链滴

Vue 实现动态路由与权限思路

作者：mtkx

原文链接：https://ld246.com/article/1561627491618

来源网站：链滴

实现目标

客户端从服务端拿到路由和权限数据后，刷新项目的路由和菜单列表，并进行权限控制。

实现逻辑

动态路由控制加载

一般来说，动态路由控制分为两种：一种是将所有路由数据存储在本地文件中，然后从服务端获取用的权限信息，在路由跳转时，添加权限判断钩子，如果用户前往的页面不在权限列表内，则禁止跳转另一种则是本地只存储基本路由，如错误处理页面、无权限控制页面等，而权限路由则从服务器获取服务器根据用户的权限下发相应的路由数据，客户端利用这些数据进行路由的动态生成和添加，本文用的是第二种方法。

iview-admin项目将路由分为三种：

1. 不作为Main组件的子页面展示的页面路由，例如login、404、403等错误页面路由；
2. 作为Main组件的子页面展示但是不在左侧菜单显示的路由 otherRouter，比如首页路由；
3. 作为Main组件的子页面展示并且在左侧菜单显示的路由 appRouter;

拿到路由数据后，我们主要进行两部分操作，第一部分是遍历数据，利用组件异步加载的方法，加载个路由节点对应的组件，之后利用router.addRoutes(routes)完成路由列表的动态添加；第二部分是为iview-admin框架下的页面标签和面包屑导航，需要遍历appRouter获取路由信息，所以我们也需将路由数据存入vuex，以便全局访问。

需要特别注意的是，如果404页面为静态路由，那么第一次进入页面时，这时动态路由还未加载，找到路由地址会默认跳转到404错误页，体验很差，所以404路由先不写入路由规则中，和动态路由一加载。

主要代码实现如下：

数据请求及路由节点生成

```
//util.js

//生成路由
util.initRouter = function (vm) {
    const constRoutes = [];
    const otherRoutes = [];

    // 404路由需要和动态路由一起注入
    const otherRouter = [{
        path: '/*',
        name: 'error-404',
        meta: {
            title: '404-页面不存在'
        },
        component: 'error-page/404'
    }];
    // 模拟异步请求
    util.ajax('menu.json').then(res => {
```

```
        var menuData = res.data;
        util.initRouterNode(constRoutes, menuData);
        util.initRouterNode(otherRoutes, otherRouter);
        // 添加主界面路由
        vm.$store.commit('updateAppRouter', constRoutes.filter(item => item.children.length >
));
        // 添加全局路由
        vm.$store.commit('updateDefaultRouter', otherRoutes);
        // 刷新界面菜单
        vm.$store.commit('updateMenulist', constRoutes.filter(item => item.children.length > 0))

    });
};

//生成路由节点
util.initRouterNode = function (routers, data) {
    for (var item of data) {
        let menu = Object.assign({}, item);
        menu.component = lazyLoading(menu.component);

        if (item.children && item.children.length > 0) {
            menu.children = [];
            util.initRouterNode(menu.children, item.children);
        }
        //添加权限判断
        meta.permission = menu.permission ? menu.permission : null;
        //添加标题
        meta.title = menu.title ? menu.title : null;
        menu.meta = meta;
    }
};
```

动态加载组件

```
//lazyLoading.js

export default (url) =>()=>import(`@/views/${url}.vue`)
```

Store缓存实现

```
//app.js

 // 动态添加主界面路由，需要缓存
updateAppRouter (state, routes) {
    state.routers.push(...routes);
    router.addRoutes(routes);
},

// 动态添加全局路由，不需要缓存
updateDefaultRouter (state, routes) {
    router.addRoutes(routes);
},
// 接受前台数组，刷新菜单
updateMenulist (state, routes) {
```

```
    state.menuList = routes;
}
```

最后在main.js中进行调用

```
//main.js
 mounted () {
    // 调用方法，动态生成路由
    util.initRouter(this);
  }
```

权限控制

同动态路由实现方法类似，操作权限控制也一般也分为两种，第一种是页面显示时不控制权限，所有操作，比如按钮全部展现，然后在操作发起时，进行权限判断，如果用户拥有该操作的权限，则通过否则提醒用户无权限，第二种则是在页面加载的时候，就进行权限判断，无权限的操作不进行显示。人更喜欢第二种方法，这样不会对用户进行误导，个人认为用户看到的应该就行可操作的，不然点下钮再提示无权限的感觉一定很不爽。

本项目的思路来源见参考博文，原博主的具体思路是：在路由结构的meta字段中，添加用户操作权列表，然后注册全局指令，当节点初次渲染时，判断该页面是否存在权限，如果存在，并且传入的参不在权限列表中，则直接删除该节点。

主要代码实现如下：

在路由数据中添加permission字段，存放权限列表

```
//menu.json，模拟异步请求数据
[ {
    "path": "/groupOne",
    "icon": "ios-folder",
    "name": "system_index",
    "title": "groupOne",
    "component": "Main",
    "children": [
      {
        "path": "pageOne",
        "icon": "ios-paper-outline",
        "name": "pageOne",
        "title": "pageOne",
        "component": "group/page1/page1",
        "permission":["del"]
      },
     ...
    ]
  }
]
```

在遍历生成路由节点时，将permission字段数据存入路由节点meta属性中

```
//util.js

//生成路由节点
util.initRouterNode = function (routers, data) {
```

```
    for (var item of data) {
        ....
        //添加权限判断
        meta.permission = menu.permission ? menu.permission : null;
        ...
    }
};
```

定义全局命令组件，读取路由permission属性值获得权限列表，如果该不权限在权限列表中，则删除点

```
//hasPermission.js

const hasPermission = {
    install (Vue, options) {
        Vue.directive('hasPermission', {
            bind (el, binding, vnode) {
                let permissionList = vnode.context.$route.meta.permission;
                if (permissionList && permissionList.length && !permissionList.includes(binding.val
e)) {
                    el.parentNode.removeChild(el);
                }
            }
        });
    }
};

export default hasPermission;
```

权限组件使用示例：

```
<template>
    <div>
        <h1>page1</h1>
        <Button v-hasPermission="'add'">添加</Button>
        <Button v-hasPermission="'edit'">修改</Button>
        <Button v-hasPermission="'del'">删除</Button>
    </div>
</template>
```

全局注册组件

```
// main.js

import hasPermission from '@/libs/hasPermission.js';
Vue.use(hasPermission);
```

这种权限控制方法的优点就是，不管是管理配置还是页面处理逻辑都相对简单，没有很多重复的代码断和节点处理，在参考对比了网上几种实现方式后，个人比较推荐这一种方法。

页面标签和面包屑导航

在我看来，页面标签和面包屑都属于系统中锦上添花的页面相关控件，提高页面管理的便捷性，在iview官方admin项目中已经实现了这两个组件。所以这个项目中，只是将其进行移植，实现了组件功能

没有深入了解，感兴趣的可以仔细研究。

原文链接：Vue 实现动态路由与权限思路