**关于跨域请求和django处理跨域请求最佳解决方案的总结**

2018年03月26日 20:21:46 [严登闳](https://me.csdn.net/yandenghong123) 阅读数：1889 标签： [django](http://so.csdn.net/so/search/s.do?q=django&t=blog" \t "_blank)[跨域](http://so.csdn.net/so/search/s.do?q=%E8%B7%A8%E5%9F%9F&t=blog) 更多

个人分类： [django](https://blog.csdn.net/yandenghong123/article/category/7540232" \t "_blank)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/yandenghong123/article/details/79703082

一、什么是跨域请求？

**跨域：**

简单来说就是 A 网站的 javascript 代码试图访问 B 网站，包括提交内容和获取内容。这显然是不安全的。为此，浏览器的鼻祖：网景（Netscape）公司提出了优秀的解决方案：**著名的浏览器同源策略**。现在所有支持JavaScript的浏览器都会使用这个策略。

**同源：**域名、协议、端口均相同的网站即为同源。

**流程：**

当一个浏览器的两个Tab页分别打开百度和谷歌页面时，百度发起一个脚本执行，此时浏览器会检查该脚本属于哪个页面。即检查是否同源。只有和百度同源的脚本才会被执行。若非同源，在请求数据时，浏览器会在控制台报一个异常。提示拒绝访问。

ps: 以上关于同源策略解释参考自[百度百科-同源策略](https://baike.baidu.com/item/%E5%90%8C%E6%BA%90%E7%AD%96%E7%95%A5/3927875?fr=aladdin" \t "_blank)。

二、关于解决跨域请求

解决跨域请求从前端到后端有N种解决方式。我不在此一一列举。只分享一个目前看来对django处理跨域请求的最佳方案。

1.安装django-cors-headers

pip install django-cors-headers

2.配置settings.py文件

1. INSTALLED\_APPS = [
2. ...
3. 'corsheaders'，
4. ...
5. ]
7. MIDDLEWARE\_CLASSES = (
8. ...
9. 'corsheaders.middleware.CorsMiddleware',
10. 'django.middleware.common.CommonMiddleware', *# 注意顺序*
11. ...
12. )
13. *#跨域增加忽略*
14. CORS\_ALLOW\_CREDENTIALS = True
15. CORS\_ORIGIN\_ALLOW\_ALL = True
16. CORS\_ORIGIN\_WHITELIST = (
17. '\*'
18. )
20. CORS\_ALLOW\_METHODS = (
21. 'DELETE',
22. 'GET',
23. 'OPTIONS',
24. 'PATCH',
25. 'POST',
26. 'PUT',
27. 'VIEW',
28. )
30. CORS\_ALLOW\_HEADERS = (
31. 'XMLHttpRequest',
32. 'X\_FILENAME',
33. 'accept-encoding',
34. 'authorization',
35. 'content-type',
36. 'dnt',
37. 'origin',
38. 'user-agent',
39. 'x-csrftoken',
40. 'x-requested-with',
41. )

通过以上配置，即可完美解决django跨域请求处理。唯一需要注意的就是cors-headers的中间件CorsMiddleware在注册时必须放在django-common中间件的前一个。

如有漏洞还请指正。



1. 修改views.py文件

修改views.py中对应API的实现函数，允许其他域通过Ajax请求数据：

todo\_list = [

{"id": "1", "content": "吃饭"},

{"id": "2", "content": "吃饭"},

]

class Query(View):

@staticmethod

def get(request):

response = JsonResponse(todo\_list, safe=False)

response["Access-Control-Allow-Origin"] = "\*"

response["Access-Control-Allow-Methods"] = "POST, GET, OPTIONS"

response["Access-Control-Max-Age"] = "1000"

response["Access-Control-Allow-Headers"] = "\*"

return response

@staticmethod

def post(request):

print(request.POST)

return HttpResponse()

---------------------

作者：风格色

来源：CSDN

原文：https://blog.csdn.net/qq\_27068845/article/details/73007155

版权声明：本文为博主原创文章，转载请附上博文链接！

GitHub地址: <https://github.com/ottoyiu/django-cors-headers>

随笔 - 70, 文章 - 0, 评论 - 0, 引用 - 0

## [vue-cli proxyTable中跨域中pathRewrite配置](https://www.cnblogs.com/luziluck/p/10028122.html)

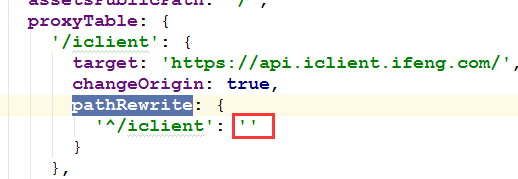
1. vue本地项目调试线上接口出现跨域问题  
2. 通过在 config/index.js 配置文件中找到proxyTable配置项  
dev: {

// Paths  
assetsSubDirectory: 'static',  
assetsPublicPath: '/',  
proxyTable: {  
'/api': {  
target: 'http://XX.XX.XX.XX:8083',  
changeOrigin: true,  
pathRewrite: {  
'^/api': '/api' // 这种接口配置出来 http://XX.XX.XX.XX:8083/api/login  
          //'^/api': '/' 这种接口配置出来 http://XX.XX.XX.XX:8083/login  
}  
}  
}  
},

3. 配置完成后需要重新编译一遍 , 调用接口的时候

        // 获取菜单权限  
getPermission(){  
this.$ajaxget({  
url: '/api/getPermission',  
data: {},  
isLayer: true,  
successFc: data => {  
console.log(data.data)  
}  
})  
},  
4 . 因为我的项目接口中 是存在 '/api' 这个字段的 所以我选择的是第一种配置  
---------------------   
作者：敌人-别人家的孩子   
来源：CSDN   
原文：https://blog.csdn.net/jikangjian/article/details/80798677   
版权声明：本文为博主原创文章，转载请附上博文链接

# [vue-cli proxyTable中跨域中pathRewrite 怎么用](https://www.cnblogs.com/hanguidong/p/9460495.html)



问：proxyTable 里面的pathRewrite里面的‘^/iclient’:'' 什么意思？

答：用代理, 首先你得有一个标识, 告诉他你这个连接要用代理. 不然的话, 可能你的 html, css, js这些静态资源都跑去代理. 所以我们只要接口用代理, 静态文件用本地.

'/iclient': {}, 就是告诉node, 我接口只要是'/iclient'开头的才用代理.所以你的接口就要这么写 /iclient/xx/xx. 最后代理的路径就是 http://xxx.xx.com/iclient/xx/xx.

可是不对啊, 我正确的接口路径里面没有/iclient啊. 所以就需要 pathRewrite,用''^/iclient'':'', 把'/iclient'去掉, 这样既能有正确标识, 又能在请求接口的时候去掉iclient.

# [vue下axios和fetch跨域请求](https://www.cnblogs.com/lhl66/p/8856815.html)



[复制代码](javascript:void(0);)

**1.在config的index.js下面进行常用跨域配置代码；**  
proxyTable: {

'/apis': { //使用"/api"来代替"http://xxxx.cn"

target: 'http://xxxx.cn', //源地址 （接口域名）

changeOrigin: true, //改变源 （是否跨域）

pathRewrite: {

'^/apis': 'http://xxxx.cn' //路径重写 (正常请求接口的简写)

}

}

}

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

**2.利用axios的post方式组件内发起请求**  
this.$axios.post("/apis/test/testToken.php",{username:"hello",password:"123456"})

.then(res => {

console.log(res)

}).catch(err=>{

console.log(err)

})

[复制代码](javascript:void(0);)

**3.这几个axios常用设置可在封装axios的api中写上也可以在main.js写上**  
axios.defaults.headers.common['token'] = "f4c902c9ae5a2a9d8f84868ad064e706" //设置header里面的token依据需求设置

axios.defaults.headers.post["Content-type"] = "application/json" //设置请求头可要可不要

Vue.prototype.$http = axios //全局可要使用this.$http来发起请求

[复制代码](javascript:void(0);)

**4.使用fetch API的形式请求接口数据**  
 fetch("/apis/test/testToken.php", {

method: "post",

headers: {

"Content-type": "application/json",

token: "f4c902c9ae5a2a9d8f84868ad064e706"

},

body: JSON.stringify({ username: "henry", password: "321321" })

})

.then(result => {

console.log(result)

return result.json()

})

.then(data => {

console.log(data)

}