VUE-CLI

VUE-CLI作为vue的脚手架，可以帮助我们在实际开发中自动生成vue.js的模板工程。

> CLI（command-line interface）命令行界面，是在图形用户界面得到普及之前使用最为广泛的用户界面，它通常不支持鼠标，用户通过键盘输入指令，计算机接收到指令后，予以执行。也有人称之为字符用户界面（CUI）。命令行界面要较图形用户界面节约计算机系统的资源、操作速度快。所以，图形用户界面的操作系统中，都保留着可选的命令行界面。

第一节 项目环境

1. 搭建项目环境

开发环境配置项及配置方法

1. 安装node.js
2. 安装vue-cli

```
vue要安装在全局，因为要使用vue命令，安装完成后，可使用“vue”查看是否安装成功
npm install vue-cli -g
```

3. 全局安装webpack：

```
npm install webpack -g
```

4. 选择项目目录，例如：要在d盘下创建项目，cmd命令：d：
5. 创建（初始化）项目（项目名不能使用大写，例如：vue_demo)：会在d盘下创建名为vue_demo目录

```
$ vue init <template-name> <project-name>
```

init：表示我要用vue-cli来初始化项目

<template-name>：表示模板名称，vue-cli官方为我们提供了5种模板：

模板	说明
webpack	全面的webpack+vue-loader的模板，功能包括热加载，linting（语法检测）、单元检测和CSS扩展。
webpack-simple	简单webpack+vue-loader的模板，不包含其他功能，让你快速的搭建vue的开发环境。
browserify	全面的Browserify+vueify 的模板，功能包括热加载，linting、单元检测。
browserify-simple	简单Browserify+vueify的模板，不包含其他功能，让你快速的搭建vue的开发环境
simple	最简单的单页应用模板

<project-name>：标识项目名称，这个你可以根据自己的项目来起名字。

我们使用webpack模板，安装命令如下：

```
vue init webpack <project-name>
```

> vue会将依赖的webpack包一同安装到项目中的node_modules目录中
>
> 提问部分：一直按“enter”键， 选择安装的部分，只有router选择'y'，其他都按'n'

第二节 目录结构

```
├── index.html                  入口页面
├── build                       构建脚本目录
│   ├── build.js                  生产环境构建脚本
│   ├── check-versions.js         检查依赖工具版本是否适用，如nodejs、npm，若版本太低会提示。
│   ├── utils.js                  构建相关工具方法
│   ├── vue-loader.conf.js        webpack的vue-loader配置
│   ├── webpack.base.conf.js      wabpack基础配置
│   ├── webpack.dev.conf.js       wabpack开发环境配置
│   └── webpack.prod.conf.js      wabpack生产环境配置
├── config                      项目配置
│   ├── dev.env.js                开发环境变量
│   ├── index.js                  项目配置文件
│   └── prod.env.js               生产环境变量
├── src                         项目源码目录
│   ├── main.js                   入口js文件
│   ├── app.vue                   根组件
│   ├── components                公共组件目录
│   │   └── title.vue
│   ├── assets                    资源目录（会被wabpack构建）
│   │   └── images
│   │       └── logo.png
```

```
│   └── routes                  前端路由
│       └── index.js            路由js文件
├── static                      纯静态资源，不会被wabpack构建。
└── package.json                npm包配置文件
```

assets和static目录的区别

目录	处理	路径	其他
assets	会被webpack处理解析为模块依赖	只支持相对路径形式	放置会变动的文件
static	不会被Webpack处理，直接被复制到最终的打包目录（默认是dist/static）下	需要以绝对路径的形式引用/static/[filename]。	static放不会变动的文件

第三节 项目开发

1. 启动服务器

```
npm run dev
```

开启本地开发服务器，监控项目文件的变化，实时构建并自动刷新浏览器，浏览器访问:'http://localhost:8080

浏览器的console窗口中显示：

```
[HMR] Waiting for update signal from WDS...
```

> HMR：模块热替换(Hot Module Replacement)
>
> WDS：webpack dev server
>
> **[模块热替换] 等待webpack dev server的更新信号.....**

2. main.js

文件位置

```
├── src                         项目源码目录
    └── main.js                   入口js文件
```

入口js文件

```
import Vue from 'vue';
import App from './App';
import router from './router';
```

```
import vue_resource from"vue-resource";
Vue.use(vue_resource);

import ElementUI from 'element-ui' ;
Vue.use(ElementUI);

Vue.config.productionTip = false

//全局实例
new Vue({
  el: '#app',
  router,
  components: {App},
  template: '<App/>'
})
```

3. VUE模块

文件位置

```
├── src                 项目源码目录
    └── components      公共组件目录
        └── xxx.vue     vue模块
```

xxx.vue文件，是模块文件，该文件内部，分为三个部分：

```
<!--html模板----------------------------------------->
<template>

</template>

<!--javascript脚本----------------------------------->
import 名称 from '模块'; <!--引入其他模块（js文件）-->
<script>
    export default{
        name：'本模块名称',
        data:function(){
            ruturn{
                msg:"hxsd"
            }
        },
        methods:{
            run:function(){
                console.log(this.msg);
            }
        }
    }
</script>

<!--CSS样式----------------------------------------->
<style>
```

```
    p{ color：red}
</style>
```

app.vue

文件位置

```
├── src                          项目源码目录
    └── app.vue                  根组件
```

根组件，该组件通过main.js引入

```
//模板文件
<template>
  <div>
      <header>
        <div class="title">
            <img src="./assets/images/logo.jpg">
        </div>
      </header>
      <div id="app">
        <router-view/>
        <tabbar/>
      </div>
  </div>
</template>

<script>
import tabbar from '@/components/tabbar';

export default {
  name: 'App',
  components:{  tabbar }
}
</script>

<style>  ......  </style>
```

注意：

模板内的html语法、css语法、img路径等都要正确无误，否则会打包失败

4. 路由配置

文件位置

```
├── src
    └── router
        └── index.js
```

index.js 是路由模块，一个典型的路由模块的内容：

```
import Vue from 'vue';
import Router from 'vue-router';
Vue.use(Router);

//引入vue模块 （vue后缀名可以省略）
import page1 from '@/components/page1';
import page2 from '@/components/page2';
import page3 from '@/components/page3';

export default new Router({
  routes: [
    {
      path: '/',
      redirect:'/page1'
    },
    {
      path: '/p1',
      name: 'route1',
      component:page1
    },
    {
      path: '/p2',
      name: 'route2',
      component:page2
    },
    {
      path: '/p3',
      name: 'route3',
      component:page3
    }
  ]
})
```

5. 其他

@与./的区别

路径符	说明
@	webpack配置的引用路径，表示项目中的**src目录** （webpack.base.conf.js中设置）
./	html路径语法，表示当前目录

dev阶段跨域调试接口

dev开发阶段，是运行在webpack的webserver上（内存中运行），此时的开发环境，与后台不在同一个域名下，无法直接进行同域的数据请求，需要进行如下设置，进行跨域接口请求：

修改配置文件config目录中的index.js文件，编辑proxyTable：

```
├── config                        项目配置
    └── index.js                  项目配置文件
```

```
//代理
proxyTable: {
    '/api': {
        target: 'http://127.0.0.1:8090', // 接口域名:别忘了写http://
        changeOrigin: true, //是否跨域
    }
}
```

模块文件中的ajax请求：

```
onSubmit() {
    //url的 "/api" 将会被代理到跨域的 http://127.0.0.1:8090
    axios.get("/api?name=zhangsan").then(
        (res)=>{
            console.log(res.data);
        },
        ()=>{
            alert("连接失败！！");
        }
    );
}
```

第四节 打包发布

```
npm run build
```

该命令，将src中需要合并打包的文件，打包到dist目录中：

```
> node build/build.js

Hash: 95733af968436fd432b4
Version: webpack 3.11.0
Time: 22852ms
                                          Asset       Size  Chunks                    Chunk Names
          static/js/vendor.2180b39aba75dd358f0b.js.map    2.84 MB       0  [emitted]         vendor
                static/fonts/element-icons.6f0a763.ttf      11 kB          [emitted]
                  static/js/app.2e81e786d436ecbf269e.js    14.4 kB       1  [emitted]         app
            static/js/manifest.2ae2e69a05c33dfc65f8.js  857 bytes       2  [emitted]         manifest
    static/css/app.ad4aab7b8afb5e78f36ed7009a497ddb.css     186 kB       1  [emitted]         app
static/css/app.ad4aab7b8afb5e78f36ed7009a497ddb.css.map     269 kB          [emitted]
               static/js/vendor.2180b39aba75dd358f0b.js     737 kB       0  [emitted]  [big]  vendor
             static/js/app.2e81e786d436ecbf269e.js.map    37.1 kB       1  [emitted]         app
        static/js/manifest.2ae2e69a05c33dfc65f8.js.map    4.97 kB       2  [emitted]         manifest
                                       index.html  513 bytes          [emitted]
                                  static/info.json    0 bytes          [emitted]
                                  static/data.json   40 bytes          [emitted]

  Build complete.

  Tip: built files are meant to be served over an HTTP server.
  Opening index.html over file:// won't work.
```

提示：构建文件是要在HTTP服务器上环境下运行index.html，不能以 " file:// " 形式打开文件。

浏览器查看：localhost:8080/dist/index.html