一、项目创建过程

1.各种安装

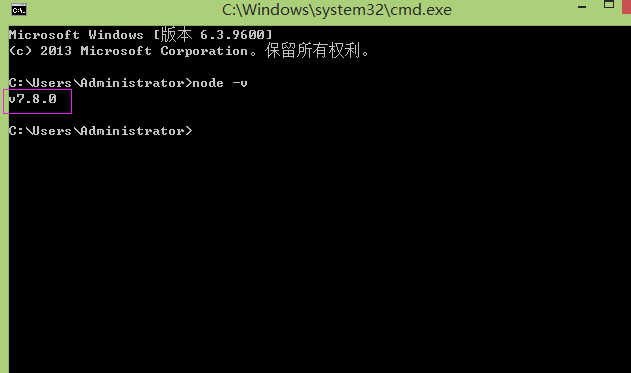
node.js环境（npm包管理器）

vue-cli 脚手架构建工具

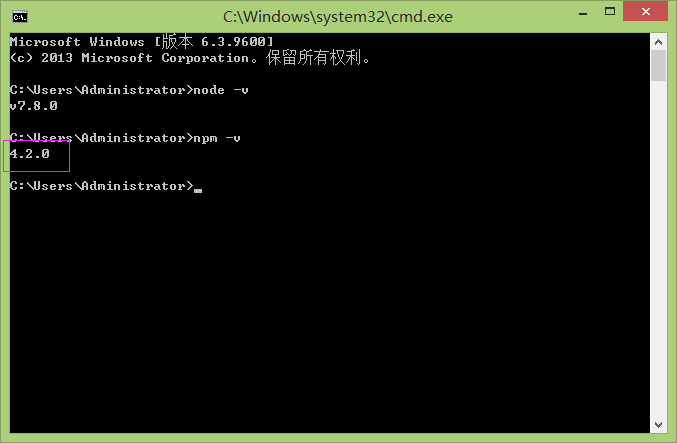
cnpm npm的淘宝镜像

1. 从node.js官网下载并安装node，安装过程很简单，一路“下一步”就可以了（傻瓜式安装）。

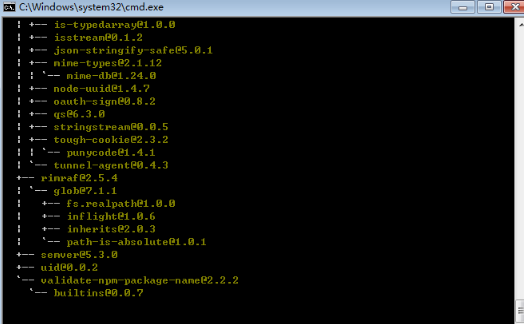
安装完成之后，打开命令行工具，输入 node -v,出现下图所示



1. npm包管理器，是集成在node中的，所以，直接输入 npm -v就会如下图所示，显示出npm的版本信息。



1. 在命令行中输入 npm install -g cnpm --registry=http://registry.npm.taobao.org 然后等待，安装完成如下图。



安装vue-cli脚手架构建工具 在命令行中运行命令 npm install -g vue-cli ,等待安装完成

1. 项目建立

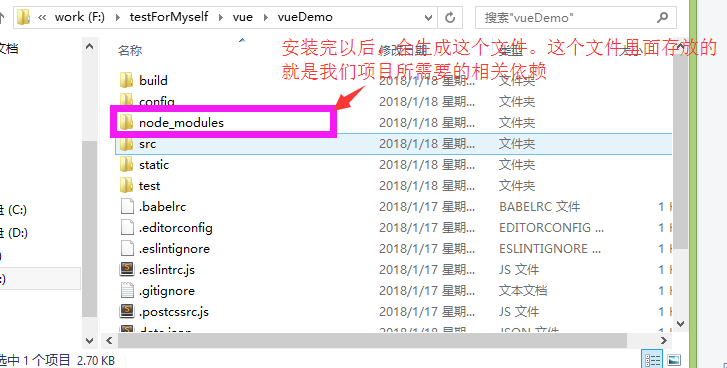
1）首先确定你要放置的文件位置，我的在F盘的prioject目录下面



在命令行中运行命令 vue init webpack myVue .这个命令是初始化一个自己的项目。运行初始化命令的时候回让用户输入几个基本的选项，如项目名称，描述，作者等信息，如果不想填直接回车默认就好。

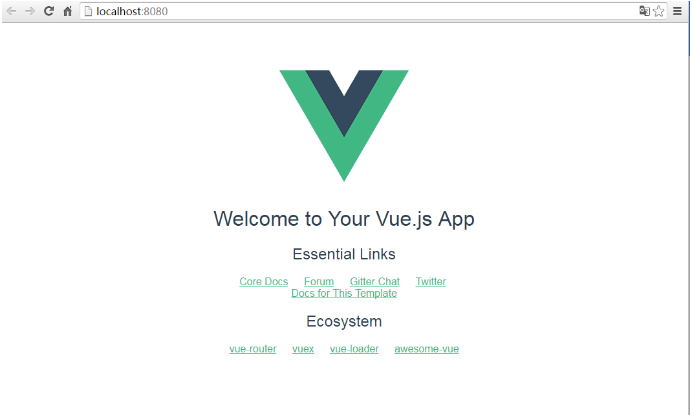
1. 安装项目中所需要的依赖





3）在项目目录中，运行命令 npm run dev ，会用热加载的方式运行我们的应用，热加载可以让我们在修改完代码后不用手动刷新浏览器就能实时看到修改后的效果。

在浏览器上面输入localhost:8080，就可以看到如下图所示的效果



1. vue-router的使用

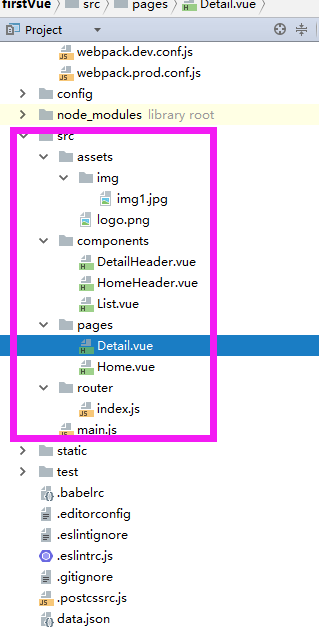
router-view 和 router-link 来控制路由

transition 控制页面之间的跳转动画

1. 更改原来的src目录结构

在 src 目录下，删除 App.vue ，增加 pages 文件夹，用来放置我们的.vue文件,

修改后的结构是这样的。



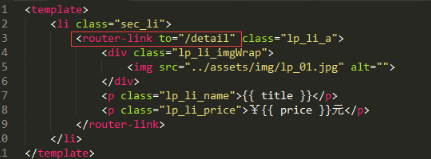
两个页面，Home和detail和其中用到的三个组件。单文件组件，由三大部分组成，template、style、script。

1. 安装vue-router 2.0。打开命令行，cd到当前项目的目录，运行 cnpm install vue-router --save 。等待安装完成即可。

1）router-view是现实路由内容的地方，即如果有多个“页面”需要切换，显示当前“页面“的地方。需要注意的是，使用vue-router控制路由则必须router-view作为容器



2）router-link有一个to属性，其属性值是目标路由，在运行项目的时候，router-link表现为a标签，to属性则表现为a标签的href属性。



修改根目录下的index.html如下：

**[html]** [view plain](http://blog.csdn.net/u013182762/article/details/53027883" \o "view plain) [copy](http://blog.csdn.net/u013182762/article/details/53027883" \o "copy)

1. <!-- index.html -->
2. <!DOCTYPE html**>**
3. **<html>**
4. **<head>**
5. **<meta** charset="utf-8"**>**
6. **<title>**first-vue**</title>**
7. **<style** type="text/css"**>**
8. \* { margin: 0; padding: 0; }
9. html,body { background: #eee; }
10. ul,li { list-style: none; }
11. a { text-decoration: none; }
12. img { vertical-align: middle; }
13. /\* 跳转页面动画 \*/
14. .slide-enter,
15. .slide\_back-enter {
16. position: absolute;
17. width: 100%;
18. }
19. .slide-leave,
20. .slide\_back-leave {
21. position: absolute;
22. width: 100%;
23. }
24. .slide-enter-active,
25. .slide\_back-enter-active {
26. transition: all 0.3s linear;
27. }
28. .slide-leave-active {
29. position: absolute;
30. transition: all 0.3s linear;
31. transform: translate(-100%);
32. }
33. .slide-enter{
34. transform: translateX(100%);
35. }
36. .slide\_back-leave-active {
37. position: absolute;
38. transition: all 0.3s linear;
39. transform: translate(100%);
40. }
41. .slide\_back-enter {
42. transform: translateX(-100%);
43. }
44. **</style>**
45. **</head>**
46. **<body>**
47. **<div** id="app"**>**
48. **<transition** :name="transitionName"**>**
49. **<router-view></router-view>**
50. **</transition>**
51. **</div>**
53. **<script** type="text/javascript"**>**
54. **</script>**
55. **</body>**
56. **</html>**

修改src目录下的main.js如下：

**[javascript]** [view plain](http://blog.csdn.net/u013182762/article/details/53027883" \o "view plain) [copy](http://blog.csdn.net/u013182762/article/details/53027883" \o "copy)

1. // main.js
3. // 导入Vue，这个是必需的，在使用Vue之前，必须先导入
4. **import** Vue from 'vue'
6. // 导入 vue-router，并使用
7. **import** VueRouter from 'vue-router'
8. Vue.use(VueRouter)
10. // 导入 pages 下的 Home.vue
11. **import** Home from './pages/Home'
12. **import** Detail from './pages/Detail'
14. // 定义路由配置
15. **const** routes = [
16. {
17. path: '/',
18. component: Home
19. },
20. {
21. path: '/detail',
22. component: Detail
23. }
24. ]
26. // 创建路由实例
27. **const** router = **new** VueRouter({
28. routes
29. })
31. // 创建 Vue 实例
32. **new** Vue({
33. el: '#app',
34. data(){
35. **return** {
36. transitionName: 'slide'
37. }
38. },
39. router, // 在vue实例配置中，用router
40. watch: {
41. // 监视路由，参数为要目标路由和当前页面的路由
42. '$route' (to, from){
43. **const** toDepth = to.path.substring(0, to.path.length-2).split('/').length
44. **const** fromDepth = from.path.substring(0, from.path.length-2).split('/').length
45. **this**.transitionName = toDepth < fromDepth ? 'slide\_back' : 'slide'
46. // 根据路由深度，来判断是该从右侧进入还是该从左侧进入
47. }
48. }
49. })

HomeHeader.vue代码

**[plain]** [view plain](http://blog.csdn.net/u013182762/article/details/53027883" \o "view plain) [copy](http://blog.csdn.net/u013182762/article/details/53027883" \o "copy)

1. <!-- HomeHeader.vue -->
2. <template>
3. <header class="header">
4. <div class="header\_inner">
5. <div class="header\_cont">主页</div>
6. </div>
7. </header>
8. </template>
10. <style>
11. .header {
12. height: 50px;
13. }
14. .header\_inner {
15. position: fixed;
16. top: 0;
17. left: 0;
18. right: 0;
19. z-index: 99;
20. max-width:800px;
21. height: 50px;
22. box-sizing: border-box;
23. margin: 0 auto;
24. border-bottom: 1px solid #aaa;
25. background-color: #fff;
26. }
27. .header\_cont {
28. text-align: center;
29. line-height: 50px;
30. width: 100%;
31. font-size: 15px;
32. overflow: hidden;
33. text-overflow: ellipsis;
34. white-space: nowrap;
35. }
36. </style>

List.vue代码

**[plain]** [view plain](http://blog.csdn.net/u013182762/article/details/53027883" \o "view plain) [copy](http://blog.csdn.net/u013182762/article/details/53027883" \o "copy)

1. <!-- List.vue -->
2. <template>
3. <li class="sec\_li">
4. <router-link to="/detail" class="lp\_li\_a">
5. <div class="lp\_li\_imgWrap">
6. <img src=".../assets/img/img1.jpg" alt="">
7. </div>
8. <p class="lp\_li\_name">{{ title }}</p>
9. <p class="lp\_li\_price">￥{{ price }}元</p>
10. </router-link>
11. </li>
12. </template>
14. <style scoped>
15. .sec\_li {
16. float: left;
17. width: 50%;
18. margin-bottom: 5px;
19. }
20. .lp\_li\_a {
21. display: block;
22. padding: 20px 0;
23. margin: 0 2px;
24. text-align: center;
25. background: #fff;
26. }
27. .lp\_li\_imgWrap {
28. padding: 10px 0;
29. }
30. .lp\_li\_imgWrap > img {
31. width: auto;
32. height: 200px;
33. }
34. .lp\_li\_name {
35. height: 50px;
36. line-height: 50px;
37. font-size: 16px;
38. color: #333;
39. }
40. .lp\_li\_price {
41. height: 40px;
42. line-height: 40px;
43. font-size: 16px;
44. color: #fb3b3b;
45. }
46. </style>
47. <script>
48. export default {
49. props: ['price', 'title']
50. }
51. </script>

DetailHeader.vue代码

**[plain]** [view plain](http://blog.csdn.net/u013182762/article/details/53027883" \o "view plain) [copy](http://blog.csdn.net/u013182762/article/details/53027883" \o "copy)

1. <!-- DetailHeader.vue -->
2. <template>
3. <header class="header">
4. <div class="header\_inner">
5. <div
6. id="header\_btn\_nav"
7. class="header\_btn header\_btn\_back"
8. v-on:click="goBack"
9. >返回
10. </div>
11. <div class="header\_cont flex">详情</div>
12. </div>
13. </header>
14. </template>
15. <style scoped>
16. .header {
17. height: 50px;
18. }
19. .header\_inner {
20. position: fixed;
21. top: 0;
22. left: 0;
23. right: 0;
24. z-index: 99;
25. max-width: 800px;
26. height: 50px;
27. margin: 0 auto;
28. border-bottom: 1px solid #eee;
29. background-color: #fff;
30. }
31. .header\_btn {
32. width: 50px;
33. height: 100%;
34. background-repeat: no-repeat;
35. float: left;
36. margin-left: 20px;
37. }
38. .header\_btn\_back {
39. line-height: 50px;
40. }
41. .header\_cont {
42. text-align: center;
43. float: right;
44. width: 100px;
45. padding: 0 20px;
46. line-height: 50px;
47. font-size: 15px;
48. overflow: hidden;
49. text-overflow: ellipsis;
50. white-space: nowrap;
51. }
52. .header\_btn:active {
53. opacity: 0.7;
54. }
55. </style>
56. <script>
57. export default {
58. methods: {
59. goBack(){
60. window.history.back();
61. }
62. }
63. }
64. </script>

Home.vue代码

**[plain]** [view plain](http://blog.csdn.net/u013182762/article/details/53027883" \o "view plain) [copy](http://blog.csdn.net/u013182762/article/details/53027883" \o "copy)

1. <!-- Home.vue -->
2. <template>
3. <div class="container">
4. <!-- 由于html不区分大小写，所以js中驼峰命名方式在html中要改成用短横线连接的形式 -->
5. <home-header></home-header>
6. <div class="content">
7. <ul class="cont\_ul">
8. <list
9. v-for="item in items"
10. :price="item.price"
11. :title="item.title">
12. </list>
13. </ul>
14. </div>
15. </div>
16. </template>
17. <style>
18. .container {
19. max-width: 800px;
20. margin: 0 auto;
21. overflow-x: hidden;
22. }
23. .cont\_ul {
24. padding-top: 10px;
25. }
26. .cont\_ul:after {
27. content: "";
28. display: block;
29. width: 0;
30. height: 0;
31. clear: both;
32. }
33. </style><script>
34. // 导入要用到的子组件
35. import HomeHeader from '../components/HomeHeader'
36. import List from '../components/List'
38. export default {
39. data () {
40. return {
41. items: [
42. { price: "129.00", title: "大学" },
43. { price: "256.00", title: "中庸" },
44. { price: "399.00", title: "论语" },
45. { price: "998.00", title: "孟子" },
46. { price: "99.00", title: "道德经" },
47. { price: "89.00", title: "老子" },
48. { price: "188.00", title: "金刚经" },
49. { price: "209.00", title: "易筋经" },
50. ]
51. }
52. },
53. // 在components字段中，包含导入的子组件
54. components: {
55. HomeHeader,
56. List
57. }
58. }
59. </script>

Detail.vue代码

**[plain]** [view plain](http://blog.csdn.net/u013182762/article/details/53027883" \o "view plain) [copy](http://blog.csdn.net/u013182762/article/details/53027883" \o "copy)

1. <!-- Detail.vue -->
2. <template>
3. <div class="detail">
4. <detail-header></detail-header>
5. <div class="content">
6. <img src="../assets/img/img1.jpg" alt="">
7. <p>崇贤馆始置唐代太宗朝。1999年，李克先生及志同道合者复兴其宗旨。以积累、传播中华优秀传统文化，提供全新国学体验馆为宏愿。</p>
8. <p>其间，在季羡林、冯其庸等国学大师及著名文史学家傅璇琮、毛佩琦先生指导下，耕注先贤原典，以宣纸线装精品形式呈奉世人。作为一家国学传播机构，崇贤馆始终致力于中华传统文化的传承和推广，以古籍线装宣纸书的形式，对浩繁的史海巨著进行经典复刻。不仅如此，崇贤馆还延请了傅璇琮、毛佩奇等诸位在国学界内享有盛誉的专家和学者担纲学术顾问，以精益求精的治学态度面对每一部崇贤馆的作品，使之成为学术史中无尚的精品。</p>
9. </div>
10. </div>
11. </template>
12. <style scoped>
13. .detail {
14. font-size: 12px;
15. }
16. .content {
17. max-width: 800px;
18. background-color: #fff;
19. margin: 0 auto;
20. overflow: hidden;
21. margin-top: 10px;
22. min-height: 500px;
23. }
24. img {
25. float: left;
26. width:300px;
27. }
28. p {
29. font-size: 14px;
30. float: right;
31. overflow: auto;
32. width: 480px;
33. display: inline-block;
34. vertical-align: middle;
35. text-indent: 25px;
36. line-height: 45px;
37. padding: 5px 10px;
38. }
39. </style><script>
40. import DetailHeader from '../components/DetailHeader'
42. export default {
43. components: {
44. DetailHeader
45. }
46. }
47. </script>
48. 模拟服务器返回数据
49. 在项目根目录下，新建一个data.json。这个文件里的内容就是我们要模拟的服务端返回的数据。
50. {
51. "books": [
52. { "price": "111.00", "title": "语文", "img": "http://imgsrc.baidu.com/forum/w%3D580/sign=688d19b6422309f7e76fad1a420f0c39/0e143912b31bb051dfdebf0a327adab44bede0f9.jpg" },
53. { "price": "123.00", "title": "数学", "img": "http://shopimg.kongfz.com.cn/20130826/1746508/1746508YTGtm0\_b.jpg"},
54. { "price": "113.00", "title": "英语", "img": "http://www.shiyi910.com/UploadFiles/2014-01/170/2014011322414554305.jpg"},
55. { "price": "112.00", "title": "物理", "img": "http://f.hiphotos.baidu.com/zhidao/wh%3D450%2C600/sign=b62ac53d4b36acaf59b59ef849e9a126/f603918fa0ec08fa2cf895a359ee3d6d55fbda3e.jpg"}
56. ]
57. }

2）VUE开发请求本地数据的配置，早期的vue-lic下面有dev-server.js和dev-client.js两文件，请求本地数据在dev-server.js里配置，最新的vue-webpack-template 中已经去掉了dev-server.js和dev-client.js 改用webpack.dev.conf.js代替，所以 配置本地访问在webpack.dev.conf.js里配置即可。

1）、旧版dev-server.js配置本地数据访问：

在const app = express()后添加如下代码：

var appData = require('../data.json')

var books = appData.books

var apiRoutes = express.Router()

apiRoutes.get('/book', function(req, res){

res.json({

data: books

})

})

app.use('/api', apiRoutes)

2）新版在const portfinder = require('portfinder')后添加如下

首先

const express = require('express')

const app = express()

const appData = require('../data.json')//加载本地数据文件

const books = appData.books

const apiRoutes = express.Router()

app.use('/api', apiRoutes)

/然后找到devServer,在里面添加

before(app) {

app.get('/api/book', (req, res) => {

res.json({

errno: 0,

data: books

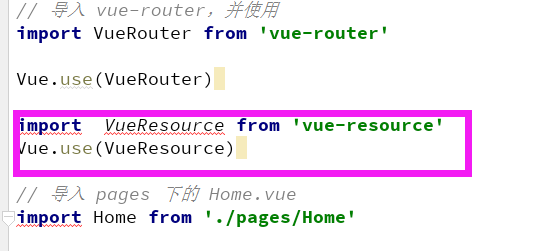
})//接口返回json数据，上面配置的数据seller就赋值给data请求后调用

})

}, 有的同学说报错，不能识别before,注意看下你的配置。

1. 打开命令行，cd到项目目录，运行 npm install vue-resource --save或者cnpm install vue-resource --save,等待安装完成即可

在main.js里导入并使用vue-resource



1. 修改Home.vue和List.vue来接收服务端返回的数据

**[html]** [view plain](http://blog.csdn.net/u013182762/article/details/53488870" \o "view plain) [copy](http://blog.csdn.net/u013182762/article/details/53488870" \o "copy)

1. <!-- Home.vue -->
2. **<template>**
3. **<div** class="container"**>**
4. <!-- 由于html不区分大小写，所以js中驼峰命名方式在html中要改成用短横线连接的形式 -->
5. **<home-header></home-header>**
6. **<div** class="content"**>**
7. **<ul** class="cont\_ul"**>**
8. ***<list***
9. ***v-for="item in items"***
10. ***:price="item.price"***
11. ***:title="item.title"***
12. ***:img="item.img">***
13. <!-- 通过v-bind（简写为“:”）绑定props来从父组件给子组件传值 -->
14. **</list>**
15. **</ul>**
16. **</div>**
17. **</div>**
18. **</template>**
19. **<style>**
20. .container {
21. max-width: 640px;
22. margin: 0 auto;
23. overflow-x: hidden;
24. }
25. .cont\_ul {
26. padding-top: 0.05rem;
27. margin: 0 -0.12rem;
28. }
29. .cont\_ul:after {
30. content: "";
31. display: block;
32. width: 0;
33. height: 0;
34. clear: both;
35. }
36. **</style>**
37. **<script>**
38. // 导入要用到的子组件
39. import HomeHeader from '../components/HomeHeader'
40. import List from '../components/List'
41. export default {
42. data () {
43. return {
44. ***items: []***
45. }
46. },
47. // 在components字段中，包含导入的子组件
48. components: {
49. HomeHeader,
50. List
51. },
52. // 在组件创建完成时，执行的钩子函数
53. ***created (){***
54. ***// 在main.js里导入并使用vue-resource之后，就可以通过this.$http来使用vue-resource了，这里我们用到了get方法***
55. ***this.$http.get('/api/books').then((data) => {***
56. ***// 由于请求成功返回的是Promise对象，我们要从data.body.data拿到books数组***
57. ***this.items = data.body.data;***
58. ***})***
59. ***}***
60. }
61. **</script>**
62. <!-- List.vue -->
63. **<template>**
64. **<li** class="sec\_li"**>**
65. **<router-link** to="/detail" class="lp\_li\_a"**>**
66. **<div** class="lp\_li\_imgWrap"**>**
67. ***<img :src="img" alt="">***
68. **</div>**
69. ***<p class="lp\_li\_name">{{ title }}</p>***
70. ***<p class="lp\_li\_price">￥{{ price }}元</p>***
71. **</router-link>**
72. **</li>**
73. **</template>**
75. **<style** scoped**>**
76. .sec\_li {
77. float: left;
78. width: 50%;
79. margin-bottom: 0.1rem;
80. }
81. .lp\_li\_a {
82. display: block;
83. padding: 0.3rem 0;
84. margin: 0 0.05rem;
85. text-align: center;
86. background: #fff;
87. }
88. .lp\_li\_imgWrap {
89. padding: 0.24rem 0;
90. }
91. .lp\_li\_imgWrap **>** img {
92. width: auto;
93. height: 2.3rem;
94. }
95. .lp\_li\_name {
96. height: 0.5rem;
97. line-height: 0.5rem;
98. font-size: 16px;
99. color: #333;
100. }
101. .lp\_li\_price {
102. height: 0.5rem;
103. line-height: 0.5rem;
104. font-size: 16px;
105. color: #fb3b3b;
106. }
107. **</style>**
109. **<script>**
110. export default {
111. // 接收父组件传过来的以下几个属性
112. ***props: ['price', 'title', 'img'],***
113. }
114. **</script>**

然后，再回到浏览器。