template.content;
        let { code } = compileTemplate({ id: filePath, source: templateContent });
        code = code.replace(/export function render/, 'function _sfc_render');
        targetCode += code;
      }
      targetCode += `\n_sfc_main.render=_sfc_render`;
      targetCode += `\nexport default _sfc_main`;
      ctx.type = 'js';
      ctx.body = targetCode;
    }
  });
}
async function transformStyle(code, descriptor, index) {
  const block = descriptor.styles[index];
  const result = await compileStyleAsync({
    filename: descriptor.filename,
    source: code,
    id: `data-v-${descriptor.id}`,
    scoped: block.scoped
  })
  return result;
}
async function getDescriptor(filePath) {
  if (descriptorCache.has(filePath)) {
    return descriptorCache.get(filePath);
  }
  const content = await fs.readFile(filePath, 'utf8');
  const { descriptor } = parse(content, { filename: filePath });
  descriptorCache.set(filePath, descriptor);
  return descriptor;
}
module.exports = vuePlugin;
```