08-vue-router: vue3路由管理器

一、基本介绍

1. 单页应用程序介绍

1.1 概念

单页应用程序：SPA(Single Page Application)是指所有的功能都在**一个HTML页面**上实现

1.2 具体示例

单页应用网站： 网易云音乐 https://music.163.com/

多页应用网站： 京东 https://jd.com/

1.3 单页应用 VS 多页应用

开发分类	实现方式	页面性能	开发效率	用户体验	学习成本	首屏加载	SEO
单页	一个html页面	按需更新 性能高	高	非常好	高	慢	差
多页	多个html页面	整页更新 性能低	中等	一般	中等	快	优

单页应用类网站：系统类网站 / 内部网站 / 文档类网站 / 移动端站点

多页应用类网站：公司官网 / 电商类网站

1.4 总结

1. 什么是单页面应用程序?

   答：所有的功能都在一个html页面上

2. 单页面应用优缺点?

   答：1、优点：体验好、开发效率高

   2、缺点：首屏加载相对较慢、不利于SEO

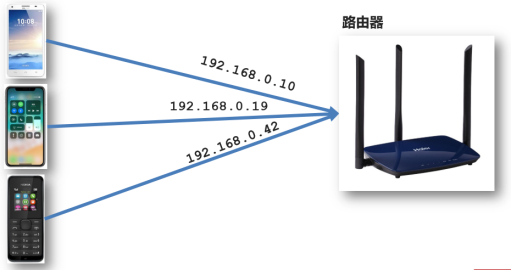
3. 单页应用场景?

   答：`系统类网站 / 内部网站 / 文档类网站 / 移动端站点`

2. 路由介绍

2.1 路由的介绍

生活中的路由：设备和ip的映射关系



Vue中的路由：**路径和组件**的**映射**关系

单页面应用程序, 之所以开发效率高, 性能好, 用户体验好， 最大的原因就是: **页面按需更新**

比如当点击【发现音乐】和【关注】时, **只是局部更新内容,** 对于头部是不更新的，要按需更新, 首先就需要明确: **访问路径** 和 **组件**的对应关系！访问路径和页面的对应关系如何确定呢？ **路由**

2.2 如何实现路由

借助 vue3 的好基友 vue-router, 当修改地址栏路径时，切换显示的组件

2.3 介绍vue-router

Vue Router | Vue.js 的官方路由

Vue 官方的一个路由插件, 是一个第三方包

2.4 总结

1. 什么是路由?

   答：一种映射(对应)关系

2. Vue中的路由是什么

   答：路径和页面的映射关系

3. vue-router是什么？

   答：vue的官方路由

3. 组件存放目录

3.1 组件分类

人为的把 `.vue` 文件分为两类，仅仅是为了便于理解和管理，但二者本质无区别

- 页面组件: 配合路由切换, 展示整个页面, 不复用的
- 复用组件: 用于组装页面组件, 可复用的

3.2 存放目录

1. 页面组件 - 页面展示 - 配合路由使用

   放置在 src/views 目录下

2. 复用组件 - 用于组装页面组件

   放置在 src/components 目录下

3.3 总结

1. 组件分为哪两类？分类的目的？

   答：页面组件、复用组件；便于管理

2. 不同分类的组件应该放在什么文件夹？作用分别是什么？

   答： 1、 `页面组件 -> src/views -> 配合路由切换`

   2、 `复用组件 -> src/components -> 组装页面组件`

二、基本使用和模块封装

1. 基本使用(4+2)

1.1 四个固定步骤

如下4个固定的步骤（不用背）

1. 下载 VueRouter 模块

```
yarn add vue-router
```

2. 导入相关函数

```
import { createRouter, createWebHashHistory } from 'vue-router'
```

3. 创建路由实例

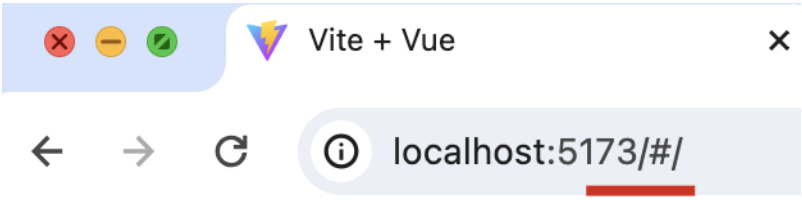
```
const router = createRouter({
    // 哈希模式，路径带 #
    history: createWebHashHistory(),
    routes: [
      // 路由表规则，即 path 与 component 的对应关系
    ]
})
```

4. 注册，将路由实例注册到应用中，让规则生效

```
app.use(router)
```

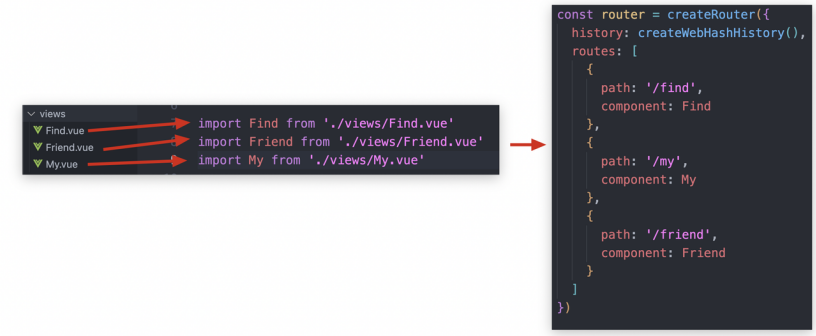

当完以上 4 步之后，就可以看到浏览器地址栏中的路径变成了 /#/的形式。

表示项目的路由已经被 Vue-Router 管理了



1.2 两个核心步骤

1. views目录下，创建需要的页面组件，并配置路由规则



views/Find.vue

```vue
<script setup></script>

<template>
  <div class="find">
    <p>发现音乐</p>
    <p>发现音乐</p>
    <p>发现音乐...</p>
  </div>
</template>

<style scoped></style>
```

views/My.vue

```vue
<script setup></script>

<template>
  <div class="my">
    <p>我的音乐</p>
    <p>我的音乐</p>
    <p>我的音乐...</p>
  </div>
</template>

<style scoped></style>
```

views/Friend.vue

```vue
<script setup></script>

<template>
  <div class="friend">
```

```
    <p>朋友</p>
    <p>朋友</p>
    <p>朋友...</p>
  </div>
</template>

<style scoped></style>
```

main.js

```
// 导入两个相关函数
// createRouter(): 创建路由实例
// createWebHashHistory(): 创建哈希模式的路由，路径带 #
import { createRouter, createWebHashHistory } from 'vue-router'

// 导入 3 个页面组件
import Find from '@/views/Find.vue'
import Friend from '@/views/Friend.vue'
import My from '@/views/My.vue'

// 创建路由实例
const router = createRouter({
  history: createWebHashHistory(),
  routes: [
    {
      path: '/find',
      component: Find
    },{
      path: '/my',
      component: My
    },{
      path: '/friend',
      component: Friend
    }
  ]
})

// 注册
app.use(router)
```

2. 给路由出口(路径匹配的组件, 显示的位置)

App.vue

```vue
<script setup></script>

<template>
  <!-- 路由出口 -->
  <router-view />
</template>
```

1.3 路由运作原理

当浏览器url改变时, 匹配路由表数组中的path值，如果命中了，

则把相应的component渲染到 `<router-view />` 的位置；否则显示空白

1.4 总结

1. 如何实现 路径改变，对应组件 切换,应该使用哪个插件?

   答： `vue-router`

2. vue-router的使用步骤是什么(4+2)?

   答：下载->导入->创建->注册-> `配置规则表->给出口`

2. 抽离封装路由模块

2.1 问题

路由配置代码都写在 main.js 中合适吗？显然不合适, 会让 main.js 代码变得臃肿

2.2 目标

将路由模块抽离出来。 好处：**利于管理和维护**

2.3 代码示例

新建 `router/index.js`

```vue
import { createRouter, createWebHashHistory } from 'vue-router'

// 注意：脚手架环境下 @ 代指 src 目录，可以用于快速引入组件
import Find from '@/views/Find.vue'
import Friend from '@/views/Friend.vue'
```

```
import My from '@/views/My.vue'

const router = createRouter({
  history: createWebHashHistory(),
  routes: [
    {
      path: '/find',
      component: Find
    },
    {
      path: '/my',
      component: My
    },
    {
      path: '/friend',
      component: Friend
    }
  ]
})

// 导出路由实例
export default router
```

main.js

```
import { createApp } from 'vue'
import App from './App.vue'

// 导入路由实例
import router from './router'

const app = createApp(App)

// 注册
app.use(router)

app.mount('#app')
```

2.4 总结

1. 封装抽离路由模块的好处是什么？

   答：便于管理、维护

2. 以后如何快速引入组件？

答：可以借助 `@` 代指 `src 目录`

三、声明式导航与传参

1. 声明式导航

1.1 问题

要手动输入地址，切换页面，不合理吧？能否点击链接进行跳转

1.2 解决方案

vue-router 提供了一个全局组件 router-link, 用于点击跳转，需添加 to 属性指定路径，

其本质还是 a 标签

语法： `<router-link to="path值"> xxx </router-link>`

```vue
<script setup></script>

<template>
  <nav>
    <router-link to="/find">发现音乐</router-link>
    <router-link to="/my">我的音乐</router-link>
    <router-link to="/friend">朋友</router-link>
  </nav>
  <!-- 一级路由出口 -->
  <router-view />
</template>

<style>
  nav a {
    color: #333;
    text-decoration: none;
  }
  nav a:nth-child(2) {
    margin: 0 80px;
  }
  nav a.router-link-active {
    background: red;
    color: #fff;
  }
</style>
```

1.3 自带高亮类名

使用router-link跳转后，我们发现。当前点击的链接默认加了两个class的值

`router-link-exact-active` 和 `router-link-active`，

我们可以给任意一个class属性添加高亮样式即可实现功能

1.4 总结

1. router-link是什么？

   答： `声明式导航`，点击跳转路由的

2. router-link怎么用？

   答：添加 `to` 属性

3. router-link的好处是什么？

   答：自带 `激活类名`，方便实现高亮样式

2. 两个类名

当我们使用跳转时，自动给当前导航加了**两个类名**

```
<nav> == $0
  <a href="#/find" class="router-link-active router-link-exact-active" aria-current="page">发现音乐</a>
  <a href="#/my" class>我的音乐</a>
  <a href="#/friend" class>朋友</a>
</nav>
```

2.1 router-link-active

模糊匹配

只要是以/find开头的路径都可以和 to="/find"匹配到

2.2 router-link-exact-active

精确匹配

to="/find" 仅可以匹配 /find

2.3 总结

1. router-link 会自动给当前导航添加两个类名，有什么区别呢？

   答：1、 `router-link-active`：模糊匹配

   2、 `router-link-exact-active`：精确匹配

3. 声明式导航-传查询参

3.1 目标

在跳转路由时，进行传参，比如：现在我们在发现音乐页，点击去朋友页，并携带id，方便后续查询详情，如何传参？

3.2 跳转传参

我们可以通过两种方式，在跳转的时候把所需要的参数传到其他页面中

1. 查询参数传参
2. 动态路由传参

3.3 查询参数传参

- 传参(有2种格式)

  a. 字符串

```
<router-link to="/path?参数名=值"> xxx  </router-link>
```

  a. 对象

```
<router-link :to="{
    path: '/path',
    query: {
      参数名: 值
      ...
    }
}"> xxx </router-link>
```

- 接收参数

```
<script setup>
  import { useRoute } from 'vue-router'
  // 得到当前激活的路由对象
  const route = useRoute()
  // 获取查询参数
  console.log(route.query)
</script>
```

3.4 代码示例

App.vue

```
<!-- 字符串 -->
<router-link to="/friend?id=10086"> 朋友 </router-link>

<!-- 对象 -->
```

```vue
<router-link :to="{
  path: '/friend',
  query: {
    id: 10086
  }
}"> 朋友 </router-link>
```

Friend.vue

```vue
<script setup>
  import { useRoute } from 'vue-router'
  const route = useRoute()
  console.log(route.query.id)
</script>
```

3.5 总结

1. 路由查询参数如何传递?

   答：字符串: `"/path?参数名=值"` 或 对象: `path+query`

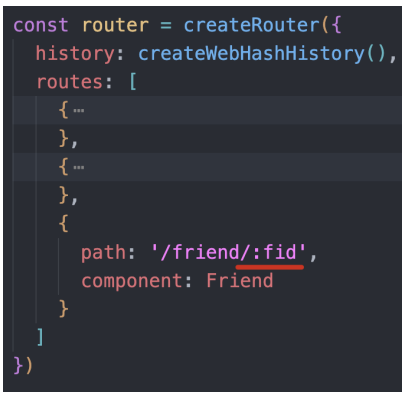
2. 查询参如何接收?

   答： `route.query.参数名`

4. 声明式导航-传动态参

4.1 动态路由传参方式

- 配置动态路由
- 动态路由后面的参数可以随便起名，但要有语义




```vue
const router = createRouter({
  ...
  routes: [
```

```
    ...
    {
      path: '/friend/:fid',
      component: Friend
    }
  ]
})
```

- 传递参数(有2种格式)

  a. 字符串

```
<router-link to="/path/具体值"> xxx </router-link>
```

  a. 对象

```
<router-link :to="{
  name: 'Friend',
  params: {
    参数名: 具体值
  }
}"> xxx </router-link>
```

- 接收参数

```
<script setup>
  import { useRoute } from 'vue-router'
  const route = useRoute()
  // 获取动态路由参数
  console.log(route.params)
</script>
```

4.2 代码示例

router/index.js

```
createRouter({
  routes: [
    {
      name: 'Friend',
```

```
    path: '/friend/:fid',
    component: Friend
  }
 ]
})
```

App.vue

```
<!-- 字符串 -->
<router-link to="/friend/10010"> 朋友 </router-link>

<!-- 对象 -->
<router-link :to="{
  name: 'Friend',
  params: {
    fid: 10010
  }
}"> 朋友 </router-link>
```

Friend.vue

```
<script setup>
  import { useRoute } from 'vue-router'
  const route = useRoute()
  console.log(route.params.fid)
</script>
```


4.3 总结

1. 路由动态参数如何传递?

   答: 1、 `/:参数名` 先占位

   2、 字符串: `"/path/具体值"` 或 对象: `name+params`

2. 查询参如何接收?

   答: `route.params.参数名`

5. 查询参 VS 动态参

5.1 对比

1. 查询参数传参 (比较适合传**多个参数**)

a. 跳转：to="/path?参数名=值&参数名2=值"

b. 获取：route.query.参数名

2. 动态路由传参 (**优雅简洁**，传单个参数比较方便)

a. 配置动态路由：path: "/path/:参数名"

b. 跳转：to="/path/参数值"

c. 获取：route.params.参数名

3. 注意：动态路由也可以传多个参数，但一般只传一个

5.2 总结

1. 声明式导航跳转时, 有几种方式传值给路由页面?

答：2种，查询参数传参 和 动态路由传参

四、更多配置

1. 重定向

1.1 问题

网页打开时，url 默认是 / 路径, 未匹配到组件时, 会出现空白

1.2 解决方案

重定向: 匹配 / 后, 比如强制跳转 /find 路径, 避免页面空白

1.3 语法

```
{ path: 匹配路径, redirect: 要重定向的路径 }
```

1.4 代码示例

```
const router = createRouter({
  ...

```

```
  routes: [
    // 访问 / , 自动跳转到 /find
    { path: '/', redirect: '/find'},
    ...
  ]
})
```

2. 404

2.1 作用

当路径找不到匹配时，给个提示页面

2.2 位置

404的路由，虽然配置在任何一个位置都可以，但一般都**配置在其他路由规则的最后面**

2.3 语法

path: "*" (任意路径) – 前面不匹配就命中最后这个

```
import _404 from '@/views/404.vue'

const router = new VueRouter({
  routes: [
    ...
    { path: '*', component: _404 } // 404配置，推荐放在路由表的最后一个
  ]
})
```

2.4 代码

`views/404.vue`

```
<script setup> </script>

<template>
  <div>
    <h3>404</h3>
    <p> 你访问的页面去了月球 </p>
    <router-link to="/"> 去首页 </router-link>
  </div>
</template>
```

router/index.js

```
...
import _404 from '@/views/404'

const router = createRouter({
  routes: [
    ...

    { path: '*', component: _404 }
  ]
})

export default router
```

3. 模式

3.1 问题

路由的路径看起来不好看，有#, 能否切成真正路径形式?

- hash路由(默认) 例如: http://localhost:5173/#/find
- history路由(常用) 例如: http://localhost:8080/find (上线需要服务器端支持，开发环境Vite给规避掉了history模式的问题)

3.2 语法

```
const router = new VueRouter({
  // 历史模式: createWebHistory(), 路径带 #
  // 哈希模式: createWebHashHistory(), 路径不带 #
  history: createWebHashHistory(),

  routes:[]
})
```

五、编程式导航与传参

1. 编程式导航

1.1 问题

如何主动做路由跳转？比如：登录成功自动跳转至首页

1.2 解决方案

编程式导航： 用JS代码来进行跳转

1.3 语法

`路由实例 router.push(路径)`

```
<script setup>
  import { useRouter } from 'vue-router'
  const router = useRouter()
  router.push(字符串)
  router.push(对象)
</script>
```

1.4 代码示例

Find.vue

```
<script setup>
  import { useRouter } from 'vue-router'
  import { onMounted } from 'vue'
  const router = useRouter()
  // 组件挂载后，延迟2秒，自动从 发现音乐页 跳转至 朋友页
  onMounted(() => {
    setTimeout(() => {
      router.push('/friend')

      router.push({
         path: '/friend'
      })

      router.push({
        name: 'Friend'
      })
    }, 2000)
  })
</script>
```


1.5 总结

1. 编程式导航如何做路由跳转?

   答： `router.push(字符串/对象)`

2. 编程式导航传参

2.1 问题

编程式导航如何传参呢?

2.2 语法

与`声明式导航<router-link>`传参和接参方式完全一样，既支持字符串, 也支持对象

```
router.push('/path?参数名=值')

router.push({
  path: '/path',
  query: {
    参数名: 值
    ...
  }
})

router.push({
  name: '路由名称',
  params: {
    参数名: 值
  }
})
```

2.3 代码示例

2.3.1 查询参数

router/index.js

```
createRouter({
  routes: [
    { path: '/friend', component: Friend }
  ]
})
```

Find.vue

```
router.push('/friend?fid=110')

```

```
router.push({
  path: '/friend',
  query: {
    fid: 101
  }
})
```

Friend.vue

```
<script setup>
  import { useRoute } from 'vue-router'

  const route = useRoute()
  console.log(route.query.fid)
</script>
```

2.3.2 动态参数

router/index.js

```
createRouter({
  routes: [{
    // 命名路由
    name: 'Friend',
    path: '/friend/:id',
    component: Friend
  }]
})
```

Find.vue

```
router.push('/friend/110')

router.push({
  name: 'Friend'
  params: {
    id: 101
  }
})
```

Friend.vue

```vue
<script setup>
  import { useRoute } from 'vue-router'
  const route = useRoute()
  console.log(route.params.id)
</script>
```

2.4 总结

1. 编程式导航如何传参和接参?

   答：同声明式导航完全一样，`router.push(字符串/对象)`

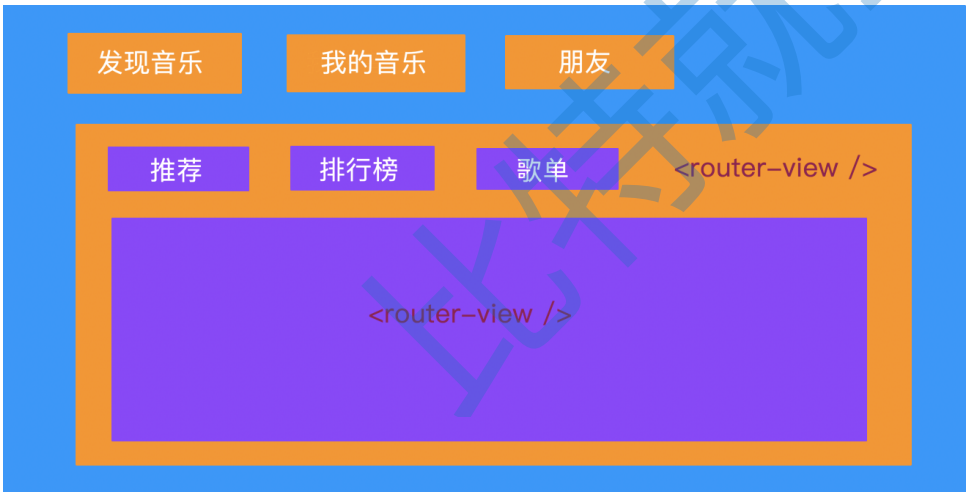
六、嵌套与守卫

1. 嵌套

1.1 问题

能否在一个路由页面中，再进行一套路由的切换呢？比如 网易云音乐 的发现音乐页

1.2 效果图



1.3 步骤

1. 创建3个二级路由页面组件(Recommend、Ranking、SongList)
2. 路由表数组中, 在相应的一级路由规则中, 配置 `children` 规则
3. 在相应的一级路由页面组件中, 给二级路由 `<router-view/>` 出口和导航链接

注意：1、 二级路由的 path 推荐 `不加 /`

2、 做路由跳转的时候要写 `完整路径(父路径+当前路径)`

3、 防止二级路由页面空白, 给默认显示的一级路由 `添加重定向`

1.4 代码示例

views/Recommend.vue

```vue
<script setup></script>

<template>
  <div class="recommend">
    <h5>推荐</h5>
    <h5>推荐</h5>
    <h5>推荐</h5>
  </div>
</template>

<style scoped></style>
```

views/Ranking.vue

```vue
<script setup></script>

<template>
  <div class="ranking">
    <h5>排行榜</h5>
    <h5>排行榜</h5>
    <h5>排行榜</h5>
  </div>
</template>

<style scoped></style>
```

views/SongList.vue

```vue
<script setup></script>

<template>
  <div class="songlist">
    <h5>歌单</h5>
    <h5>歌单</h5>
    <h5>歌单</h5>
  </div>
</template>

<style  scoped></style>
```

router/index.js

```
import Recommend from '@/views/Recommend.vue'
import Ranking from '@/views/Ranking.vue'
import SongList from '@/views/SongList.vue'

const router = createRouter({
  history: createWebHashHistory(),
  routes: [{
    path: '/find',
    component: Find,
    // 重定向，防止二级路由空白
    redirect: '/find/recommend',
    // 配置嵌套路由
    children: [{
      // 二级路由开始，推荐 path 不加 /
      path: 'recommend',
      component: Recommend
    },{
      path: 'ranking',
      component: Ranking
    },{
      path: 'songlist',
      component: SongList
    }]
  }]
})
```

views/Find.vue

```
<script setup></script>

<template>
  <div class="find">
    <nav>
      <!-- 二级路由导航链接 -->
      <router-link to="/find/recommend">推荐</router-link>
      <router-link to="/find/ranking">排行榜</router-link>
      <router-link to="/find/songlist">歌单</router-link>
    </nav>
    <!-- 二级路由出口 -->
    <router-view />
  </div>
```

```
</template>

<style scoped></style>

```

1.5 总结

1. 如何配置路由嵌套?

   答：配 `children` ，给路由 `出口<router-view/>`

2. 需要注意什么?

   答：二级 path 不加 /；跳转要写完成路径; 避免空白、要添加重定向

2. 路由守卫

2.1 问题

能否在访问某个路由前, 添加权限判断? 比如 我的音乐页, 只有登录了才可以访问

2.2 解决方案

路由(导航)全局前置守卫: 每个路由在跳转前都会触发回调函数

```
router.beforeEach((to, from) => {
  // to: 即将要进入的路由
  // from: 当前正要离开的路由

  // false 取消导航(不发生跳转)
  return false
  // 正常放行、正常跳转
  return undefined | true
  // 重定向到指定的路由
  return '/路径'
})
```

2.3 代码示例

```
// 用来模拟是否登录
const isLogin = true

router.beforeEach((to, from) => {
  // 如果没有登录，并且还要去 我的音乐页
  if(!isLogin && to.path === '/my') {
    // 进行提示
```

```
    alert('请先登录')
    // 不放行(不跳转)
    return false
  }
  // 正常放行
  return true
})
```

2.4 总结

1. 如何在访问路由前添加权限校验?

   答: `全局前置守卫 router.beforeEach((to, from) => { })`

2. 参数 to、from 和 回调函数返回值 都表示什么?

```
to: 即将进入的路由
from: 正要离开的路由
回调函数返回值:
  return false: 不放行
  return true/undefined: 放行
  return 路径: 重定向到指定路由
```