**使用Vue快速开发单页应用**

原创 2016年09月03日 17:09:41

http://blog.csdn.net/u013063153/article/details/52424224

本文所涉及代码全在[vue-cnode](https://github.com/wszgxa/vue-cnode)

单页应用，即在一个页面集成系统中所有功能，整个应用只有一个页面。因为路由的控制在前端，单页面应用在页面切换时比传统页面更快，从而在前端体验更好。

将逻辑从后端转移到前端，提升了性能减少了页面加载时间，前后逻辑更扁平。但是当页面复杂度变高时，你会发现，数据处理，UI交互将变得难以维护，所以应运而生，出现了很多MV框架和类库。Vue就是其中之一，个人觉得（非喜勿喷）Vue类库相对于其他MV框架上整体的api更为简洁，提供的api很平衡，解决了问题的同时，没有增加复杂度。另外个人觉得vue在大型应用，开发中使用vue-loader将组件分成template,style,script结构更为清晰。

本文以及后面相应文章，主要是vue相关技术栈来快速的实现单页应用开发。系列文章将以一个实际项目进行讲解，项目的github地址为：

[vue-cnode demo](https://github.com/wszgxa/vue-cnode)

这是一个以[cnodejs.org](http://hiluluke.cn/2016/08/04/vue-spa/cnodejs.org)提供的api来开发的单页，主要使用的modules有vue、vue-router、vuex、vue-resource。为了快速开发，我们还使用了vue-cli脚手架工具，下文会做介绍。

vue-cli

自从node的兴起，前端项目中就开始出现各种预处理工具，当我们开始一个新项目时，我们都会先编写一些预处理文件，和构建项目目录。

而vue-cli就是为了做这方面工作的，生成一套提前定义好的构建文件，和相应的文件。

vue-cli有5个对应的项目结构。我们使用的是[vue-webpack-boilerplate](https://github.com/vuejs-templates/webpack)。

|  |  |
| --- | --- |
|  | $ npm install -g vue-cli  $ vue init webpack my-project  $ cd my-project  $ npm install  $ npm run dev |

执行上面命令后，我们将生成下面的文件结构，并开一个服务，你可以打开[http://localhost:8080看看。](http://localhost:8080%E7%9C%8B%E7%9C%8B%E3%80%82/)

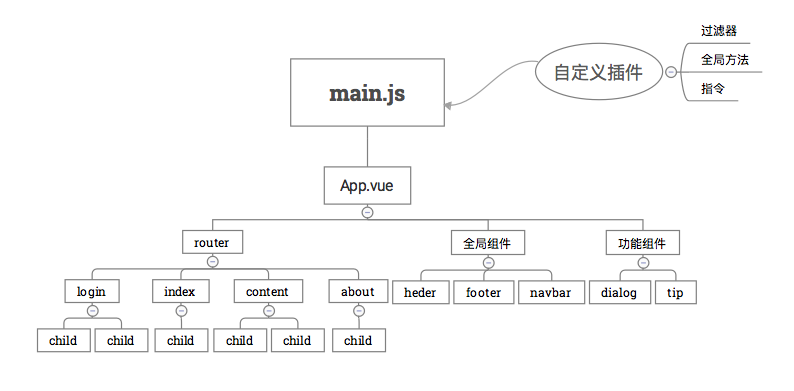


具体的使用建议看[文档](http://vuejs-templates.github.io/webpack/)。

项目结构

如果你之前就了解vue和vue-router，可以先看这部分。如果你了解vue不了解vue-router,可以先看这篇文章[vue-router](http://hiluluke.cn/2016/08/05/vue-router/)。如果你连vue都不是很理解我建议，抽5个小时左右把[文档教程](http://cn.vuejs.org/guide/)过一遍。

你可以看到项目根目录下面有一个html，仅有的一个html。



上图的结构是我自己琢磨的，主要是结合vue-router、vuex两使用方法来考虑的。另外对于组件的复用，将一些功能组件和全局组件都放在根部，通过vuex来控制组件属性实现一些功能。

下面我就结构由上至下的介绍。

main.js

main.js 是我们的入口文件，主要作用是初始化vue实例并使用需要的插件。

|  |  |
| --- | --- |
|  | import Vue from 'vue'  import App from './App'  import VueRouter from 'vue-router'  import VueResource from 'vue-resource'  import filter from './filter'  import store from './vuex/store'  import { sync } from 'vuex-router-sync'  import { configRouter } from './config\_router'  import resourceGlobalSet from './resource\_set'    Vue.use(VueResource)  Vue.use(VueRouter)  // 初始化自定义过滤器  Vue.use(filter)    const router = new VueRouter({  history: true,  saveScrollPosition: true  })  configRouter(router)  Vue.http.options.emulateJSON = true  Vue.http.interceptors.push(resourceGlobalSet) // ajax 拦截    sync(store, router)  router.start(App, 'app') |

就如同上面所示，主要是使用和配置相应插件，并初始化一个vue，上面的初始化在router.start(App, 'app')，是以App.vue为组要组件，并以html中的为挂载替换点。

APP.vue

App.vue是我们的主组件，所有页面都是在App.vue下进行切换的。其实你也可以理解为所有的路由也是App.vue的子组件。所以我将router标示为App.vue的子组件。

下面是App.vue的template

|  |  |
| --- | --- |
|  | <template>  <div id="app">  <cn-header></cn-header>  <sidebar></sidebar>  <router-view></router-view>  <tip></tip>  <loading></loading>  </div>  </template> |

你可以看到route-view和其它全局功能组件，全局组件在一个层级。

另外由于APP.vue在所有页面都有，我们将会在APP.vue上面写一些初始化全局方法。

router

router 是具体的业务组件，比如index,login,content等组件都是具体业务相关的。下面就是再和业务相关的组件。

全局组件

全局组件是页面共用的部分，比如header，footer，navbar，你可能在想如果我有一些header是独特的怎么办，这种情况下可以通过路由做判断，渲染不同的html，如果判断条件不是路由，也可以在vuex写一个store记录组件的state。

功能组件

功能组件是比如dialog,tip等组件，使用来与用户交互的。

通常情况下，功能组件是各个组件都需要的一些组件。在一个页面里如果有两个组件，两个组件都同时引了一个tip组件作为子组件是纯在的。为了避免这种情况，我们将功能组件提到App.vue然后通过vuex进行组件交互，从而就讲一个功能组件变成了全局方法。

自定义插件

vue还能自己写插件。对于一些公用的方法和逻辑，我们可以提出来写在插件里面。

小节

可以看到，我们项目整体结构非常清晰。入口加载初始化，主组件挂载路由全局控制，然后全局组件功能组件借助vuex进行数据控制。