React后台管理系统
金渡教育VIP内部资料
请勿外传 违者必究
主讲老师：Casey

一、项目搭建

1、yarn init -y #初始化项目
2、yarn add -D create-react-app #使用本地安装
3、npx create-react-app --version #查看脚手架版本
4、npx create-react-app react-project #创建项目
5、配置项目相关环境：
React路由，从4.0开始,路由就不再集中在一个包管理器中，需要我们手动安装
yarn add -D react-router-dom

二、安装CSS预处理器SCSS

在老版本里面我们如果想要使用Scss语法，需要把webpack配置暴露出来去做对应的修改才行，但是现在的React版本不需要我们去处理webpack配置就能够添加和使用Scss语法。
直接安装SCSS就可以了。把文件名改为.scss就可以直接使用
cnpm add node-sass@5 -D

三、UI框架：ANTD

Antd是蚂蚁金服开源的一款react-ui组件库,兼容性很不错,常用的浏览器以及IE11都兼容。(兼容IE11需要polyfiles,它是一个JS库,主要作用就是解决兼容问题,解决不同浏览器之间对JS实现的一个差异),它还支持服务端渲染(SSR)
cnpm i antd -D

在index.js中引入使用
import 'antd/dist/antd.css'

使用方法：
import {Button} from 'antd'
<Button>按钮</Button>

四、组件化和模块化

我们设计结构的时候需要根据页面来拆分为不同的组件,组件是构成页面独立的功能模块。我们要尽可能的将我们的项目进行组件化、模块化。
组件比较侧重于UI层面，模块化比较侧重功能层面。

五、Axios二次封装

yarn add -D axios

2、使用，在项目开发过程中我们都会对axios进行二次封装，方便我们进行开发。

二次封装，新建一个axios文件夹来进行二次封装的模块处理

config.js:
```
// axios的配置文件, 可以在这里去区分开发环境和生产环境等全局一些配置
const devBaseUrl = 'http://api.k780.com/'
const proBaseUrl = 'http://xxxxx.com/'

// process.env返回的是一个包含用户的环境信息,它可以去区分是开发环境还是生产环境
console.log(process.env)
export const BASE_URL =  process.env.NODE_ENV === 'development' ? devBaseUrl : proBaseUrl

export const TIMEOUT = 5000
```

request.js:
```
// 封装后的axios
import axios from 'axios'
import { BASE_URL, TIMEOUT } from './config'

const instance = axios.create({
    baseURL: BASE_URL,
    timeout: TIMEOUT
})

// 请求拦截器 在发起http请求之前的一些操作
// 1、发送请求之前，加载一些组件
// 2、某些请求需要携带token，如果说没有没有携带，直接跳转到登录页面
instance.interceptors.request.use((
config) => {
    console.log('被拦截做的一些操作')
    return config
}, err => {
    return err
})

// 响应拦截器
instance.interceptors.response.use((res) => {
    return res
}, err => {
    if (err && err.response) {
        switch(err.response.status) {
            case 400:
                console.log('请求错误')
                break
            case 401:
                console.log('未认证')
                break
            default:
                console.log('其他信息错误')
        }
    }
})

export default instance
```

六、路由使用

路由的使用

1、安装路由：yarn add -D react-router-dom

BrowserRouter：使用是history模式

```
import { BrowserRouter, Link, Route } from '
react-router-dom'
<BrowserRouter>
        <ul>
          <li><Link to='/'>首页</Link></li>
          <li><Link to='/about'>关于</Link></li>
          <li><Link to='/user'>会员中心</
Link></li>
        </ul>
        {/* exact是精确匹配 */}
        <Route path='/' exact component={
Home} />
        <Route path='/about' component={
About} />
        <Route path='/user' component={
User} />
      </BrowserRouter>
```

HashRouter：使用的是hash模式

```
import { HashRouter, Link, Route } from '
react-router-dom'
<HashRouter>
      <ul>
        <li><Link to='/'>首页</Link></li>
        <li><Link to='/about'>关于</Link></li>
        <li><Link to='/user'>会员中心</
Link></li>
      </ul>
      {/* exact是精确匹配 */}
      <Route path='/' exact component={
Home} />
      <Route path='/about' component={
About} />
      <Route path='/user' component={
User} />
    </HashRouter>
```

还可以给路由起别名：import { BrowserRouter as Router, Link, Route } from 'react-router-dom'

Redirect：路由的重定向,当这个组件出现的时候,就会执行到对应的to路径中

路由嵌套：嵌套：就是在子页面中再设置一层新的路由导航规则(/color/red)

Switch：做精确匹配，匹配到一个就会停止向下匹配。'/about'的link里的to包含了'/','/'就会被匹配到,Switch的作用是为了解决route的唯一渲染,它只是为了保证路由只渲染一个路径。

手动跳转路由：
1、如果这个组件是通过路由跳转来的，那么就拥有路由信息
2、如果这个组件是一个普通渲染的组件,那么是不可以获取到路由信息，如果需要获取到路由信息，就需要用到高阶组件withRouter

Route：
1、Route是用于路径的哦匹配(用于定义组价和路径的映射关系，相当于一个占位符,在哪里占位就在哪里渲染)
2、path属性:用于上设置匹配到的路径
3、component:用了关于匹配带路径渲染组件
4、exact:精确匹配,会匹配到所有能匹配到的路由组件，它能让路由匹配更严格些

路由传参：
动态路由：路由中路径并不固定,那么可以在path上写/user/:id，这种时候使用this.props.match.params

如果是比较复杂的数据,比如'/user?name=casey&age=18'这种时候需要使用location中的search获取,并且需要下载安装query-string进行处理,这个方法React已经不推荐
cnpm i -D query-string

```
<Link to={
                {
                    pathname: '/
userdetail', // 跳转的路径
                    state: {name: 'casey',
age: 18} // 传递的数据

                }
            }>用户详情</Link>
```

路由的集中管理：cnpm i -D react-router-config

七、在项目中集中路由管理

1、左侧菜单导航栏加上对应的路由

cnpm i -D react-router-dom

去nav组件加上对应的Link
```
import  {Link} from 'react-router-dom'

<Link  to={item.key}>{item.title}</Link>
```

此时会报错，需要去最外层加Route
在项目入口文件Index.js里
```
import { BrowserRouter } from 'react-router-
dom'
<BrowserRouter>
    <App />
  </BrowserRouter>
```

2、路由的集中处理

cnpm i -D react-router-config
新建router文件夹，去写全局的路由配置文件内容

然后去app.js中
```
import { renderRoutes } from 'react-router-
config' // 引入渲染路由的方法
import routes from './router/' // 引入全局配置的路由规则

{renderRoutes(routes)}
```

3、使用ANTD的图标

使用图标必须要先安装
cnpm i -D @ant-design/icons

```
<Button icon={<PlusOutlined/>}>添加</
Button>
```

4、处理UI模块里的子路由

index.js里面最核心的处理是用renderRoutes去渲染子路由

1、在UI文件下新建了一个Index.js文件，来统一管理整个UI里面的其他子页面,然后去建议对应的子页面内容

2、需要去router里面修改对应的路由信息

```
{
    path: '/ui',
    component: UI,
    routes: [
      {
        path: '/ui/buttons',
        component: Button
      },
      {
        path: '/ui/modals',
        component: Modals
      },
      {
        path: '/ui/messages',
        component: Messages
      },
      {
        path: '/ui/carousel',
        component: Carousel
      },
    ]
  }
```

八、antd样式按需引入

目前项目中用是全局引入形式,会导致打包体积变大,所以我们用按需引入

这个时候我们需要对create-react-app的默认配置进行定义,所以需要使用craco

1、安装：yarn add @craco/craco
2、修改package.json配置
```
"scripts": {
    "start": "craco start",
    "build": "craco build",
    "test": "craco test"
  }
```

3、创建一个craco.config.js,用于修改默认配置
4、按需引入需要用到babel-plugin-import,这里需要下载一下：cnpm i -D babel-plugin-import
5、修改craco.config.js中的配置内容：
```
module.exports = {
    babel: {
      plugins: [
        [
          "import",
          {
            "libraryName": "antd",
            "libraryDirectory": "es",
            "style": "css" // 设置为true默认是less
          }
        ]
      ]
    }
```
6、项目核心入口index.js页面中的全局引入antd的样式代码注释掉，重启项目即可
}

九、表格的使用

对于列很多的数据，我们可以去固定前后的列，横向滚动查看其他数据,需要和scroll.x配合使用
```
<Table dataSource={data} columns={
columns} scroll={{x: 1000}} />
```

需要对表格写对应的操作按钮的时候
```
{
        title: '操作',
        key: 'operaion',
        // text:row的值
        render: (text, record) => {
            return <Button onClick={() =>
  removeItem(text, record.key)}>删除</Button>
        }
    }
```

表格的分页功能
```
<Table
        dataSource={state}
        columns={columns}
        pagination={{pageSize: 20}}
        scroll={{ y: 260 }} />
```

函数式组件的话要记得用useEffect请求数据
```
  useEffect(() => {
    // 请求接口
    const fetchData = async () => {
      const { data } = await axios.get('
http://localhost:3001/user')
      console.log(data)
      const { user } = {...data}
      setData(user)
    }
    fetchData()
  }, []) // 传入一个空数组让它只渲染一次
```

类组件中要在生命周期中请求数据
```
    async componentDidMount() {
        const { data } = await axios.get('http://
  localhost:3001/user')
        this.setState({data: data.user})
    }
```

后端接口：拿到moke项目之后，npm i安装依赖包，node index.js把后端接口跑起来

十、Redux

npm i redux

使用步骤：
1、创建一对象,作为状态state
2、创建store存储state
3、通过action修改state
4、使用reducer连接state和action
5、派发action,派发之前还可以监听store变化

三大原则：
1、单一数据源
整个应用state都是被存储在一棵object tree中,这个object tree只存储在一个store中,单一数据源可以让应用中state变得很方便去维护、追踪、修改
2、state是只读的
唯一修改state的方法是：触发action，不用视图在其他的地方通过其他的方式来修改state，可以保证所有的修改都被集中处理，并且严格按照顺序来执行
3、使用纯函数来执行修改
通过reducer将state和action联系在一起,并且返回一个新的state

什么时候用redux：业务场景需要涉及到不同层次的组件进行组件通信的话就是我们使用redux的绝佳时机

redux工作流：组件想要获取state,用ActionCreate创建一个请求交给store,Store借助reducer确认该state的状态,reducer会返回一个结果,store再把这个state转给组件。

中间件：需要异步处理的时候,就需要用到中间件。所谓的中间件，其实就是一个函数,函数的作用是增强或者扩展功能,可以在我们的请求和响应之间做一些操作，redux推荐的网络请求中间件是redux-thunk,这个中间件的目的就是在dispatch和action最终到大reducer之间扩展一些代码。

react-redux：
1、cnpm i -D react-redux
2、去Index.js页面,用Provider包裹根组件，这样就能让所有的子组件都能拿到store中的数据

十一、路由懒加载

简单来说，路由懒加载就是延迟加载或者是按需加载，react也是可以直接进行处理的,在router里进行处理就可以.

```
1、outer文件夹下Index.js

const Home = lazy(() => import('../pages/
home'))

2、在项目根index.js中
<Suspense fallback={<div>Loading.....</
div>}>
      <App />
    </Suspense>
```