

【架构师（第七篇）】脚手架之准备阶段编写

一尾流莺 2022-04-01 09:00 3549

关注

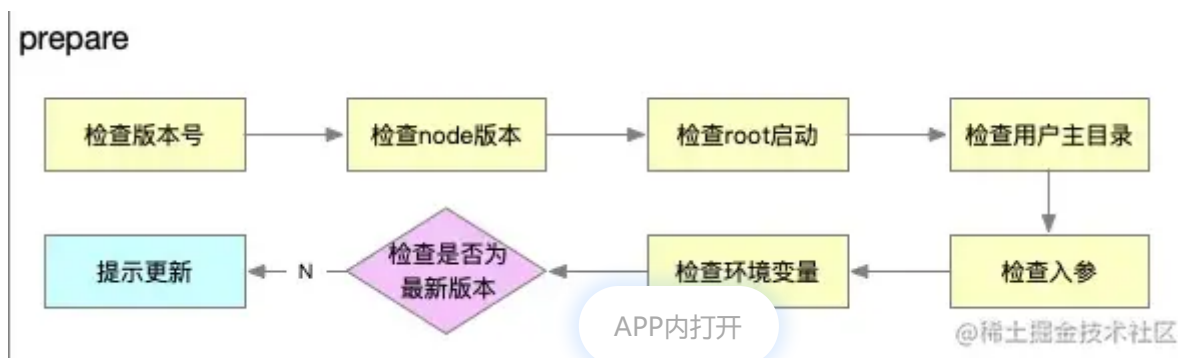
根据模块划分调整工程结构

- 核心模块: **Core**
- 命令模块: **Command**
- 模型模块: **Model**
- 工具模块: **Utils**

js 复制代码

```
1 hzw-cli-dev
2 |— command // 命令模块
3 |— core // 核心模块
4 |— utils // 工具模块
5 |— models // 模型模块
6 |— node_modules // 依赖
7 |— package-lock.json
8 |— package.json
9 |— lerna.json // lerna 配置文件
10
```

准备阶段流程图



```
1 // core\cli\bin\index.js
2 #! /usr/bin/env node
3
4 const importLocal = require('import-local')
5
6 // 如果当前项目中的 node_modules 中存在一个脚手架命令，全局的 node 环境中也存在一个脚手架命令的时候
7 // import-local 会优先选用项目中 node_modules 的版本，然后输出 log
8 if (importLocal(__filename)) {
9   require('npmlog').info('提示', '正在使用当前项目中 hzw-cli-dev 的版本')
10 } else {
11   // 使用全局下的脚手架命令
12   require('../lib')(process.argv.slice(2))
13 }
14
```

`require` 支持加载的模块类型 `.js/.json/.node`。

当加载 `.js` 模块时，需要使用 `module.exports/exports` 进行导出。

当加载 `.json` 模块时，会调用 `JSON.parse` 对模块进行解析，并返回一个对象。

当加载 `.node` 模块时，会使用一个 `c++` 插件，基本不用。

当加载 **任意类型的文件** 模块时，会当作 `.js` 去执行，如果内容不是 `js` 代码，那么会报错。

| log 工具

创建 `log` 包



js 复制代码

```
1 // 使用 lerna 创建包
2 lerna create @hzw-cli-dev/log
3 // 使用 lerna 给 log 包 安装依赖
4 lerna add npmlog utils/log
5 // 修改入口文件名为 index.js
6 // 修改 package.json 中的 main 字段为 lib/index.js
7 // 给 core\cli\package.json 添加依赖
8 "dependencies": {
9   "@hzw-cli-dev/utils": "^1.0.1",
```

APP内打开

定制 log



js 复制代码

```
1 'use strict';
2
3 // 引入 npmlog 模块
4 const log = require('npmlog');
5
6 // 从环境变量中读取 log.level
7 // log.level 的作用是：只有超过 level 设置的权重，log 才会生效
8 log.level = process.env.LOG_LEVEL || 'info';
9
10 // 定制 log 的 level 参数：(名称,权重,配置)
11 log.addLevel('success', 2000, { fg: 'red', bg: 'yellow', bold: true });
12
13 // 定制 log 的前缀
14 log.heading = 'hzw';
15 // 定制 log 前缀的样式
16 log.headingStyle = { fg: 'blue', bg: 'green', bold: true };
17
18 module.exports = log;
```

调用 log



js 复制代码

```
1 // core\cli\lib\index.js
2 // 引入我们封装的 npmlog 工具
3 const log = require('@hzw-cli-dev/log');
4 log.success('test', 'success...');
```

效果如下，还是挺好玩的，这样就可以根据自己的喜好高度定制 log 了

APP内打开

```
PS D:\imooc-learn\hzw-cli-dev\hzw-cli-dev> hzw-cli-dev
hzw success 友情提示,当前的版本是: 1.0.1
PS D:\imooc-learn\hzw-cli-dev\hzw-cli-dev> |
```

@稀土掘金技术社区

检查版本号

js 复制代码

```
1 'use strict';
2
3 // 引入当前脚手架的 package.json 文件
4 const pkg = require('../package.json');
5
6 // 引入我们封装的 npmlog 工具
7 const log = require('@hzw-cli-dev/log');
8
9 /**
10  * @description: 核心方法
11  * @param {*}
12  * @return {*}
13  */
14 function core() {
15   // 检查版本号
16   checkPkgVersion();
17 }
18
19 /**
```

APP内打开

 38 21 收藏

```
23 /
24 function checkPkgVersion() {
25   log.success('友情提示,当前的版本是:', pkg.version);
26 }
27
28 module.exports = core;
```

检查 node 版本

- 安装版本对比的第三方库 `semver`
- 安装定义脚手架输出颜色的库 `colors`

js 复制代码

```
1 lerna add semver core/cli
2 lerna add colors core/cli
```

定义最低 node 版本号

js 复制代码

```
1 // core\cli\lib\const.js
2 const LOWEST_NODE_VERSION = '17.0.0';
3
4 module.exports = {
5   LOWEST_NODE_VERSION,
6 };
7
```

检查 node 版本号是否符合要求

js 复制代码

```
1 // core\cli\lib\index.js
2 'use strict';
3
4 // 引入版本比对第三方库 semver
5 const semver = require('semver');
6 // 引入颜色库 colors
7 const colors = require('colors/safe');
8 // 引入当前脚手架的 package.json 文件
9 const pkg = require('../package.json');
```

APP内打开

```
13 const constant = require('./const');
14
15 /**
16  * @description: 核心方法
17  * @param {*}
18  * @return {*}
19  */
20 function core() {
21   try {
22     // 检查版本号
23     checkPkgVersion();
24     // 检查 node 版本
25     checkNodeVersion();
26   } catch (error) {
27     log.error(error.message);
28   }
29 }
30
31 /**
32  * @description: 检查当前的 node 版本,防止 node 版本过低调用最新 api 出错
33  * @param {*}
34  * @return {*}
35  */
36 function checkNodeVersion() {
37   // 获取当前 node 版本号
38   const currentVersion = process.version;
39   log.info('友情提示,当前的node版本是:', process.version);
40   // 获取最低 node 版本号
41   const lowestVersion = constant.LOWEST_NODE_VERSION;
42   // 对比最低 node 版本号
43   if (!semver.gte(currentVersion, lowestVersion)) {
44     throw new Error(colors.red('错误:node版本过低'));
45   }
46 }
47
48 module.exports = core;
```

效果如下

APP内打开



```
PS D:\imooc-learn\hzw-cli-dev\hzw-cli-dev> hzw-cli-dev
hzw success 友情提示,当前的脚手架版本是: 1.0.1
hzw info 友情提示,当前的node版本是: v16.14.0
hzw ERR! 错误:node版本过低
PS D:\imooc-learn\hzw-cli-dev\hzw-cli-dev> |
```

@稀土掘金技术社区

检查 root 账号启动

安装第三方库 `root-check` , 要指定版本, 不然 `2.0` 是用 `es module` 写的, 会报错。



js 复制代码

```
1 lerna add root-check@1.0.0 core/cli/
```



js 复制代码

```
1 // core\cli\lib\index.js
2 /**
3  * @description: 检查root账户并自动降级
4  * @param {*}
5  * @return {*}
6  */
7 function checkRoot() {
8   // 检查 root 等级并自动降级
9   const rootCheck = require('root-check');
10  rootCheck();
11 }
```

APP内打开



安装第三方库 `user-home` ，跨操作系统获取用户主目录。

安装第三方库 `path-exists` ，检查文件是否存在。



js 复制代码

```
1 lerna add user-home core/cli/
2 lerna add path-exists@4.0.0 core/cli/
```



js 复制代码

```
1 // core\cli\lib\index.js
2 /**
3  * @description:检查用户主目录
4  * @param {*}
5  * @return {*}
6  */
7 function checkUserHome() {
8   // 引入user-home 跨操作系统获取用户主目录
9   const userHome = require('user-home');
10  // 检查文件是否存在
11  const pathExists = require('path-exists').sync;
12  // 如果主目录不存在,抛出异常
13  if (!userHome || !pathExists(userHome)) {
14    throw new Error(colors.red('当前登录用户主目录不存在'));
15  }
16 }
```

检查入参

安装第三方库 `minimist` ，解析参数。



js 复制代码

```
1 lerna add minimist core/cli/
```



js 复制代码

```
1 /**
2  * @description: 解析参数,判断是否开启 ( APP内打开 全局变量中设置 log 等级
3  * @param {*}
```



```
8   const minimist = require('minimist');
9   args = minimist(process.argv.slice(2));
10  // 判断是否开启 debug 模式,并在全局变量中设置 log 等级
11  checkArgs();
12 }
13
14 /**
15  * @description: 判断是否开启 debug 模式,并在全局变量中设置 log 等级
16  * @param {*}
17  * @return {*}
18  */
19 function checkArgs() {
20   if (args.debug) {
21     process.env.LOG_LEVEL = 'verbose';
22   } else {
23     process.env.LOG_LEVEL = 'info';
24   }
25   // 设置 log 的等级
26   log.level = process.env.LOG_LEVEL;
27 }
```

检查环境变量

安装第三方库 `dotenv` 。

[js 复制代码](#)

```
1 lerna add dotenv core/cli/
```

[js 复制代码](#)

```
1 /**
2  * @description: 检查环境变量
3  * @param {*}
4  * @return {*}
5  */
6 function checkEnv() {
7   // 引入解析环境变量的库 dotenv
8   const dotenv = require('dotenv');
9   // 环境变量的路径
10  const dotenvPath = path.resolve(user
11  // 如果路径存在
```

[APP内打开](#)

```
16  }  
17  }  
18  // 创建默认的环境变量配置  
19  createDefaultConfig();  
20  log.verbose('环境变量', process.env.CLI_HOME_PATH);  
21  }  
22  
23  /**  
24   * @description: 创建默认的环境变量配置  
25   * @param {*}  
26   * @return {*}  
27   */  
28  function createDefaultConfig() {  
29    const cliConfig = {  
30      home: userHome,  
31    };  
32    // 如果 CLI_HOME 存在 使用CLI_HOME  
33    if (process.env.CLI_HOME) {  
34      cliConfig['cliHome'] = path.join(userHome, process.env.CLI_HOME);  
35    } else {  
36      // 如果 CLI_HOME 不存在 使用默认配置  
37      cliConfig['cliHome'] = path.join(userHome, constant.DEFAULT_CLI_HOME);  
38    }  
39    // 设置 process.env.CLI_HOME_PATH  
40    process.env.CLI_HOME_PATH = cliConfig.cliHome;  
41  }  
42
```

通用 npm API 模块封装

通过 **npm API**: <https://registry.npmjs.org/模块名>

可以获取到某个模块的信息.

也可以换成其他的镜像 比如淘宝源 <https://registry.npmmirror.com/模块名>



js 复制代码

```
1 // 通过 lerna 新建一个包 放在 utils 下面  
2 lerna create @hzw-cli-dev/get-npm-info ./utils/get-npm-info  
3 // 修改文件名和 main 属性为 index.js  
4 // core模块引入
```

APP内打开

```
9 lerna add url-join utils/get-npm-info
10 // 安装 semver 用来做版本比对
11 lerna add semver utils/get-npm-info
```

封装工具包 `get-npm-info`

[js](#) 复制代码

```
1 // utils\get-npm-info\lib\index.js
2 'use strict';
3
4 const axios = require('axios');
5 const urlJoin = require('url-join');
6 const semver = require('semver');
7
8 /**
9  * @description: 获取 npm 模块的信息
10  * @param {*} npmName npm 模块名称
11  * @param {*} register npm 镜像地址
12  * @return {*}
13  */
14 async function getNpmInfo(npmName, register) {
15   // 如果 npmName 不存在直接返回
16   if (!npmName) {
17     return null;
18   }
19   // 获取镜像地址 ,如果没有传递参数则默认使用 npm 源
20   const registerUrl = register || getRegister('taobao');
21   // 拼接url
22   const npmInfoUrl = urlJoin(registerUrl, npmName);
23   // 调用 npm API 获取数据
24   return axios
25     .get(npmInfoUrl)
26     .then((res) => {
27       if (res.status === 200) {
28         return res.data;
29       }
30       return null;
31     })
32     .catch((e) => {
33       return Promise.reject(e);
34     });
35 }
36
37 /**
```

APP内打开

```
42 function getRegister(origin) {
43   const originList = {
44     npm: 'https://registry.npmjs.org/',
45     taobao: 'https://registry.npmirror.com/',
46   };
47   return originList[origin];
48 }
49
50 /**
51  * @description: 获取模块版本号数组
52  * @param {*} npmName npm 模块名称
53  * @param {*} register npm 镜像地址
54  * @return {*} 模块的版本号
55  */
56 async function getNpmVersions(npmName, register) {
57   const data = await getNpmInfo(npmName, register);
58   if (data) {
59     return Object.keys(data.versions);
60   }
61   return [];
62 }
63
64 /**
65  * @description: 获取符合条件的版本号(大于当前版本的版本号)
66  * @param {*} baseVersion 当前版本
67  * @param {*} versions 版本号数组
68  * @return {*} 大于当前版本的版本号数组
69  */
70 function getNpmSemverVersions(baseVersion, versions) {
71   if (!versions || versions.length === 0) {
72     return [];
73   }
74   return versions
75     .filter((version) => semver.satisfies(version, `>=1.0.0`))
76     .sort((a, b) => semver.gt(b, a));
77 }
78
79 /**
80  * @description: 从 npm 获取符合条件的版本号(大于当前版本的最新版本号)
81  * @param {*} npmName npm 模块名称
82  * @param {*} register npm 镜像地址
83  * @param {*} baseVersion 当前版本
84  * @return {*} 最新版本号
85  */
86 async function getNpmSemverVersion(baseVersion, npmName, register) {
87   const versions = await getNpmVersions(npmName, register);
```

APP内打开

```
92 module.exports = {
93   getNpmInfo,
94   getNpmVersions,
95   getNpmSemverVersion,
96 };
97
```

获取 npm 包的信息



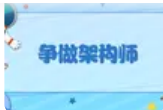
js 复制代码

```
1 // core\cli\lib\index.js
2
3 /**
4  * @description: 检查是否需要全局更新
5  *   1. 获取当前版本号 and 模块名
6  *   2. 调用 npm API , 获取所有的版本号
7  *   3. 提取所有的版本号, 比对哪些版本号是大于当前版本号的
8  *   4. 获取最新的版本号, 提示用户更新到该版本
9  * @param {*}
10 * @return {*}
11 */
12 async function checkGlobalUpdate() {
13   const currentVersion = pkg.version;
14   const npmName = pkg.name;
15   const { getNpmSemverVersion } = require('@hzw-cli-dev/get-npm-info');
16   // getNpmInfo(npmName);
17   const lastVersion = await getNpmSemverVersion(currentVersion, 'vue');
18   // 如果最新版本存在并且大于当前版本
19   if (lastVersion && semver.gt(lastVersion, currentVersion)) {
20     log.warn(
21       '友情提示',
22       colors.yellow('请更新版本: 当前的版本是:', lastVersion),
23     );
24     log.warn(
25       '友情提示',
26       colors.yellow('更新命令:', `npm install -g ${npmName}`),
27     );
28   }
29 }
```

APP内打开

标签: 前端 架构 前端框架

 38 21 收藏



我要当架构师

为了提升自己的能力，我在某课网买了一个架构师的课程。 本专栏...

已订阅

相关小册



VIP 现代 Web 布局

大漠_w3cplu...

2762购买

¥49.9



基于 Vite 的 SSG 框架开发实战

神三元

879购买

¥199

评论

输入评论 (Enter换行, Ctrl + Enter发送)

全部评论 21

最新

最热



feng_cc

7月前

lernaadd会更新其它的包，这样会干掉上次npm install file:xxx的本地包，lerna add xxx后，还要npm i一次才有效

👍 点赞 1



feng_cc

7月前

add后面添加参数--no-bootstrap就不用再npm install

👍 点赞 回复

APP内打开



buchiyu

1年前

👍 38


💬 21

★ 收藏

sort只会识别>0 <0 和=0
true和false不会识别的
developer.mozilla.org

👍 1 💬 12



一尾流莺 

1年前

你看到哪了

👍 点赞 💬 回复



buchiyu

1年前

子进程

“你看到哪了”

👍 点赞 💬 回复

查看更多回复 ▾

APP内打开



我终于知道那个file:xxx怎么使用了!!!!

需要手动cnpm install

每次老师做到core/cli下使用npm link的时候 我们使用link，还是不行 但是我尝试使用cnpm i 就可以

【但是貌似每次lerna add xxx 任何包 都需要手动在引入包的package.json同级文件夹下cnpm install 一下，所以可以使用cnpm i 代替lerna add 就好】

👍 点赞 💬 2



一尾流莺



1年前

这个我当时也是没好用，因为我一直没用cnpm，然后我就放弃了file: xxx这种方式，我还是每次都正常引入依赖，每次添加新的依赖以后，执行一次 lerna link 就好了。。。

👍 点赞 💬 回复



buchiyu

1年前

嗯，也行

“这个我当时也是没好用，因为我一直没用cnpm，然后我就放弃了file: xxx这种方...”

👍 点赞 💬 回复



燕小乙

LV.3

JY.5

1年前

好熟悉的代码 😊

👍 点赞 💬 1



一尾流莺



1年前

哈哈

👍 点赞 💬 回复



ATIP

JY.4



1年前

希望每天保持更新

👍 1 💬 回复

APP内打开

相关推荐

👍 38

💬 21

☆ 收藏



👁 11.5w 🍏 2201 💬 206

一尾流莺 2年前

【初学者笔记】整理的一些Vue3知识点

👁 3.9w 🍏 1005 💬 137

一尾流莺 1年前

【今天你更博学了么】一个神奇的前端动画 API requestAnimationFrame

👁 1.8w 🍏 370 💬 42

一尾流莺 1年前

【面试题解】CSS盒子模型与margin负值

👁 1.4w 🍏 109 💬 8

一尾流莺 1年前

【图文并茂】六十多个 vscode 插件，助你打造最强编辑器

👁 8650 🍏 227 💬 36

一尾流莺 1年前

【面试题解】你了解JavaScript常用的的十个高阶函数么？

👁 1.1w 🍏 101 💬 14

一尾流莺 2年前

【流莺书签】Vue3+TS的收藏网址小项目

👁 7653 🍏 124 💬 12

一尾流莺 1年前

【面试题解】JavaScript数据类型相关的六个面试题

👁 8618 🍏 38 💬 4

一尾流莺 1年前

【面试题解】vue-router有几种钩子函数？具体是什么及执行流程是怎样的？

👁 6594 🍏 122 💬 10

一尾流莺 1年前

【面试题解】你了解call, apply, bind吗？那你可以手写一个吗？

👁 7305 🍏 75 💬 14

APP内打开

一尾流莺 1年前

🍏 38

💬 21

☆ 收藏



一尾流莺 1年前

【初学者笔记】前端工程化必须要掌握的 webpack

👁 6304 👍 71 💬 11

一尾流莺 1年前

【实战技巧】Vue3+Vite工程常用工具的接入方法

👁 3440 👍 126 💬 25

一尾流莺 1年前

【面试题解】谈一谈JavaScript数据类型判断

👁 5713 👍 34 💬 1

一尾流莺 2年前

【实战技巧】VUE3.0实现简易的可拖放列表排序

👁 4614 👍 78 💬 3

一尾流莺 2年前

【年中总结】还是要继续努力呀 | 2021 年中总结

👁 2963 👍 104 💬 47

一尾流莺 2年前

【初学者笔记】一文学会使用Vuex

👁 3632 👍 109 💬 2

一尾流莺 1年前

【面试题解】初识 JavaScript 闭包

👁 4033 👍 47 💬 41

一尾流莺 1年前

【源码学习】Vue 初始化过程 (附思维导图)

👁 3006 👍 79 💬 6

一尾流莺 1年前

【前端财富】前端工程师装机指南 (windows10)

👁 3295 👍 56 💬 10

APP内打开