

自己动手打造一款React路由守卫

引言

用过vue的小伙伴都知道，vue自带路由守卫钩子并且巨他妈的好用，而对于react开发者来说，在需要路由权限校验时常常存在许多痛点问题。今天我将为大家打造一款属于我们reacter的路由守卫方法，希望可以为大家提供帮助。

react路由

大家先不要急，我们先温习下react基本的路由搭建过程。由于react路由统一管理不唯一，此处列举的是基于useRoutes的路由管理。

1. 下载安装

```
npm install react-router-dom@6
```

css 复制代码

2. index.js挂载

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { BrowserRouter, HashRouter } from 'react-router-dom';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(

  <BrowserRouter>
    <App />
  </BrowserRouter>

);
```

javascript 复制代码

3. 定义路由数组

/router/index.js

```
import Page1 from '../views/Page1';
import Page2 from '../views/Page2';
```

javascript 复制代码

```
import Login from "../../views/Login";
import NotFound from "../../views/404";
export default [
  {
    path: '/',
    element:<Page1/>,
    auth:false
  },
  {
    path: '/page2',
    element:<Page2/>,
    auth:true
  },
  {
    path: '/login',
    element:<Login/>
  },
  {
    path: '/404',
    element:<NotFound/>
  }
]
```

4.App.js通过useRoutes统一管理路由

```
import {useRoutes} from "react-router-dom"
import router from "./router/index";
function App() {
  return useRoutes(router);
}
export default App;
```

javascript 复制代码

经过上面四步，react简单的路由就搭建成功了，大家就轻松可以完成路由页面的切换。

| 路由守卫

终于要开车了，大家准备好安全带。经过上面简单的介绍，我们已经知道了如何搭建简单的路由管理，那么如何基于上面的知识点完成路由守卫？嗯不卖关子了，请继续往下看。

1.路由数组保持不变，同上面的 /router/index.js一样

2.创建函数路由组件，模拟路由守卫 /router/beforeEnter.js

scss 复制代码

```
import {useLocation,useNavigate,useRoutes} from "react-router-dom";
import {useState,useEffect} from "react";
const BeforeEnter = ({routers}) => {

  //1.在路由数组中找当前页面路由的对应路由项
  const fineRouter = (routes,path) => {
    for(let val of routers) {
      if(val.path===path) return val;
      if(val.children) return fineRouter(val.children,path);
    }
    return null;
  }

  //2.路由守卫判断
  const judgeRouter = (location,navigate) => {
    const {pathname} = location;
    //2.1路由数组找路由项
    const findRoute = fineRouter(routers,pathname);
    //2.2没找到，说明没有这个路由，直接404
    if(!findRoute) {
      navigate("/404");
      return ;
    }
    //2.3路由项如果有权限需求，进行逻辑验证
    if(findRoute.auth) {
      //用户未登陆，挑战登陆页面
      if(!localStorage.getItem("user")) navigate("/login");
    }
  }

  //3.基于useEffect监听页面路由改变。然后组件重新加载，又重新校验权限。
  const navigate = useNavigate();
  const location = useLocation();
  const router = useRoutes(routers);
  useEffect(()=>{
    //路由守卫判断
    judgeRouter(location,navigate)
  },[navigate,location])
  return router;
}
export default BeforeEnter;
```

3.App.js

App.js只需要简单的几行代码就完事了。

```
import {useRoutes} from "react-router-dom"
import BeforeEnter from "../router/beforeEnter";
import router from "../router/index";
function App() {
  return <BeforeEnter routers={router}/>
}
export default App;
```

javascript 复制代码

4.效果展示

为了加深效果，我们还是基于上面的路由组件进行讲解。我逐一列举路由各个页面的功能。在路由数组中我们约定，如果路由项添加了auth:true表示该路由需要进行权限校验。此处我们就是检验是否登陆。

Page1.js

根据路由数组，我们很清楚，这个组件是首页功能不需要权限校验

```
export default () => {
  return <div>首页</div>
}
```

javascript 复制代码

Page2.js

根据路由数组，我们很清楚，这个组件是需要权限认证

```
export default () => {
  return <div>page2</div>
}
```

javascript 复制代码

404.js

根据路由数组，这个组件就是当路由没有匹配到时，跳到404页面

javascript 复制代码

```
export default () => {  
  return <div>404</div>  
}
```

login.js

根据路由数组，这个组件就是模拟登陆页面，如果某个路由需要校验权限，并且权限失败时，就跳转到登陆页面。其次我们在登陆页面模拟登陆效果，登陆点击后，自动跳转到首页。

javascript 复制代码

```
import {useNavigate} from "react-router-dom";  
  
export default () => {  
  
  const nav = useNavigate();  
  return <div>  
    login页面  
    <div onClick={()=>{  
      localStorage.setItem("user","dzp");  
      setTimeout(() => {  
        nav('/');  
      }, 1000);  
    }}>  
    点击登陆  
  </div>  
</div>  
  
}
```

完整效果展示



[markdown](#) [复制代码](#)

1. 初始没有登陆，我们直接来到首页，正常
2. 我们访问page1, 正常
3. 我们随意访问page99. 自动跳转404页面
4. 我们访问page2, page2需要权限认证。由于未登陆，自动跳转登陆页面，然后我们点击登陆挑战到首页，接着我们继续访