

Vue工程化开发

Vue工程化是为了应付在单页面开发里面的大型项目，它提全套的解决方案，使用Vue全家桶完成整个项目的开发，期间可能会使用到第三方的框架

在进行工程化开发的时候，Vue需要借用于webpack来实现工程化，因为SPA本质上面只有一个页面，所有页面跳都是虚拟的，通过组件来进行的，而这些组件也不能够直接在浏览器里面运行，所以需要借用于webpack来进行统译

webpack与vue的结合配置非常麻烦，对于初学者为说相当复杂，但是VUE提供一种便捷的方案去创建工程化的项目，这种东西叫“脚手架”

安装Vue脚手架

Vue的脚手架是快速构建Vue单页面开发平台。目的使用到的脚手架是4.X。本次授课主要以4.0及以上为主之前的版本3以前的版本，它的创建方式与现在的不同，我们后期做对比

```
1 $ npm install @vue/cli -g
```

安装完成以后，我们可以在DOS里面执行如下命令去看版本

```
1 $ vue -V
```

使用脚手架来创建Vue的工程化项目

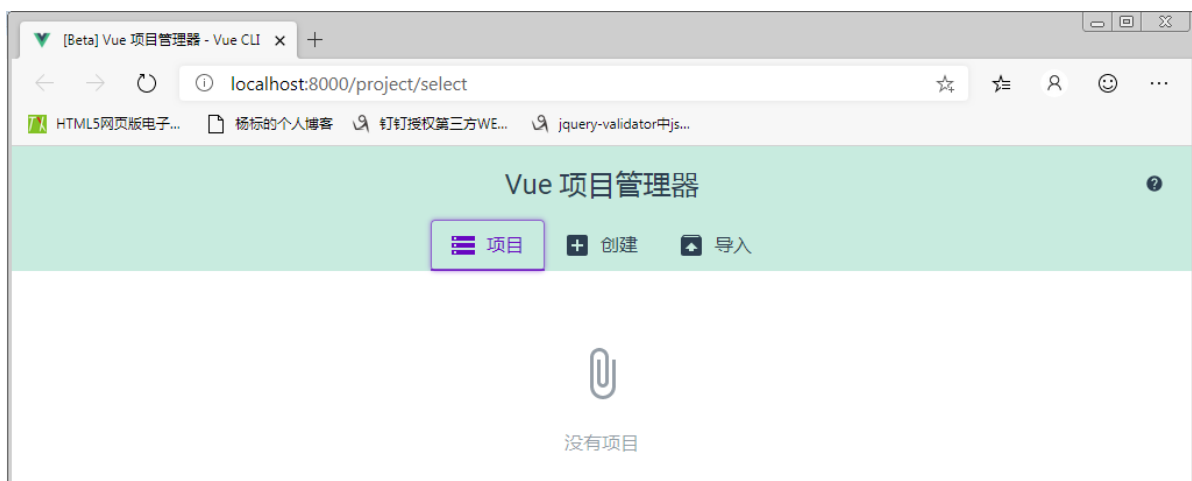
Vue的工程化项目它的创建方式有两种，第一种是图形界面创建，第二种是命令控制台创建

图形创建方式

第一步：首先，在你需要创建的文件夹下面打开控制台输入如下命令

```
1 $ vue ui
```

这个时候会启动一个浏览器，如下图所示



第二步：切换到创建项目，根据提示进行创建

Vue 项目管理器

项目

+ 创建

导入



D: H1904 1030 code



103001

103002

+ 在此创建新项目

创建新项目

详情

预设

功能

配置

项目文件夹

sms_spa

D:/H1904/1030/code/sms_spa



包管理器

yarn



更多选项

若目标文件夹已存在则将其覆盖



无新手指引的脚手架项目



Git

初始化 git 仓库 (建议)



取消

下一步

填写项目的相关信息，选择包管理器为 `npm` 或 `yarn`

创建新项目

详情

预设

功能

配置

预设就是一套定义好的插件和配置。你也可以将自己的配置保存成预设，方便以后创建项目使用。

选择一套预设

默认

babel, eslint

手动

手动配置项目

远程预设

从 git 仓库拉取预设

← 上一步

下一步 →

在这里，我们选择手动创建项目配置

创建新项目

详情

预设

功能

配置

在项目创建之后，你仍然可以通过安装插件来增加功能。

选择功能

Babel

Transpile modern JavaScript to older versions (for compatibility) [查看详情](#)

TypeScript

Add support for the TypeScript language [查看详情](#)

Progressive Web App (PWA) Support

Improve performances with features like Web manifest and Service workers [查看详情](#)

Router

Structure the app with dynamic pages [查看详情](#)

Vuex

Manage the app state with a centralized store [查看详情](#)

CSS Pre-processors

Add support for CSS pre-processors like Sass, Less or Stylus [查看详情](#)

Linters / Formatter

Check and enforce code quality with ESLint or Prettier [查看详情](#)

Unit Testing

Add a Unit Testing solution like Jest or Mocha [查看详情](#)

E2E Testing

Add an End-to-End testing solution to the app like Cypress or Nightwatch [查看详情](#)

使用配置文件

将插件的配置保存在各自的配置文件 (比如 '.babelrc') 中。

← 上一步

下一步 →

在这一步里面，根据要应的需求，开启相应的插件

创建新项目

详情

预设

功能

配置

Use history mode for router? (Requires proper server setup for index fallback in production)
By using the HTML5 History API, the URLs don't need the '#' character anymore. [查看详情](#)

Pick a CSS pre-processor:
PostCSS, Autoprefixer and CSS Modules are supported by default.

Sass/SCSS (with node-sass)

← 上一步

✓ 创建项目

在这一步里面，根据要应的需求，开启相应的插件

创建新项目

详情

预设

功能

配置

Use history mode for router? (Requires proper server setup for index fallback in production)
By using the HTML5 History API, the URLs don't need the '#' character anymore. [查看详情](#)

Pick a CSS pre-processor:
PostCSS, Autoprefixer and CSS Modules are supported by default.

Sass/SCSS (with node-sass)

这里换成dart-sass

根据上一步的配置，选择相应的配置

保存为新预设

×

预设名

将功能和配置保存为一套新的预设

取消

创建项目，不保存预设

保存预设并创建项目

在这里，我们不保存配置，下次如果要创建项目重新来一遍

当点击完成以后在控制台就会创建项目

```
管理员: C:\Windows\system32\cmd.exe - vue ui
D:\H1904\1030\code>vue ui
❑ Starting GUI...
❑ Ready on http://localhost:8000
❑ Creating project in D:\H1904\1030\code\sms_spas.
```

想过漫长的等待，我们就看到一个项目了

命令行方式创建

它是一种快速的创建试，我们只需要文件夹下面打开控制台，执行下面的命令就可以了

```
1 $ vue create sms_spas1
```

执行命令以后出现如下界面，需要我选择配置

```
管理员: C:\Windows\system32\cmd.exe - vue create sms_spas1
Vue CLI v4.0.5
? Please pick a preset:
> default (babel, eslint)
  Manually select features
```

在此，我们选择第二个 **Manually select features** ,手动配置

```
管理员: Windows PowerShell
Vue CLI v4.0.5
? Please pick a preset: Manually select features
? Check the features needed for your project:
  (<*) Babel
  < > TypeScript
  < > Progressive Web App (PWA) Support
  (<*) Router
  (<*) Vuex
  (<*) CSS Pre-processors
>< > Linter / Formatter
  < > Unit Testing
  < > E2E Testing
```

出现这个界面以后，按空格选择自己所需要的插件

```
管理员: Windows PowerShell
Vue CLI v4.0.5
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors
? Use history mode for router? (Requires proper server setup for index fallback in production) <Y/n> _
```

在这一步，问我们的Vue-router是否启用 **history** 模式，我们选NO。因为我们使用 **hash** 模式（hash模式会在地址栏产生一个#号）

```
管理员: Windows PowerShell
Vue CLI v4.0.5
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default):
  Sass/SCSS (with dart-sass)
> Sass/SCSS (with node-sass)
  Less
  Stylus
```

```
管理员: Windows PowerShell
Vue CLI v4.0.5
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default):
  Sass/SCSS (with dart-sass)
> Sass/SCSS (with node-sass)
  Less
  Stylus
```

这里选这个东西dart-sass

在这里，我们选择 `node-sass` 来进行

在这里，我们选择 `dart-sass` 来进行

```
管理员: Windows PowerShell
Vue CLI v4.0.5
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Sass/SCSS (with node-sass)
? Where do you prefer placing config for Babel, PostCSS, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

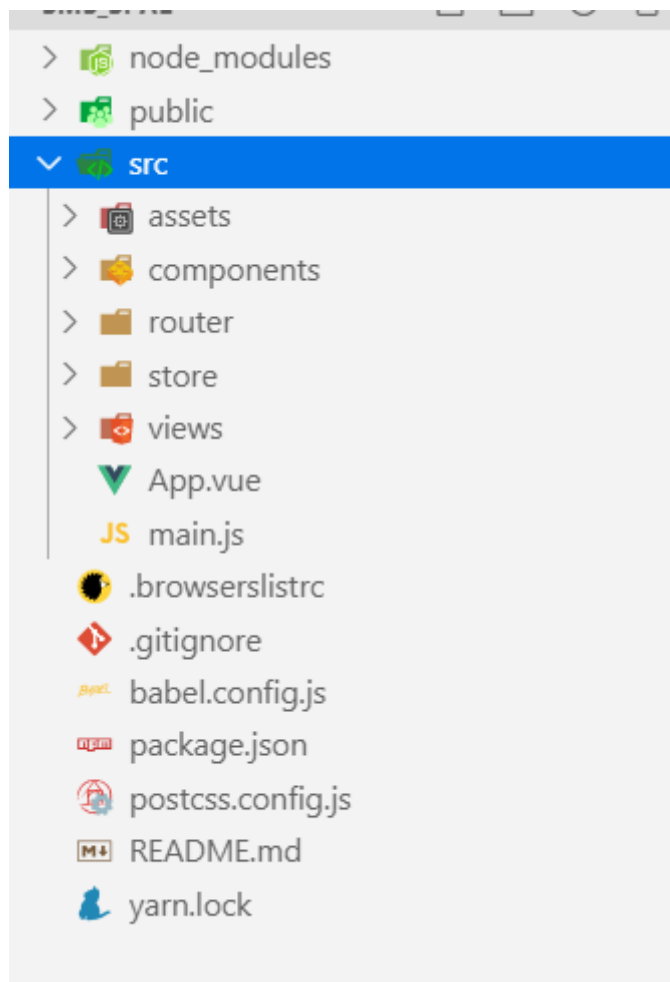
这一步询问我们配置文件写在什么地方。我们选第一个，把配置文件写在单独的文件里面

```
管理员: Windows PowerShell
Vue CLI v4.0.5
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, CSS Pre-processors
? Use history mode for router? (Requires proper server setup for index fallback in production) No
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Sass/SCSS (with node-sass)
? Where do you prefer placing config for Babel, PostCSS, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? (y/N) _
```

询问是否保存这个配置，我们不保存，下次重新创建

注意：使用控制台创建的时候，如果发现系统里面已经安装了yarn,那么优先使用yarn

分析脚手架创建的目录结构



1. public公共目录，存放了一个index.html文件，这是单页面的html文件
2. src目录是我们的程序员的目录，我们所开发的vue代码全在这里面
3. `.babel.config.js` 这是 `babel` 的配置文件
4. `postcss.config.js` 这是 `postcss` 的配置文件
5. `.browserslistrc` 这个是为了配置浏览器列表兼容性的，以前是写在 `package.json` 里面的

作为一个程序员，它主要使用的目录就是 `src` 目录

src目录结构

1. assets静态资源目录，在vue开发过程当，我们可以把图片，CSS以及SCSS等放在里面
2. components目录用于存放vue的组件（模块组件，也叫公共组件）
3. router文件夹里面存放路由文件，一般情况下，里面只有一个 `index.js` 文件
4. store文件夹里面存放vuex的状态，action以及mapper，mutation等
5. views文件夹，存放vue的虚拟页面（Login.vue或register.vue）
6. App.vue 项目当中的根组件，入口组件，相当于我们之前学习的 `<div id="app"></div>`
7. main.js这个是整个程序的入口文件，也是webpack的入口文件，所有程序从这个文件启动

脚手架创建的项目启动

使用 `@vue/cli` 创建的项目，它有两种，一个是开发环境，一个是生产环境，针对不同的环境，我们要启动不同的命令，这个时候，我们可以打开 `package.json` 这个文件，里面详细的配置了两个环境的启动过程

```
1 {
2   "name": "sms_spa1",
3   "version": "0.1.0",
```



```
4   "private": true,
5   "scripts": {
6     "serve": "vue-cli-service serve",
7     "build": "vue-cli-service build"
8   },
9   "dependencies": {
10    "core-js": "^3.3.2",
11    "vue": "^2.6.10",
12    "vue-router": "^3.1.3",
13    "vuex": "^3.0.1"
14  },
15  "devDependencies": {
16    "@vue/cli-plugin-babel": "^4.0.0",
17    "@vue/cli-plugin-router": "^4.0.0",
18    "@vue/cli-plugin-vuex": "^4.0.0",
19    "@vue/cli-service": "^4.0.0",
20    "node-sass": "^4.12.0",
21    "sass-loader": "^8.0.0",
22    "vue-template-compiler": "^2.6.10"
23  }
24 }
```

- serve代表开发环境
- build代表生产环境

我们可以根据这两个东西去启动项目

```
1  $ npm run serve
```

或

```
1  $ yarn run serve
```

特别注意

如果使用脚手创建项目失败的，可以在其它的电脑上面创建一个以后，然后去掉 `node_modules` 文件夹，然后复制到自己电脑，再根据自己的电脑去安装依赖就可以了 `npm install`