

微服务框架说明 v1.0

本文档主要介绍 qiankun 与 vue-element-admin 完美结合实现微服务框架。

1	前言	2
1.1	开源 git 地址	3
2	微服务基座说明	3
2.1	基座说明	3
2.2	注入依赖	4
2.3	安装 qiankun	4
2.4	安装 tmvc-base	4
2.4.1	导入子应用配置文件至 src 目录下	5
2.5	修改页面操作	6
2.5.1	添加子应用挂载节点	6
2.5.2	修改路由配置文件	6
2.6	路由说明	7
2.6.1	基座路由说明	7
3	微应用注入说明	8
3.1	注入应用说明	9
3.2	注入依赖方式	9

3.2.1	安装依赖 tmvc.....	9
3.2.1	main.js 改造.....	9
3.3	vue-config.js 依赖文件.....	10
3.3.1	micro-config.js 文件下载地址.....	11
3.3.2	注入方式	11
3.4	获取基座通讯信息.....	11
3.4.1	通讯取值方法	12
3.4.2	Vuex 例子（仅供参考）	12
3.5	左侧菜单隐藏处理.....	13
3.5.1	Data 里声明以下：	13
3.5.2	view 层展示以下：	13

1 前言

使用技术栈 qiankun 将 vue-elment-admin 框架打造为微前端基座处理。

本项目文件目录包含：micro-main 是主要项目、micro-con 是子项目。

- 简单介绍一下使用的技术栈，此架构所使用主要两个开源项目以下是详细介绍：

(1) qiankun

是蚂蚁金服开源的一套完整的微前端解决方案。具体描述可查看 [文档](#) 和 [Github](#)。

(2) vue-elment-admin

是用于管理界面的生产就绪前端解决方案。它基于 vue 并使用 UI Toolkit [element-ui](#)，具体可查看[文档](#)和 [Github](#)

结合以上内容，下面将通过一个微服务 Demo 介绍 Vue 项目如何接入 qiankun，
和微应用如何完美注入到微服务中代码地址：

1.1 开源 git 地址

2、微前端开源下载地址

<https://gitee.com/tmvc/tmvc>

2、微前端工具包

<https://gitee.com/tmvc/tmvc-utils.git>

2 微服务基座说明

将普通的 vue-elment-admin 框架改造为一个微服务框架需要以下流程：

2.1 基座说明

名称	技术栈	框架
micro-main	VUE、Qiankun	vue-elment-admin

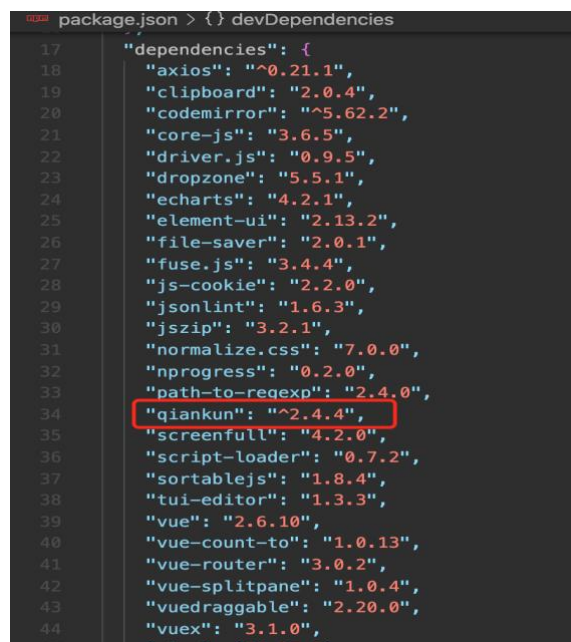
本文使用的是 qiankun2.0 之后的版本和方法，底层使用 vue-elment-admin、

详细描述可查看上方连接。

2.2 注入依赖

2.3 安装 qiankun

```
npm install qiankun
```



```
package.json > {} devDependencies
17  "dependencies": {
18    "axios": "^0.21.1",
19    "clipboard": "2.0.4",
20    "codemirror": "^5.62.2",
21    "core-js": "3.6.5",
22    "driver.js": "0.9.5",
23    "dropzone": "5.5.1",
24    "echarts": "4.2.1",
25    "element-ui": "2.13.2",
26    "file-saver": "2.0.1",
27    "fuse.js": "3.4.4",
28    "js-cookie": "2.2.0",
29    "jsonlint": "1.6.3",
30    "jszip": "3.2.1",
31    "normalize.css": "7.0.0",
32    "nprogress": "0.2.0",
33    "path-to-regexp": "2.4.0",
34    "qiankun": "^2.4.4",
35    "screenfull": "4.2.0",
36    "script-loader": "0.7.2",
37    "sortablejs": "1.8.4",
38    "tui-editor": "1.3.3",
39    "vue": "2.6.10",
40    "vue-count-to": "1.0.13",
41    "vue-router": "3.0.2",
42    "vue-splitpane": "1.0.4",
43    "vuedraggable": "2.20.0",
44    "vuex": "3.1.0",
```

图 1 qiankun 注入

2.4 安装 tmvc-base

```
npm install tmvc-base
```

2.4.1 导入子应用配置文件至 src 目录下

复制 microRouter 文件夹到 src 目录下

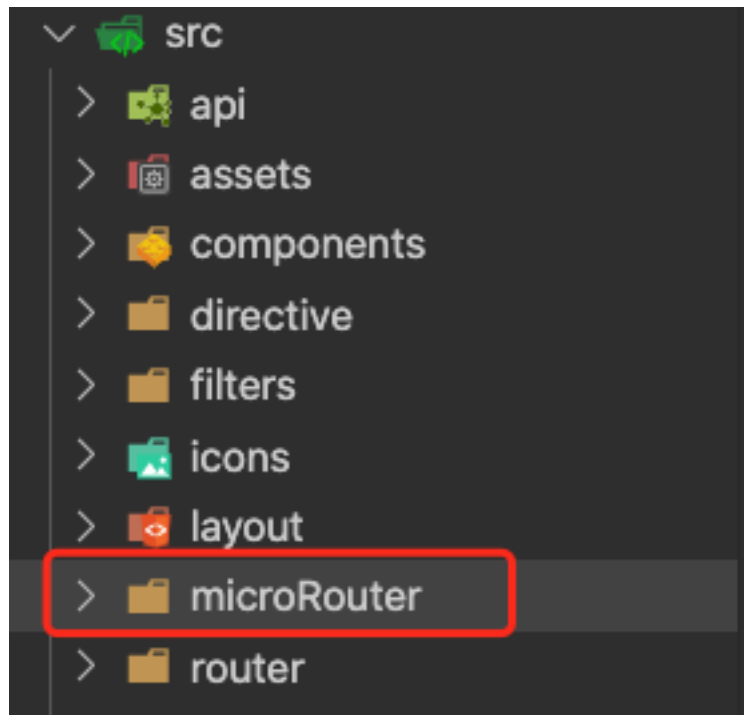


图 2 依赖文件注入

下载地址: <https://gitee.com/tmvc/tmvc-utils.git>

2.5 修改页面操作

2.5.1 添加子应用挂载节点

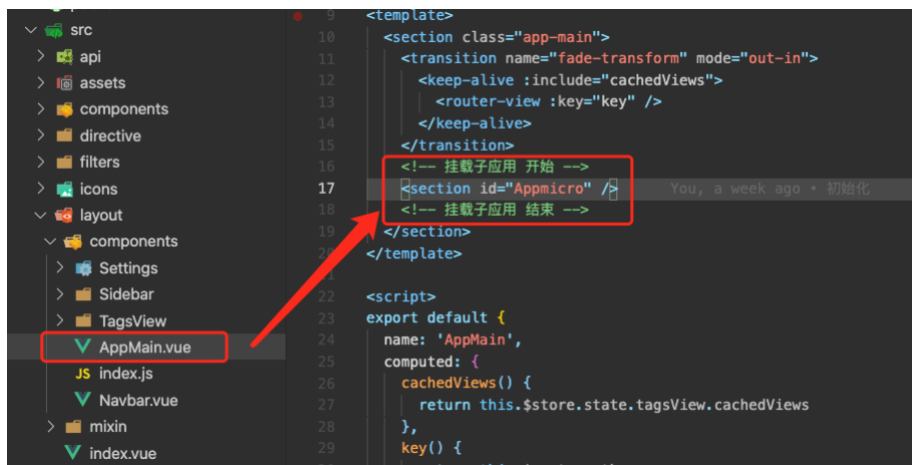


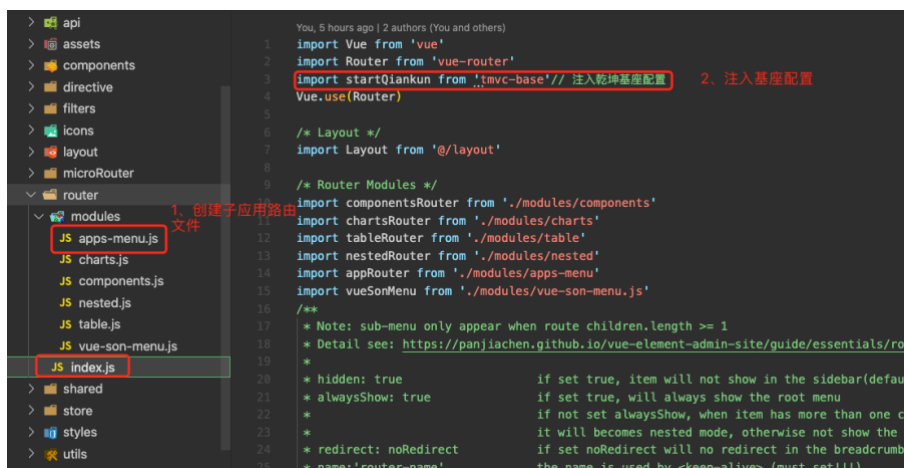
图 3 挂载节点

2.5.2 修改路由配置文件

微应用配置路由下载地址：

请下载“工具包/基座”内容使用

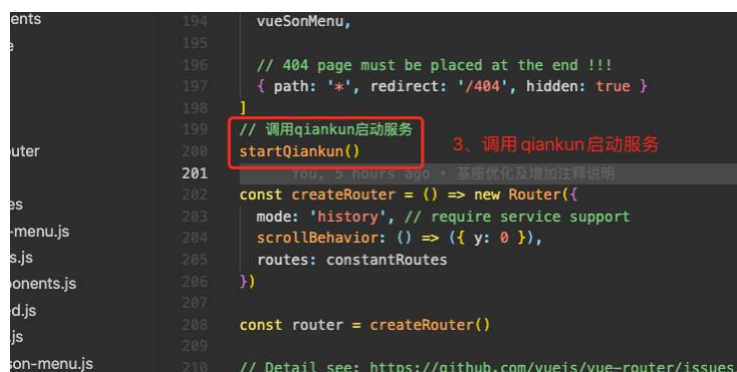
<https://gitee.com/tmvc/tmvc-utils.git>



博客地址：<https://blog.csdn.net/tamil2021/article/details/119332074>

著作：TMVC

图 4 路由配置文件 1



```

194 vueSonMenu,
195
196 // 404 page must be placed at the end !!!
197 { path: '*', redirect: '/404', hidden: true }
198 }
199 // 调用qiankun启动服务
200 startQiankun()
201 // 3. 调用 qiankun 启动服务
202
203 const createRouter = () => new Router({
204   mode: 'history', // require service support
205   scrollBehavior: () => ({ y: 0 }),
206   routes: constantRoutes
207 })
208
209 const router = createRouter()
210
211 // Detail see: https://github.com/vuejs/vue-router/issues/

```

图 5 路由配置文件 2

2.6 路由说明

2.6.1 基座路由说明

在基座配置子项目文件当中路径为：src/microRouter.js

文件内容的 name 与 activeRule 的名称必须一致如以下：

```

{
  name: 'vue-son', //微应用名称
  entry: VUE_SON_APP, //微应用服务地址
  container: '#Appmicro', //基座 id
  activeRule: '/vue-son' //访问名称，注意必须和微应用名称相同
},

```

2、微应用路由设置如以下

路径为：src/modules/vue-son-menu.js

对应的基座当中设置的微应用名称，微应用路由配置必须与配置的

microRouter.js 当中的 “name” 对应起来，可参考以下例子：

```
import Layout from '@layout'
const appRouter = {
  path: '/vue-con',
  component: Layout,
  redirect: 'doc1',
  name: 'vue-con',
  meta: {
    title: '微·子系统2',
    icon: 'table'
  },
},
children: [
  {
    path: '/vue-son',
    children: [
      {
        path: 'dashboard',
        name: 'Dashboard',
        meta: { title: '子应用·测试页面4', icon: 'menuSon' }
      }
    ]
  },
  {
    path: '/vue-son/form',
    children: [
      {
        path: 'form',
        name: 'Form',
        meta: { title: '子应用·测试页面5', icon: 'menuSon' }
      }
    ]
  }
]
}
```

3 微应用注入说明

微应用例子是以目前主流的框架 vue-elment-admin 框架为主，基于 qiankun 底层的微前端框架，进行了生命周期、通讯、依赖等一系列封装，方便维护及快速注入到微应用，仅供参考。详细框架介绍可参考前言篇里内容。

3.1 注入应用说明

微应用底层使用 vue-elment-admin 框架，详情可查看前言里的介绍。

项目名称	技术栈	框架
micro-son	VUE	vue-elment-admin

3.2 注入依赖方式

***注意：**需要本地安装有 npm 与 node 环境

3.2.1 安装依赖 tmvc

- (1) 将代码从 npm 下导入到项目当中
- (2) npm install tmvc

在 package.json 文件里插入 "tmvc": "^1.0.13"

在执行 npm install 也可以注入

3.2.1 main.js 改造

(1) 文件注入

在 mian.js 注入以下文件

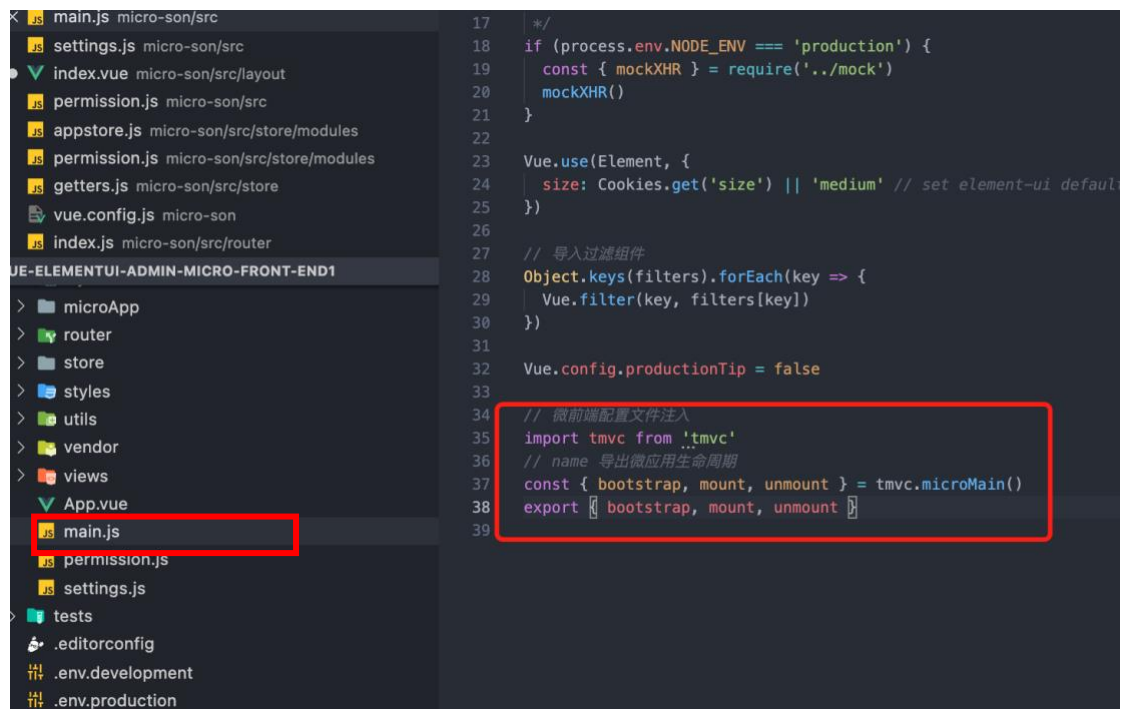


图 为 src/main.js

源码以下供使用参考：

```

// 微前端配置文件注入
import tmvc from 'tmvc'
// name 导出微应用生命周期
const { bootstrap, mount, unmount } = tmvc.microMain()
export { bootstrap, mount, unmount }

```

3.3 vue-config.js 依赖文件

在 vue-config.js 配置底部复制以下代码，供微应用使用。

*注意路径一定要写对否则会启动报错

```
150     })
151     // https:// webpack.js.org/configuration/optimization/#optimizationruntimechunk
152     config.optimization.runtimeChunk('single')
153   }
154 }
155 }
156 }
157 // 微前端子项目配置注入
158 const { microConfig, headers } = require('./src/microApp/micro-config.js')
159 module.exports.devServer.headers = headers // 配置跨域 必须
160 module.exports.configureWebpack.output = microConfig // umd 格式注入 必须
161 module.exports.publicPath = `//localhost:${port}` // 端口号配置 不是必须设置
162
163
```

图 为 vue.config.js

3.3.1 micro-config.js 文件下载地址

请下载工具包内 “子应用 microApp 文件”

Git 地址: <https://gitee.com/tmvc/tmvc-utils.git>

3.3.2 注入方式

```
// 微前端子项目配置注入
const { microConfig, headers } = require('./src/microApp/micro-config.js')
module.exports.devServer.headers = headers // 配置跨域 必须
module.exports.configureWebpack.output = microConfig // umd 格式注入 必须
module.exports.publicPath = `//localhost:${port}` // 端口号配置 不是必须设置
```

3.4 获取基座通讯信息

目前需要在生命周期中取 props 基座的内容，在通过存储等操作把值存起来供微应用使用。注意：存储方式仅供参考，取值才是重要的。下面是

博客地址: <https://blog.csdn.net/tamil2021/article/details/119332074>

著作: TMVC

目前我全局存储了 Bootstrap 和 mount 的值仅供使用。

3.4.1 通讯取值方法

```
this.$MicroBootstrap //这里是全局注入的值
```

注意：只会在微应用初始化的时候调用一次，下次微应用重新进入时会直接调用 mount 钩子，不会再重复触发如果需要重复调用 请使用---

```
this.$MicroMount 取值和方法
```

3.4.2 Vuex 例子（仅供参考）

将全局存储的值共享给 vuex?

appStore.js 文件是使用的 vuex 存储通讯 onGlobalStateChange 是父级存储的函数（观察者函数可自行动娘）

例子：props 是全局传输过来的值

```
const appStore = props => {
  console.log(props)
  /**
   * @name 监听应用间通信，并存入 store
   */
  props?.onGlobalStateChange?.(
    (value, prev) => {
      console.log(props.name, value, prev)
      store.dispatch('appstore/setMsg', 'name 为---' + props.name)
    },
    true
  )
}
```

我们将取到的值同时存储到 store 里，这里的 appstore 必须创建到 src/store 文件里的否则会报错，目前已 vuex 传输为例子

```
1 <template>
2   <div class="dashboard-container">
3     <h1>子应用的dashboard</h1>
4     <h4>父级数据验证: {{ msg }}</h4>
5     <div class="dashboard-text">name: {{ name }}</div>
6     <button class="msg-btn" @click="handleVuexMsgChange">父级传过来的方法</button>
7   </div>
8 </template>
9
10
11 <script>
12 import { mapGetters } from 'vuex'
13 export default {
14   name: 'Dashboard',
15   data() {
16     return {
17       msg: this.$store.getters.msg // 父级传过来的值
18     }
19   },
20   computed: {
21     ...mapGetters([
22       'name'
23     ])
24   }
25 }
```

3.5 左侧菜单隐藏处理

每个独立的微应用都有属于自己的菜单栏，那么我们注入到“微服务”里时需要将菜单隐藏处理，这里我做了以下操作：

以目前项目里“/src/layout/index.vue”为实例：

3.5.1 Data 里声明以下：

```
isMicroApp: window.__POWERED_BY_QIANKUN__ // 配置微前端菜单显示与否
```

3.5.2 view 层展示以下：

```
<div v-if="!isMicroApp" :class="classObj" class="app-wrapper">
  <div v-if="device==='mobile'&&sidebar.opened" class="drawer-bg"
  @click="handleClickOutside" />
```

```
<sidebar class="sidebar-container" />
<div :class="{hasTagsView:needTagsView}" class="main-container" >
  <div :class="{fixed-header:fixedHeader}">
    <navbar />
    <tags-view v-if="needTagsView" />
  </div>
  <app-main />
  <right-panel v-if="showSettings">
    <settings />
  </right-panel>
</div>
</div>
<app-main v-else />
```

*注意：根据 public-path 里取到的 `window.__POWERED_BY_QIANKUN__` 值判断当前是否在微应用还是独立运行，仅供参考。