**HTML5 History 模式**

vue-router 默认 hash 模式 —— 使用 URL 的 hash 来模拟一个完整的 URL，于是当 URL 改变时，页面不会重新加载。

如果不想要很丑的 hash，我们可以用路由的 **history 模式**，这种模式充分利用 history.pushState API 来完成 URL 跳转而无须重新加载页面。

const router = new VueRouter({

mode: 'history',

routes: [...]

})

当你使用 history 模式时，URL 就像正常的 url，例如 http://yoursite.com/user/id，也好看！

不过这种模式要玩好，还需要后台配置支持。因为我们的应用是个单页客户端应用，如果后台没有正确的配置，当用户在浏览器直接访问 http://oursite.com/user/id 就会返回 404，这就不好看了。

所以呢，你要在服务端增加一个覆盖所有情况的候选资源：如果 URL 匹配不到任何静态资源，则应该返回同一个 index.html 页面，这个页面就是你 app 依赖的页面。

**后端配置例子**

**Apache**

<IfModule mod\_rewrite.c>

RewriteEngine On

RewriteBase /

RewriteRule ^index\.html$ - [L]

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule . /index.html [L]

</IfModule>

除了 mod\_rewrite，你也可以使用 [FallbackResource](https://httpd.apache.org/docs/2.2/mod/mod_dir.html#fallbackresource)。

方法二：（这个比较好用避免冲突）

<IfModule mod\_rewrite.c>

**ErrorDocument 404 /index.html**

</IfModule>

**原生 Node.js**

const http = require('http')

const fs = require('fs')

const httpPort = 80

http.createServer((req, res) => {

fs.readFile('index.htm', 'utf-8', (err, content) => {

if (err) {

console.log('We cannot open "index.htm" file.')

}

res.writeHead(200, {

'Content-Type': 'text/html; charset=utf-8'

})

res.end(content)

})

}).listen(httpPort, () => {

console.log('Server listening on: http://localhost:%s', httpPort)

})

**基于 Node.js 的 Express**

对于 Node.js/Express，请考虑使用 [**connect-history-api-fallback 中间件**](https://github.com/bripkens/connect-history-api-fallback)。

**Internet Information Services (IIS)**

1. 安装 [**IIS UrlRewrite**](https://www.iis.net/downloads/microsoft/url-rewrite)
2. 在你的网站根目录中创建一个 web.config 文件，内容如下：

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<system.webServer>

<rewrite>

<rules>

<rule name="Handle History Mode and custom 404/500" stopProcessing="true">

<match url="(.\*)" />

<conditions logicalGrouping="MatchAll">

<add input="{REQUEST\_FILENAME}" matchType="IsFile" negate="true" />

<add input="{REQUEST\_FILENAME}" matchType="IsDirectory" negate="true" />

</conditions>

<action type="Rewrite" url="/" />

</rule>

</rules>

</rewrite>

</system.webServer>

</configuration>

**Caddy**

rewrite {

regexp .\*

to {path} /

}

**Firebase 主机**

在你的 firebase.json 中加入：

{

"hosting": {

"public": "dist",

"rewrites": [

{

"source": "\*\*",

"destination": "/index.html"

}

]

}

}

**警告**

给个警告，因为这么做以后，你的服务器就不再返回 404 错误页面，因为对于所有路径都会返回 index.html 文件。为了避免这种情况，你应该在 Vue 应用里面覆盖所有的路由情况，然后在给出一个 404 页面。

const router = new VueRouter({

mode: 'history',

routes: [

{ path: '\*', component: NotFoundComponent }

]

})

**nginx配置**

最近想把所有的前端项目都整合在两台机器上，通过负载访问如下路径  
Https://域名/gameCenter/  
Https://域名/member/  
Https://域名/loan/  
这样的方式访问

按照平时的打包方式，直接将webpack打包好的包放到服务器上  
服务器路径/H5/gameCenter(并非平时的/gameCenter)

在nginx中做好配置如下：

location ^~ /gameCenter/{

alias /H5/gameCenter/;

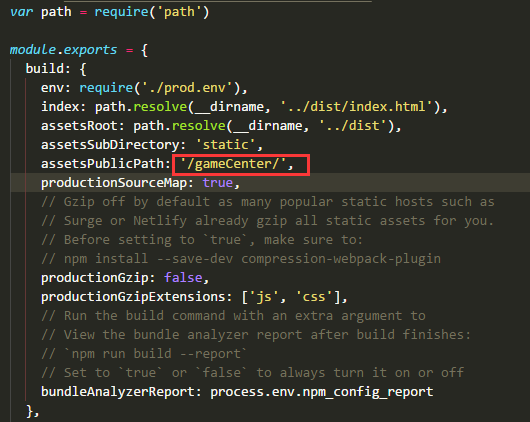
try\_files $uri $uri/ /gameCenter/index.html;

gzip on;

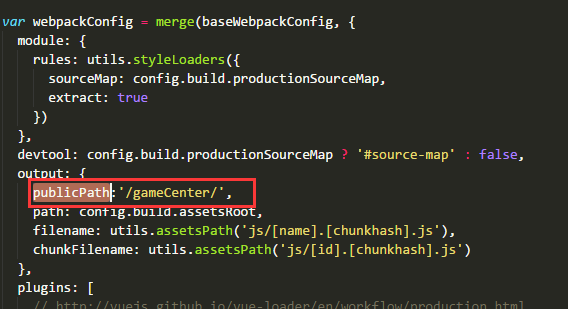
gzip\_types application/javascript text/css image/jpeg image/png;

}

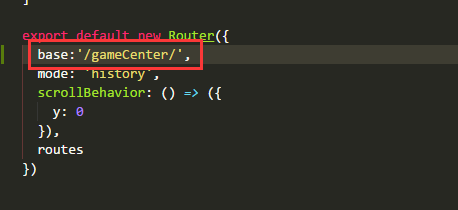
1、修改config/index.js配置文件



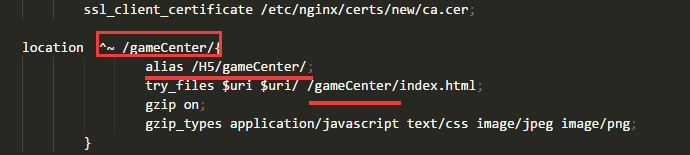
2、修改build/webpack.prod.conf.js配置文件



3、修改router/index.js



4、修改nginx配置



这几个改好了，部署就没有问题啦~ 分享给大家