

vue-router安装及使用方法 (路径跳转、单页面)

安装：npm install vue-router --save

使用Vue-Router 2实现路由功能

常用属性

- router-link是用来控制跳转的
- router-view是用来控制要显示的组件的
- tag属性 可以控制 router-link 转化成的html标签；默认转成 a标签
 - tag="button"
- router-link-active 当前路由下 对应的标签具有类名
- active-class: 修改默认选中的类名；默认类名是 router-link-active

内置组件

- transition 过渡动画

```
<transition
  enter-active-class="animated zoomIn"
  leave-active-class="animated zoomOut"
  mode="out-in">
  <router-view></router-view>
</transition>
```

- keep-alive 组件不销毁
路由的来回跳转 会触发组件的销毁；组件的生命周期钩子函数会重新走一遍；
若需要组件不销毁；在 router-view外层套一个 keep-alive标签

```
<keep-alive>
  <router-view></router-view>
</keep-alive>
```

一、使用路由

在main.js中，需要明确安装路由功能：

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import App from './App.vue'
Vue.use(VueRouter)
```

- 1.定义组件，这里使用从其他文件import进来

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

- 2.定义路由 [{},{}]

```
const routes = [
  { path: '/index', component: index },
  { path: '/hello', component: hello },
]
```

- 3.创建 router 实例，然后传 routes 配置

```
const router = new VueRouter({
  routes
})
```

- 4.创建和挂载根实例。通过 router 配置参数注入路由，从而让整个应用都有路由功能

```
const app = new Vue({
  router,
  render: h => h(App)
}).$mount('#app')
```

- 经过上面配置后，路由匹配到的组件将会渲染到App.vue里的 `<router-view>`
`</router-view>`
- App.vue里应该这样写：

```
<template>
  <div id="app">
    <router-view></router-view>
  </div>
</template>
```

- index.html里呢要这样写：

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

这样就把渲染出来的页面挂载到这个id为app的div里了。

二、重定向 redirect

```
const routes = [
  { path: '/', redirect: '/index'},      // 这样进/ 就会跳转到/index
  { path: '/index', component: index }
]
```

三、嵌套路由

```
const routes = [
  { path: '/index', component: index,
    children: [
      { path: 'info', component: info}
    ]
  }
]
```

通过/index/info可以访问到info组件

四、懒加载

```
const routes = [
  { path: '/index', component: resolve => require(['./index.vue'], resolve) },
  { path: '/hello', component: resolve => require(['./hello.vue'], resolve) },
]
```

懒加载不会一次性把所有组件都加载进来，而是访问到那个组件时才加载那一个。组件比较多的应用会提高首次加载速度。

五、<router-link>

`<router-link></router-link>` 替换1版本中的a标签

```
<!-- 字符串 -->
<router-link to="home">Home</router-link>
<!-- 渲染结果 -->
<a href="home">Home</a>

<!-- 使用 v-bind 的 JS 表达式 -->
<router-link v-bind:to="'home'">Home</router-link>

<!-- 不写 v-bind 也可以，就像绑定别的属性一样 -->
<router-link :to="'home'">Home</router-link>

<!-- 同上 -->
<router-link :to="{ path: 'home' }">Home</router-link>

<!-- 命名的路由 -->
<router-link :to="{ name: 'user', params: { userId: 123 } }">User</router-link>

<!-- 带查询参数，下面的结果为 /register?plan=private -->
<router-link :to="{ path: 'register', query: { plan: 'private' } }">Register</router-link>
```

六、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）。

七/ 检测与管理需要登录的页面，全局路由守卫

配合本地存储判断是否跳转

```
{
  path: '/cart',
  component: Cart,
  name: 'Cart',
  meta: {
    title: '购物车',
    keepAlive: true, // 需要被缓存
    needLogin: true // 需要登录
  }
},
```

```
// 全局路由守卫
router.beforeEach((to, from, next) => {
  let isLogin = localTake('userMsg')
  if (to.meta.needLogin) { // 判断该路由是否需要登录权限
    if (isLogin) { // 判断是否已经登录
      next()
    }
    else {
      next({
        path: '/login',
        query: {redirect: to.fullPath} // 将跳转的路由path作为参数，登
        录成功后跳转到该路由
      })
    }
  }
  else {
    next()
  }
})
```

综合上述，一个包含重定向、嵌套路由、懒加载的main.js如下：

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import App from './App'
Vue.use(VueRouter)
const router = new VueRouter({
  routes:[
    { path: '/', redirect: '/index' },
    { path: '/index', component: resolve => require(['./component
```

```
s/index.vue'], resolve),
  children:[
    { path: 'info', component: resolve => require(['./components/info.vue'], resolve) }
  ]
},
{ path: '/hello', component: resolve => require(['./components/hello.vue'], resolve) },
]
})
const app = new Vue({
  router,
  render: h => h(App)
}).$mount('#app')
```