快手内部文档请勿外传



手动搭建vue+node单页面（一） <https://segmentfault.com/a/1190000014368466>

手动搭建vue+node单页面（二） <https://segmentfault.com/a/1190000014384817?utm_source=channel-newest>

项目地址 <https://github.com/liubingyang/vue-spa>

1 跨域解决方法

# 1.1 为什么会出现跨域问题

可以说Web是构建在同源策略基础之上的，浏览器只是针对同源策略的一种实现。同源策略会阻止一个域的javascript脚本和另外一个域的内容进行交互。所谓同源（即指在同一个域）就是两个页面具有相同的协议（protocol），主机（host）和端口号（port）

# 1.2 什么是跨域

当一个请求url的协议、域名、端口三者之间任意一个与当前页面url不同即为跨域

# 1.3 非同源限制

无法读取非同源网页的 Cookie、LocalStorage 和 IndexedDB

无法接触非同源网页的 DOM

无法向非同源地址发送 AJAX 请求

# 1.4 跨域解决方法

## 1.4.1 设置document.domain解决无法读取非同源网页的 Cookie问题

因为浏览器是通过document.domain属性来检查两个页面是否同源，因此只要通过设置相同的document.domain，两个页面就可以共享Cookie（此方案仅限主域相同，子域不同的跨域应用场景。）

|  |
| --- |
| // 两个页面都设置  document.domain = 'test.com'; |

## 1.4.2 跨文档通信 API：window.postMessage()

调用postMessage方法实现父窗口http://test1.com向子窗口http://test2.com发消息（子窗口同样可以通过该方法发送消息给父窗口）

它可用于解决以下方面的问题：

* 页面和其打开的新窗口的数据传递
* 多窗口之间消息传递
* 页面与嵌套的iframe消息传递
* 上面三个场景的跨域数据传递

|  |
| --- |
| // 父窗口打开一个子窗口  var openWindow = window.open('http://test2.com', 'title');  // 父窗口向子窗口发消息(第一个参数代表发送的内容，第二个参数代表接收消息窗口的url)  openWindow.postMessage('Nice to meet you!', 'http://test2.com'); |

调用message事件，监听对方发送的消息

|  |
| --- |
| // 监听 message 消息  window.addEventListener('message', function (e) {  console.log(e.source); // e.source 发送消息的窗口  console.log(e.origin); // e.origin 消息发向的网址  console.log(e.data); // e.data 发送的消息  },false); |

## 1.4.3 JSONP

JSONP 是服务器与客户端跨源通信的常用方法。最大特点就是简单适用，兼容性好（兼容低版本IE），缺点是只支持get请求，不支持post请求。

核心思想：网页通过添加一个<script>元素，向服务器请求 JSON 数据，服务器收到请求后，将数据放在一个指定名字的回调函数的参数位置传回来。

### ①原生实现：

|  |
| --- |
| <script src="http://test.com/data.php?callback=dosomething"></script>  // 向服务器test.com发出请求，该请求的查询字符串有一个callback参数，用来指定回调函数的名字  // 处理服务器返回回调函数的数据  <script type="text/javascript">  function dosomething(res){  // 处理获得的数据  console.log(res.data)  }  </script> |

### ② jQuery ajax：

|  |
| --- |
| $.ajax({  url: 'http://www.test.com:8080/login',  type: 'get',  dataType: 'jsonp', // 请求方式为jsonp  jsonpCallback: "handleCallback", // 自定义回调函数名  data: {}  }); |

### ③ Vue.js

|  |
| --- |
| this.$http.jsonp('http://www.domain2.com:8080/login', {  params: {},  jsonp: 'handleCallback'  }).then((res) => {  console.log(res);  }) |

## 1.4.4 CORS

CORS 是跨域资源分享（Cross-Origin Resource Sharing）的缩写。它是 W3C 标准，属于跨源 AJAX 请求的根本解决方法。

1、普通跨域请求：只需服务器端设置Access-Control-Allow-Origin

2、带cookie跨域请求：前后端都需要进行设置

### 前端设置

根据xhr.withCredentials字段判断是否带有cookie

①原生ajax

|  |
| --- |
| var xhr = new XMLHttpRequest(); // IE8/9需用window.XDomainRequest兼容  // 前端设置是否带cookie  xhr.withCredentials = true;  xhr.open('post', 'http://www.domain2.com:8080/login', true);  xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  xhr.send('user=admin');  xhr.onreadystatechange = function() {  if (xhr.readyState == 4 && xhr.status == 200) {  alert(xhr.responseText);  }  }; |

② jQuery ajax

|  |
| --- |
| $.ajax({  url: 'http://www.test.com:8080/login',  type: 'get',  data: {},  xhrFields: {  withCredentials: true // 前端设置是否带cookie  },  crossDomain: true, // 会让请求头中包含跨域的额外信息，但不会含cookie  }); |

③vue-resource

|  |
| --- |
| Vue.http.options.credentials = true |

④ axios

|  |
| --- |
| axios.defaults.withCredentials = true |

### 服务端设置

服务器端对于CORS的支持，主要是通过设置Access-Control-Allow-Origin来进行的。如果浏览器检测到相应的设置，就可以允许Ajax进行跨域的访问。

① Java后台

|  |
| --- |
| /\*  \* 导入包：import javax.servlet.http.HttpServletResponse;  \* 接口参数中定义：HttpServletResponse response  \*/  // 允许跨域访问的域名：若有端口需写全（协议+域名+端口），若没有端口末尾不用加'/'  response.setHeader("Access-Control-Allow-Origin", "http://www.domain1.com");  // 允许前端带认证cookie：启用此项后，上面的域名不能为'\*'，必须指定具体的域名，否则浏览器会提示  response.setHeader("Access-Control-Allow-Credentials", "true");  // 提示OPTIONS预检时，后端需要设置的两个常用自定义头  response.setHeader("Access-Control-Allow-Headers", "Content-Type,X-Requested-With"); |

② Nodejs后台

|  |
| --- |
| var http = require('http');  var server = http.createServer();  var qs = require('querystring');  server.on('request', function(req, res) {  var postData = '';  // 数据块接收中  req.addListener('data', function(chunk) {  postData += chunk;  });  // 数据接收完毕  req.addListener('end', function() {  postData = qs.parse(postData);  // 跨域后台设置  res.writeHead(200, {  'Access-Control-Allow-Credentials': 'true', // 后端允许发送Cookie  'Access-Control-Allow-Origin': 'http://www.domain1.com', // 允许访问的域（协议+域名+端口）  /\*  \* 此处设置的cookie还是domain2的而非domain1，因为后端也不能跨域写cookie(nginx反向代理可以实现)，  \* 但只要domain2中写入一次cookie认证，后面的跨域接口都能从domain2中获取cookie，从而实现所有的接口都能跨域访问  \*/  'Set-Cookie': 'l=a123456;Path=/;Domain=www.domain2.com;HttpOnly' // HttpOnly的作用是让js无法读取cookie  });  res.write(JSON.stringify(postData));  res.end();  });  });  server.listen('8080');  console.log('Server is running at port 8080...'); |

2 项目结构

# 2.1 项目目录

|  |
| --- |
| .  ├── README.md  ├── bitbug\_favicon.ico  ├── build  │ └── webpack.config.js  ├── index.html  ├── node\_modules  ├── package-lock.json  ├── package.json  ├── server.js  ├── service  │ ├── get.js  │ ├── juejinResources.js  │ └── search.js  └── src  ├── App.vue  ├── common  │ ├── base.css  │ └── reset.css  ├── main.js  ├── router  │ └── index.js  ├── store  │ └── index.js  └── views  ├── browserInfo.vue  ├── forum.vue  ├── home.vue  └── juejinResources.vue |

test下创建src文件夹（前端所有代码）；

build文件夹(打包相关的webpack配置文件)；

service文件夹(存放后端代码)；

data文件夹(存放json数据)；

server.js(node服务)；

在src目录中创建

views(存放所有.vue组件)

router(存放前端路由配置),

store(存放vuex配置),

man.js(单页面入口文件),

app.vue（单页面首页组件）,

index.html作为载体

# 2.2 server.js

## 主要功能

## 示例代码

|  |
| --- |
| //server.js  const webpack=require('webpack')  const webpackConfig=require('./build/webpack.config')  webpack(webpackConfig,function(err,stats){  if(err) throw err  //输出打包信息（不用在意这些细节）  process.stdout.write(stats.toString({  colors: true,  modules: false,  children: false,  chunks: false,  chunkModules: false  }) + '\n\n')  }) |

# 2.2 main.js

## 主要功能

入口文件，创建Vue应用实例

## 示例代码

|  |
| --- |
| //main.js  import Vue from 'vue'  import App from './app'  import router from "./router"//默认加载index文件  new Vue({  el: '#app',  router,//注册到vue实例  render: h => h(App)  }) |

# 2.3 app.vue

## 主要功能

单页面主题

## 示例代码

|  |
| --- |
| //app.vue  <template>  <div>  {{msg}}  </div>  </template>  <script>  export default{  data(){  return {  msg:'this is my app!'  }  }  }  </script>  <style lang='less'>  </style> |

# 2.3 index.js

## 主要功能

## 示例代码

|  |
| --- |
| //router/index.js  import Vue from 'vue'  import Router from 'vue-router'  import Home from '../views/home'  Vue.use(Router)  export default new Router({  routes: [{  path: '/',  name: 'home',  component: Home  }]  }) |

# 2.4 build/webpack.config.js

## 主要功能

## 示例代码

|  |
| --- |
| //webpack.config.js  const path = require('path')  module.exports = {  mode:'development',//webpack4新加的，不写会报黄  entry: {  main: path.resolve(\_\_dirname, '../src/main.js')  },  output: {  path: path.resolve(\_\_dirname, '../dist'),  publicPath: 'http://localhost:3000/',  filename: '[name].js'  },  module: {  rules: [{  test: /\.vue$/,  loader: 'vue-loader',  options: {  loaders: {  'less': [  'vue-style-loader',  'css-loader',  'less-loader'  ]  }  }  },  {  test: /\.js$/,  loader: 'babel-loader',  exclude: /node\_modules/  },]  },  resolve: {  alias: {//开发环境使用vue.esm.js',官网有说明  'vue$': 'vue/dist/vue.esm.js'  },  //reuqire加载文件会自动为以下后缀的文件（如：./app.vue可以写成./app了）  extensions: ['\*', '.js', '.vue', '.json']  },  } |

快手内部文档请勿外传