大麦网项目——Vue-rooter实现单页面跳转

vue-router是Vue.js官方的路由插件，它和vue.js是深度集成的，适合用于构建单页面应用。vue的单页面应用是基于路由和组件的，路由用于设定访问路径，并将路径和组件映射起来。传统的页面应用，是用一些超链接来实现页面切换和跳转的。在vue-router单页面应用中，则是路径之间的切换，也就是组件的切换。

# 1、安装路由

1）直接下载vue文件，网址：[https://unpkg.com/vue-router/dist/vue-router.js](https://unpkg.com/vue-router/dist/vue-router.js" \t "https://router.vuejs.org/zh-cn/_blank)**。**

2）使用 CDN 链接

[Unpkg.com](https://unpkg.com/" \t "https://router.vuejs.org/zh-cn/_blank) 提供了基于 NPM 的 CDN 链接。上面的链接会一直指向在 NPM 发布的最新版本。你也可以像https://unpkg.com/vue-router@2.0.0/dist/vue-router.js 这样指定 版本号 或者 Tag。

在 Vue 后面加载 vue-router，它会自动安装的：

<script src="/path/to/vue.js"></script>

<script src="/path/to/vue-router.js"></script>

2）NPM安装（推荐使用npm安装）。

npm install vue-router

注意：一般如果使用vue-cli创建项目的话，可以直接就安装vue-router。

# **2、使用路由**

## 1）建立一个路由器模块

先建立一个路由器模块，主要用来配置和绑定相关信息。首先要使用import引入相关的包如vue和vue-router，然后用use方法加载路由。最后new Router创建一个路由，一个路由是一个对象，一条路由的实现需要三部分组成，name、path和component。name是命名，path是路径，component是组件，具体示例如下所示。

import Vue from 'vue';

import Router from 'vue-router';

// 把对应的组件或页面引入进来

import Login from '@/pages/login''

// 加载模块

Vue.use(Router);

const router = new Router({

routes: [{

path: '/', // 默认是跳转到首页

name: 'index',

component: Index

},

path: '/login', // 登录页路径

name: 'Login',

component: Login

} // 后续如果还有页面要跳转就按照这样的格式继续书写

]

})

export default router; //将路由器导出

## 启动路由器

在main.js入口启动文件中启用该路由器，main.js作为入口文件，使用import把所有要用到的包等都引用到这里。然后在vue中去运用。创建和挂载到根实例。通过 router 配置参数注入路由，从而让整个应用都有路由功能。

import Vue from 'vue'

import App from './App'

import router from './router'

//将store, router加入并生成应用

new Vue({

el: '#app',

router,

store,

template: '<App/>',

components: { App }

})

经过上面的配置之后呢，路由匹配到的组件将会渲染到App.vue里的<router-view></router-view>

那么这个App.vue里应该这样写：

<template>

<div id="app">

<router-view/>

</div>

</template>

index.html里应该这样写：

<body>

<div id="app"></div>

</body>

这样就会把渲染出来的页面挂载到这个id为app的div里了。

## 重定向 redirect

项目应用通常都会有一个首页，例如：index页，默认首先打开的也是首页，要跳转到哪个页面都可以根据上面的方法设置路由路径发生跳转，一些有的时候也需要路由器定义全局的重定向规则，全局的重定向会在匹配当前路径之前执行。重定向也是通过 routes 配置来完成，简单的从下面的一个例子了解一下。

下面例子是从 /a 重定向到 /b：

const router = new VueRouter({

routes: [

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

]

})

重定向的目标也可以是一个命名的路由：

const router = new VueRouter({

routes: [

{ path: '/a', redirect: { name: 'foo' }}

]

})

甚至是一个方法，动态返回重定向目标：

const router = new VueRouter({

routes: [

{ path: '/a', redirect: to => {

// 方法接收 目标路由 作为参数

// return 重定向的 字符串路径/路径对象

}}

]

})

## 懒加载

当打包构建应用时，Javascript 包会变得非常大，影响页面加载。如果我们能把不同路由对应的组件分割成不同的代码块，然后当路由被访问的时候才加载对应组件，这样就更加高效了。

结合 Vue 的[异步组件](https://cn.vuejs.org/guide/components.html" \l "异步组件" \t "https://router.vuejs.org/zh-cn/advanced/_blank)和 Webpack 的[代码分割功能](https://doc.webpack-china.org/guides/code-splitting-async/" \l "require-ensure-/" \t "https://router.vuejs.org/zh-cn/advanced/_blank)，轻松实现路由组件的懒加载。

首先，可以将异步组件定义为返回一个 Promise 的工厂函数 (该函数返回的 Promise 应该 resolve 组件本身)：

const Foo = () => Promise.resolve({ /\* 组件定义对象 \*/ })

第二，在 Webpack 2 中，我们可以使用[动态 import](https://github.com/tc39/proposal-dynamic-import" \t "https://router.vuejs.org/zh-cn/advanced/_blank)语法来定义代码分块点 (split point)：

import('./Foo.vue') // 返回 Promise

结合这两者，这就是如何定义一个能够被 Webpack 自动代码分割的异步组件。

const Foo = () => import('./Foo.vue')

在路由配置中什么都不需要改变，只需要像往常一样使用 Foo：

const router = new VueRouter({

routes: [

{ path: '/foo', component: Foo }

]

})

通过懒加载就不会一次性把所有组件都加载进来，而是当你访问到那个组件的时候才会加载那一个。对于组件比较多的应用会提高首次加载速度。

具体示例如下：

// 引入组件header

const Header = () => import( '@/components/header');

// 引入页面中的首页

const Index = () => import( '@/pages/index');

const Login = () => import( '@/pages/login');

## <router-link>

<router-link> 组件支持用户在具有路由功能的应用中（点击）导航。 通过 to 属性指定目标地址，默认渲染成带有正确链接的 <a> 标签，可以通过配置 tag 属性生成别的标签.。另外，当目标路由成功激活时，链接元素自动设置一个表示激活的 CSS 类名。

<router-link> 比起写死的 <a href="..."> 会好一些，理由如下：

* 无论是 HTML5 history 模式还是 hash 模式，它的表现行为一致，所以，当你要切换路由模式，或者在 IE9 降级使用 hash 模式，无须作任何变动。
* 在 HTML5 history 模式下，router-link 会守卫点击事件，让浏览器不再重新加载页面。
* 当你在 HTML5 history 模式下使用 base 选项之后，所有的 to 属性都不需要写（基路径）了。

具体使用就通过如下示例进行分析，点击名为ad-card这个模块以后会跳转到ticketDesc路径下，并且携带参数id值传过去。其中**To**的类型: string | Location。**required**表示目标路由的链接。当被点击后，内部会立刻把 to 的值传到 router.push()，所以这个值可以是一个字符串或者是描述目标位置的对象。

<div class="ad-card">

<router-link :to="{ path: '/ticketDesc', query: { id: headCard.id }}">

<img :src="headCard.imgUrl" alt="">

<div class="ad-hide">

<h3>{{headCard.itemName}}</h3>

<p>{{headCard.minPrice}}元</p>

</div>

</router-link>

</div>

更多的关于rooter-link的配置可以参考https://router.vuejs.org/zh-cn/api/router-link.html进行学习。

## 6）路由信息对象

下面是常用到的路由信息对象

1.$route.path  
字符串，对应当前路由的路径，总是解析为绝对路径，如 "/foo/bar"。  
2.$route.params  
一个 key/value 对象，包含了 动态片段 和 全匹配片段，如果没有路由参数，就是一个空对象。  
3.$route.query  
一个 key/value 对象，表示 URL 查询参数。例如，对于路径 /foo?user=1，则有 $route.query.user == 1，如果没有查询参数，则是个空对象。  
4.$route.hash  
当前路由的 hash 值 (不带 #) ，如果没有 hash 值，则为空字符串。  
5.$route.fullPath  
完成解析后的 URL，包含查询参数和 hash 的完整路径。  
6.$route.matched  
一个数组，包含当前路由的所有嵌套路径片段的 路由记录 。路由记录就是 routes 配置数组中的对象副本（还有在 children 数组）。

综合上述，一个包含重定向、懒加载的main.js如下：

import Vue from 'vue'

import Router from 'vue-router'

import store from '@/store'

const Index = () => import( '@/pages/index');

const Login = () => import( '@/pages/login');

// 加载模块

Vue.use(Router)

// 创建路由

const router = new Router({

routes: [{

path: '/',

name: 'index',

component: Index

},

{

path: '/login',

name: 'Login',

component: Login

}

]})

//获取当前页面路由信息

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

if (to.matched.some(r => r.meta.requireAuth)) {

let isLoginValid = false

const isLogin = store.getters.gIsLogin

if (isLogin) {

const { genTime, extTime } = store.state.sLoginInfo

isLoginValid = Date.now() - (genTime + extTime) \* 1000 < 0

if (!isLoginValid) store.commit('mLogout')

}

if (isLoginValid) {

next()

} else {

next({

path: '/login',

query: {redirect: to.fullPath}

})

}

} else {

next()

}

})

export default router