

- React基础
- 组件化开发
- 全家桶
- 项目实战开发

## 项目描述

---

1. 此项目为一个前后台分离的后台管理的SPA, 包括前端PC应用和后端应用
2. 包括用户管理 / 商品分类管理 / 商品管理 / 权限管理等功能模块
3. 前端: 使用React全家桶 + Antd + Axios + ES6+ + Webpack等技术
4. 后端: 使用Node + Express + Mongodb等技术
5. 采用模块化、组件化、工程化的模式开发

## 收获

---

### 流程&开发方式

1. 熟悉一个项目的开发流程
2. 学会模块化、组件化、工程化的开发模式
3. 掌握使用**create-react-app**脚手架初始化react项目开发
4. 了解使用**node+express+mongoose+mongodb**搭建后台应用

### React插件或第三方库

1. 掌握使用**react-router-dom**开发单页应用
2. 学会使用**redux+react-redux+redux-thunk**管理应用组件状态
3. 掌握使用**axios/jsonp**与后端进行数据交互
4. 掌握使用**antd**组件库构建界面
5. 学会使用**echarts/echarts-for-react**实现数据可视化展现

## 6. 学会使用wangeditor实现富文本编辑器

# npm/yarn常用命令

[yarn命令文档](#)

[npm命令文档](#)

## 项目初始化

---

1. create-react-app是react官方提供的用于搭建基于react+webpack+es6项目的脚手架

```
1 npm install -g create-react-app : 全局下载工具
2 create-react-app react-admin : 下载模板项目
3 cd react-admin
4 npm start
```

## 引入路由

---

```
1 yarn add react-router-dom
```

新建pages/Login/Login.js

```

1 import React, { Component } from 'react'
2 class Login extends Component {
3   render() {
4     return (
5       <div className='login'>
6         Login组件
7       </div>
8     )
9   }
10 }
11
12 export default Login;

```

pages/Admin/Admin.jsx

```

1 import React, { Component } from 'react'
2 export default class Admin extends Component {
3   render() {
4     return (
5       <div>
6         Admin组件
7       </div>
8     )
9   }
10 }
11

```

App.js

```

1 import React, { Component } from 'react'
2 import { Button, message } from "antd";
3 import { BrowserRouter, HashRouter, Switch, Route
4   ,Redirect} from "react-router-dom";
5 import Login from './pages/Login/Login';

```

```

5 import Admin from './pages/Admin/Admin';
6 export default class App extends Component {
7   render() {
8     return (
9       <BrowserRouter>
10        <Switch>
11          <Route path='/login' component={Login} />
12          <Route path='/admin' component={Admin} />
13        </Switch>
14      </BrowserRouter>
15    )
16  }
17 }
18

```

重置样式&引入静态资源

**重置样式 reset.css**

public/style/reset.css

```

1  /*! minireset.css v0.0.6 | MIT License |
   github.com/jgthms/minireset.css */
2  html,
3  body,
4  p,
5  ol,
6  ul,
7  li,
8  dl,
9  dt,
10 dd,
11 blockquote,
12 figure,
13 fieldset,

```

```
14 legend,
15 textarea,
16 pre,
17 iframe,
18 hr,
19 h1,
20 h2,
21 h3,
22 h4,
23 h5,
24 h6 {
25     margin: 0;
26     padding: 0;
27 }
28
29 h1,
30 h2,
31 h3,
32 h4,
33 h5,
34 h6 {
35     font-size: 100%;
36     font-weight: normal;
37 }
38
39 ul {
40     list-style: none;
41 }
42
43 button,
44 input,
45 select,
46 textarea {
47     margin: 0;
48 }
```

```
49
50 html {
51     box-sizing: border-box;
52 }
53
54 *,
55 *::before,
56 *::after {
57     box-sizing: inherit;
58 }
59
60 img,
61 video {
62     height: auto;
63     max-width: 100%;
64 }
65
66 iframe {
67     border: 0;
68 }
69
70 table {
71     border-collapse: collapse;
72     border-spacing: 0;
73 }
74
75 td,
76 th {
77     padding: 0;
78 }
79
80 td:not([align]),
81 th:not([align]) {
82     text-align: left;
83 }
```

```
84
85 html,body{
86     height: 100%;
87     width: 100%;
88 }
89 #root{
90     width: 100%;
91     height: 100%;
92 }
```

注意：必须在index.html中通过link标签引入

## Login组件(无交互)

在src目录下，新建assets/images文件夹存放图片资源

新建Login/Login.less

```
1  .login {
2      width: 100%;
3      height: 100%;
4      background-image: url('../assets/images/bg.jpg');
5
6      .login-header {
7          height: 80px;
8          background-color: rgba(0, 0, 0, 0.5);
9          display: flex;
10         align-items: center;
11
12         img {
13             width: 40px;
14             height: 40px;
15             margin-left: 30px;
16             margin-right: 20px;
```

```

17     }
18
19     h1 {
20         font-size: 30px;
21         color: #fff;
22         margin-bottom: 0;
23     }
24 }
25
26 .login-content {
27     width: 400px;
28     height: 300px;
29     background-color: #fff;
30     margin: 100px auto;
31     padding: 30px;
32     border-radius: 3px;
33     h1 {
34         font-size: 24px;
35         font-weight: bold;
36         text-align: center;
37         color: #000;
38     }
39 }
40 }

```

注意：使用less前，先安装less和less-loader插件：`yarn add less less-loader`

## Login/Login.jsx

```

1 import React, { Component } from 'react'
2 import { Form, Icon, Input, Button } from 'antd';
3 import './Login.less'
4 import logo from '../assets/images/logo.svg'
5 class Login extends Component {

```



```
6   handleSubmit = e => {
7     e.preventDefault();
8     this.props.form.validateFields((err, values) => {
9       if (!err) {
10         // console.log('Received values of form: ',
values);
11         const { username, password } = values;
12       }
13     });
14   };
15   /**
16    * 自定义校验规则
17    */
18   validatorPwd = (rule, value, callback)=>{
19     value = value.trim();
20     if(!value){
21       callback('密码必须输入')
22     }else if(value.length < 4){
23       callback('密码不能小于4位')
24     } else if (value.length > 12) {
25       callback('密码不能大于12位')
26     }else if(!/^[a-zA-Z0-9_]+$/.test(value)){
27       callback('密码必须是英文、数字或下划线组成')
28     }else{
29       // 验证通过一定要callback,如果callback,验证无法响应
30       callback();
31     }
32   }
33   render() {
34     const { getFieldDecorator } = this.props.form;
35     return (
36       <div className='login'>
37         <div className='login-header'>
38           <img src={logo} alt="" />
39           <h1>React后台管理系统</h1>
```

```
40         </div>
41         <div className="login-content">
42             <h1>用户登录</h1>
43             <Form onSubmit={this.handleSubmit}
44             className="login-form">
45                 <Form.Item>
46                     {getFieldDecorator('username', {
47                         // 设置初始值
48                         initialValue: 'admin',
49                         // 声明式校验规则
50                         rules: [
51                             { required: true, whitespace: true,
52                             message: '用户名是必须的!' },
53                             { min: 4, message: '密码不能小于4位!' },
54                             { max: 12, message: '密码不能大于12
55                             位!' },
56                             { pattern: /^[a-zA-Z0-9_]+$/,
57                             message: '必须是英文、数字或下划线组成!' },
58                         ],
59                     })(
60                         <Input
61                             prefix={<Icon type="user" style={{
62                             color: 'rgba(0,0,0,.25)' }} />}
63                             placeholder="用户名"
64                         />,
65                     )}
66                 </Form.Item>
67                 <Form.Item>
68                     {getFieldDecorator('password', {
69                         rules: [{ validator: this.validatorPwd
70                         }],
71                     })(
72                         <Input
```

```

67         prefix={<Icon type="lock" style={{
color: 'rgba(0,0,0,.25)' }} />}
68         type="password"
69         placeholder="密码"
70     />,
71     )}
72     </Form.Item>
73     <Form.Item>
74         <Button type="primary" htmlType="submit"
className="login-form-button" style={{ width: '100%'
}}>
75             登录
76         </Button>
77     </Form.Item>
78 </Form>
79 </div>
80 </div>
81 )
82 }
83 }
84
85 export default Form.create({ name: 'normal_login' })
(Login);;

```

## Login组件(交互)

### 封装ajax请求模块

下载axios库

```
1 yarn add axios
```

新建api/ajax.js

```
1  /*
2    封装的能发ajax请求的函数
3  */
4  import axios from 'axios'
5  import qs from 'qs'
6  import { message } from "antd";
7  /**
8   * 请求拦截器 在发送请求数据之前，提前对数据做处理
9   */
10 axios.interceptors.request.use(function (config) {
11     const { method, data } = config;
12     if (method.toLocaleLowerCase() === 'post' && typeof
13 data === 'object') {
14         config.data = qs.stringify(data);
15     }
16     return config;
17 }, function (error) {
18     // Do something with request error
19     return Promise.reject(error);
20 });
21 /*
22 响应拦截器，响应数据之前做工作
23 */
24 axios.interceptors.response.use(function (response) {
25     return response.data;
26 }, function (error) {
27     message.error('请求失败了:'+error.message);
28     // return Promise.reject(error);
29     // 让错误处于pending状态，不再往下执行，
30     // 返回一个pending的promise，中断promise链
31     return new Promise(()=>{})
32 });
33 export default axios;
```

新建api/index.js

```
1  /*
2  包含应用中所有请求接口的函数：接口请求函数
3  */
4  import ajax from './ajax'
5  const BASE = '';
6  // 请求登录
7  export const reqLogin = (username, password) => {
8      return ajax({
9          method: 'post',
10         url: BASE + '/login',
11         data: {
12             // 默认使用json格式的请求体携带参数数据
13             username,
14             password
15         }
16     })
17 }
18
```

## 配置代理

package.json

```
1  "proxy": "http://localhost:5000"
```

## 登录功能

Login/Login.js

```
1  import React, { Component } from 'react'
```

```
2 import { Form, Icon, Input, Button, message } from
  'antd';
3 import { Redirect } from "react-router-dom";
4 import './Login.less'
5 import logo from '../assets/images/logo.svg'
6 import { reqLogin } from '../api';
7 class Login extends Component {
8   handleSubmit = e => {
9     e.preventDefault();
10    this.props.form.validateFields(async (err, values)
=> {
11      if (!err) {
12
13        const { username, password } = values;
14        const res = await reqLogin(username, password);
15        if (res.status === 0) {
16          // 将用户信息保存到local中
17          const user = res.data;
18          // 保存用户信息
19          localStorage.setItem('user_key',
JSON.stringify(user))
20          // 跳转到管理页面
21          this.props.history.replace('/')
22          message.success('登录成功')
23        } else {
24          message.error(res.msg);
25        }
26      } else {
27
28      }
29    });
30  };
31  /**
32   * 自定义校验规则
33   */
```

```

34   validatorPwd = (rule, value, callback) => {
35     ....
36   }
37   render() {
38     // 读取保存的user 如果存在, 直接跳转到管理界面
39     const user =
JSON.parse(localStorage.getItem('user_key')) || {};
40     if (user._id) {
41       // render中不能这样写 这种方法很一般在事件回调函数中
进行路由跳转
42       // this.props.history.replace('/login')
43       return <Redirect to="/" />
44     }
45     const { getFieldDecorator } = this.props.form;
46     return (
47       <div className='login'>
48         ...
49       </div>
50     )
51   }
52 }
53
54 export default Form.create({ name: 'normal_login' })(Login);

```

## Admin/Admin.js

```

1  import React, { Component } from 'react'
2  import { Redirect } from "react-router-dom";
3  export default class Admin extends Component {
4    render() {
5      // 读取保存的user 如果不存在, 直接跳转到登录及诶面
6      const user =
JSON.parse(localStorage.getItem('user_key'));

```

```

7     if (!user._id) {
8         // render中不能这样写 这种方法很一般在事件回调函数中
        进行路由跳转
9         // this.props.history.replace('/login')
10        return <Redirect to='/login' />
11    }
12    return (
13        <div>
14            hello,Admin组件
15        </div>
16    )
17 }
18 }
19

```

## 封装storageUtils模块

新建utils/storageUtils.js

```

1 export default {
2     saveUser(user){
3
4         localStorage.setItem('user_key',JSON.stringify(user))
5     },
6     getUser(){
7         return JSON.parse(localStorage.getItem('user_key'))
8     } || {},
9     },
10    removeUser(){
11        localStorage.removeItem('user_key')
12    }
13 }

```



## 修改Admin.js

```
1 import React, { Component } from 'react'
2 import { Redirect } from "react-router-dom";
3 import storageUtils from '../utils/storageUtils.js';
4 export default class Admin extends Component {
5   render() {
6     // 读取保存的user 如果不存在, 直接跳转到登录及诶面
7     const user = storageUtils.getUser();
8     if (!user._id) {
9       // render中不能这样写 这种方法很一般在事件回调函数中
10      // this.props.history.replace('/login')
11      return <Redirect to='/login' />
12    }
13    return (
14      <div>
15        Admin组件
16      </div>
17    )
18  }
19 }
20 }
21
```

## 修改Login/Login.js

```
1 import React, { Component } from 'react'
2 import { Form, Icon, Input, Button, message } from
  'antd';
3 import { Redirect } from "react-router-dom";
4 import './Login.less'
5 import logo from '../assets/images/logo.svg'
6 import { reqLogin } from '../api';
```

```
7 class Login extends Component {
8   handleSubmit = e => {
9     e.preventDefault();
10    this.props.form.validateFields(async (err, values)
=> {
11      if (!err) {
12        const { username, password } = values;
13        const res = await reqLogin(username, password);
14        if (res.status === 0) {
15          // 将用户信息保存到local中
16          const user = res.data;
17          // 保存用户信息
18          storageUtils.saveUser(user)
19          // 跳转到管理页面
20          this.props.history.replace('/')
21          message.success('登录成功')
22        } else {
23          message.error(res.msg);
24        }
25      } else {
26
27      }
28    });
29  };
30  /**
31   * 自定义校验规则
32   */
33  validatorPwd = (rule, value, callback) => {
34    ....
35  }
36  render() {
37    // 读取保存的user 如果存在, 直接跳转到管理界面
38    const user = storageUtils.getUser();
39    if (user._id) {
```

```

40      // render中不能这样写 这种方法很一般在事件回调函数中
    进行路由跳转
41      // this.props.history.replace('/login')
42      return <Redirect to="/" />
43    }
44    const { getFieldDecorator } = this.props.form;
45    return (
46      <div className='login'>
47        ...
48      </div>
49    )
50  }
51 }
52
53 export default Form.create({ name: 'normal_login' })(Login);;

```

## 使用store插件封装存储模块

### 安装

```
1 npm i store -S
```

### 修改storageUtils.js

```

1  /*
2   操作local数据的工具函数模块
3   */
4  import store from 'store'
5  const USER_KEY = 'user_key'
6  export default {
7    saveUser(user){
8      store.set(USER_KEY,user);
9    },

```

```

10   getUser(){
11       return store.get(USER_KEY) || {}
12   },
13   removeUser(){
14       store.remove(USER_KEY)
15   }
16
17 }

```

## 封装内存中存储用户信息模块

每次读取用户信息都得从local中读取，多次的获取本地的用户数据会让运行效率更慢。我们可以把用户数据放入内存，放入内存的好处就是在你使用的时候从内存中获取，而且只会被获取一次。

新建utils/memoryUtils.js

```

1   import storageUtils from "../storageUtils"
2
3   // 初始时取一次并保存为user
4   const user = storageUtils.getUser()
5   export default {
6       user, // 用来存储登陆用户的信息，初始值为local中读取的
7       user
8   }

```

修改Amdin.js

```

1   import React, { Component } from 'react'
2   import { Redirect } from "react-router-dom";
3   import storageUtils from '../utils/storageUtils.js';
4   import memoryUtils from '../utils/memoryUtils'
5
6   export default class Admin extends Component {

```

```

7   render() {
8     // 读取保存的user 如果不存在, 直接跳转到登录及诶面
9     // const user = storageUtils.getUser();
10    const user = memoryUtils.user;
11    if (!user._id) {
12      // render中不能这样写 这种方法很一般在事件回调函数中
      进行路由跳转
13      // this.props.history.replace('/login')
14      return <Redirect to='/login' />
15    }
16    return (
17      <div>
18        Admin组件
19      </div>
20    )
21  }
22 }
23 }
24

```

## 修改Login.js

```

1  import React, { Component } from 'react'
2  import { Form, Icon, Input, Button, message } from
    'antd';
3  import { Redirect } from "react-router-dom";
4  import './Login.less'
5  import logo from '../assets/images/logo.svg'
6  import { reqLogin } from '../api';
7  import storageUtils from '../utils/storageUtils.js';
8  import memoryUtils from '../utils/memoryUtils';
9
10 class Login extends Component {
11   handleSubmit = e => {

```

```
12     e.preventDefault();
13     this.props.form.validateFields(async (err, values)
=> {
14         if (!err) {
15             // console.log('Received values of form: ',
values);
16             const { username, password } = values;
17             const res = await reqLogin(username, password);
18             if (res.status === 0) {
19                 // 将用户信息保存到local中
20                 const user = res.data;
21                 // localStorage.setItem('user_key',
JSON.stringify(user))
22                 storageUtils.saveUser(user)
23                 // 不要忘记将用户信息保存到内存中
24                 memoryUtils.user = user;
25                 // 跳转到管理页面
26                 this.props.history.replace('/')
27                 message.success('登录成功')
28             } else {
29                 message.error(res.msg);
30             }
31         } else {
32
33         }
34     });
35 };
36 /**
37  * 自定义校验规则
38  */
39 validatorPwd = (rule, value, callback) => {
40     value = value.trim();
41     if (!value) {
42         callback('密码必须输入')
43     } else if (value.length < 4) {
```

```

44     callback('密码不能小于4位')
45   } else if (value.length > 12) {
46     callback('密码不能大于12位')
47   } else if (!/^[a-zA-Z0-9_]+$/.test(value)) {
48     callback('密码必须是英文、数字或下划线组成')
49   } else {
50     // 验证通过一定要callback,如果callback,验证无法响应
51     callback();
52   }
53 }
54 render() {
55   // 读取保存的user 如果存在, 直接跳转到管理界面
56   // const user =
JSON.parse(localStorage.getItem('user_key')) || {};
57   // const user = storageUtils.getUser();
58   const user = memoryUtils.user
59   if (user._id) {
60     // render中不能这样写 这种方法很一般在事件回调函数中
进行路由跳转
61     // this.props.history.replace('/login')
62     return <Redirect to="/" />
63   }
64   const { getFieldDecorator } = this.props.form;
65   return (
66     <div className='login'>
67       ....
68     </div>
69   )
70 }
71 }
72
73 export default Form.create({ name: 'normal_login' })(Login);

```

将来的以上解决方案我们会使用redux来处理

## Admin组件

---

安装antd

```
1 npm i antd -S
```

按需加载引入

```
1 cnpm i react-app-rewired customize-cra -S
```

package.json修改

```
1 {  
2   "scripts": {  
3     "start": "react-app-rewired start",  
4     "build": "react-app-rewired build",  
5     "test": "react-app-rewired test",  
6     "eject": "react-app-rewired eject"  
7   },  
8 }
```

根目录下新建config-overrides.js用于修改默认配置

支持装饰器

```
1 cnpm install --save-dev babel-plugin-transform-  
  decorators-legacy @babel/plugin-proposal-decorators
```

```
1 const {  
2   override,  
3   fixBabelImports, //按需加载配置函数
```



```

4   addBabelPlugins //babel插件配置函数
5 } = require('customize-cra');
6 module.exports = override(
7   fixBabelImports('import', {
8     libraryName: 'antd',
9     libraryDirectory: 'es',
10    style: 'css',
11  }),
12  addBabelPlugins( // 支持装饰器
13    [
14      '@babel/plugin-proposal-decorators',
15      {
16        legacy: true
17      }
18    ]
19  )
20 );

```

## Admin/Admin.jsx

```

1  import React, { Component } from 'react'
2  import { Redirect } from "react-router-dom";
3  import storageUtils from '../utils/storageUtils.js';
4  import memoryUtils from '../utils/memoryUtils'
5  import './Admin.less'
6  import LeftNav from '../components/LeftNav'
7  import MHeader from '../components/MHeader'
8  import { Switch,Route } from "react-router-dom";
9
10
11  import Home from '../Home/Home';
12  import Category from '../Category/Category';
13  import Product from '../Product/Product';
14  import Role from '../Role/Role';

```

```
15 import User from '../User/User';
16 import Bar from '../Charts/Bar';
17 import Line from '../Charts/Line';
18 import Pie from '../Charts/Pie';
19
20 import { Layout } from 'antd';
21
22 const { Content, Footer } = Layout;
23
24 export default class Admin extends Component {
25   state = {
26     collapsed: false,
27   };
28
29   toggle = () => {
30     this.setState({
31       collapsed: !this.state.collapsed,
32     });
33   };
34   render() {
35     // 读取保存的user 如果不存在, 直接跳转到登录及诶面
36     // const user = storageUtils.getUser();
37     const user = memoryUtils.user;
38     if (!user._id) {
39       // render中不能这样写 这种方法很一般在事件回调函数中
       进行路由跳转
40       // this.props.history.replace('/login')
41       return <Redirect to='/login' />
42     }
43     return (
44       <Layout style={{height: '100%'}}>
45         <LeftNav collapsed={this.state.collapsed}>
46         </LeftNav>
47         <Layout>
```

```
48         <MHeader collapsed = {this.state.collapsed}/>
49         <Content
50             style={{
51                 margin: '24px 16px',
52                 padding: 24,
53                 background: '#fff',
54                 minHeight: 280,
55             }}
56         >
57             { /* 路由配置 */ }
58             <Switch>
59                 <Route path='/home' component={Home} />
60                 <Route path='/category' component=
{Category} />
61                 <Route path='/product' component=
{Product} />
62                 <Route path='/role' component={Role} />
63                 <Route path='/user' component={User} />
64                 <Route path='/charts/bar' component={Bar}
/>
65                 <Route path='/charts/line' component=
{Line}/>
66                 <Route path='/charts/pie' component=
{Pie}/>
67                 <Redirect to='/home' />
68             </Switch>
69         </Content>
70         <Footer style={{ textAlign: 'center' }}>Ant
Design ©2018 Created by Ant UED</Footer>
71     </Layout>
72 </Layout>
73 )
74
75 }
76 }
```

## LeftNav组件

```
1 import React, { Component } from 'react'
2 import { Link } from "react-router-dom";
3 import { Layout, Menu, Icon } from "antd";
4 import Logo from '../assets/images/logo192.png';
5 import './index.less'
6 const { Sider } = Layout;
7 const { SubMenu } = Menu;
8 export default class LeftNav extends Component {
9   render() {
10     return (
11       <Sider collapsible collapsed=
12 {this.props.collapsed}>
13       <div className='leftnav'>
14         <div className="logo">
15           <Link to='/home' className='left-nav-
16 link'>
17             <img src={Logo} alt="" />
18             <h1>小马哥后台</h1>
19           </Link>
20           <Menu theme="dark" defaultSelectedKeys=
21 {[ '1' ]} mode="inline">
22             <Menu.Item key="/home">
23               <Link to='/home'>
24                 <Icon type="home" />
25                 <span>首页</span>
26               </Link>
27             </Menu.Item>
28             <SubMenu
29               key="sub1"
```

```
27         title={
28             <span>
29                 <Icon type="project" />
30                 <span>
31                     商品
32                 </span>
33             </span>
34         }
35     >
36         <Menu.Item key="/category">
37             <Link to="/category">
38                 <Icon type="menu-fold" />
39                 <span>
40                     品类管理
41                 </span>
42             </Link>
43
44         </Menu.Item>
45         <Menu.Item key="/product">
46             <Link to="/product">
47                 <Icon type="pic-right" />
48                 <span>
49                     商品管理
50                 </span>
51             </Link>
52
53         </Menu.Item>
54     </SubMenu>
55
56     <Menu.Item key="/user">
57         <Link to="/user">
58             <Icon type="user" />
59             <span>
60                 用户管理
61             </span>
```

```
62         </Link>
63
64     </Menu.Item>
65     <Menu.Item key="/role">
66         <Link to="/role">
67             <Icon type="ant-design" />
68             <span>
69                 角色管理
70             </span>
71         </Link>
72
73     </Menu.Item>
74     <SubMenu
75         key="sub3"
76         title={
77             <span>
78                 <Icon type="line-chart" />
79                 <span>
80                     图形图标
81                 </span>
82             </span>
83         }
84     >
85         <Menu.Item key="/charts/bar">
86             <Link to="/charts/bar">
87                 <Icon type="bar-chart" />
88                 <span>
89                     柱形图
90                 </span>
91             </Link>
92         </Menu.Item>
93         <Menu.Item key="/charts/line">
94             <Link to="/charts/line">
95                 <Icon type="line-chart" />
96                 <span>
```

```
97         折线图
98         </span>
99         </Link>
100     </Menu.Item>
101     <Menu.Item key="/charts/pie">
102         <Link to="/charts/pie">
103             <Icon type="pie-chart" />
104             <span>
105                 饼形图
106             </span>
107         </Link>
108     </Menu.Item>
109 </SubMenu>
110 </Menu>
111 </div>
112 </div>
113 </Sider>
114
115     )
116 }
117 }
```

## LeftNav/index.less

```
1  .logo {
2    height: 32px;
3    margin: 16px;
4
5    .left-nav-link {
6      display: flex;
7      align-items: center;
8      height: 80px;
9
10     img {
```

```
11     width: 40px;
12     height: 40px;
13     margin: 0 10px;
14 }
15
16 h1 {
17     color: #fff;
18     font-size: 18px;
19     margin-bottom: -5px;
20 }
21 }
22 }
23
24 a {
25     color: rgba(255, 255, 255, 0.65);
26
27 }
```

## 侧边栏配置数据

config/menConfig.js

```
1  const menuList = [
2    {
3      title: '首页', // 菜单标题名称
4      key: '/home', // 对应的path
5      icon: 'home', // 图标名称
6      public: true, // 公开的
7    },
8    {
9      title: '商品',
10     key: '/products',
11     icon: 'appstore',
12     children: [ // 子菜单列表
```



```
13     {
14         title: '品类管理',
15         key: '/category',
16         icon: 'bars'
17     },
18     {
19         title: '商品管理',
20         key: '/product',
21         icon: 'tool'
22     },
23 ]
24 },
25
26 {
27     title: '用户管理',
28     key: '/user',
29     icon: 'user'
30 },
31 {
32     title: '角色管理',
33     key: '/role',
34     icon: 'safety',
35 },
36
37 {
38     title: '图形图表',
39     key: '/charts',
40     icon: 'area-chart',
41     children: [
42         {
43             title: '柱形图',
44             key: '/charts/bar',
45             icon: 'bar-chart'
46         },
47         {
```

```

48         title: '折线图',
49         key: '/charts/line',
50         icon: 'line-chart'
51     },
52     {
53         title: '饼图',
54         key: '/charts/pie',
55         icon: 'pie-chart'
56     },
57 ]
58 },
59 ]
60
61 export default menuList

```

## 封装侧边栏动态菜单

```

1  import React, { Component } from 'react'
2  import { Link } from "react-router-dom";
3  import { Layout, Menu, Icon } from "antd";
4  import Logo from '../assets/images/logo192.png';
5  import './index.less'
6  import menuList from '../config/menuConfig';
7
8  const { Sider } = Layout;
9  const { SubMenu } = Menu;
10 export default class LeftNav extends Component {
11     /*
12     根据指定菜单数据列表产生<Menu>的子节点数组
13     使用 reduce() + 递归
14     */
15     getMenuNodes2 = (menulist) => {

```

```
16     return menulist.reduce((pre, item) => {
17         if (!item.children) {
18             pre.push((
19                 <Menu.Item key={item.key}>
20                     <Link to={item.key}>
21                         <Icon type={item.icon} />
22                         <span>{item.title}</span>
23                     </Link>
24                 </Menu.Item>
25             ))
26         } else {
27             pre.push((
28                 <SubMenu
29                     key={item.key}
30                     title={
31                         <span>
32                             <Icon type={item.icon} />
33                             <span>
34                                 {item.title}
35                             </span>
36                         </span>
37                     }
38                 >
39                     {this.getMenuNodes(item.children)}
40                 </SubMenu>
41             ))
42         }
43         return pre;
44     }, [])
45 }
46
47 /*
48     根据指定菜单数据列表产生<Menu>的子节点数组
49     使用map()+递归
50 */
```

```
51 getMenuNodes = (menuList) => {
52   return menuList.map(item => {
53     if (!item.children) {
54       return (
55         <Menu.Item key={item.key}>
56           <Link to={item.key}>
57             <Icon type={item.icon} />
58             <span>{item.title}</span>
59           </Link>
60         </Menu.Item>
61       )
62     } else {
63       return (
64         <SubMenu
65           key={item.key}
66           title={
67             <span>
68               <Icon type={item.icon} />
69               <span>
70                 {item.title}
71               </span>
72             </span>
73           }
74         >
75           {this.getMenuNodes(item.children)}
76         </SubMenu>
77       )
78     }
79   })
80 }
81
82 componentWillMount() {
83   this.menuNodes = this.getMenuNodes2(menuList);
84 }
85
```

```

86     render() {
87         return (
88             <Sider collapsible collapsed=
{this.props.collapsed}>
89                 <div className='leftnav'>
90                     <div className="logo">
91                         <Link to='/home' className='left-nav-
link'>
92                             <img src={Logo} alt="" />
93                             <h1>小马哥后台</h1>
94                         </Link>
95                         <Menu theme="dark" defaultSelectedKeys=
{['1']} mode="inline">
96                             {this.menuNodes}
97                         </Menu>
98                     </div>
99                 </div>
100             </Sider>
101         )
102     }
103 }
104 }
105

```

## 包装LeftNav组件拥有路由的能力

使用 `withRouter` 高阶组件包装普通组件 `LeftNav`

```

1 import React, { Component } from 'react'
2 import { Link, withRouter } from "react-router-dom";
3 //.....
4 class LeftNav extends Component{
5     //.....
6     render(){
7         let defaultKey = this.props.location.pathname;
8         console.log(defaultKey);
9     }
10
11 }
12 export default withRouter(LeftNav);

```

## 自动打开二级菜单

```

1 import React, { Component } from 'react'
2 import { Link, withRouter } from "react-router-dom";
3 import { Layout, Menu, Icon } from "antd";
4 import Logo from '../assets/images/logo192.png';
5 import './index.less'
6 import menuList from '../config/menuConfig';
7
8 const { Sider } = Layout;
9 const { SubMenu } = Menu;
10 class LeftNav extends Component {
11     /*
12     根据指定菜单数据列表产生<Menu>的子节点数组
13     使用 reduce() + 递归
14     */
15     getMenuNodes2 = (menulist) => {
16         const path = this.props.location.pathname;
17         return menulist.reduce((pre, item) => {

```

```
18     if (!item.children) {
19         pre.push((
20             <Menu.Item key={item.key}>
21                 <Link to={item.key}>
22                     <Icon type={item.icon} />
23                     <span>{item.title}</span>
24                 </Link>
25             </Menu.Item>
26         ))
27     } else {
28
29         // 如果当前请求路由与当前菜单的某个子菜单的key匹
30         // 配，将菜单的key保存在openKey
31         /*
32         判断当前item的key是否是我需要的key
33         查找item的所有children中cItem的key, 按是否有一个跟请求的path匹配
34         */
35         const cItem = item.children.find(cItem =>
36             cItem.key === path)
37         if (cItem) {
38             this.openKey = item.key;
39         }
40
41         pre.push((
42             <SubMenu
43                 key={item.key}
44                 title={
45                     <span>
46                         <Icon type={item.icon} />
47                         <span>
48                             {item.title}
49                         </span>
46                     </span>
47                 </SubMenu>
48             )
49         )
```

```
50         >
51         {this.getMenuNodes(item.children)}
52     </SubMenu>
53     ))
54 }
55 return pre;
56 }, [])
57 }
58
59 render() {
60     let defaultKey = this.props.location.pathname;
61     return (
62         <Sider collapsible collapsed=
{this.props.collapsed}>
63         <div className='leftnav'>
64             <div className="logo">
65                 <Link to='/home' className='left-nav-link'>
66                     <img src={Logo} alt="" />
67                     <h1>小马哥后台</h1>
68                 </Link>
69                 <Menu
70                     theme="dark"
71                     defaultSelectedKeys={[defaultKey]}
72                     mode="inline"
73                     defaultOpenKeys={[this.openKey]}
74                 >
75                     {this.menuNodes}
76                 </Menu>
77
78             </div>
79         </div>
80     </Sider>
81
82     )
83 }
```



```

84 }
85 /*
86   向外暴露 使用高阶组件 withRouter() 来包装非路由组件
87   新组建向 LeftNav 传递3个特别属性: history/location/match
88   结果: LeftNav 可以操作路由相关方法了
89 */
90 export default withRouter(LeftNav)
91

```

退出登录后，用户登录时，默认选中首页

只需要将 `defaultSelectedKeys` 改为 `selectedKeys` 即可

- 1 `defaultSelectedKeys`: 总是根据第一次指定的key进行显示
- 2 `selectedKeys`: 总是根据指定的key进行显示

## Header组件

MHeader/index.js

```

1  import React, { Component } from 'react'
2  import { Layout, Button } from 'antd';
3  import './index.less'
4  const { Header } = Layout;
5  export default class MHeader extends Component {
6    logout = ()=>{
7      alert('退出了')
8    }
9    render() {
10      return (
11        <Header style={{ background: '#fff', padding: 0
12        }}>
13          <div className="header">
14            <h2 className='header-title'>首页</h2>

```

```

14         <div className='header-user'>
15             <div className='currentTime'>2020-02-30
16             12:00:08</div>
17             <div className='weather'>天气晴</div>
18             <div class='userInfo'>
19                 欢迎,admin
20                 <Button style={{ marginLeft: '20px' }}
21                 onClick={this.logout}>退出</Button>
22             </div>
23         </div>
24     </Header>
25 )
26 }
27

```

## MHeader/index.less

```

1  .header{
2    display: flex;
3    justify-content: space-between;
4    .header-title{
5      margin-left: 100px;
6      font-size: 30px;
7      color: #666;
8    }
9    .header-user{
10     display: flex;
11     margin-right: 100px;
12     .currentTime{
13       margin-right: 10px;
14     }
15     .weather{

```

```
16     margin-right: 10px;
17   }
18 }
19 }
```

## 退出登录

MHeader/index.js

```
1 import React, { Component } from 'react'
2 import { Layout, Button, Modal } from 'antd';
3 import { withRouter } from "react-router-dom";
4 import './index.less'
5 import memoryUtils from '../utils/memoryUtils';
6 import storageUtils from '../utils/storageUtils';
7 const { Header } = Layout;
8 const { confirm } = Modal;
9 class MHeader extends Component {
10   logout = () => {
11     confirm({
12       title: '确定要退出登录吗？',
13       content: '',
14       onOk: () => {
15         storageUtils.removeUser();
16         memoryUtils.user = {};
17         this.props.history.replace('/home');
18       },
19       onCancel: () => {
20         console.log('Cancel');
21       },
22     });
23   };
24 }
25 render() {
```

```

26     const user = memoryUtils.user
27     return (
28         <Header style={{ background: '#fff', padding: 0
29         }}>
30             <div className="header">
31                 <h2 className='header-title'>首页</h2>
32                 <div className='header-user'>
33                     <div className='currentTime'>2020-02-30
34                     12:00:08</div>
35                     <div className='weather'>天气晴</div>
36                     <div class='userInfo'>
37                         欢迎,{user.username}
38                         <Button style={{ marginLeft: '20px' }}
39                         onClick={this.logout}>退出</Button>
40                     </div>
41                 </div>
42             </div>
43         </Header>
44     )
45 }
46 }
47 export default withRouter(MHeader)

```

## 动态显示标题

MHeader/index.js

```

1  getTitle = () => {
2      // 根据当前请求的path得到对应的title
3      let title = '';
4      const path = this.props.location.pathname;
5      menuList.forEach(item => {
6          if (item.key === path) {
7              title = item.title;

```

```

8         } else if (item.children) {
9             const cItem =
item.children.find(cItem=>path===cItem.key);
10             if(cItem){
11                 title = cItem.title;
12             }
13         }
14     })
15     return title;
16 }

```

## 动态显示当前时间

```

1  //...
2  class MHeader extends Component {
3      constructor(props) {
4          super(props);
5          this.state = {
6              currentTime: new Date().toLocaleString()
7          }
8      }
9
10     // 动态显示当前时间
11     componentDidMount(){
12         // 开启定时器
13         this.timer = setInterval(() => {
14             this.setState({
15                 currentTime: new Date().toLocaleString()
16             })
17         }, 1000);
18     }
19     componentWillUnmount(){
20         clearInterval(this.timer)
21     }

```

```

22   render() {
23     const user = memoryUtils.user
24     return (
25       <Header style={{ background: '#fff', padding: 0
26     }}>
27       <div className="header">
28         <h2 className='header-title'>
29           {this.getTitle()}</h2>
30         <div className='header-user'>
31           <div className='currentTime'>
32             {this.state.currentTime}</div>
33           ....
34         </div>
35       </div>
36     </Header>
37   )
38 }
39
40 export default withRouter(MHeader)

```

## 显示当前天气

postman测试: <http://api.map.baidu.com/telematics/v3/weather?location=北京&output=json&ak=3p49MVra6urFRGOT9s8UBWr2>

api/index.js

```

1  import jsonp from 'jsonp'
2  import { message } from "antd";
3  // 发送jsonp请求得到天气信息
4  export const reqWeather = (city) => {
5    // 执行器函数:内部执行异步任务
6    // 成功了调用resolve(),失败了不调用reject,直接提示错误
7    return new Promise((resolve, reject) => {

```

```

8      const url =
`http://api.map.baidu.com/telematics/v3/weather?
location=${city}&output=json&ak=3p49MVra6urFRGOT9s8UBWr
2`
9      jsonp(url, {}, (error, data) => {
10        if (!error && data.error === 0) {
11          const { dayPictureUrl, weather } =
data.results[0].weather_data[0]
12          // 成功的
13          resolve({ dayPictureUrl, weather });
14        } else {
15          message.error('获取天气信息失败')
16        }
17      })
18    })
19
20  }

```

## MHeader/index.js

```

1  import React, { Component } from 'react'
2  import { reqWeather } from '../api';
3  //....
4  class MHeader extends Component {
5    constructor(props) {
6      super(props);
7      this.state = {
8        dayPictureUrl: '', // 图片url
9        weather: '', // 天气文本
10      }
11
12    }
13    // 动态显示当前时间
14    componentDidMount(){

```

```

15      // 发jsonp请求获取天气信息显示
16      this.getWeather();
17  }
18
19  // 获取天气信息显示
20  getWeather = async ()=>{
21      const { dayPictureUrl, weather } = await
reqWeather('北京');
22      // 更新状态
23      this.setState({
24          dayPictureUrl,
25          weather
26      })
27
28  }
29  render() {
30      return (
31          <Header style={{ background: '#fff', padding: 0
}}>
32              <div className="header">
33                  <div className='header-user'>
34                      <div className='weather'>天气
{this.state.weather}</div>
35                      <div className='weatherPic'>
36                          <img src={this.state.dayPictureUrl}
alt=""/>
37                      </div>
38                  </div>
39              </Header>
40          )
41      }
42  }
43  export default withRouter(MHeader)

```



# 分类组件

api/index.js

```
1 // 获取分类列表
2 export const reqCategorys = () => {
3   return ajax(BASE + '/manage/category/list')
4 }
5 // 修改分类
6 export const reqUpdateCategory =
  ({categoryId, categoryName})=>{
7   return ajax.post(BASE + '/manage/category/update',{
8     categoryId,
9     categoryName
10  })
11 }
12 // 添加分类
13 export const reqAddCategory = ({ categoryName }) => {
14   return ajax.post(BASE + '/manage/category/add', {
15     categoryName
16   })
17 }
```

Category/category.js

```
1 import React, { Component } from 'react'
2 import { Card, Button, Icon, Table, message, Modal }
  from "antd";
3 import { reqCategorys, reqUpdateCategory,
  reqAddCategory } from "../api";
4 import AddUpdateForm from './add-update-form'
5 export default class Category extends Component {
6   state = {
7     categorys: [],
```

```

8      showStatus: 0, //0:不显示 1:显示添加 2:显示修改
9  }
10  getCategories = async () => {
11      const res = await reqCategories();
12      if (res.status === 0) {
13          // 成功了
14          this.setState({
15              categories: res.data,
16
17          })
18      } else {
19          message.error('获取分类列表失败了')
20      }
21
22  }
23  initColumns = () => {
24      this.columns = [
25          {
26              title: '分类的名称',
27              dataIndex: 'name',
28          },
29          {
30              title: '操作',
31              width: 300,
32              render: (category) => <Button type='primary'
onClick={() => {
33                  console.log(category);
34
35                  this.category = category; //保存当前分类，其
它地方都可以读取到
36                  this.setState({
37                      showStatus: 2
38                  })
39              }}>
40                  修改分类

```

```
41         </Button>
42     }
43 ]
44 }
45 componentWillMount() {
46     this.initColumns();
47 }
48
49 componentDidMount() {
50
51     this.getCategorys();
52 }
53 handleOk = e => {
54     // 进行表单验证
55     this.form.validateFields(async (err, values) => {
56         if (!err) {
57             // 验证通过后,得到输入数据
58             // console.log(values);
59             const { categoryName } = values;
60             const { showStatus } = this.state;
61             let result
62             if (showStatus === 1) {
63                 // 添加
64                 result = await reqAddCategory({ categoryName
65
66             } else {
67                 // 修改
68                 const categoryId = this.category._id;
69                 result = await reqUpdateCategory({
70                     categoryId, categoryName })
71             }
72
73             // 重置输入的数据(变成初始值)
74             this.form.resetFields();
```

```

74         this.setState({ showStatus: 0 })
75         const action = showStatus === 1 ? '添加' : '修
    改';
76         // 根据响应结果，做不同处理
77         if (result.status === 0) {
78             // 重新获取分类列表显示
79             this.getCategorys();
80             message.success(action + '分类成功');
81         } else {
82             message.error(action + '分类失败')
83         }
84     }
85 })
86 };
87
88 handleCancel = e => {
89
90     // 重置表单
91     this.form.resetFields();
92     this.setState({
93         showStatus: 0
94     });
95 };
96 render() {
97     // Cart右上角的结构
98     const extra = (
99         <Button type='primary' onClick={() => {
100             this.setState({
101                 showStatus: 1
102             })
103         }}>
104             <Icon type="plus" />
105             添加
106         </Button>
107     )

```

```

108      // 读取更新的分类名称
109      const category = this.category || {}
110
111
112      return (
113        <Card extra={extra} style={{ width: '100%' }}>
114          <Table
115            dataSource={this.state.categories}
116            columns={this.columns}
117            rowKey='_id'
118            bordered
119            pagination={{ defaultPageSize: 6,
showQuickJumper: true }}
120          />
121          <Modal
122            title={this.state.showStatus === 1 ? '添加分
类' : '修改分类'}
123            visible={this.state.showStatus !== 0}
124            onOk={this.handleOk}
125            onCancel={this.handleCancel}
126          >
127
128            <AddUpdateForm setForm={form => this.form =
form} categoryName={category.name}></AddUpdateForm>
129          </Modal>
130
131        </Card>
132      )
133    }
134  }
135

```

Category/add-update-form.js

```
1 import React, { Component } from 'react'
2 import PropTypes from 'prop-types';
3 import { Form, Input } from "antd";
4 class AddUpdateForm extends Component {
5   static propTypes = {
6     setForm: PropTypes.func.isRequired,
7     categoryName: PropTypes.string
8   }
9   componentWillMount() {
10     this.props.setForm(this.props.form)
11   }
12   render() {
13     const { getFieldDecorator } = this.props.form;
14     const { categoryName } = this.props;
15     return (
16       <Form>
17         <Form.Item>
18           {
19             getFieldDecorator('categoryName',{
20               initialValue:categoryName || '',
21               rules:[
22                 {
23                   required:true,
24                   message:'分类名称必须输入'
25                 }
26               ]
27             })(
28               <Input type='text' placeholder='请输入分
29 类名称' />
30             )
31           }
32         </Form.Item>
33       </Form>
34     )
35   }
36 }
```

```
34   }
35 }
36 export default Form.create()(AddUpdateForm);
37
```

## 商品分页组件

### 获取商品分页显示

接口

```
1 export const reqProducts = (pageNum, pageSize) => {
2   return ajax(BASE + '/manage/product/list', {
3     params: {
4       pageNum,
5       pageSize
6     }
7   })
8 }
```

Product/product.js

```
1 import React, { Component } from 'react'
2 import './Product.less'
3 import { Card, Select, Input, Button, Icon, Table }
4   from "antd";
5 import { reqProducts, reqSearchProducts } from
6   "../api";
7 const Option = Select.Option;
8 const PAGE_SIZE = 2;
9 export default class Product extends Component {
10   state = {
```

```
9      loading: false,
10      searchType: 'productName', //默认是按商品名称搜索
11      searchName: '', //搜索的关键字
12      products: [], //商品列表
13      total: 0, //商品的总数量
14    }
15    /*
16    异步获取指定页码商品分页(可能带搜索)列表显示
17    */
18    getProducts = async (pageNum) => {
19      // 保存当前请求的页码
20      this.pageNum = pageNum;
21      const { searchName, searchType } = this.state;
22      let result = await reqProducts(pageNum, PAGE_SIZE)
23      if (result.status === 0) {
24        const { list, total } = result.data;
25
26        this.setState({
27          total,
28          products: list
29        })
30      }
31    }
32    updateStatus = (id, status) => {
33      console.log(id, status);
34
35    }
36    initColumns = () => {
37      this.columns = [
38        {
39          title: '商品名称',
40          dataIndex: 'name'
41        },
42        {
43          title: '商品描述',
```



```
44     dataIndex: 'desc'
45   },
46   {
47     title: '价格',
48     width: 100,
49     dataIndex: 'price',
50     render: price => '¥' + price
51   },
52   {
53     title: '状态',
54     width: 100,
55     // dataIndex: 'status'
56     render: ({ _id, status }) => {
57       let btnText = '下架', text = '在售';
58       if (status === 2) {
59         btnText = '上架';
60         text = '已下架';
61       }
62       return (
63         <span>
64           <Button onClick={this.updateStatus(_id,
status)}>{btnText}</Button>
65           <span className='status'>{text}</span>
66         </span>
67       )
68     }
69   },
70   {
71     title: '操作',
72     width: 100,
73     dataIndex: 'done',
74     render: (product) => {
75       return <span>
76         <Button
```

```
78         type='link'
79         onClick={() => {
80             // 跳转商品详情页面
81         }}
82     >详情</Button>
83     <Button
84         type='link'
85         onClick={() => {
86             // 跳转商品修改页面
87         }}
88     >修改</Button>
89 </span>
90     }
91 }
92 ]
93 }
94 componentWillMount() {
95     this.initColumns()
96 }
97 componentDidMount() {
98     // 获取第一页显示
99     this.getProducts(1);
100 }
101 render() {
102     const { total, searchType, searchName, products,
103 loading } = this.state;
104     const title = (
105         <span>
106             <Select value={searchType} style={{ width: 200
107 }} onChange={value => this.setState({ searchType:
108 value })}>
109                 <Option value="productName">按名称搜索
110             </Option>
111                 <Option value="productDesc">按描述搜索
112             </Option>
```

```
108         </Select>
109         <Input
110             placeholder="关键字"
111             style={{ width: 200, margin: '0 10px' }}
112             value={searchName}
113             onChange={e => this.setState({ searchName:
e.target.value })}
114         />
115         <Button type='primary' onClick={() => {
116             //搜索功能
117         }}>搜索</Button>
118     </span>
119 )
120     const extra = (
121         <Button type='primary' onClick={() => {
122             //添加商品操作
123         }}>
124             <Icon type="plus" />
125             添加商品
126         </Button>
127     )
128     return (
129         <Card title={title} extra={extra}>
130             <Table
131                 bordered
132                 loading={loading}
133                 rowKey='_id'
134                 columns={this.columns}
135                 dataSource={products}
136                 pagination={{
137                     total,
138                     defaultPageSize: PAGE_SIZE,
139                     showQuickJumper: true,
140                     onChange: this.getProducts,
141                     current: this.pageNum
```

```

142         }}
143     >
144     </Table>
145 </Card>
146 )
147 }
148 }

```

## 按描述/名称搜索

接口

```

1 // 根据搜索关键字获取商品分页列表
2 export const reqSearchProducts = ({ pageNum, pageSize,
  searchName, searchType }) => {
3     return ajax(BASE + '/manage/product/search', {
4         params: {
5             pageNum,
6             pageSize,
7             [searchType]: searchName
8         }
9     })
10 }

```

```

1 export default class ProductHome extends Component {
2     state = {
3         loading: false,
4         searchType: 'productName', //默认是按商品名称搜索
5         searchName: '', //搜索的关键字
6         products: [], //商品列表
7         total: 0, //商品的总数量
8     }

```

```
9      /*
10      异步获取指定页码商品分页(可能带搜索)列表显示
11      */
12      getProducts = async (pageNum) => {
13          // 保存当前请求的页码
14          this.pageNum = pageNum;
15          const { searchName, searchType } = this.state;
16          let result;
17          // 发请求获取数据
18          if (!this.isSearch) {
19              result = await reqProducts(pageNum, PAGE_SIZE)
20          } else {
21              //搜索功能
22              result = await reqSearchProducts({
23                  pageNum,
24                  pageSize: PAGE_SIZE,
25                  searchName,
26                  searchType
27              })
28          }
29          if (result.status === 0) {
30              const { list, total } = result.data;
31              this.setState({
32                  total,
33                  products: list
34              })
35          }
36      }
37
38      render() {
39          const { total, searchType, searchName, products,
40              loading } = this.state;
41
42          const title = (
43              <span>
```

```

43         ....
44         <Button type='primary' onClick={() => {
45             // 保存搜索的标记
46             this.isSearch = true; // 保存搜索的标记
47             // 获取搜索的商品
48             this.getProducts(1)
49         }}>搜索</Button>
50     </span>
51 )
52     const extra = (
53         <Button type='primary' onClick={() => {
54             // 添加商品操作
55         }}>
56             <Icon type="plus" />
57             添加商品
58         </Button>
59     )
60     return (
61         <Card title={title} extra={extra}>
62             ...
63         </Card>
64     )
65 }
66 }
67

```

## 路由显示

在当前商品页面,需要对添加商品(/product/addupdate)/商品详情(/product/detail/5e167d0835c6d482a9f0e2c0)/修改商品(/product/addUpdate)按钮做相应操作,所以需要路由控制

- 新建Product/home.js,将 product.js 代码迁移
- 新建Product/detail.js

- 新建Product/add-update.js

原Product/product.js修改

```
1 import React, { Component } from 'react'
2 import { Switch,Route,Redirect } from "react-router-dom";
3 import './Product.less'
4 import ProductHome from './home';
5 import ProductAddUpdate from './add-update'
6 import ProductDetail from './detail'
7
8 export default class Product extends Component {
9   render() {
10     return (
11       <Switch>
12         <Route path='/product' exact component=
{ProductHome}></Route>
13         <Route path='/product/addupdate' component=
{ProductAddUpdate}></Route>
14         <Route path='/product/detail/:id' component=
{ProductDetail}></Route>
15         <Redirect to='/product' />
16       </Switch>
17     )
18   }
19 }
```

add-update.js

```

1 import React from 'react'
2 class ProductAddUpdate extends React.Component {
3   render() {
4     return (
5       <div>商品修改</div>
6     );
7   }
8 }
9
10 export default ProductAddUpdate;

```

## detail.js

```

1 import React from 'react'
2 class ProductDetail extends React.Component {
3   render() {
4     return (
5       <div>商品详情</div>
6     );
7   }
8 }
9
10 export default ProductAddUpdate;

```

## home.js

```

1 //商品详情和商品修改页面 因为商品的详情和修改都需要获取当前
  商品的数据,所以在跳转路由之前先将当前商品数据保存到缓存中
2 <span>
3   <Button
4     type='link'
5     onClick={() => {
6       // 保存当前商品的数据
7       memoryUtils.product = product

```



```

8          // 跳转商品详情页面
9          this.props.history.push('/product/detail/'
+ product._id)
10      }}
11      >详情</Button>
12      <Button
13          type='link'
14          onClick={() => {
15              // 保存当前商品的数据
16              memoryUtils.product = product
17              // 跳转商品修改页面
18
19              this.props.history.push('/product/addUpdate')
20              }}
21      >修改</Button>
22  </span>
23  //添加商品
24  <Button type='primary' onClick={() => {
25      //添加商品,不需要获取当前商品的数据,先把商品数据置
26      空
27      memoryUtils.product = {}
28      //跳转商品添加页面
29      this.props.history.push('/product/addupdate');
30  }}>
31  <Icon type="plus" />
    添加商品
32 </Button>

```

## 查看商品详情

接口

```

1 // 根据商品ID获取该商品
2 export const reqProduct = (productId) => {

```

```

3   return ajax(BASE + '/manage/product/info', {
4     params: {
5       productId
6     }
7   })
8 }
9 // 根据分类ID获取该商品分类
10 export const reqCategory = (categoryId) => {
11   return ajax(BASE + '/manage/category/info',{
12     params:{
13       categoryId
14     }
15   })
16 }

```

## detail.js

```

1   import React, { Component } from 'react'
2   import { Card, Icon, Button, List } from "antd";
3   // import memoryUtils from '../utils/memoryUtils'
4   import { reqProduct, reqCategory } from "../api";
5   import './detail.less'
6   export default class detail extends Component {
7     state = {
8       product: {},
9       categoryName: ''
10    }
11    getCategory = async (categoryId) => {
12      const res = await reqCategory(categoryId);
13      console.log(res);
14
15      if (res.status === 0) {
16        this.setState({
17          categoryName: res.data.name

```

```
18     })
19   }
20 }
21 async componentDidMount() {
22   let product = this.state.product;
23   if (product._id) {
24     // 如果商品中有数据,获取对应的分类
25   } else {
26     // 如果当前product状态没有数据,根据id参数中请求获取
商品并更新
27     const id = this.props.match.params.id;
28     const res = await reqProduct(id);
29     if (res.status === 0) {
30       product = res.data;
31       this.setState({ product })
32       //获取对应的分类
33       this.getCategory(product.categoryId);
34     }
35   }
36 }
37 render() {
38   const title = (
39     <span>
40       <Button type='link' onClick={() =>
this.props.history.goBack()}>
41         <Icon type="arrow-left" />
42       </Button>
43       <span>商品详情</span>
44     </span>
45   )
46   const { product, categoryName } = this.state;
47
48   return (
49     <Card title={title}>
50       <List>
```

```
51     <List.Item>
52         <p>
53             <span>商品名称:</span>
54             <span>{product.name}</span>
55         </p>
56     </List.Item>
57
58     <List.Item>
59         <p>
60             <span>商品描述:</span>
61             <span>{product.desc}</span>
62         </p>
63     </List.Item>
64
65     <List.Item>
66         <p>
67             <span>商品价格:</span>
68             <span>{product.price}</span>
69         </p>
70     </List.Item>
71
72     <List.Item>
73         <p>
74             <span>所属分类:</span>
75             <span>{categoryName}</span>
76         </p>
77     </List.Item>
78
79     <List.Item>
80         <p>
81             <span>商品分类:</span>
82             <span>
83                 {
```

```

84         product.imgs && product.imgs.map(img
=> <img className="detail-img" key={img} src=
{'http://localhost:5000/upload/' + img} alt="img" />)
85     }
86
87     </span>
88     </p>
89     </List.Item>
90     <List.Item>
91         <span>商品详情:</span>
92         <div dangerouslySetInnerHTML=
{{__html:product.detail}}></div>
93     </List.Item>
94
95     </List>
96     </Card>
97 )
98 }
99 }
100

```

## 修改/添加商品

接口 api/index.js

```

1 // 请求所有商品
2 export const reqProducts = (pageNum, pageSize) => {
3     return ajax(BASE + '/manage/product/list', {
4         params: {
5             pageNum,
6             pageSize
7         }
8     })
9 }

```

```
10 // 如果有商品的id 则是更新 否则是添加
11 export const reqUpdateProduct = (product) => {
12   const a = product._id ? 'update' : 'add'
13   return ajax.post(`${BASE}/manage/product/${a}`,
14     product)
15 }
```

## Product/add-update.js

```
1  import React from 'react'
2  import {
3    Form,
4    Input,
5    Icon,
6    Select,
7    Button,
8    Card,
9    message
10 } from 'antd';
11 import PicturesWall from '../pictures-wall';
12 import RichTextEditor from '../rich-text-editor';
13 import memoryUtils from '../../utils/memoryUtils';
14 import { reqCategorys, reqUpdateProduct } from
15   "../../api";
16
17
18 class ProductAddUpdate extends React.Component {
19   state = {
20     categorys: []
21   };
22   constructor(props) {
23     super(props)
```

```
24     // 创建ref容器，并保存到组件对象
25     this.pwRef = React.createRef()
26     this.editorRef = React.createRef()
27 }
28
29 handleSubmit = e => {
30     e.preventDefault();
31     this.props.form.validateFields(async (err, values)
=> {
32         if (!err) {
33             const { name, desc, price, categoryId } =
values;
34             // 收集上传的图片文件名的数组
35             const imgs = this.pwRef.current.getImgs();
36             const detail =
this.editorRef.current.getDetail();
37             // 封装product对象
38             const product =
{name,desc,price,categoryId,imgs,detail}
39             if(this.isUpdate){
40                 // 如果是更新，获取商品的id
41                 product._id = this.product._id;
42             }
43             console.log(product);
44
45             // 发请求添加或修改
46             const res = await reqUpdateProduct(product);
47             if(res.status === 0){
48                 message.success(`${this.isUpdate ? '修改' :
'添加'}商品成功` )
49                 this.props.history.replace('/product')
50             }else{
51                 message.error(res.msg)
52             }
53 }
```

```
54     }
55   });
56 };
57 componentWillMount() {
58   // 组件将要加载的时候将商品的数据取出 存起来
59   this.product = memoryUtils.product;
60   this.isUpdate = !!this.product._id
61 }
62 componentDidMount() {
63   this.getCategorys()
64 }
65 getCategorys = async () => {
66   const res = await reqCategorys();
67   if (res.status === 0) {
68     this.setState({
69       categorys: res.data
70     })
71   }
72
73 }
74 render() {
75   const { getFieldDecorator } = this.props.form;
76
77   const formItemLayout = {
78     labelCol: { span: 2 },
79     wrapperCol: { span: 8 }
80   };
81   const { categorys } = this.state
82   const { isUpdate, product } = this
83   const title = (
84     <span>
85       <Button type='link' onClick={() =>
this.props.history.goBack()}>
86         <Icon type="arrow-left" />
87       </Button>
```



```
88         <span>{isUpdate ? '修改商品' : '添加商品'}
</span>
89     </span>
90 )
91
92     return (
93         <Card title={title}>
94             <Form {...formItemLayout} onSubmit=
{this.handleSubmit}>
95                 <Form.Item label="商品名称" >
96                     {getFieldDecorator('name', {
97                         initialValue: product.name,
98                         rules: [
99                             {
100                                 required: true,
101                                 message: '必须输入商品名称',
102                             },
103                         ],
104                     )}<Input placeholder='商品名称' />)}
105                 </Form.Item>
106                 <Form.Item label="商品描述">
107                     {getFieldDecorator('desc', {
108                         initialValue: product.desc,
109                         rules: [
110                             {
111                                 required: true,
112                                 message: '必须输入商品描述',
113                             },
114                         ],
115                     )}<Input placeholder='商品描述' />)}
116                 </Form.Item>
117                 <Form.Item label="商品价格">
118                     {getFieldDecorator('price', {
119                         initialValue: product.price,
120                         rules: [
```

```

121         {
122             required: true,
123             message: '必须输入价格',
124         },
125         {
126             validator: this.validatePrice,
127         },
128     ],
129     ))(<Input type="number" placeholder="商品
价格" addonAfter="元" />})
130 </Form.Item>
131 <Form.Item label='商品分类'>
132     {getFieldDecorator('categoryId', {
133         initialValue: product.categoryId || '',
134         rules: [{ required: true, message: '必须
输入商品分类' }]},
135     ))(
136         <Select>
137             <Option value=''>未选择</Option>
138             {
139                 categorys.map(c => <Option value=
{c._id} key={c._id}>{c.name}</Option>)
140             }
141         </Select>
142     )}
143 </Form.Item>
144 <Form.Item label="商品图片">
145     <PicturesWall ref={this.pwRef} imgs=
{product.imgs} />
146 </Form.Item>
147 <Form.Item label="商品详情" wrapperCol={{
col: 20 }}>
148     <br />
149     <RichTextEditor ref={this.editorRef}
detail={product.detail}></RichTextEditor>

```

```

150         </Form.Item>
151         <Form.Item>
152             <Button type='primary' htmlType="submit">
提交</Button>
153         </Form.Item>
154     </Form>
155 </Card>
156 );
157 }
158 }
159
160 export default Form.create()(ProductAddUpdate);

```

## 上传图片

Product/pictures-wall.jsx

```

1  import React from "react";
2  import { Upload, Icon, Modal } from 'antd';
3  import './pictures-wall.less'
4  function getBase64(file) {
5      return new Promise((resolve, reject) => {
6          const reader = new FileReader();
7          reader.readAsDataURL(file);
8          reader.onload = () => resolve(reader.result);
9          reader.onerror = error => reject(error);
10     });
11 }
12
13 class PicturesWall extends React.Component {
14     state = {
15         previewVisible: false,
16         previewImage: '', //当前上传图片
17         fileList: [],

```

```
18     };
19
20     handleCancel = () => this.setState({ previewVisible:
false });
21
22     handlePreview = async file => {
23         if (!file.url && !file.preview) {
24             file.preview = await
getBase64(file.originFileObj);
25         }
26         this.setState({
27             previewImage: file.url || file.preview,
28             previewVisible: true,
29         });
30     };
31     // 获取所有已上传图片文件名的数组
32     getImgs = () => this.state.fileList.map(file =>
file.name)
33
34     handleChange = async ({ file, fileList }) => {
35         // file与fileList中最后一个file代表同个图片的不
同对象
36         console.log('handleChange()', file.status,
file===fileList[fileList.length-1])
37         // 如果上传成功
38         if (file.status==='done') {
39             // 将数组最后一个file保存到file变量
40             file = fileList[fileList.length - 1]
41             // 取出响应数据中的图片文件名和url
42             const {name, url} = file.response.data
43             // 保存到上传的file对象
44             file.name = name
45             file.url = url
46         } else if (file.status==='removed') { // 删除
47             const result = await reqDeleteImg(file.name)
```

```
48         if (result.status===0) {
49             message.success('删除图片成功')
50         } else {
51             message.error('删除图片失败')
52         }
53     }
54
55     // 更新状态
56     this.setState({ fileList })
57 }
58 //组件将要加载时
59 componentWillMount() {
60     // 根据传入的imgs生成fileList并更新
61     const imgs = this.props.imgs;
62     if (imgs && imgs.length) {
63         const fileList = imgs.map((img, i) => ({
64             uid: -i, //唯一标识
65             name: img, //文件名
66             status: 'done', //状态有:uploading done error
removed
67             url: 'http://localhost:5000/upload/' + img
68         }))
69         this.setState({ fileList })
70     }
71 }
72
73 render() {
74     const { previewVisible, previewImage, fileList } =
this.state;
75     const uploadButton = (
76         <div>
77             <Icon type="plus" />
78             <div className="ant-upload-text">Upload</div>
79         </div>
80     );
```

```

81     return (
82         <div className="clearfix">
83             <Upload
84                 action="/manage/img/upload" //上传图片的url
85                 name="image" // 图片文件对应参数名
86                 listType="picture-card" //显示图片风格
87                 fileList={fileList} //已上传的所有图片文件信息
对象的数组
88                 onPreview={this.handlePreview}
89                 onChange={this.handleChange}
90             >
91                 {fileList.length >= 3 ? null : uploadButton}
92             </Upload>
93             <Modal visible={previewVisible} footer={null}
onCancel={this.handleCancel}>
94                 <img alt="example" style={{ width: '100%' }}
src={previewImage} />
95             </Modal>
96         </div>
97     );
98 }
99 }
100
101 export default PicturesWall;

```

## 富文本编辑器

### [文档链接](#)

Product/rich-text-editor.js

```

1  import React, { Component } from 'react'
2  import E from 'wangeditor'
3  export default class RichTextEditor extends Component {

```

```
4   constructor(props) {
5     super(props);
6     this.state = {
7       editorContent: '',
8       editor:null
9     };
10  }
11  getDetail = ()=> this.state.editorContent;
12
13  componentDidMount() {
14    const elemMenu = this.refs.editorElemMenu;
15    const elemBody = this.refs.editorElemBody;
16    const editor = new E(elemMenu, elemBody)
17    editor.create()
18    const detail = this.props.detail;
19    if(detail){
20      editor.txt.html(detail)
21    }
22    // 使用 onchange 函数监听内容的变化，并实时更新到
state 中
23    editor.customConfig.onChange = html => {
24      console.log(editor);
25
26      console.log(editor.txt.html())
27      this.setState({
28        // editorContent: editor.txt.text()
29        editorContent: editor.txt.html()
30      })
31    }
32    editor.customConfig.menus = [
33      'head', // 标题
34      'bold', // 粗体
35      'fontSize', // 字号
36      'fontName', // 字体
37      'italic', // 斜体
```

```
38     'underline', // 下划线
39     'strikeThrough', // 删除线
40     'foreColor', // 文字颜色
41     'backColor', // 背景颜色
42     'link', // 插入链接
43     'list', // 列表
44     'justify', // 对齐方式
45     'quote', // 引用
46     'emoticon', // 表情
47     'image', // 插入图片
48     'table', // 表格
49     'video', // 插入视频
50     'code', // 插入代码
51     'undo', // 撤销
52     'redo' // 重复
53 ]
54 editor.customConfig.uploadImgShowBase64 = true
55 editor.create()
56
57 };
58 render() {
59     return (
60         <div className="shop">
61             <div className="text-area" >
62                 <div ref="editorElemMenu"
63                     style={{ backgroundColor: '#f1f1f1',
border: "1px solid #ccc" }}
64                     className="editorElem-menu">
65
66                 </div>
67                 <div
68                     style={{
69                     padding: "0 10px",
70                     overflowY: "scroll",
71                     height: 300,
```



```

72         border: "1px solid #ccc",
73         borderTop: "none"
74     }}
75     ref="editorElemBody" className="editorElem-
body">
76         {this.state.editorContent}
77     </div>
78 </div>
79 </div>
80 );
81 }
82 }

```

## 角色管理组件

api/index.js

```

1  // 获取所有角色列表
2  export const reqRoles = () => {
3      return ajax(BASE + '/manage/role/list')
4  }
5
6  // 添加角色
7  export const reqAddRole = (roleName) => {
8      return ajax.post(BASE + '/manage/role/add', {
9          roleName
10     })
11 }
12
13 // 修改权限
14 export const reqAddAuth = (role) => {
15     return ajax.post(BASE + '/manage/role/update', role)
16 }

```

## Role/role.js

```
1  import React, { Component } from 'react'
2  import { Card, Button, Table, message, Modal } from
   "antd";
3  import AddRole from './add-role'
4  import AddAuth from './add-auth'
5  import { reqRoles, reqAddRole, reqAddAuth } from
   "../..../api";
6  import memoryUtils from "../..../utils/memoryUtils";
7  export default class Role extends Component {
8    state = {
9      isShowAdd: false, //是否显示添加界面
10     isShowAuth: false, //是否显示设计权限界面
11     roles: [], //所有角色的列表
12   }
13   addRole = () => {
14     // 进行表单验证, 只能通过了才向下处理
15     this.form.validateFields(async (error, value) => {
16       if (!error) {
17         // 隐藏确认框
18         this.setState({
19           isShowAdd: false
20         })
21         // 添加用户交互
22         const result = await
reqAddRole(value.roleName)
23         if (result.status === 0) {
24           message.success('添加角色成功');
25           const role = result.data;
26           // 修改状态时, 必须这样修改
27           this.setState(state => ({
28             roles: [...state.roles, role]
29           })))
```

```
30         } else {
31             message.error(result.msg);
32         }
33
34     }
35 })
36 }
37 // 增加用户权限操作
38 addAuth = async () => {
39     // 隐藏确认框
40     this.setState({
41         isShowAuth: false
42     })
43
44     // 更新role对象相关属性
45     const { role } = this;
46     role.menus = this.refs.authRef.getMenus();
47     role.auth_time = Date.now();
48     role.auth_name = memoryUtils.user.username;
49     // 请求更新角色
50     const result = await reqAddAuth(role);
51     if(result.status === 0){
52         message.success('角色授权成功');
53         // 重新获取所有角色
54         this.getRoles();
55     }else{
56         message.error(result.msg);
57     }
58
59
60 }
61 // 获取所有的角色
62 getRoles = async () => {
63     const result = await reqRoles();
64     if (result.status === 0) {
```

```
65     const roles = result.data;
66     this.setState({
67         roles
68     })
69 }
70 }
71 /*
72 初始化table列数组
73 */
74 initColumn = () => {
75     this.columns = [
76         {
77             title: '角色名称',
78             dataIndex: 'name'
79         },
80         {
81             title: '创建时间',
82             dataIndex: 'create_time',
83             // render: create_time =>
formateDate(create_time)
84             render: (create_time) => {
85                 if (create_time) {
86                     return new
Date(create_time).toLocaleString()
87                 } else {
88                     return;
89                 }
90             }
91         },
92         {
93             title: '授权时间',
94             dataIndex: 'auth_time',
95             render: (auth_time) => {
96                 if (auth_time) {
```

```
97         return new
Date(auth_time).toLocaleString()
98     } else {
99         return;
100     }
101
102     },
103     {
104         title: '授权人',
105         dataIndex: 'auth_name'
106     },
107     {
108         title: '操作',
109         render: (role) => <Button type='primary'
110         onClick={() => this.showAuth(role)}>设置权限</Button>
111     },
112 ]
113 }
114 showAuth = (role) => {
115     console.log(role);
116     // 将当前需要设置的角色保存到组件对象上
117     this.role = role;
118     this.setState({
119         isShowAuth: true
120     })
121 }
122 componentWillMount() {
123     this.initColumn()
124 }
125 componentDidMount() {
126     this.getRoles();
127 }
128
129 render() {
```

```

130     const { isShowAdd, roles, isShowAuth } =
this.state;
131
132     const title = (
133       <Button type='primary' onClick={() =>
this.setState({ isShowAdd: true })}>创建角色</Button>
134     )
135     return (
136       <Card title={title}>
137         <Table
138           dataSource={roles}
139           columns={this.columns}
140           bordered
141           rowKey='_id'
142           pagination={{ defaultPageSize: 2 }}
143         ></Table>
144         <Modal
145           title="添加角色"
146           visible={isShowAdd}
147           onOk={this.addRole}
148           onCancel={() => {
149             this.setState({ isShowAdd: false })
150           }}
151         >
152           <AddRole setForm={form => this.form = form}>
</AddRole>
153
154         </Modal>
155
156         <Modal
157           title="设置角色权限"
158           visible={isShowAuth}
159           onOk={this.addAuth}
160           onCancel={() => {
161             this.setState({ isShowAuth: false })

```

```

162         }}
163     >
164     <AddAuth ref='authRef' setForm={form =>
this.form = form} role={this.role}></AddAuth>
165
166     </Modal>
167 </Card>
168 )
169 }
170 }
171

```

## 添加角色

Role/add-role.js

```

1  import React, { Component } from 'react'
2  import {
3    Form,
4    Input,
5  } from 'antd';
6
7  class AddRole extends Component {
8    state = {
9    };
10
11    componentWillMount(){
12      this.props.setForm(this.props.form);
13    }
14    render() {
15      const { getFieldDecorator } = this.props.form;
16      const formItemLayout = {
17        labelCol: { span: 4 },
18        wrapperCol: { span: 15 }

```

```

19     };
20     return (
21       <Form {...formItemLayout} onSubmit=
{this.handleSubmit}>
22         <Form.Item label="角色名称" >
23           {getFieldDecorator('roleName', {
24             rules: [
25               {
26                 required: true,
27                 message: '必须输入角色名称',
28               },
29             ],
30           })(<Input placeholder='请输入角色名称' />)}
31         </Form.Item>
32
33       </Form>
34     );
35   }
36 }
37
38 export default Form.create({ name: 'role' })(AddRole);

```

## 修改权限

```

1  import React, { Component, PureComponent } from
"react";
2  import { Tree, Form, Input } from 'antd';
3  import menuConfig from "../../config/menuConfig";
4  const { TreeNode } = Tree;
5  const Item = Form.Item
6
7
8  class AddAuth extends PureComponent {
9    state = {

```



```

10     checkedKeys: [],
11 };
12
13 onCheck = checkedKeys => {
14     console.log('onCheck', checkedKeys);
15     this.setState({ checkedKeys });
16 };
17
18 // 给当前组件定制一个getMenus方法，目的供父级组件Role在
提交的时候，获取到当前最新的所有勾选的node的key数组
19 getMenus = () => this.state.checkedKeys;
20 renderTreeNodes = data =>
21     data.map(item => {
22         if (item.children) {
23             return (
24                 <TreeNode title={item.title} key={item.key}
dataRef={item}>
25                     {this.renderTreeNodes(item.children)}
26                 </TreeNode>
27             );
28         }
29         return <TreeNode key={item.key} {...item} />;
30     });
31 componentWillMount() {
32     // 根据传入角色的menus来更新默认选中checkedKeys状态
33     console.log(this.props.role);
34     const menus = this.props.role.menus;
35     this.setState({
36         checkedKeys: menus
37     });
38
39 }
40 /*
41 组件接收到新的标签属性时就会执行(初始显示时不会调用)
42 nextProps: 接收到的包含新的属性的对象

```

```
43  */
44  componentWillReceiveProps(nextProps) {
45      const menus = nextProps.role.menus;
46      this.setState({
47          checkedKeys: menus
48      })
49  }
50  render() {
51      const { role } = this.props;
52      const { checkedKeys } = this.state;
53      // 指定Item布局的配置对象
54      const formItemLayout = {
55          labelCol: { span: 4 }, // 左侧label的宽度
56          wrapperCol: { span: 15 }, // 右侧包裹的宽度
57      }
58      return (
59          <div>
60              <Item label='角色名称' {...formItemLayout}>
61                  <Input value={role.name} disabled />
62              </Item>
63              <Tree
64                  checkable
65                  onCheck={this.onCheck}
66                  checkedKeys={checkedKeys}
67                  defaultExpandAll
68              >
69                  <TreeNode title='平台权限' key='all'>
70                      {this.renderTreeNodes(menuConfig)}
71                  </TreeNode>
72              </Tree>
73          </div>
74      );
75  }
76 }
77
```

## 用户管理组件

User/user.js

```
1 import React, { Component } from 'react'
2 import { Card, Table, Button, Modal, message } from
  "antd";
3 import UserForm from './user-form';
4 import { reqUsers, reqRemoveUser, reqAddOrUpdateUser }
  from '../api';
5 // 用户路由
6 export default class User extends Component {
7   state = {
8     visible: false, // 是否显示确认框
9     users: [], // 所有用户列表
10    roles: [], // 所有角色列表
11  };
12  // 初始化表头数据
13  initColumns = () => {
14    this.columns = [
15      {
16        title: '用户名',
17        dataIndex: 'username'
18      },
19      {
20        title: '邮箱',
21        dataIndex: 'email'
22      },
23      {
24        title: '电话',
```

```
25         dataIndex: 'phone'
26     },
27     {
28         title: '注册时间',
29         dataIndex: 'create_time',
30         render: (create_time) => {
31             if (create_time) {
32                 return new
Date(create_time).toLocaleString()
33             } else {
34                 return;
35             }
36         }
37     },
38     {
39         title: '所属角色',
40         dataIndex: 'role_id',
41         render: role_id => this.state.roles.find(role
=> role._id === role_id).name
42     },
43     {
44         title: '操作',
45         render: user => (
46             <span>
47                 <Button onClick={() =>
this.showUpdate(user)}>修改</Button>
48                 <Button onClick={() =>
this.deleteUser(user)}>删除</Button>
49
50             </span>
51         )
52     }
53 ]
54 ]
55 }
```

```

56
57  /*
58    显示修改界面
59  */
60  showUpdate = (user) => {
61    this.user = user; //保存user
62    this.setState({
63      visible: true
64    })
65  }
66  // 删除指定的用户
67  deleteUser = async (user) => {
68    Modal.confirm({
69      title: `确认删除${user.username}用户吗?`,
70      onOk: async () => {
71        const res = await reqRemoveUser(user._id);
72        if (res.status === 0) {
73          message.success('删除用户成功');
74          this.getUsers();
75        } else {
76          message.error(res.msg)
77        }
78      }
79    })
80
81  }
82  // 点击OK之后, 增加用户还是更新用户
83  addOrUpdateUser = () => {
84    this.setState({ visible: false });
85
86    this.form.validateFields(async (err, values) => {
87      if (!err) {
88        // 如果this有user, 表示要更新了
89        if (this.user) {
90          values._id = this.user._id

```

```
91         }
92         const res = await reqAddOrUpdateUser(values);
93         if (res.status === 0) {
94             message.success('添加/更新用户成功!')
95             this.getUsers();
96         } else {
97             message.error(res.msg);
98         }
99
100     }
101 })
102 };
103 // 获取所有用户
104 getUsers = async () => {
105     const res = await reqUsers();
106     console.log(res);
107     if (res.status === 0) {
108         const { users, roles } = res.data;
109         this.setState({
110             users,
111             roles
112         })
113     }
114
115 }
116 componentWillMount() {
117     this.initColumns();
118 }
119 componentDidMount() {
120     this.getUsers();
121 }
122
123
124 // 取消之后的操作
125 handleCancel = e => {
```

```

126     this.user = null;
127     this.setState({
128         visible: false,
129     });
130 };
131 /*
132     显示添加界面 不需要user
133 */
134 showAdd = () => {
135
136     this.user = null; //去除前面保存的user
137     this.setState({
138         visible: true,
139     });
140 }
141 render() {
142     const createUserBtn = (
143         <Button type='primary' onClick={this.showAdd}>创
建用户</Button>
144     )
145     const { users, roles } = this.state;
146     const user = this.user || {}
147     return (
148         <Card title={createUserBtn} style={{ width:
'100%' }}>
149             <Table
150                 bordered
151                 rowKey='_id'
152                 dataSource={users}
153                 columns={this.columns}
154                 pagination={{ defaultPageSize: 2 }}
155             >
156
157             </Table>
158             <Modal

```

```

159         title={user._id ? '修改用户' : '添加用户'}
160         visible={this.state.visible}
161         onOk={this.addOrUpdateUser}
162         onCancel={this.handleCancel}
163     >
164         <UserForm
165             setForm={form => this.form = form}
166             roles={roles}
167             user={user}
168         ></UserForm>
169     </Modal>
170 </Card>
171 )
172 }
173 }
174

```

## User/user-form.js

```

1  import React, { Component } from 'react'
2  import {
3      Form,
4      Input,
5      Select,
6  } from 'antd';
7
8  const { Option } = Select;
9  class UserForm extends Component {
10     state = {
11         confirmDirty: false,
12         autoCompleteResult: [],
13     };
14
15     componentWillMount() {

```



```
16     this.props.setForm(this.props.form)
17   }
18   render() {
19     const { getFieldDecorator } = this.props.form;
20     const { roles, user } = this.props;
21     const formItemLayout = {
22       labelCol: { span: 4 },
23       wrapperCol: { span: 15 }
24     };
25     const prefixSelector = getFieldDecorator('prefix',
26   {
27     initialValue: '86',
28   }))(
29     <Select style={{ width: 70 }}>
30       <Option value="86">+86</Option>
31       <Option value="87">+87</Option>
32     </Select>,
33   );
34   return (
35     <Form {...formItemLayout}>
36       <Form.Item label="用户名">
37         {getFieldDecorator('username', {
38           initialValue:user.username,
39           rules: [
40             {
41               required: true,
42               message: '请输入用户名',
43             },
44           ],
45         })(
46           <Input placeholder='请输入用户名' />
47         </Form.Item>
48         {
49           user._id ? null : (
```

```
50         <Form.Item label="密码" hasFeedback>
51             {getFieldDecorator('password', {
52                 initialValue: user.password,
53                 rules: [
54                     {
55                         required: true,
56                         message: '请输入密码',
57                     },
58                 ],
59             })(<Input.Password placeholder='请输入密
60 码' />)}
61         </Form.Item>
62     )
63 }
64 <Form.Item label="手机号">
65     {getFieldDecorator('phone', {
66         initialValue: user.phone,
67         rules: [{ required: true, message: '请输入
68 正确的手机号码' }],
69     })(<Input addonBefore={prefixSelector}
70 style={{ width: '100%' }} placeholder='请输入手机号'
71 />)}
72 </Form.Item>
73
74 <Form.Item label="邮箱">
75     {getFieldDecorator('email', {
76         initialValue: user.email,
77         rules: [
78             {
79                 pattern: /^[a-zA-Z][0-9](\w|\.|-
80 )+@[a-zA-Z0-9]+\.[a-zA-Z]{2,4})$/,
81                 // required: true,
82                 message: '邮箱格式不正确',
83             },
84         ],
85     )}
```

```

80
81         ],
82         })(<Input placeholder='请输入邮箱' />)}
83     </Form.Item>
84
85     <Form.Item label="角色">
86         {getFieldDecorator('role_id', {
87             initialValue: user.role_id,
88             rules: [
89                 {
90                     required: true,
91                     message: '必须制定角色',
92                 },
93             ],
94         })(
95             <Select>
96                 {
97                     roles.map(role=><Option key={role._id}
value={role._id}>{role.name}</Option>)
98                 }
99             </Select>
100         )}
101     </Form.Item>
102 </Form>
103 );
104 }
105 }
106
107 export default Form.create({ name: 'user_form' })(
    UserForm);

```

## 图形图表组件

下载 echarts 和 echarts-for-react 插件

```
npm i echarts -S
```

```
npm i echarts-for-react -S
```

## 柱形组件

```
1 import React, { Component } from 'react'
2 import { Card, Button } from "antd";
3 import ReactEcharts from 'echarts-for-react';
4 export default class Charts extends Component {
5   state = {
6     sales: [30, 20, 36, 10, 10, 20], // 销量的数组
7     stores: [20, 10, 25, 20, 15, 10], // 库存的数组
8   }
9   update = ()=>{
10     this.setState(state=>({
11       sales:state.sales.map(sale=>sale+1),
12       stores:state.stores.reduce((pre,store)=>{
13         pre.push(store-1)
14         return pre;
15       },[])
16     })))
17   }
18   getOption = (sales, stores) => {
19     return {
20       title: {
21         text: 'ECharts 入门示例'
22       },
23       tooltip: {},
24       legend: {
25         data: ['销量', '库存'],
26       },
27       xAxis: {
```

```

28         data: ["Vue", "React", "Angular", "Nodejs", "微
信小程序", "自动化测试"]
29     },
30     yAxis: {},
31     series: [{
32         name: '销量',
33         type: 'bar',
34         data: sales
35     }, {
36         name: "库存",
37         type: 'bar',
38         data: stores
39     }]
40 }
41 }
42 render() {
43     const { sales, stores } = this.state;
44     return (
45         <div>
46             <Card>
47                 <Button type='primary' onClick={this.update}>
更新</Button>
48             </Card>
49             <ReactEcharts option={this.getOption(sales,
stores)}></ReactEcharts>
50         </div>
51     )
52 }
53 }
54

```

## 折线组件

```

1 import React, { Component } from 'react'

```

```
2 import { Card, Button } from 'antd'
3 import ReactEcharts from 'echarts-for-react'
4 export default class Line extends Component {
5   state = {
6     sales: [30, 20, 36, 10, 10, 20], // 销量的数组
7     stores: [20, 10, 25, 20, 15, 10], // 库存的数组
8   }
9   getOption = (sales, stores) => {
10     return {
11       xAxis: {
12         type: 'category',
13         data: ["Vue", "React", "Angular", "Nodejs", "微信小程序", "自动化测试"]
14       },
15       legend: {
16         data: ['销量', '库存']
17       },
18       yAxis: {
19         type: 'value'
20       },
21       series: [{
22         name: '销量',
23         type: 'line',
24         data: sales
25       }, {
26         name: '库存',
27         type: 'line',
28         data: stores
29       }]
30     };
31   }
32   update = () => {
33     this.setState(state => ({
34       sales: state.sales.map(sale => sale + 1),
35       stores: state.stores.reduce((pre, store) => {
```

```

36         pre.push(store - 1)
37         return pre
38     }, []),
39     )))
40 }
41 render() {
42     const {sales,stores} = this.state;
43     return (
44         <div>
45             <Card>
46                 <Button type='primary' onClick={this.update}>
更新</Button>
47             </Card>
48             <ReactEcharts option=
{this.getOption(sales,stores)}></ReactEcharts>
49         </div>
50     )
51 }
52 }
53

```

## 饼形组件

```

1  import React, { Component } from 'react'
2  import { Card } from "antd";
3  import ReactEcharts from 'echarts-for-react'
4  export default class Pie extends Component {
5      getOption = ()=>{
6          return {
7              title: {
8                  text: '某站点用户访问来源',
9                  subttext: '纯属虚构',
10                 left: 'center'
11             },

```

```

12     tooltip: {
13         trigger: 'item',
14         formatter: '{a} <br/>{b} : {c} ({d}%)'
15     },
16     legend: {
17         orient: 'vertical',
18         left: 'left',
19         data: ['直接访问', '邮件营销', '联盟广告', '视频
广告', '搜索引擎']
20     },
21     series: [
22         {
23             name: '访问来源',
24             type: 'pie',
25             radius: '55%',
26             center: ['50%', '60%'],
27             data: [
28                 { value: 335, name: '直接访问' },
29                 { value: 310, name: '邮件营销' },
30                 { value: 234, name: '联盟广告' },
31                 { value: 135, name: '视频广告' },
32                 { value: 1548, name: '搜索引擎' }
33             ],
34             emphasis: {
35                 itemStyle: {
36                     shadowBlur: 10,
37                     shadowOffsetX: 0,
38                     shadowColor: 'rgba(0, 0, 0, 0.5)'
39                 }
40             }
41         }
42     ]
43 };
44 }
45 getOption2 = ()=>{

```



```
46     return {
47         backgroundColor: '#2c343c',
48
49         title: {
50             text: 'Customized Pie',
51             left: 'center',
52             top: 20,
53             textStyle: {
54                 color: '#ccc'
55             }
56         },
57
58         tooltip: {
59             trigger: 'item',
60             formatter: '{a} <br/>{b} : {c} ({d}%)'
61         },
62
63         visualMap: {
64             show: false,
65             min: 80,
66             max: 600,
67             inRange: {
68                 colorLightness: [0, 1]
69             }
70         },
71         series: [
72             {
73                 name: '访问来源',
74                 type: 'pie',
75                 radius: '55%',
76                 center: ['50%', '50%'],
77                 data: [
78                     { value: 335, name: '直接访问' },
79                     { value: 310, name: '邮件营销' },
80                     { value: 274, name: '联盟广告' },
```

```
81         { value: 235, name: '视频广告' },
82         { value: 400, name: '搜索引擎' }
83     ].sort(function (a, b) { return a.value -
177 b.value; })),
84     roseType: 'radius',
85     label: {
86         color: 'rgba(255, 255, 255, 0.3)'
87     },
88     labelLine: {
89         lineStyle: {
90             color: 'rgba(255, 255, 255, 0.3)'
91         },
92         smooth: 0.2,
93         length: 10,
94         length2: 20
95     },
96     itemStyle: {
97         color: '#c23531',
98         shadowBlur: 200,
99         shadowColor: 'rgba(0, 0, 0, 0.5)'
100     },
101
102     animationType: 'scale',
103     animationEasing: 'elasticOut',
104     animationDelay: function (idx) {
105         return Math.random() * 200;
106     }
107 }
108 ]
109 };
110 }
111 render() {
112     return (
113         <div>
114             <Card title='饼图一'>
```

```

115         <ReactEcharts option={this.getOption()}
style={{ height: 300 }} />
116     </Card>
117     <Card title='饼图二'>
118         <ReactEcharts option={this.getOption2()}
style={{ height: 300 }} />
119     </Card>
120 </div>
121 )
122 }
123 }
124

```

## 使用React-redux

下载安装:

```
npm i redux -S;npm i react-redux
```

新建store/index.js

```

1 import { createStore } from "redux";
2 import userer from './user.reducer'
3
4 // 创建store 有state和reducer的store
5 const store = createStore(userer);
6 export default store;

```

新建store/user.reducer.js

```

1 // 创建reducer
2 function userReducer(user = {}, action) {
3     switch (action.type) {
4         case 'SETUSER':

```

```
5     user = action.user
6     return user;
7   case 'REMOVEUSER':
8     user = {};
9     return user;
10  default:
11    return user;
12  }
13 }
14 export const mapStateToProps = (state /*, ownProps*/)
=> {
15   return {
16     user: state
17   }
18 }
19 export const mapDispatchToProps = dispatch => {
20   return {
21     setUser: (user) => {
22       console.log(user);
23
24       dispatch({ type: 'SETUSER', user: user })
25     },
26     removeUser: () => {
27       dispatch({ type: 'REMOVEUSER' })
28     }
29   }
30 }
31 export default userReducer;
```

index.js中连接使用

```

1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import './api'
6 import { Provider } from "react-redux";
7 import store from './store'
8 ReactDOM.render((
9   <Provider store = {store}>
10     <App />
11   </Provider>
12 ), document.getElementById('root'));

```

修改Login/login.jsx

```

1 import React, { Component } from 'react'
2 import { Form, Icon, Input, Button, message } from
  'antd';
3 import { Redirect } from "react-router-dom";
4 import './Login.less'
5 import logo from '../assets/images/logo.svg'
6 import { reqLogin } from '../api';
7 import storageUtils from
  '../utils/storageUtils.js';
8 import memoryUtils from '../utils/memoryUtils';
9 import { connect } from "react-redux";
10 import { mapStateToProps, mapDispatchToProps } from
  '../store/user.reducer';
11 @connect(mapStateToProps, mapDispatchToProps)
12 class Login extends Component {
13   handleSubmit = e => {
14     e.preventDefault();
15     this.props.form.validateFields(async (err, values)
=> {

```

```
16         if (!err) {
17             // console.log('Received values of form: ',
values);
18             const { username, password } = values;
19             const res = await reqLogin(username,
password);
20             if (res.status === 0) {
21                 // 将用户信息保存到local中
22                 const user = res.data;
23                 // localStorage.setItem('user_key',
JSON.stringify(user))
24                 storageUtils.saveUser(user)
25
26                 // 保存到内存中
27                 // memoryUtils.user = user;
28                 // 修改
29                 this.props.setUser(user);
30                 // 跳转到管理页面
31                 this.props.history.replace('/')
32                 message.success('登录成功')
33             } else {
34                 message.error(res.msg);
35             }
36         } else {
37
38         }
39     });
40 };
41 /**
42  * 自定义校验规则
43  */
44 validatorPwd = (rule, value, callback) => {
45     value = value.trim();
46     if (!value) {
47         callback('密码必须输入')
```

```

48     } else if (value.length < 4) {
49         callback('密码不能小于4位')
50     } else if (value.length > 12) {
51         callback('密码不能大于12位')
52     } else if (!/^([a-zA-Z0-9_]+)$/.test(value)) {
53         callback('密码必须是英文、数字或下划线组成')
54     } else {
55         // 验证通过一定要callback,如果callback,验证无法响
    应
56         callback();
57     }
58 }
59 render() {
60     // 读取保存的user 如果存在, 直接跳转到管理界面
61     // const user =
JSON.parse(localStorage.getItem('user_key')) || {};
62     // const user = storageUtils.getUser();
63     //const user = memoryUtils.user
64     //修改
65     const user = this.props.user
66     if (user._id) {
67         // render中不能这样写 这种方法很一般在事件回调函数
    中进行路由跳转
68         // this.props.history.replace('/login')
69         return <Redirect to="/" />
70     }
71     const { getFieldDecorator } = this.props.form;
72     return (
73         <div className='login'>
74             <div className='login-header'>
75                 <img src={logo} alt="" />
76                 <h1>React后台管理系统</h1>
77             </div>
78             <div className="login-content">
79                 <h1>用户登录</h1>

```

```
80         <Form onSubmit={this.handleSubmit}
className="login-form">
81         <Form.Item>
82             {getFieldDecorator('username', {
83                 // 设置初始值
84                 initialValue: 'admin',
85                 // 声明式校验规则
86                 rules: [
87                     { required: true, whitespace: true,
message: '用户名是必须的!' },
88                     { min: 4, message: '密码不能小于4位!'
},
89                     { max: 12, message: '密码不能大于12
位!' },
90                     { pattern: /^[a-zA-Z0-9_]+$/,
message: '必须是英文、数字或下划线组成!' },
91                 ],
92             })(
93                 <Input
94                     prefix={<Icon type="user" style={{
color: 'rgba(0,0,0,.25)' }} />}
95                     placeholder="用户名"
96                 />,
97             )}
98         </Form.Item>
99         <Form.Item>
100             {getFieldDecorator('password', {
101                 rules: [{ validator: this.validatorPwd
}],
102             })(
103                 <Input
104                     prefix={<Icon type="lock" style={{
color: 'rgba(0,0,0,.25)' }} />}
105                     type="password"
106                     placeholder="密码"
```



```

107         />,
108     })
109     </Form.Item>
110     <Form.Item>
111         <Button type="primary" htmlType="submit"
112             className="login-form-button" style={{ width: '100%'
113             }}>
114             登录
115         </Button>
116     </Form.Item>
117 </Form>
118 </div>
119 </div>
120 )
121 }
122 }
123
124 export default Form.create({ name: 'normal_login' })(Login);

```

Admin/admin.jsx

```

1  import {connect} from 'react-redux';
2  import { mapStateToProps, mapDispatchToProps } from
3  "../../store/user.reducer";
4
5  @connect(mapStateToProps, mapDispatchToProps)
6  class Admin extends Component {
7      state = {
8          collapsed: false,
9      };
10
11     toggle = () => {
12         this.setState({

```

```
12     collapsed: !this.state.collapsed,
13   });
14 };
15 render() {
16   // 读取保存的user 如果不存在, 直接跳转到登录界面
17   // const user = storageUtils.getUser();
18   // const user = memoryUtils.user;
19   //修改
20   const user = this.props.user;
21   if (!user._id) {
22     // render中不能这样写 这种方法很一般在事件回调函数中
    进行路由跳转
23     // this.props.history.replace('/login')
24     return <Redirect to='/login' />
25   }
26   return (
27     <Layout style={{height: '100%'}}>
28       <LeftNav collapsed={this.state.collapsed}>
    </LeftNav>
29
30       <Layout>
31         <MHeader collapsed = {this.state.collapsed}
    toggle={this.toggle}/>
32         <Content
33           style={{
34             margin: '24px 16px',
35             padding: 24,
36             background: '#fff',
37             minHeight: 280,
38           }}
39         >
40           {/* 路由配置 */}
41           <Switch>
42             <Route path='/home' component={Home} />
```

```

43         <Route path='/category' component=
{Category} />
44         <Route path='/product' component=
{Product} />
45         <Route path='/role' component={Role} />
46         <Route path='/user' component={User} />
47         <Route path='/charts/bar' component={Bar}
/>
48         <Route path='/charts/line' component=
{Line}/>
49         <Route path='/charts/pie' component=
{Pie}/>
50         <Redirect to='/home' />
51     </Switch>
52 </Content>
53     <Footer style={{ textAlign: 'center' }}>Ant
Design ©2018 Created by Ant UED</Footer>
54 </Layout>
55 </Layout>
56 )
57
58 }
59 }
60 export default Admin;

```

## MHeader/index.js

```

1 import React, { Component } from 'react'
2 import { Layout, Button, Modal } from 'antd';
3 import { withRouter } from "react-router-dom";
4 import './index.less'
5 import memoryUtils from '../utils/memoryUtils';
6 import storageUtils from '../utils/storageUtils';
7 import menuList from '../config/menuConfig';

```

```
8 import { reqWeather } from '../..api';
9 import {connect} from 'react-redux'
10 import {mapStateToProps, mapDispatchToProps} from
  '../..store/user.reducer'
11 const { Header } = Layout;
12 const { confirm } = Modal;
13 @connect(mapStateToProps, mapDispatchToProps)
14 class MHeader extends Component {
15   constructor(props) {
16     super(props);
17     this.state = {
18       currentTime: new Date().toLocaleString(),
19       dayPictureUrl: '', // 图片url
20       weather: '', // 天气文本
21     }
22
23   }
24
25   logout = () => {
26     confirm({
27       title: '确定要退出登录吗？',
28       content: '',
29       onOk: () => {
30         storageUtils.removeUser();
31         // memoryUtils.user = {};
32         // 修改
33         this.props.removeUser()
34         this.props.history.replace('/home');
35       },
36       onCancel: () => {
37         console.log('Cancel');
38
39       },
40     });
41   }
```

```
42     //....
43     // 获取天气信息显示
44     getWeather = async ()=>{
45         const { dayPictureUrl, weather } = await
reqWeather('北京');
46         // 更新状态
47         this.setState({
48             dayPictureUrl,
49             weather
50         })
51     }
52 }
53
54 render() {
55     // const user = memoryUtils.user
56     // 修改
57     const user = this.props.user;
58     return (
59         <Header style={{ background: '#fff', padding: 0
60 }}>
61         <div className="header">
62             <h2 className='header-title'>
63 {this.getTitle()}</h2>
64             <div className='header-user'>
65                 <div className='currentTime'>
66 {this.state.currentTime}</div>
67                 <div className='weather'>天气
68 {this.state.weather}</div>
69                 <div className='weatherPic'>
70                     <img src={this.state.dayPictureUrl}
71 alt=""/>
72                 </div>
73                 <div className='userInfo'>
74                     欢迎,{user.username}
```

```

70         <Button style={{ marginLeft: '20px' }}
onClick={this.logout}>退出</Button>
71     </div>
72 </div>
73 </div>
74 </Header>
75 )
76 }
77 }
78 export default withRouter(MHeader)

```

保证刷新时状态不变，在App.js组件中更新user

```

1  import React, { Component } from 'react'
2  import { BrowserRouter, Switch, Route } from "react-
router-dom";
3  import Login from './pages/Login/login';
4  import Admin from './pages/Admin/admin';
5  import { connect } from "react-redux";
6  import { mapStateToProps, mapDispatchToProps } from
"./store/user.reducer";
7  import storageUtils from './utils/storageUtils'
8  @connect(mapStateToProps, mapDispatchToProps)
9  class App extends Component {
10
11      componentWillMount() {
12          const user = storageUtils.getUser();
13          this.props.setUser(user);
14      }
15
16      render() {
17          return (
18              <BrowserRouter>
19                  <Switch>

```

```
20         <Route path='/login' component={Login}>
    </Route>
21         <Route path='/' component={Admin}></Route>
22     </Switch>
23 </BrowserRouter>
24 )
25 }
26 }
27 export default App;
28
```

## 项目部署

---

### 登录服务器

用户名: ssh root@123.206.16.61

密码: xxxx

---

### 安装node二进制文件

```
cd /tmp/
```

```
wget https://nodejs.org/download/release/v10.15.3/node-
v10.15.3-linux-x64.tar.xz
```

### 解压node

```
xz -d node-v10.15.3-linux-x64.tar.xz :去除掉.xz后缀
```

```
tar -xf node-v10.15.3-linux-x64.tar
```

### 配置环境变量

```
1 ln -s /opt/node-v10.15.3-linux-x64/bin/node  
  /usr/local/sbin/  
2 ln -s /opt/node-v10.15.3-linux-x64/bin/npm  
  /usr/local/sbin/
```

## 安装pm2进程管理工具

```
1 npm install pm2 -g
```

## 安装mongodb

```
1 yum install mongodb-server mongodb -y 安装客户端和服务端  
2  
3 systemctl stop mongod 关闭mongodb服务  
4  
5 systemctl start mongod 启动mongodb服务  
6  
7 netstat -tunlp | grep mongod 查询应用的端口号  
8  
9 t: 表示查看tcp  
10  
11 u: 表示查看udp  
12  
13 n: 表示端口以数字形式表示, 没有n直接显示服务名。  
14  
15 l: 表示显示所监听的端口  
16  
17 p: 表示占用端口的进程  
18  
19 vim /etc/mongod.conf 查看配置文件
```

## 部署Node后端

- `git pull https://www.github.com/xiaomage/server`



- 模拟本地文件上传到服务器
- 本地终端运行: `scp ./admin-server_final.zip root@123.206.16.61:/tmp`
- 服务器终端运行: `unzip admin-server_final.zip && cd admin-server_final && npm install && pm2 start app.js`

## 部署Vue前端项目

前端打包文件:

```
1 npm run build
```

- 本地终端运行: `scp ./build.zip root@123.206.16.61:/tmp`
- 服务器终端运行: `unzip build.zip`

## 部署nginx

找到nginx的安装目录

```
1 //以我的服务器为例: nginx目录
2 cd /opt/nginx112/conf
3 vim nginx.conf
4
5 //修改配置文件如下
6 server {
7     listen      80; //端口号
8     server_name  www.pythonav.cn; //域名
9     location / {
10         try_files $uri $uri/ /index.html; #匹
11         配所有的路由
12         root /tmp/dist; //填写前端的根目录
13         index index.html index.htm;
14     }
```

```
14      }
```

输入nginx的启动命令

```
1 /opt/nginx112/sbin/nginx 第一次输入是启动
2 /opt/nginx112/sbin/nginx -s reload #平滑重启, 重新读取配置
  文件, 不重启进程
3 /opt/nginx112/sbin/nginx -s stop
```

访问<http://www.pythonav.cn> 查看xmall商城项目