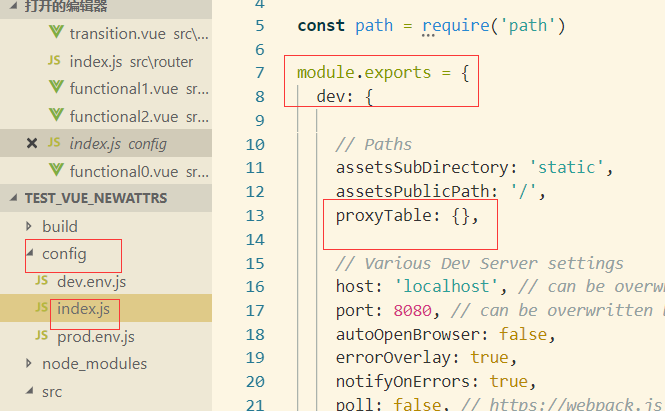
1. **在vue 的开发版中实现跨域是：**



具体的设置参考：

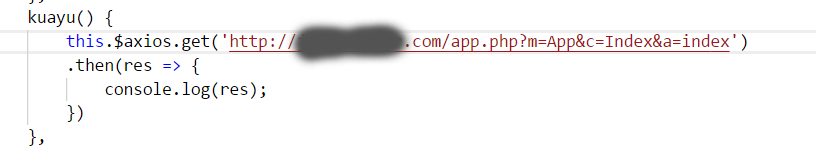
## 本地开发环境请求服务器接口跨域的问题

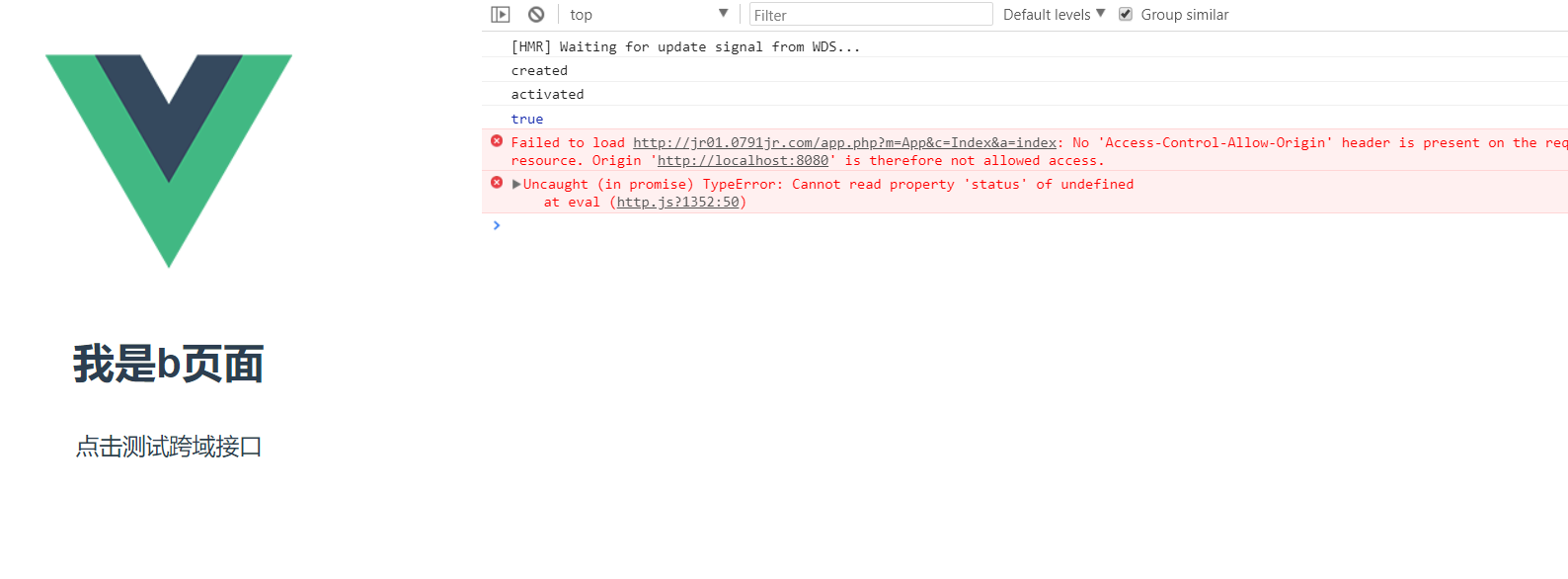
IMG_256

上面的这个报错大家都不会陌生，报错是说没有访问权限（跨域问题）。本地开发项目请求服务器接口的时候，因为客户端的同源策略，导致了跨域的问题。

下面先演示一个没有配置允许本地跨域的的情况：





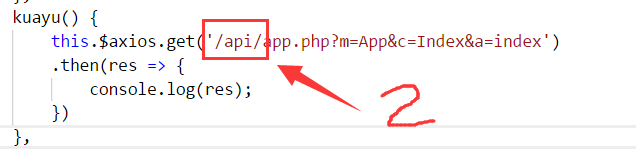


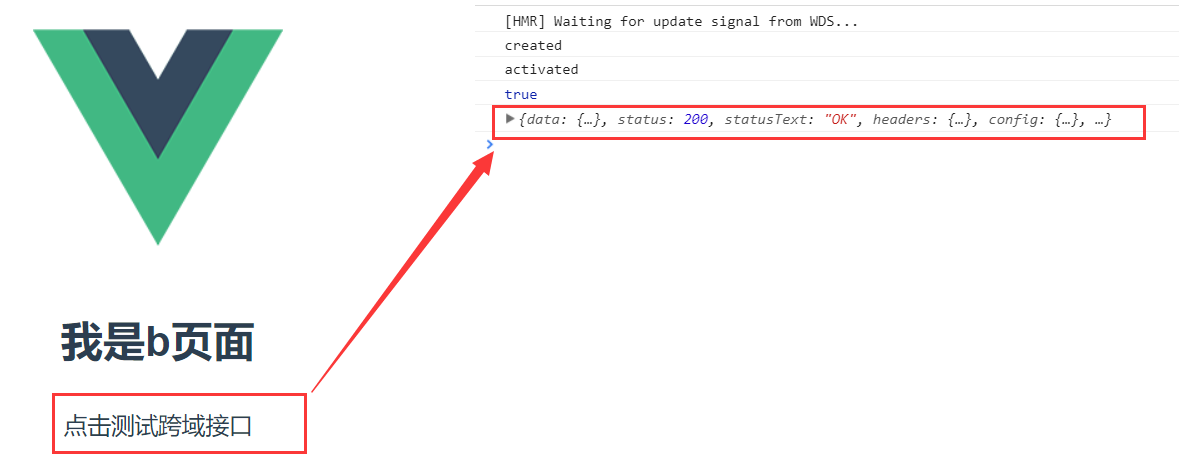
可以看到，此时我们点击获取数据，浏览器提示我们跨域了。所以我们访问不到数据。

那么接下来我们演示设置允许跨域后的数据获取情况：

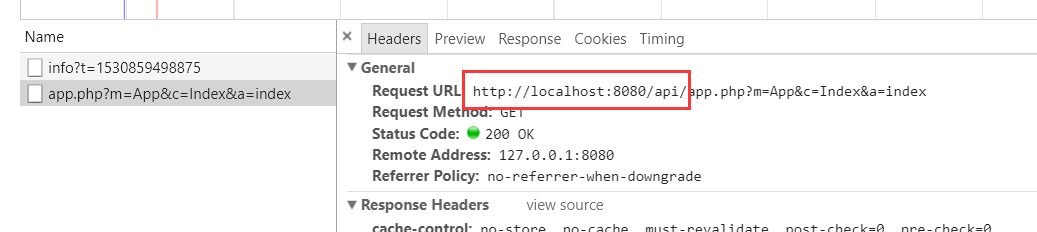


**注意：配置好后一定要关闭原来的server，重新npm run dev启动项目。不然无效。**





我们在1出设置了允许本地跨域，在2处，要注意我们访问接口时，写的是/api，此处的/api指代的就是我们要请求的接口域名。如果我们不想每次接口都带上/api，可以更改axios的默认配置axios.defaults.baseURL = '/api';这样，我们请求接口就可以直接this.$axios.get('app.php?m=App&c=Index&a=index')，很简单有木有。此时如果你在network中查看xhr请求，你会发现显示的是localhost:8080/api的请求地址。这样没什么大惊小怪的，代理而已：



好了，最后附上proxyTable的代码：

proxyTable: {

// 用‘/api’开头，代理所有请求到目标服务器

'/api': {

target: 'http://jsonplaceholder.typicode.com', // 接口域名

changeOrigin: true, // 是否启用跨域

pathRewrite: { //

'^/api': ''

}

}

}复制代码

**注意：配置好后一定要关闭原来的server，重新npm run dev启动项目。不然无效。**

1. **Vue针对开发环境（线上开发）的跨域配置：**

# 详解vue项目和普通项目如何解决开发环境与生产环境下的跨域问题

概念：

****什么是跨域？****

跨域，指的是浏览器不能执行其他网站的脚本。它是由浏览器的同源策略造成的，是浏览器对javascript施加的安全限制。所谓同源是指，域名，协议，端口均相同。

http://www.123.com/index.html 调用 http://www.123.com/server.php （非跨域）

http://www.123.com/index.html 调用 http://www.456.com/server.php （主域名不同:123/456，跨域）

http://abc.123.com/index.html 调用 http://def.123.com/server.php （子域名不同:abc/def，跨域）

http://www.123.com:8080/index.html 调用 http://www.123.com:8081/server.php （端口不同:8080/8081，跨域）

http://www.123.com/index.html 调用 https://www.123.com/server.php （协议不同:http/https，跨域）

请注意：localhost和127.0.0.1虽然都指向本机，但也属于跨域。

浏览器执行javascript脚本时，会检查这个脚本属于哪个页面，如果不是同源页面，就不会被执行。

解决方案：

****方法一：使用jsonp****

****前端代码：****

$.ajax({

url : 'XXXXXXX',

type : 'get',

dataType : 'jsonp',

jsonp : 'jsonpCallback', *//成功回调,如果不设置,默认返回callback,后端需要返回此方法,比如Java写法如下*

success : function (data) {

console.log(data);

},

error : function (error) {

console.log(error);

}

})

****后端代码(以Java为例)：****

public string ProcessCallback(string name, string email)

{

if (Request.QueryString != null)

{

string jsonpCallback = Request.QueryString["jsonpcallback"];

var user = new User

{

Name = name,

Email = email

};

*// jsonpCallback 对应前端的jsonp: 'jsonpCallback'中的jsonpCallback,如果没设置,默认callback*

return jsonpCallback + "(" + new JavaScriptSerializer().Serialize(user) + ")";

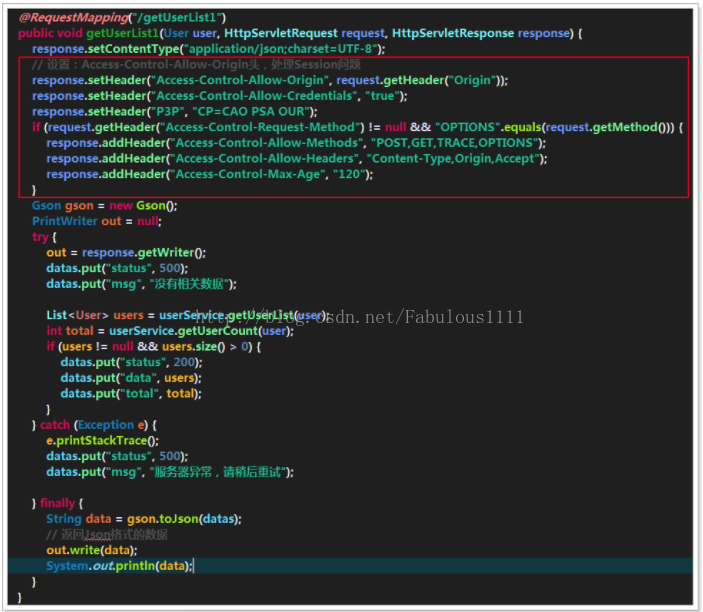
}

return "error";

}

****方法二：利用CORS（跨域资源共享）****

前端代码:  


服务器代码:  


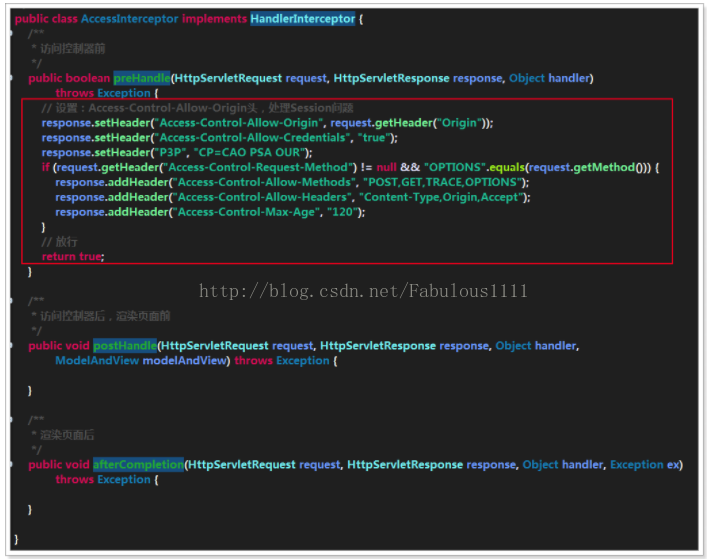
服务器端对于CORS的支持，主要就是通过设置Access-Control-Allow-Origin来进行的。如果浏览器检测到相应的设置，就可以允许Ajax进行跨域的访问。

使用该方式的特点：

与前jsonp方式相比，前端代码和未处理跨域前一样，即普通的ajax请求，但服务器代码添加了一段解决跨域的代码  
  
PS:

cors高级使用：

在springmvc中配置拦截器,创建跨域拦截器实现HandlerInterceptor接口，并实现其方法，在请求处理前设置头信息，并放行

  
在springmvc的配置文件中配置拦截器，注意拦截的是所有的文件  


例子：

****vue项目跨域举例：****

首先需要配置config下的index.js中的proxyTable，配置如下：

dev: {

*// Paths*

assetsSubDirectory: 'static',

assetsPublicPath: './',

proxyTable: {

'/api': {

target: 'http://happymmall.com',*// 请换成你需要跨域请求的地址*

changeOrigin: true,

pathRewrite: {

'^/api': ''

}

}

},

解释：proxyTable中的pathRewrite的/api理解成用‘/api’代替target里面的地址，后面组件中我们掉接口时直接用api代替 比如我要调用

****'http://hayypmmall.com/product/list.do?keyword=手机'这个接口，则换成'/api/procuct/list.do?keyword=手机'即可。****

以下为发送该请求时，axios的代码：代码中baseURL为'/api'，url为'product/list.do?keyword=手机',合起来就是'/api/procuct/list.do?keyword=手机'，

由于我们在config中的index.js设置了代理，则请求地址实际为'http://hayypmmall.com/product/list.do?keyword=手机'

axios({

method: 'get',

baseURL: '/api',

url: 'product/list.do?keyword=手机'

}).then(

res => {

console.log(res.data.data.list[0].name);

}

).catch(

err => {

console.log(err);

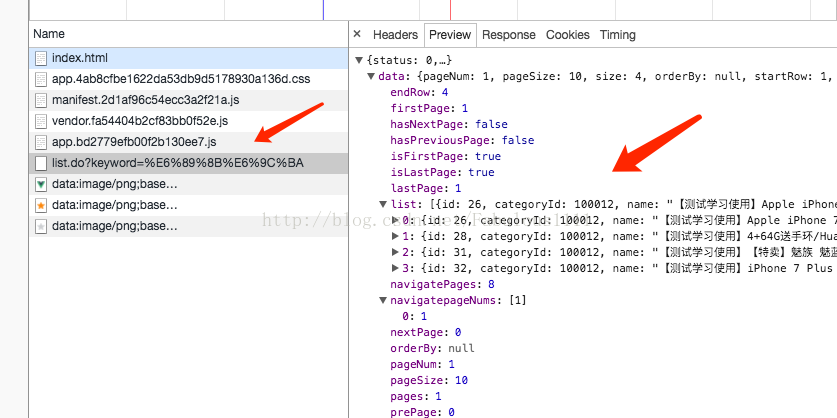
}

);

运行结果为：



****此时网络请求中也可看到成功的请求：****



然而这只是解决了开发环境下的跨域，npm run build出来后的dist代码，放到服务器中运行，还是会存在跨域问题，比如网上的这位伙伴，问题如下，相信大家也是遇到这个问题（开发环境下的跨域解决了，但是生产环境中的跨域还是没有解决）。

以下为图片：



****以上为图片****

****许多有经验的伙伴在问题下纷纷贡献上解决办法，其中一位800多声望值的伙伴回答如下：****

****以下为图片：****



****以上为图片。****

也就是问题又重新回归到了上边介绍的跨域方法二：利用CORS

****如果上线时前后端部署到同源下的话，这时只需要解决开发时的跨域，部署上线后，由于是同源，不存在跨域问题；这种情况下，本人使用的是mac下的charles，一款代理软件，轻松解决开发环境下的跨域问题；Windows下相同功能的有Fiddler。****