

vue-router路由

在之前讲 `express` 的 `node.js` 的后端框架的时候，我们已经了解过了路由的概念，路由主要就是控制请求的跳转的，那么今天我们所讲的路由其实是一个前端路由
前端路由是在浏览器端控制页面的展示

主要特点

🔊 播报 ✎ 编辑

- 速度：更好的用户体验，让用户在web app感受native app的速度和流畅，
- MVVM：经典MVVM开发模式，前后端各负其责。
- ajax：重前端，业务逻辑全部在本地操作，数据都需要通过AJAX同步、提交。

·路由：在URL中采用#号来作为当前视图的地址,改变#号后的参数，页面并不会重载。

前端路由的原理

如果我们现在想模拟前端路由通过 `#` 来改变页面展示的东西，怎么办呢？

```
1  <!DOCTYPE html>
2  <html lang="zh">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>前端路由原理</title>
9      <style>
10         * {
11             margin: 0;
12             padding: 0;
13             list-style-type: none;
14         }
15
16         html,
17         body {
18             width: 100%;
19             height: 100%;
20         }
21
22         #login,
23         #register {
24             width: 100%;
25             height: 100%;
26             display: none;
27         }
28
29         #login {
30             background-color: lightskyblue;
31         }
32         #register{
33             background-color: lightcoral;
```

```

34     }
35     #login:target,#register:target{
36         display: block;
37     }
38 </style>
39 </head>
40
41 <body>
42     <div id="login">
43         <h2>哈哈，我是一个登录页面啊</h2>
44     </div>
45     <div id="register">
46         <h1>哟嘿，小伙子，你过来注册了</h1>
47     </div>
48 </body>
49
50 </html>

```

前端路由的原理其实本质上面就是改变地址栏里面的 `hash` 值来告知系统路径变了

vue-router的安装

<https://router.vuejs.org/zh/>

因为我们目前学习是 `vue3`，所以我们使用的路由版本是 `vue-router@4`。如果以后同学们使用的是 `vue2`，则相对应的的路由文件版本是 `vue-router@3`

```
1 $ npm instal vue-router@4
```

vue-router的创建

vue-router是vue全家桶下面的一个框架，它主要实现的功能就是前端路由

```

1 //现在我们要根据路径去跳面,改变#后面的东西, #叫hash
2 const router = VueRouter.createRouter({
3     history: VueRouter.createWebHashHistory(), //以hash的形式管理路由
4     routes: []
5 });

```

把这个路由对象创建好了以后，我们就可以加载这个路由框架

```

1 <script>
2     let login = {
3         template: "#temp1"
4     };
5     let register = {
6         template: "#temp2"
7     }
8     //现在我们要根据路径去跳面,改变#后面的东西, #叫hash
9     //路由是根据路径跳转页面
10
11     const router = VueRouter.createRouter({
12         history: VueRouter.createWebHashHistory(),
13         routes: [{
14             path: "/",

```

```

15         // 重定向
16         redirect: {
17             name: "login"
18         }
19     }, {
20         path: "/login",
21         component: login,
22         name: "login"
23     }, {
24         path: "/register",
25         component: register,
26         name: "register"
27     }]
28 });
29
30 const app = Vue.createApp({
31
32 });
33 //加载路由模块
34 app.use(router);
35 app.mount("#app");
36 </script>

```

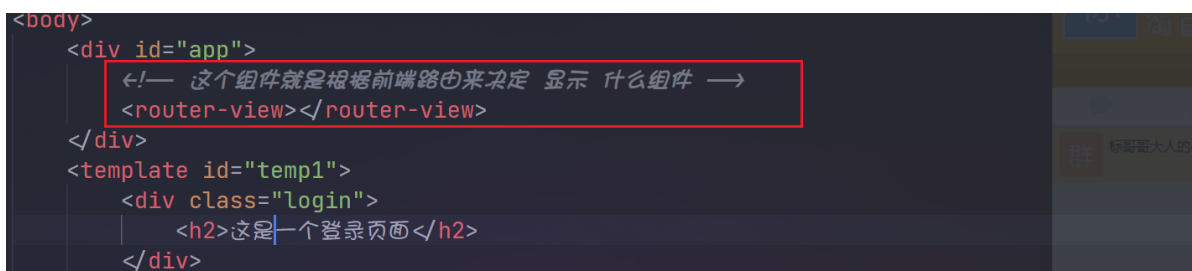
在上面的代码里面 `path` 代表的是路由的径, `component` 代表的是路由的路由匹配成功以后去找哪一个组件

当然为了后面更好的去管理路由, 我们一般都要去设置路由的名子【注意: 路由的名称不能相同】

如果我们希望由一个路由自动重定向到另一个路由, 我们可以使用 `redirect`

router-link及router-view

当我们把路由配置好了以后, 我们去访问路由的地址的时候, 没有出现效果



```

<body>
  <div id="app">
    <!-- 这个组件就是根据前端路由来决定 显示 什么组件 -->
    <router-view></router-view>
  </div>
  <template id="temp1">
    <div class="login">
      <h2>这是一个登录页面</h2>
    </div>
  </template>

```

如果希望在某一个地方根据前端路由去展示, 则可以使用 `router-view` 的形式去完成

在 `vue-router` 里面还有一个内置组件叫 `router-link`, 它决定了你跳转到哪一个路由

```

1  <template id="temp1">
2    <div class="login">
3      <h2>这是一个登录页面</h2>
4      <router-link :to="{name:'register'}">快把我带到注册页面去</router-link>
5    </div>
6  </template>
7  <template id="temp2">
8    <div class="register">
9      <h3>这是一个注册页面</h3>
10     <router-link :to="{name:'login'}">我要去登录页面.....</router-link>
11   </div>
12 </template>

```

`router-link` 默认会帮我们生成一个 `a` 标签，然后里面有一个 `to` 的属性用于决定去哪个路由

`router-link` 还需要到脚手架的项目里面去讲，在这里只是展示了基本的用法

api跳转

在上面的代码里面，我们已经可以看到使用 `router-link` 来完成页面的跳转，但是 `router-link` 帮我们生成的是一个 `a` 标签，如果我们不希望通过 `a` 标签跳转，而是想通过JS代码去实现跳转，怎么办呢？

push方式的跳转

```
1 this.$router.push({
2   name: "register"
3 });
```

直接通过路由管理对象来实现页面的跳转，跳转的时候根据路由的名子再进行跳转

push的跳转是会形成历史记录的，所以可以在浏览器里面“后退”

通过 `back()` 退回到之前的页面

```
1 this.$router.back();
```

通过 `replace()` 跳转页面

```
1 this.$router.replace({
2   name: "register"
3 });
```

这一种跳转也会去新的页面，但是它不替换掉之前的历史记录，不能够再通过 `back()` 返回到之前的页面，这一点与 `location.replace()` 所实现的效果是一样

路由传值

1.通过query进行传值

2. 通过 `params` 进行传值

路由管理对象与路由单体对象

在路由的模块里面，有两个核心对象，一个是路由的管理对象，一个是路由的单体对象

路由的管理对象

这个对象用于管理路由，它是专门负责路由的跳转，在 `vue` 的内部可以通过 `this.$router` 来拿到这个对象

路由单体对象

这个对象是用于记录当前的虚拟页面的信息的，它只负责记录当前路由的一些信息，如当前路由是否有参数，当前路由的名子是什么，当前路由的地址是什么，当前路由的meta元素信息是什么

嵌套路由的使用

嵌套路由的常规布局效果

案例



我们现在希望实现上面的效果，怎么办呢？

```
1  <!DOCTYPE html>
2  <html lang="zh">
3
4  <head>
5    <meta charset="UTF-8">
6    <meta http-equiv="X-UA-Compatible" content="IE=edge">
7    <meta name="viewport" content="width=device-width, initial-scale=1.0">
8    <title>vue-router案例</title>
9    <style>
10     * {
11       margin: 0;
12       padding: 0;
13       list-style-type: none;
14     }
15
16     html,
17     body,
18     #app {
19       width: 100%;
20       height: 100%;
21     }
22
23     #app {
24       display: flex;
25       flex-direction: column;
26     }
27
28     .content-box {
29       flex: 1;
30       overflow: auto;
31     }
32
33     .tab-bar {
34       height: 55px;
35       background-color: white;
36       display: flex;
37       justify-content: space-around;
38       border-top: 1px solid gray;
39       box-sizing: border-box;
40     }
41
42     .tab-bar>li {
43       width: 55px;
44       font-size: 14px;
45       display: flex;
46       justify-content: center;
47       align-items: center;
48     }
49
50     .page-box {
51       width: 100%;
52       height: 100%;
53       background-color: lightblue;
54     }
```

```

55         .selected{
56             color: red;
57             font-weight: bold;
58         }
59     </style>
60 </head>
61
62 <body>
63     <div id="app">
64         <div class="content-box">
65             <router-view></router-view>
66         </div>
67         <!-- 如果想通过router-link生成自定义标签,一定要下向面写 -->
68         <ul class="tab-bar">
69             <router-link :to="{name:'ChooseFood'}" custom #default="
{navigate,href,route,isActive}">
70                 <li @click="navigate" :class="{selected:isActive}">点餐</li>
71             </router-link>
72             <router-link :to="{name:'Order'}" custom #default="
{navigate,href,route,isActive}">
73                 <li @click="navigate" :class="{selected:isActive}">订单</li>
74             </router-link>
75             <router-link :to="{name:'Category'}" custom #default="
{navigate,href,route,isActive}">
76                 <li @click="navigate" :class="{selected:isActive}">分类</li>
77             </router-link>
78             <router-link :to="{name:'My'}" custom #default="
{navigate,href,route,isActive}">
79                 <li @click="navigate" :class="{selected:isActive}">我的</li>
80             </router-link>
81         </ul>
82     </div>
83     <template id="temp1">
84         <div class="page-box" style="background-color: lightseagreen;">
85             <h2>这是点餐的页面</h2>
86         </div>
87     </template>
88     <template id="temp2">
89         <div class="page-box" style="background-color: lightskyblue;">
90             <h2>这是订单的页面</h2>
91         </div>
92     </template>
93     <template id="temp3">
94         <div class="page-box" style="background-color: lightpink;">
95             <h2>这是分页的页面</h2>
96         </div>
97     </template>
98     <template id="temp4">
99         <div class="page-box" style="background-color: lightyellow;">
100             <h2>这是我的的页面</h2>
101         </div>
102     </template>
103 </body>
104 <script src="./js/vue.global.js"></script>
105 <script src="./js/vue-router.global.js"></script>
106 <script>

```

```
107 //第一步：定义了4个组件，用于表示4个虚拟的页面
108 let ChooseFood = {
109     template: "#temp1"
110 };
111 let Order = {
112     template: "#temp2"
113 };
114 let Category = {
115     template: "#temp3"
116 };
117 let My = {
118     template: "#temp4"
119 };
120 //第二步：定义了路由
121 const router = VueRouter.createRouter({
122     // 以hash的形式管理路由
123     history: VueRouter.createWebHashHistory(),
124     //配置有哪几个路由路径
125     routes: [{
126         path: "/ChooseFood",
127         component: ChooseFood,
128         name: "ChooseFood"
129     }, {
130         path: "/Order",
131         component: Order,
132         name: "Order"
133     }, {
134         path: "/Category",
135         component: Category,
136         name: "Category"
137     }, {
138         path: "/My",
139         component: My,
140         name: "My"
141     }]
142 });
143
144 const app = Vue.createApp({
145 });
146 //第三步:加载路由模块
147 app.use(router);
148
149 app.mount("#app");
150 </script>
151 </html>
```