KQGIS Portal前端二次开发手册

必要的开发环境说明(可上下浮动版本，但建议使用较新版本）：

node.js v10.15.3

npm v6.13.2

1. portal前端技术介绍

portal前端是基于es6+vue+elementui+sass进行开发的，使用的打包工具是webpack，与后端进行请求交互使用的是axios。具体版本与其他插件使用情况，请阅读package.json文件。

2、如何启动portal前端

①初次启动使用npm install命令进行包的安装，如果遇到包安装失败，请安装淘宝镜像或使用手机4G网络开热点进行安装。

②包安装完毕后，修改.env.development文件中的后端地址，然后使用npm run serve即可启动开发环境下的portal前端。

3、如何打包portal前端

①包安装完毕后，修改.env.production文件中NODE\_ENV = 'production'，然后使用npm run build进行打包，会在目录下生成dist文件夹，将dist文件夹部署至服务器即可，如使用nginx等需要配置后端代理地址。

4、portal目录结构说明



vue.config.js：前端项目打包配置【不能修改】

package.json：项目说明文件

doc文件夹：存放所有相关的文档

public文件夹：不需要经过webpack打包的文件

public/config：存放静态配置文件、图片（不经过webpack打包，打包之后可随时线上更改）

public/thirdParty：存放第三方包，如kqwebmap/IEarch这类包

public/ favicon\_kq.png：网站logo

public/index.html：网站首页【建议不要修改，修改也只增不删】其中window.portal\_config的配置一定不能删

window.portal\_config【类型Object】配置说明：

VUE\_APP\_MESSAGE\_URL：消息推送地址

NO\_SWITCH\_LANGUAGE：切换语言功能，true表示不显示，false表示显示，默认值false

NO\_SWITCH\_THEME：切换主题功能，true表示不显示，false表示显示，默认值false

WEB\_APP\_BUILDER\_STYLE：应用定制功能，主题样式

COVER\_MENU\_ROUTER\_CONFIG:{覆盖系统存在的路由，其中name属性不能覆盖，下面是示例

// "appCenter":{

// component:"webAppBuilderSC"

// }

},

"DELETE\_PORTAL\_DEFAULT\_MENU\_ROUTER\_CONFIG":[]删除portal中提供的路由菜单配置

“ADD\_MENU\_ROUTER\_CONFIG“：添加路由和菜单配置

src文件夹：存放所有前端源码

src/assets存放静态文件

src/assets/icon存放字体图标文件【需要在iconfont网站上自行维护，portal仅提供最初icon】

src/assets/images 存放图片文件【建议原始文件不要修改，可以新增文件】

src/common存放portal需要使用的公共方法【目前有的文件夹均不能删除，不能修改】

src/utils 存放portal的工具函数【目前有的文件均不能删除，不能修改】

src/style存放portal使用的样式文件，主要是公共样式和主题方案【建议原始文件不要修改，可以新增文件】

src/project存放具体页面源码

src/project/portal存放portal页面源码【建议不要修改，如果修改后期将不会再更新portal包】

src/project/independentModules存放独立项目源码，作为portal的补充与集成。【mapBuilder、webAppBuilder两个文件夹是portal在维护，建议不要修改，如果修改后期将不会更新】

5、开发前的说明

因portal是属于灵活化的二次开发框架，我们建议在进行二次开发时变量的命名最好标记上自己的项目，以免变量污染。如项目名：HelloWorld，新建一个新闻的路由，命名为：HelloWorldNews

6、如何开始开发

新建一个文件夹在src/project/independentModules路径下，文件夹命名为项目名，如HelloWorld。不建议直接在src/project/portal文件夹内新建功能，修改功能，如果发生修改portal文件夹平台将不做更新。

修改功能的两种方式：

1、复制原始代码至自己的项目文件夹下，进行修改，修改完后通过修改路由挂载组件进行相应的替换

2、通过组件继承方式修改（此方案有些代码会导致eslint报错，属正常情况。比如继承的来的data在本文档中不存在eslint就会报错。因eslint存在于正常报错，开发时需非常注意，以免产生异常与正常报错混淆），修改完后通过修改路由挂载组件进行相应的替换

7、如何新建自己项目的菜单路由

在自己的项目（如src/project/independentModules/HelloWorld）中新建menuRouterConfig.js文件（路由配置）,新建componentConfig.js文件（配置组件信息）

参考示例：src/project/portal/router/componentConf/componentConfig.js

src/project/portal/router/menuRouterConf/menuRouterConfig.js

菜单路由写在最前面的注释说明：

1、portal中菜单和路由的区别：

portal中的路由是使用的vue-router的模式，而菜单属于一种自定义的数据结构（目前基于elementui中菜单组件）。

2、路由路径和菜单路径的区别：

路由是一个树状的结构，而菜单的路径是指访问到该路由最终的完整地址。

比如：

路由结构为

[

{

path:’/index’,

children:[

{

path:’indexChildren’

}

]

}

]

那么访问到indexChildren这一层的菜单路径就是/index/indexChildren。所以在配置菜单路由的时候存在两个路径，一个叫path，一个叫menuPath，路由的路径和菜单的路径在不相同的情况下需要配置单独的menuPath。

路由配置说明：菜单和路由都是通过路由配置进行生成的

name：当前菜单路由的名字（需要唯一，并且与最终export的对象的key保持一致，如果不一致框架也会强行用key去替换name）

parent：当前菜单路由的父菜单路由的name

component：路由关联的组件，与componentConfig.js中的变量对应

mComponent：路由移动端关联的组件，与componentConfig.js中的变量对应

path:路由路径，在无菜单路径时候path默认为菜单路径

menuPath:菜单路由路径（path和menuPath不同的原因见上面注释）

sortNo：菜单排序（必须是数字，菜单排序是按照正序排的）

defaultDisplay：菜单是否默认展示（未登录状态）true/false，true为不管用户是否登录都会默认展示（有此配置项后，不会走权限）

afterLoginDefaultDisplay:登录之后默认显示true/false，true为只要用户登录了，不管是谁都会展示此菜单（有此配置项后，不会走权限）

isHomePage：菜单是否是首页true/false

isIndexMenu：菜单是否在首页菜单展示true/false

routerParentNode：路由的父节点，默认值是'/'，如果是root，则挂在与"/"路径平级，如果是其他的name，那么将挂载在其他name下面

permissionParent:根据父权限，来判定自己是否有权限，父权限的名字（示例用户：A菜单路由配置，只是单纯的挂载路由，并没有菜单，但是路由需要根据父级来鉴定权限）

indexMenuNoChild:显示在首页，但是没有子菜单，可能有子路由，true/false

redirect：重定向路径

defaultFirstRedirect:默认取第一个子菜单的路由进行重定向true/false（拥有redirect属性的时候但没有defaultFirstRedirect这个属性，会默认为true）

permissionNoIgnore:在授权时候不忽略的菜单（即使菜单设置了默认显示，登录后默认显示，主要用于下面部分子菜单需要授权，部分默认显示）true/false

meta：路由跳转时携带的参数{

//以下参数均是portal自定义参数，是与portal内部组件约定的内容，不具有强制性

indexMenuActive：第一级父菜单，用于菜单定位和选中样式

childMenuActive：第二级子菜单，用于菜单定位和选中样式

requireAuth：true/false配置需要登录（因使用了后端的权限控制，目前暂时没有具体使用场景，但是如果配置了这个会根据username进行拦截，不能仅使用此属性来做权限控制）

}

externalLinksRouter[Boolean]:外链路由，true/false，true表示是外链路由（外链路由，不配component，只用配path。添加该属性后，只会添加菜单属性，不会真实的添加进vue-router中）

customValidationRule[function]：自定义效验规则，调用时会传入三个参数（当前配置信息，登录信息，当前效验状态），返回值必须是boolean值，使用场景为登录权限效验后，仍需要做自定义效验。

tabPagesConfig[Array]:tab页签中页面配置如（组织下的用户管理、权限管理等）,配置了此属性，component属性可以配置为portal提供的通用页签容器tabPagesContainer。tabPagesConfig是不受权限控制的，默认当前路由有权限就都能看到。

tabPagesConfig配置示例如下：

[

{

name[String，必填]:"xxx",//页签唯一name，不能重复，需要是英文

menuName[String，必填]:"用户管理",//页签显示的标题名字，支持国际化

component[String，必填]:"xxx",//当前页签挂载的组件，与componentConfig.js中的变量对应

sortNo[Number，必填]:1//页签的排序，asc排，1为最前面

}

]

noMountRoute[Boolean]:不挂载路由，true/false，true表示不挂载路由（目前暂无使用场景，主要为日后扩展预留）

noMountMenu[Boolean]:不挂载菜单，true/false，true表示不挂载菜单（目前暂无使用场景，主要为日后扩展预留）

functionAuthority[Array]:功能权限（如用户管理这个路由下面，有2个功能【新增用户】【删除用户】需要走权限控制，那么就需要配置此配置）

functionAuthority配置示例：

[

{

name:“add”,

label:”新增用户”

}

]

7-1、如何删除portal自带的路由和菜单

目前提供两种配置方法：

1、在项目中配置deleteMenuRouter.js文件，配置格式为

export default {

deleteMenuRouter:[],//需要删除的菜单路由名字，类型是字符串数组

weight:1,//必须是数字，数字越大表示权重越高,portal会取权重最高的一份配置

scope:’’,//生效范围，值为independentRouter或者不填，为independentRouter表示仅在独立部署时生效

scopeSRC:’’,//生效路径，需与当前打包入口路径一致（VUE\_APP\_SRC）

}

portal作为base时，取非independentRouter的权重最大，以其他项目作为base时，取同scopeSRC且independentRouter的权重最大。

2、在index.html中的portal\_config.

DELETE\_PORTAL\_DEFAULT\_MENU\_ROUTER\_CONFIG中进行配置

portal\_config. DELETE\_PORTAL\_DEFAULT\_MENU\_ROUTER\_CONFIG(类型为字符串数组)，由于系统只有一个index.html，所以不需要配置权重，此配置自动生效。

Portal中菜单路由key说明如下：

1、资源（会删除下面所有子菜单）：service

2、资源->地图服务: serviceMap

3、资源->数据服务: serviceData

4、资源->功能服务: serviceFunction

5、资源->三维服务: serviceScene

6、资源->数据中心: serviceDatacenter

7、资源->附件: serviceFile

8、制图：mapDrafting

9、场景：scene

10、大屏：dashBoard

11、应用：app

12、群组：group

13、管理（会删除下面所有子菜单）：manage

14、管理->站点管理：siteManage

15、管理->站点管理->组织：organization

16、管理->站点管理->目录：catalog

17、管理->站点管理->安全：basicConfiguration

18、管理->站点管理->站点配置：manageSiteConfiguration

19、管理->站点管理->门户统计：portalStatistics

20、管理->站点管理->日志：manageLog

21、管理->站点管理->服务器管理：manageServer

22、管理->站点管理->参数管理：manageParamsConfig

23、管理->个人中心：personalCenter

24、管理->个人中心->我的资源：personalCenterResource

25、管理->个人中心->我的审核：personalVerify

26、管理->个人中心->我的申请：personalApply

27、管理->个人中心->我的收藏：personalCenterCollection

28、管理->个人中心->回收站：personalCenterRecycleBin

29、管理->个人中心->我的消息：myMessage

30、管理->个人中心->个人信息：personalInformation

(注意：删除管理下的菜单可能导致系统部分逻辑出现问题）

7-2、如何覆盖portal中已配置的菜单路由属性

目前提供两种配置方法：

1、在项目中配置coverMenuRouter.js文件，配置格式为

export default {

coverMenuRouter:{},//类型是对象，key为需要覆盖的菜单路由的key，value为对象，其中name属性无法覆盖。

weight:1,//必须是数字，数字越大表示权重越高,portal会取权重最高的一份配置,

scope:’’,//生效范围，值为independentRouter或者不填，为independentRouter表示仅在独立部署时生效

scopeSRC:’’,//生效路径，需与当前打包入口路径一致（VUE\_APP\_SRC）

}

portal作为base时，取非independentRouter的权重最大，以其他项目作为base时，取同scopeSRC且independentRouter的权重最大。

2、在index.html中的portal\_config. COVER\_MENU\_ROUTER\_CONFIG中进行配置 (类型为对象，key为需要覆盖的菜单路由的key，value为对象，其中name属性无法覆盖。)，由于系统只有一个index.html，所以不需要配置权重，此配置自动生效。

portal中菜单路由的key参照上面。

8、如何新建自己项目的vuex

在自己的项目（如src/project/independentModules/HelloWorld）中新建storeConfig.js，getters.js

参考示例：src/project/portal/store / storeConfig.js

src/project/portal/store / getters.js

9、如何做国际化

在自己的项目（如src/project/independentModules/HelloWorld）中新建zh.js（中文语言包）、en.js（英文语言包）文件

参考示例：src/project/portal/lang/ zh.js

src/project/portal/lang / en.js

10、portal首页说明

Portal首页采用的是可视化灵活组配模式，在界面上管理->站点管理->站点配置->首页定制中进行操作。

Portal首页如何开发组件：

正常在自己的项目（如src/project/independentModules/HelloWorld）中开发你想开发的组件。

然后在自己的项目（如src/project/independentModules/HelloWorld）中新建visualizationComponentsBoxConfig.js（组件箱配置，显示在首页定制界面的左侧组件箱区域）、visualizationComponentsConfig.js（组件配置，组件箱会从这个文件中寻找到具体的组件）文件、visualizationPropertySetConfig.js（属性设置组件，如果你想你的组件能在页面上进行动态配置）、visualizationIndexHeaderComponentsConfig.js（头部导航组件配置，portal系统头部导航是默认一直存在的，界面上也没有删除操作，默认使用的是portal中提供的头部导航，如果不想使用portal的头部，需要配置此文件）、visualizationCoverComponentsBoxConfig.js（portal中默认存在一份组件箱的配置，只会拼接visualizationComponentsBoxConfig.js的配置，如果不想使用portal默认的组件箱，需要配置此文件）

参考示例：

src/project/portal/demo/visualizationComponentsBoxConfigDemo.js

src/project/ portal/demo / visualizationComponentsConfigDemo.js

src/project/ portal/ demo/ visualizationPropertySetConfigDemo.js

src/project/ portal/demo/visualizationIndexHeaderComponentsConfigDemo.js

src/project/ portal/demo/visualizationCoverComponentsBoxConfigDemo.js

11、如何访问后端api

portal中所有访问后台接口的函数，均是传入两个参数params、config。params是传给后端的参数、config是portal中统一拦截器的配置。

config：类型object，具体属性说明如下

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 类型 | 传入值 | 默认值 | 说明 |
| noLoading | Boolean | true/false | false | 是否出现loading效果，true为不使用统一loading效果，false为使用 |
| loadingText | String | 任意字符串 | loading | 使用loading效果时需要展示的文字 |
| loadingTarget | String | dom的选择器 | document.body | 使用loading效果时出现的位置，可参照elementui详细配置 |
| loadingLock | Boolean | true/false | false | 出现loading时是否能滚动 |
| loadingBackground | String | rbg/rgba的字符串 | 'rgba(244,243,243,0.6)' | 使用loading效果时背景色 |
| loginType | String | dialog/system | system | 当需要登录时用什么方式登录，dialog使用弹出窗登录，system跳转到登录页进行登录 |
| noIntercept | Boolean | true/false | false | 是否走通用拦截器逻辑，true表示不走通用拦截器，所有返回结果自行处理，false表示走通用拦截器逻辑 |
| noTrim | Boolean | true/false | false | 去除每个参数的首尾空格，true代表不去首尾空格,false代表去除首尾空格 |

维护一个apiConfig.js，导出所有的api，以便portal集成到一起

具体api写法示例：



如何针对portal提供的统一拦截器进行扩展：

拦截器的key值如果与url第一段匹配会自动执行，匹配起始位置必须是0，否则需要每个请求指定interceptorName，noIntercept权限最高，配置了noIntercept所有拦截器全部失效

//  如何针对server.js拦截器进行扩展：

//  创建serverInterceptor.js，格式如下：

export default {

  //"xxx"与每个请求传入的config中的interceptorName对应

  "xxx":{

    //请求前拦截器，传入axios中的config，无需返回值

    before:*function*(*config*){},

    //请求前拦截器扩展，将在默认拦截器之后执行，与before不同时生效

    // 传入axios中的config，无需返回值

    beforeExtend:*function*(*config*){},

    //过滤默认请求拦截器，可配合beforeExtend使用，默认拦截器提供

    // ["beforeJwtHandle","beforeCustomizeHandle","paramsFarmatHandle"]三个步骤

    filterDefaultBefore:[],

    //请求成功拦截器，传入返回值，需返回true/false，true将调用promise的reject

    resolve:*function*(*res*){},

    //请求成功拦截器扩展，将在默认拦截器之后执行，与resolve不同时生效，

    // 传入返回值，默认拦截器的状态，需返回true/false，true将调用promise的reject

    resolveExtend:*function*(*res*,*defaultStatus*){},

    //过滤默认成功拦截器，可配合resolveExtend使用，默认拦截器提供

    // ["resolveJwtHandle","resolveStatusErrorHandle","resolveResetUserHandle","resovleConvertDateHandle"]

    filterDefaultResolve:[],

    //请求失败拦截器，传入返回值，无需返回值

    reject:*function*(*error*){},

    //请求失败拦截器扩展，将在默认拦截器之后执行，与reject不同时生效，传入返回值，无需返回值

rejectExtend:*function*(*error*){},

urlMatch:true/false,//true表示将自动做url匹配，但仅匹配第一段url，匹配到的位置必须是0

  }

 }

12、环境变量说明

NODE\_ENV = 'development'/'production'（必填，且只能为这两个值，development表示开发环境，production表示打包构建环境）

VUE\_APP\_SRC（必填，当前项目入口路径）

VUE\_APP\_ENTRY（必填，当前项目入口文件）

VUE\_APP\_BASE\_URL（开发环境必填，/api/的代理地址）

VUE\_APP\_FILE\_URL（开发环境必填，/fileapi/的代理地址）

VUE\_APP\_FILE\_DOWNLOAD\_URL（开发环境必填，/filedownloadapi/的代理地址）

VUE\_APP\_TEMPLATE（选填，独立部署时模板文件）

VUE\_APP\_ANALYZE = 'true'（production环境下选填，生成打包分析）

VUE\_APP\_PACKAGE（production环境下选填，打包生成二级域名）

VUE\_APP\_REACTIVE="true"（选填，是否执行reactiveRemConfig.js）