Vue–router && Vuex实战

课堂目标

1. vue-router基础配置
2. 路由传参
3. 子路由
4. 路由重定向
5. 路由守卫
6. vuex数据流
7. Store
8. state

9. mutation
10. action

知识要点

1. vue-router多页面
2. vuex管理数据

资源

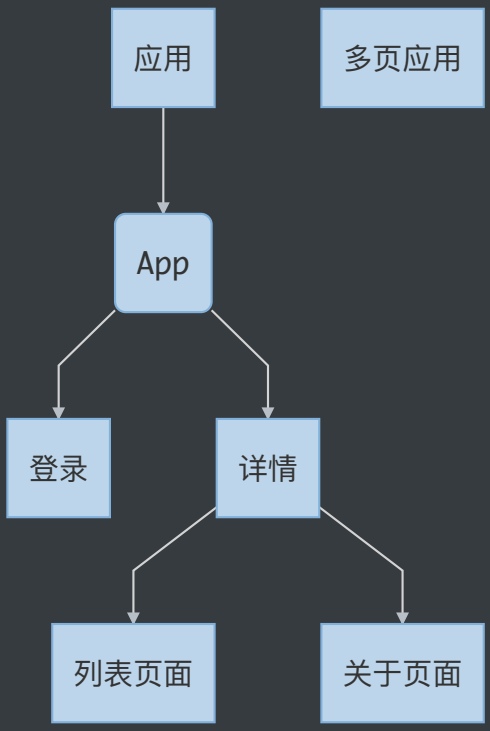
1. vue-router
2. vuex

vuex模块切分 modules 分模块(后续) getter可以理解为vuex里面的computed action 异步mutions

起步

Vue Router 是 Vue.js 官方的路由管理器。它和 Vue.js 的核心深度集成，让构建单页面应用变得易如反掌。包含的功能有：

1. 嵌套的路由/视图表
2. 模块化的、基于组件的路由配置
3. 路由参数
4. 基于 Vue.js 过渡系统的视图过渡效果
5. 细粒度的导航控制
6. 带有自动激活的 CSS class 的链接
7. HTML5 历史模式或 hash 模式



1. 新建项目 `vue create vue-router-vuex`
2. 安装vue-router `npm install vue-router --save`
3. `npm run serve`

多页面体验

新建 `routes.js`

新建两个测试组件

```
<template>
    <div>页面1</div>
</template>

<script>
export default {


}
</script>

```

```
12
13
14
15  <template>
16      <div>页面2</div>
17  </template>
18
19  <script>
20  export default {
21
22
23  }
24  </script>
25
26
27
28
```

```
1   import VueRouter from 'vue-router'
2   import Page1 from './components/Page1'
3   import Page2 from './components/Page2'
4
5   export default new VueRouter({
6       routes:[
7           {path:'/page1',component:Page1},
8           {path:'/page2',component:Page2},
9       ]
10  })
11
```

main.js

```
1
2   import Vue from 'vue'
3   import App from './App.vue'
4   import VueRouter from 'vue-router'
5   import router from './routes'
6   Vue.config.productionTip = false
7
8   Vue.use(VueRouter)
9   new Vue({
10    router,
11    render: h => h(App)
```

```
12  }).$mount('#app')
13
14
```

app.vue 使用router-view占位符

```
1   <template>
2     <div id="app">
3       <router-view></router-view>
4     </div>
5   </template>
6
7   <script>
8   // import HelloWorld from './components/HelloWorld.vue'
9
10  export default {
11    name: 'app',
12  }
13  </script>
14
15  <style>
16  </style>
17
18
```

访问 `http://localhost:8080/#/page1` 和 `http://localhost:8080/#/page2` 即可看到不同的组件内容

导航

使用App.vue中，使用router-link作为导航按钮

```
1
2       <div>
3         <router-link to="/page1">Page1</router-link>
4         <router-link to="/page2">Page2</router-link>
5
6       </div>
7
8
```

history模式

默认是hash模式，url使用#后面定位路由，对seo不利，设置history，就可以使用普通的url模式

```

// routes.js

export default new VueRouter({
    mode:"history",
    routes:[
        {path:'/page1',component:Page1},
        {path:'/page2',component:Page2},
    ]
})

```

url就变成了 `http://localhost:8080/page1`

动态路由

路由可以携带一些参数,使用this.$router获取

```
        {path:'/page3/:id',component:Page3}


```

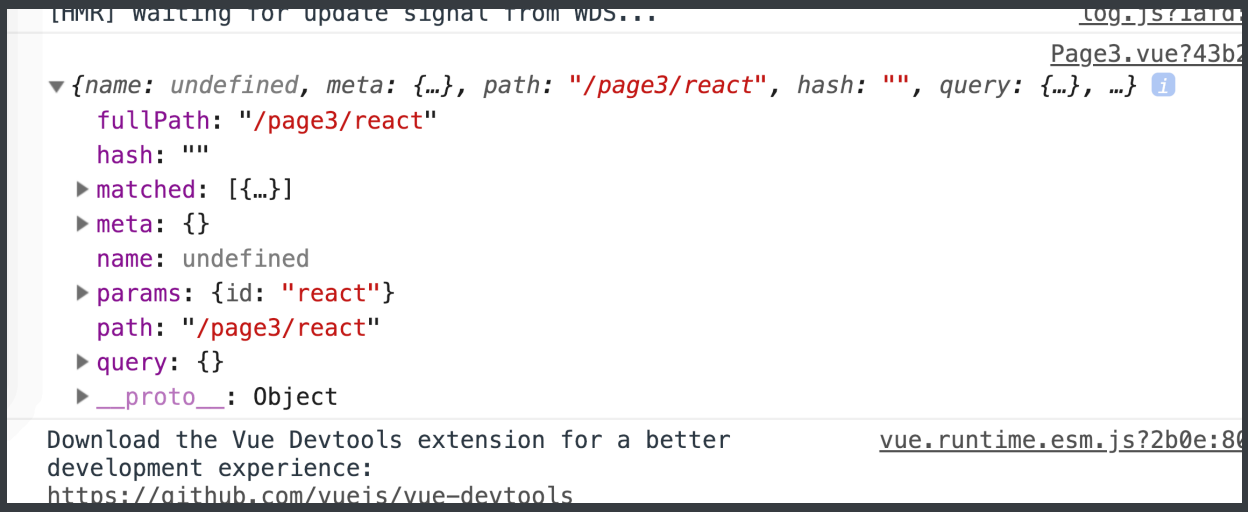
```

<template>
    <div>详情页面</div>
</template>

<script>
export default {
    created(){
        console.log(this.$route)
    }

}
</script>

```



单文件组件

.vue 是vue单文件组件，一个文件就是一个组件，由template，script和style三个标签构成，分别是html，js和css的内容部分,修改App.vue文件内容

```
<template>
    <div>
        <p>详情页面</p>
        <div>
            哈喽啊{{name}}
        </div>
    </div>
</template>

<script>
export default {
    created(){
        console.log(this.$route)
    },
    computed:{
        name(){
            return this.$route.params.id
        }
    }
}
</script>
```

```
26
```

参数属性传递

设置props属性，获取路由的变量 就和普通的属性传递没什么区别

```
1
2           { path: '/page3/:id', props: true, component: Page3 }
3
```

```
1  <template>
2      <div>
3          <p>详情页面</p>
4          <div>
5              哈喽啊{{id}}
6          </div>
7      </div>
8  </template>
9
10 <script>
11 export default {
12     created(){
13         console.log(this.$route)
14     },
15     props:['id']
16 }
17 </script>
18
19
```

路由嵌套

子路由的概念,比如页面内部的导航复用

```
1
2  export default new VueRouter({
3      mode: "history",
4      routes: [
5          {path:'/login',component:Login},
6          {
7              path:'/dashboard',
8              component:Dashboard,
```

```
            children:[
                { path: 'page1', component: Page1 },
                { path: 'page2', component: Page2 },
                { path: 'page3/:id', props: true, component: Page3 }
            ]
        },

    ]
})
```

app.vue

```
<template>
  <div id="app">
    <router-view></router-view>
    <hr>
    <div>开课吧还不错</div>
  </div>
</template>

<script>
// import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'app',
}
</script>

<style>
</style>

```

dashboard.vue

```
<template>
    <div>
        <div>
            <router-link to="/dashboard/page1">Page1</router-link>
            |
            <router-link to="/dashboard/page2">Page2</router-link>
            |
```

```
            <router-link to="/login">login</router-link>

        </div>
        <hr>

        <router-view></router-view>
    </div>
</template>

<script>
export default {

}
</script>


```

page1.vue

```

<template>
    <div>
        <p>页面1</p>
        <div>
            <router-link to='page3/react'>react</router-link>
            <br>
            <router-link to='page3/vue'>vue</router-link>
        </div>
    </div>
</template>

<script>
export default {


}
</script>


```

重定向

```
1   { path: '/', redirect: '/dashboard/page1' },
2
```

路由守卫

beforeEach 所有路由跳转前执行，next同意跳转 比如login执行2秒后跳转

```
1
2  routers.beforeEach((to, from, next) => {
3      console.log('beforeEach')
4      console.log(to)
5      if(to.path!='/login'){
6          next()
7      }else{
8          setTimeout(()=>{
9              next()
10         },2000)
11     }
12
13 })
14
15 routers.afterEach((to, from) => {
16     console.log('afterEach')
17 })
18
```

组件内部生命周期

Page3.vue

```
1
2  export default {
3    props: ["id"],
4    beforeRouteEnter(to, from, next) {
5      console.log("page3路由进入前");
6      next();
7    },
8    beforeRouteUpdate(to, from, next) {
9      console.log("page3路由，但是参数变了");
10     next();
11   },
12   beforeRouteLeave(to, from, next) {
```

```
13      console.log("page3路由离开前");
14      next();
15    }
16  };
17
```

1. 导航被触发。
2. 调用全局的 beforeEach 守卫。
3. 在重用的组件里调用 beforeRouteUpdate 守卫。
4. 在路由配置里调用 beforeEnter。
5. 在被激活的组件里调用 beforeRouteEnter。
6. 调用全局的 beforeResolve 守卫 (2.5+)。
7. 导航被确认。
8. 调用全局的 afterEach 钩子。
9. 触发 DOM 更新。

异步组件

路由懒加载 vue中配合webpack 非常简单

```
1          {
2              path:'/login',
3              component:()=>import('./components/Login')
4          }
5
```

Vuex数据管理

`npm install vuex --save` 安装

核心概念

- store
- state
- mutations

store

新建store.js

```

import Vuex from 'vuex'

export default new Vuex.Store({
    state:{
        count:0
    }
})

```

入口

```

import Vue from 'vue'
import App from './App.vue'
import VueRouter from 'vue-router'
import Vuex from 'vuex'
import router from './routes'
import store from './store'
Vue.config.productionTip = false

Vue.use(VueRouter)
Vue.use(Vuex)

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


```

Page1.vue

```
export default {
  computed: {
    count() {
      return this.$store.state.count;
    }
  }
};

```

Mutation

直接修改state的数据也可以，但是建议使用单向数据流的模式，使用Mutation来修改数据 直接改 VS 数据中心

```

    created(){
         setInterval(()=>{
             this.$store.state.count++
         },500)
     },
```

使用strict配置，可以禁用这种模式

```
(found in <Root>)
▶ Error: [vuex] Do not mutate vuex store state outside mutation handlers.    vue.runtime.esm.js?2b0e:1819
    at assert (vuex.esm.js?2f62:97)
    at Vue.store._vm.$watch.deep (vuex.esm.js?2f62:746)
    at Watcher.run (vue.runtime.esm.js?2b0e:3345)
    at Watcher.update (vue.runtime.esm.js?2b0e:3319)
    at Dep.notify (vue.runtime.esm.js?2b0e:712)
    at Object.reactiveSetter [as count] (vue.runtime.esm.js?2b0e:1037)
    at eval (Page1.vue?a479:19)
▶ [Vue warn]: Error in callback for watcher "function () {
return this._data.$$state }": "Error: [vuex] Do not mutate vuex store state outside mutation handlers."    vue.runtime.esm.js?2b0e:601

(found in <Root>)
▶ Error: [vuex] Do not mutate vuex store state outside mutation handlers.    vue.runtime.esm.js?2b0e:1819
    at assert (vuex.esm.js?2f62:97)
```

使用commit调用Mutation来修改数据

```

import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)
```

```
6
7  export default new Vuex.Store({
8      state: {
9          count: 0
10     },
11     mutations: {
12         increment(state) {
13             state.count++
14         }
15     },
16     strict: true
17 })
18
```

```
1
2  export default {
3      created(){
4          setInterval(()=>{
5              this.$store.commit('increment')
6          },500)
7      },
8    computed: {
9      count() {
10       return this.$store.state.count;
11     }
12   }
13 };
14
15
```

getters

有时候我们需要从 store 中的 state 中派生出一些状态，我们可以理解为vuex中数据的computed功能

store.js

```
1
2      getters:{
3          money: state => `￥${state.count*1000}`
4      },
5
6
```

page1.vue

```
  computed: {
    money() {
      return this.$store.getters.money;
    }
  }

```

Action

Mutation必须是同步的，Action是异步的Mutation

store.js

```
    actions: {
        incrementAsync({ commit }) {
            setTimeout(() => {
                commit('increment')
            }, 1000)
        }
    },

    this.$store.dispatch('incrementAsync')


```

传递参数

```
this.$store.dispatch('incrementAsync', {
    amount: 10
})

     actions: {
        incrementAsync(store,args) {

            setTimeout(() => {
                store.commit('incrementNum',args)
            }, 1000)
        }
    },
```

```
13
14      incrementNum(state,args) {
15          state.count += args.amount
16      }
17
18
19
```

mapState

更方便的使用api，当一个组件需要获取多个状态时候，将这些状态都声明为计算属性会有些重复和冗余。为了解决这个问题，我们可以使用 mapState 辅助函数帮助我们生成计算属性，让你少按几次键

```
1
2       ...mapState({
3           count:state=>state.count
4       }),
```

mapActions

方便快捷的使用action

```
1
2       methods:{
3           ...mapActions(['incrementAsync']),
4           ...mapMutations(['increment']),
5       },
6
7
8
```

this.$store.dispatch可以变为

```
        this.incrementAsync({
            amount: 10
        })

```

mapMutions

同理可得

```
  ...mapMutations(['increment'])

  this.increment()
```

Vue-router && Vuex实战

- 课堂目标
- 知识要点
- 资源
- 起步
- 多页面体验
- 导航
- history模式
- 动态路由
    - 单文件组件
- 参数属性传递
- 路由嵌套
- 重定向
- 路由守卫
- 组件内部生命周期
- 异步组件
- Vuex数据管理
- store
- Mutation
- getters
- Action
- mapState
- mapActions
- mapMutions
- 回顾

回顾