## 安装

npm i vue-router

src下新建一个文件夹router，存放router文件，新建main.js作为主router文件。以下是router模板。

import Vue from "vue";

import VueRouter from "vue-router";

//引入组件

import hello from '../components/hello.vue'

import hello2 from '../components/hello2.vue';

Vue.use(VueRouter);

const router = new VueRouter({

mode:'history',

base:\_\_dirname,

routes:[

{

path:'/',

component:hello,

name:'',

alias:'/index',

props:{name:''},

children:[/\*嵌套路由配置在此，配置规则不变\*/]

},

{path:'/hello2',component:hello2}

]

})

export default router;

path:路径

name:路由名称

component：组件/模版

alias：别名

children：嵌套路由配置

props:对象模式，存储路由携带的参数

## **项目mian.js引入router**

import Vue from 'vue';

import App from './App.vue';

import VueRouter from 'vue-router'//引入vueRouter

import router from './router/main.js'//引入router配置文件

Vue.use(VueRouter);

new Vue({

router,//注入

render: h => h(App),

}).$mount('#app')

## **使用router**

<router-link to="/">index</router-link>

<router-link to="/hello2">hello2</router-link>

<router-view></router-view>

### 动态路由

配置

{path:'/hello2/:id',component:hello2}//动态携带id

调用

<router-link :to="hello2">hello2</router-link>

computed:{

hello2()=> `/hello2/${this.msg}`//动态拼接

}

抽取router中携带的参数

{{$route.params.id}}

### 组件内监视路由

若路由信息没变化则不会触发watch事件，可用此方法获取路由中的变化的内容

watch:{

'$route':function(to,from){...}

}

## **导航守卫**

### 路由单独配置

const router = new VueRouter({

mode:'history',

base:\_\_dirname,

routes:[

{

path:'/',

name:'root',

children:[

/\*嵌套路由配置在此，配置规则不变\*/

],

component:hello,

**beforeEnter:(to,from,next)=>{**

**console.log('/:',to,from);**

**next();**

**}**

},

{path:'/hello2/:id',component:hello2}

]

})

### 编程式导航

//直接导航到新路由

$router.push({path:’...’,params:{...},query:{....}})

//跟 router.push 很像，唯一的不同就是，它不会向 history 添加新记录，而是跟它的方法名一样，替换掉当前的 history 记录

$router.replace({path:’...’,params:{...},query:{....}})

//前进or后退,n为负后退，为正前进，n≠0；

$router.go(n)

### 组件单独配置

【注意】

beforeRouteEnter 中不能访问this，不过可以利用next传递一些数据

<template><div>...<div></template>

<script>

export default {

name: 'hello',

components:{

},beforeRouteEnter(to,from,next){

next();

},beforeRouteUpdate(to,from,next){

next();

},beforeRouteLeave(to,from,next){

next();

}

}

</script>

<style scoped></style>

### 全局配置

to:即将进入的路由，新路由

from:即将离开的路由，旧路由

next:操作router如何继续执行

next():按to中的内容进入新路由

next(flase):中断当前导航

next('/path'):跳转到不同的地址

next(error):(2.4.0+) 如果传入next的参数是一个 Error 实例，则导航会被终止且该错误会被传递给 router.onError() 注册过的回调

//全局前置，配置在router配置文件中

import Vue from "vue";

import VueRouter from "vue-router";

import ...

Vue.use(VueRouter);

const router = new VueRouter({

mode:'history',

base:\_\_dirname,

routes:[...]

})

//router钩子函数

**router.beforeEach((to,from,next)=>{**

**...**

**next();**

**})**

export default router;

//全局后置，配置在router配置文件中

import Vue from "vue";

import VueRouter from "vue-router";

import ...

Vue.use(VueRouter);

const router = new VueRouter({

mode:'history',

base:\_\_dirname,

routes:[ ...]

})

//router钩子函数

**router.afterEach((to,from)=>{**

**...**

**})**

export default router;

### 重定向

const router = new VueRouter({

routes: [

{ path: '/b',name:’foo’,component:hello},

**{ path: '/a', redirect: '/b' },**

**{ path: '/c', redirect: { name: 'foo' }}**

]

})

## **路由模块化**

router目录下新建项目路由结构（自定义结构，符合项目规划即可）

//路由模块

import hello2 from '../../components/h2.vue';

const router = [

{path:'/hello2/:id',component:hello2}

]

export default router;

//引入主路由文件

import Vue from "vue";

import VueRouter from "vue-router";

import **Hello2** from './hello2/index.js'

Vue.use(VueRouter);

const router = new VueRouter({

mode:'history',

base:\_\_dirname,

routes:[**...Hello2**]

})

## **视图命名**

路由默认命名：default；若不对<router-view>标签指定name将无法满足一个页面同时展示多个视图的需求；视图命名同样适用于嵌套视图，只需要在children中一样语法写即可。

//页面配置

<router-view class="view two" ></router-view>

<router-view class="view two" **name="b"**></router-view>

<router-view class="view two" **name="a"**></router-view>

//Router文件配置

const router = new VueRouter({

routes: [

{path: '/',**components: {default: Foo,a: Bar,b: Baz}**

}

]

})