前端架构设计说明书

版本：1.0

编制人：Garmin

目录

[前端框架 1](#_Toc535506187)

[前后端分离 1](#_Toc535506188)

[技术说明 4](#_Toc535506187)

[工程代码结构 4](#_Toc535506188)

[安装、启动 6](#_Toc535506188)

[技术简介 6](#_Toc535506188)

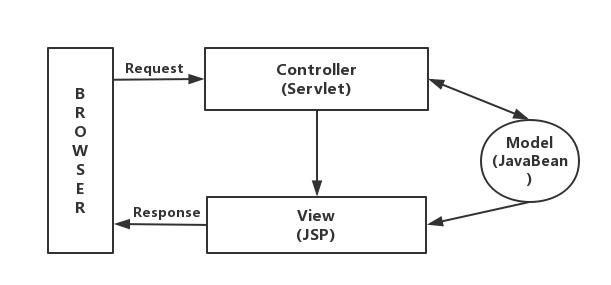
1. [VUE 6](#_Toc535506189)
2. [VUEX 7](#_Toc535506189)
3. [ELEMENT-UI 8](#_Toc535506189)
4. [MOCK 8](#_Toc535506189)
5. [更多前端技术 9](#_Toc535506189)

[前端路由规范与页面寻址 9](#_Toc535506188)

# 前端框架

* 1. **前后端分离**

早期B/S架构的应用使用jsp+servlet的方式



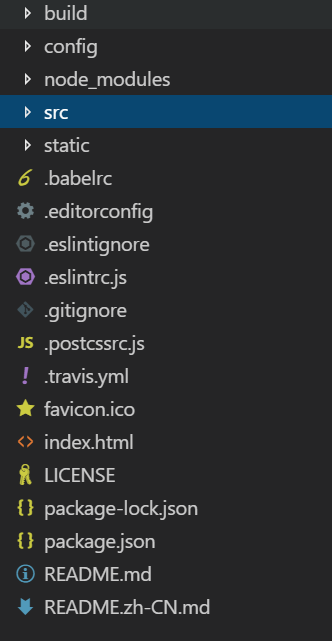
这样的缺点是显然易见的：

1. 前端在开发过程中严重依赖后端，在后端没有完成的情况下，前端根本无法干
2. 前端无法单独调试，降低开发效率
3. 前端不可避免会遇到后台代码，即使把后端业务代码单列出来，前端仍然需要管理controller，也就是说前端仍然需要懂后端框架的代码。这明显是不符合技术发展趋势的。

所以为了解决以上问题，前后端分离应运而生。

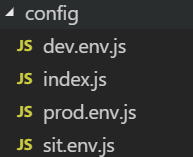
# 技术说明

* 1. **工程代码结构**



1.1 Config

配置文件



Index.js:项目的核心配置项，基本不需要修改

\*.env.js:不同环境的个性化配置，\*指不同的环境

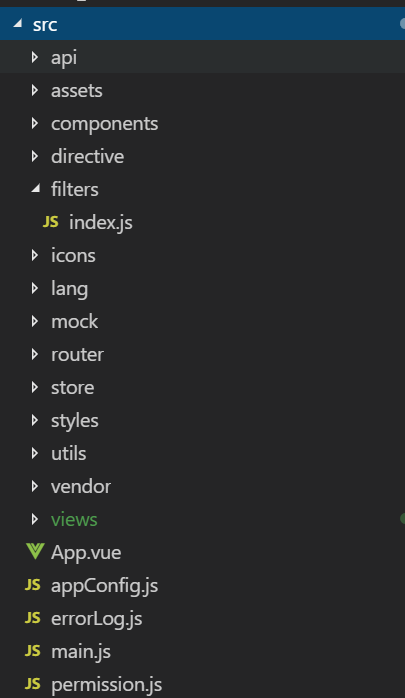
1.2 package.json,

存放了项目依赖。

1.3 Package-lock.json,

项目install后会产生的文件，如需要重新install请先把这个文件删掉。

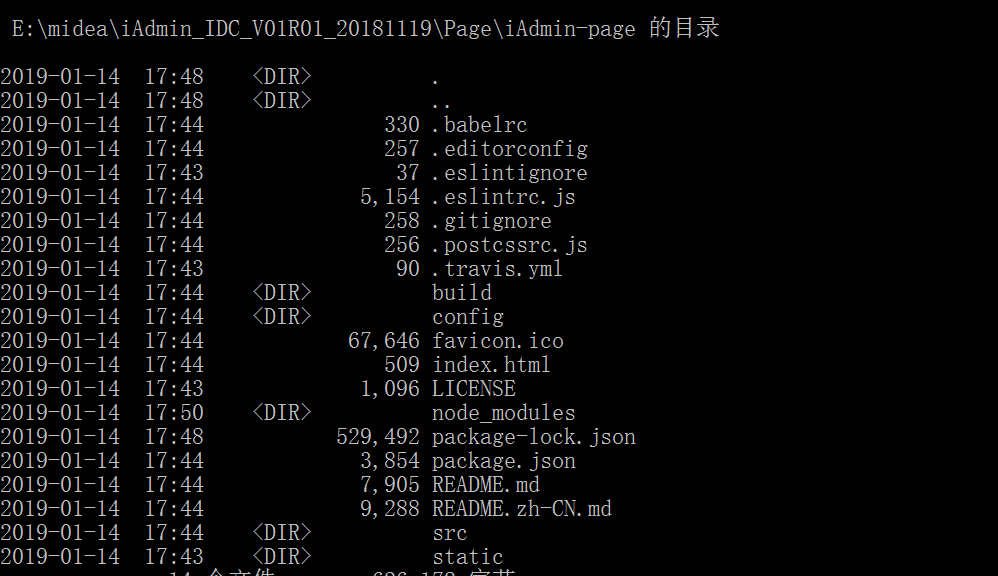
1.4 src



1. Api:存放后端接口，不建议使用
2. Components:存放自己写的控件，这个和官方文档有出入以本文档为准
3. Filters: 前端格式化使用到的函数
4. Lang:多语言
5. Mock:定义了mock请求的返回格式
6. Router:定义了框架基础路由【如：登录页、首页、404等】、
7. Store:vuex管理的状态
8. Utils:公用的工具【这个需要和Components区分，Utils是没有页面的】
9. View：业务页面
10. App.vue框架入口页面【请勿修改】
11. appConfig.js公用配置，如token、profile、appName
12. main.js入口页面对应的框架js，基本不要修改
13. permission.js权限有关的代码
    1. **安装运行**

2.1 请先安装node.js和git

2.2然后在dos打开项目文件夹【注意图片的文件，这个才是项目文件夹】



2.3 执行命令 npm -i 安装

2.4 安装成功后，执行npm run dev启动项目【其中端口在config文件夹的配置文件修改。dev是环境名，需要和配置文件的对应】

* 1. **技术简介**

**注意：这里的代码是尚未模块化的基础代码，框架代码很多是根据ES2015进行模块化，所以代码看起来会有所区别**

**3.1 vue**

Vue和前期的js框架最明显的区别就是变量的双向绑定。例如一个输入框绑定了一个js变量，当输入东西的时候就会自动把输入框即时的东西set到这个变量。

3.1.1 基本用法

let v = new Vue({

//代表这个vue对象的作用域（以下的所有东西都只在这个对象生效）

//#id和jq的选择器一致

el:’#id,’

//变量

data:{

var1 : ‘value1’

},

//格式化或转义方法

filters:{

date: function(val){

return val+’12344’

}

},

//本地方法(可以写事件响应)

methods:{

click: function(val){

return val+’12344’

}

},

//生命周期事件响应，mounted是其中一个生命周期事件

mounted:function(){

}

})

3.1.2 变量使用

如果想在js在使用以上定义的变量、方法可以直接使用this使用。如使用以上定义的var1, 可以直接this.var1使用。

页面显示vue对象的变量这样{{var1}}, 如果使用格式化可以这样{{var1 | date }}

3.1.3 更多标签

Vue还有很多用法如v-if,v-model,v-for,请到vue官网浏览。

**3.2 vuex**

Vuex就是一个全局的状态机，可以简单理解为一个常量类。在代码层面就是在store存放着很多state。如果多个控件需要使用到同样的变量【如父子控件】，就可以使用vuex，但是如果某些状态只是在单个控件内生效就使用vuex那就是增加代码的复杂度了。

3.2.1 基础用法

const store = new Vuex.Store({  
  state: {  
 //这里就是定义状态，就是常量

    count: 0  
  },  
  mutations: {  
    increment (state) {  
      state.count++  
    }

},

getters:{

doneTodos: function( state) {

return state.todos

}

}

}

3.2.2 mutations

Vuex不推荐直接请求状态来写state，需要在mutations里定义方法去写state，类似于java里边的setter。当然vuex支持通过传入变量的方式改变state，这个叫做载荷，这里不详细说明。

Vuex也提供的调用mutations的方式，如store.commit('increment')。

3.2.3 getters

Vuex不推荐直接请求状态来读state，需要在getters里定义方法去读state，类似于java里边的getter,当然大家也可以在里边对变量做处理了再返回。

**3.3 element-ui**

Element-UI封装了一些常用的控件和布局，详细请到elementUI官网查看。

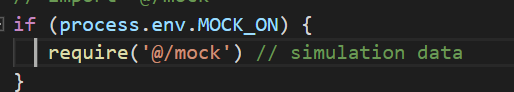
**3.4 mock**

Mock提供了一个方案，拦截了后端请求并指定返回，可以在后端还没开发完接口

的时候根据协议写前端。

3.4.1 使用

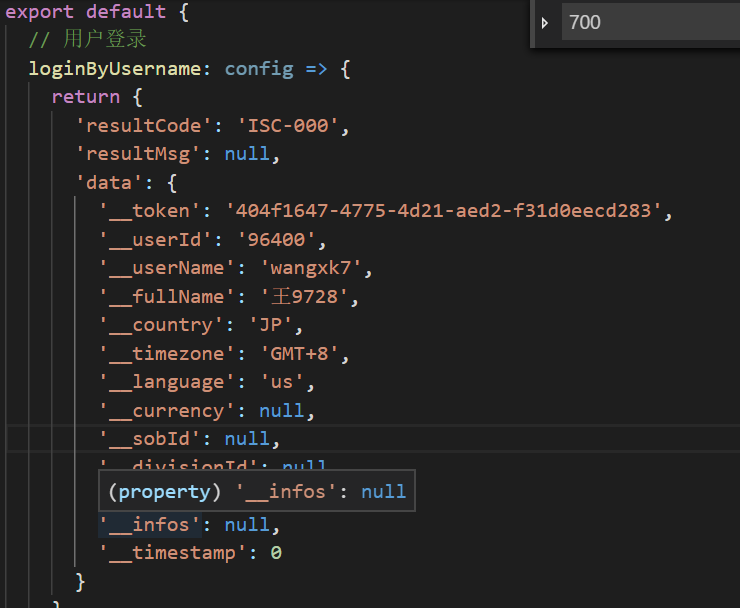
A． 在main.js开启mock



B．在mock/index.js对某个请求打开mock拦截



C．模拟请求返回



**3.5 前端路由规范和页面寻址**

为了方便维护，views文件夹里边的路径与url和前端层级一致。

如/a/b/c，对应的vue就是/views/a/b/c.vue，前端层级也是a->b->c