# **步骤四调整过程**

**注意：**

**1）微前端的请求是由基座代理的，需要在基座中增加代理内容；**

**2）子应用与主应用之间的window和相互影响，不要向window上面去绑定内容**

**3）禁止私自向localStorage中添加内容**

主要调整内容

1）基座 与 子项目之间不存在路由、方法等交互

2）子项目的路由与基座的路由是同步的，子项目只需要按照vue-router语法调整，基座可自动监听路由变化

3）子项目的数据都通过localStorage中获取

修改步骤：

1：安装vite-plugin-qiankun; npm install vite-plugin-qiankun --save

2：全局搜索，将 mgtportal-tc.mobje.cn 改成 manager-tc.mobje.cn

3：package中增加"type": "module",

4：

在 vite.config.ts 中增加配置：

1）：import qiankun from 'vite-plugin-qiankun'

2）：plugins中增加

mode === 'development'

? qiankun('csvehicle', {

useDevMode: true

})

: qiankun('csvehicle'), // csvehicle是具体的项目名称，有这些内容 nb、ddd、zb、5d、wq、csvehicle、cs、appsupport、sink

3）：去掉 htmlPlugin

5：

index.html在修改id，并增加引入，调整如下

<div id="micro-app">

<div class="micro-loading"></div>

</div>

<script type="module" src="/src/main.ts"></script>

增加样式：

<style>

.micro-loading {  
 position: fixed;  
 top: 50%;  
 left: 50%;  
 width: 32px;  
 height: 32px;  
 border-radius: 50%;  
 border: 2px solid #1677ff;  
 border-top-color: transparent;  
 animation: circle infinite 2s linear;  
}  
  
@keyframes circle {  
 0% {  
 transform: rotate(0);  
 }  
 100% {  
 transform: rotate(360deg);  
 }  
}

</style>

在style/index.less中增加

#micro-app {

width: 100%;

height: 100%;

}

6：对src中的main.ts进行修改 ：

1）新增 import { renderWithQiankun, qiankunWindow } from 'vite-plugin-qiankun/dist/helper'

2）if (isRunningAsSubApp()) { ... } else { ... } 改成如下内容（如果引入新的内容，需要自己调整）：

// @ts-ignore  
let instance = null  
  
function render() {  
 // @ts-ignore  
 instance = createApp(WrappedApp)  
 .use(router)  
 .use(createPinia())  
 // @ts-ignore  
 loadAntIcons(instance)  
 // @ts-ignore  
 loadDirectives(instance)  
 // @ts-ignore  
 instance.mount('#micro-app')  
}  
  
if (qiankunWindow.\_\_POWERED\_BY\_QIANKUN\_\_) {  
 renderWithQiankun({  
 async mount() {  
 render()  
 return Promise.resolve()  
 },  
 async bootstrap() {},  
 async update() {},  
 async unmount() {  
 // @ts-ignore  
 await instance.unmount()  
 return Promise.resolve()  
 }  
 })  
} else {  
 render()  
}

7：页面中的路由跳转修改：

common/utils/navigation文件中的 navigateTo 函数修改为本地跳转，即

if (path.startsWith('/sub-app')) {  
 router.push({  
 path: path,  
 query  
 })  
 } else {  
 router.push({  
 path: '/sub-app' + path,  
 query  
 })   
 }

**本地所有的路由都不能缺少前缀，无论是vue-router中注册的，还是跳转的，均以sub-app开头**

即modules中所有的内容都要以sub-app开头

8：

把router/index.ts中的path为/、/:pathMatch(.\*)\*、/redirect/:pathMatch(.\*)\*、/401、/404、/login注释掉

去掉router.beforeEach

9：权限内容修改，通过localStorage，获取基座传递过来的数据，包括buttonAuths（按钮权限）、details（用户数据详情）、token

App.vue修改

onMounted 中只保留 usePlatformUserStore().initAllCities() 和 platfUserStore.reuseState()

loadAuthorizedItems 中去掉

authorityStore.loadAuthorizedMenus()

template中只保存<router-view />

authorityStore 的 loadAuthorizedButtonKeys 修改为

const storage = localStorage.getItem('manager.mobje.cn')  
// @ts-ignore  
const storageObj = JSON.parse(storage)  
const buttonAuths = storageObj.buttonAuths  
this.setAuthorizedButtons(buttonAuths)

authorityStore：

最上面增加

const authsMap = {}  
function setFlattenItems(routesList) {  
 routesList.forEach(  
 item => {  
 if (item.children) {  
 return setFlattenItems(item.children)  
 }  
  
 authsMap[item.path] = item.authMapKey  
 }  
 )  
}

在loadAuthorizedButtonKeys的最后一行增加setFlattenItems(storageObj.routesList)

isAuthorized中的代码调整为

let temp = []  
 const contentName = \_key || authsMap[location.pathname]  
 if (contentName && state.AuthorizedButtonMap[contentName]) {  
 temp = state.AuthorizedButtonMap[contentName].split(',')  
 // @ts-ignore  
 return temp.includes(authName)  
 }  
 return false

buttons中的内容调整为

let returnValue = {}  
 const contentName = \_key || authsMap[location.pathname]  
 if (contentName && state.AuthorizedButtonMap[contentName]) {  
 const temp = state.AuthorizedButtonMap[contentName].split(',')  
 Object.values(temp).forEach(  
 item => {  
 // @ts-ignore  
 returnValue[item] = item  
 }  
 )  
 }  
 return returnValue

platform-user.store

reuseState 中的内容改为

const storage = localStorage.getItem('manager.mobje.cn')  
// @ts-ignore  
const storageObj = JSON.parse(storage)  
  
const token = storageObj.token  
const username = storageObj.username  
const nickname = storageObj.nickname  
  
this.setState(username, nickname, token)

setState 中的内容改为

this.$state.username = username  
this.$state.nickname = nickname  
this.$state.token = token

10：将cookie-token.ts文件中的内容修改为：

import { usePlatformUserStore } from '@/common/store/platform-user.store'  
  
const getCookie = (name: string): string => {  
 const platfUserStore = usePlatformUserStore()  
 const token = platfUserStore.$state.token  
 return token  
}  
  
const tokenExists = (): boolean => !!getCookie('Admin-Token')  
  
export { getCookie, tokenExists }

11：要看下每个页面的table展示是否正常，因为有的table高度会仅展示表头；在某些项目中增加 :scroll="false" 全部展示出数据即可，需要视情况而定

12：

MobjeLoading.vue的.mobje-loading { 中的svg样式调整为

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

> div的样式调整为

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

13：

CommonFootBar.vue 的template修改为：

<div style='position: sticky; bottom: 0;padding: 10px 0;border-top: 1px solid #d9d9d9;background-color: #fff; display: flex; justify-content: center;'>

<slot name="content"></slot>

</div>

script中的内容全部注释掉

14：

将axios.config.ts中的下面的内容注释掉

if (config.url?.includes('tsl/api/web') && config.method === 'post') {  
 signParams(config)  
}

15：

所有和wujie有关系的字符串和内容要调整，不再需要wujie，不确定的可以和我沟通

所有和getElementFromMain相关的都要调整

16：

自测页面及功能

17：

**提交的前要创建个stash，因为gerrti有bug，一出问题代码就没了**

**然后再unshift出来**

18：

要先build一下

19：

commit前要把右侧的对号都去掉