- 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

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

引入路由

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

新建pages/Login/Login.js

```
import React, { Component } from 'react'
class Login extends Component {
  render() {
    return (
      <div className='login'>
        Login组件
      </div>
    )
  }
}

export default Login;
```

pages/Admin/Admin.jsx

```
import React, { Component } from 'react'
export default class Admin extends Component {
  render() {
    return (
      <div>
        Admin组件
      </div>
    )
  }
}

```

App.js

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

```
import Admin from './pages/Admin/Admin';
export default class App extends Component {
  render() {
    return (
      <BrowserRouter>
        <Switch>
          <Route path='/login' component={Login} />
          <Route path='/admin' component={Admin} />
        </Switch>
      </BrowserRouter>
    )
  }
}

```

重置样式&引入静态资源

重置样式 reset.css

public/style/reset.css

```
/*! minireset.css v0.0.6 | MIT License |
github.com/jgthms/minireset.css */
html,
body,
p,
ol,
ul,
li,
dl,
dt,
dd,
blockquote,
figure,
fieldset,
```

```
legend,
textarea,
pre,
iframe,
hr,
h1,
h2,
h3,
h4,
h5,
h6 {
  margin: 0;
  padding: 0;
}

h1,
h2,
h3,
h4,
h5,
h6 {
  font-size: 100%;
  font-weight: normal;
}

ul {
  list-style: none;
}

button,
input,
select,
textarea {
  margin: 0;
}
```

```
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

```less
.login {
  width: 100%;
  height: 100%;
  background-image: url('../../assets/images/bg.jpg');

  .login-header {
    height: 80px;
    background-color: rgba(0, 0, 0, 0.5);
    display: flex;
    align-items: center;

    img {
      width: 40px;
      height: 40px;
      margin-left: 30px;
      margin-right: 20px;
```

```
    }

    h1 {
      font-size: 30px;
      color: #fff;
      margin-bottom: 0;
    }
  }

  .login-content {
    width: 400px;
    height: 300px;
    background-color: #fff;
    margin: 100px auto;
    padding: 30px;
    border-radius: 3px;
    h1 {
      font-size: 24px;
      font-weight: bold;
      text-align: center;
      color: #000;
    }
  }
}
```

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

Login/Login.jsx

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

```
  handleSubmit = e => {
    e.preventDefault();
    this.props.form.validateFields((err, values) => {
      if (!err) {
        // console.log('Received values of form: ', values);
        const { username, password } = values;
      }
    });
  };
  /**
   * 自定义校验规则
   */
  validatorPwd = (rule, value, callback)=>{
    value = value.trim();
    if(!value){
      callback('密码必须输入')
    }else if(value.length < 4){
      callback('密码不能小于4位')
    } else if (value.length > 12) {
      callback('密码不能大于12位')
    }else if(!/^[a-zA-Z0-9_]+$/.test(value)){
      callback('密码必须是英文、数字或下划线组成')
    }else{
      // 验证通过一定要callback,如果callback,验证无法响应
      callback();
    }
  }
  render() {
    const { getFieldDecorator } = this.props.form;
    return (
      <div className='login'>
        <div className='login-header'>
          <img src={logo} alt="" />
          <h1>React后台管理系统</h1>
```

```
        </div>
        <div className="login-content">
          <h1>用户登录</h1>
          <Form onSubmit={this.handleSubmit}
className="login-form">
            <Form.Item>
              {getFieldDecorator('username', {
                // 设置初始值
                initialValue: 'admin',
                // 声明式校验规则
                rules: [
                  { required: true, whitespace: true,
message: '用户名是必须的!' },
                  { min: 4, message: '密码不能小于4位!'
},
                  { max: 12, message: '密码不能大于12
位!' },
                  { pattern: /^[a-zA-Z0-9_]+$/,
message: '必须是英文、数字或下划线组成! ' },
                ],
              })(
                <Input
                  prefix={<Icon type="user" style={{
color: 'rgba(0,0,0,.25)' }} />}
                  placeholder="用户名"
                />,
              )}
            </Form.Item>
            <Form.Item>
              {getFieldDecorator('password', {
                rules: [{ validator: this.validatorPwd
}],
              })(
                <Input
```

```
                  prefix={<Icon type="lock" style={{ color: 'rgba(0,0,0,.25)' }} />}
                  type="password"
                  placeholder="密码"
                />,
              )}
            </Form.Item>
            <Form.Item>
              <Button type="primary" htmlType="submit" className="login-form-button" style={{ width: '100%' }}>
                登录
              </Button>
            </Form.Item>
          </Form>
        </div>
      </div>
    )
  }
}

export default Form.create({ name: 'normal_login' })(Login);;
```

Login组件(交互)

封装ajax请求模块

下载axios库

```
yarn add axios
```

新建api/ajax.js

```
/*
  封装的能发ajax请求的函数
*/
import axios from 'axios'
import qs from 'qs'
import { message } from "antd";
/**
 * 请求拦截器 在发送请求数据之前，提前对数据做处理
 */
axios.interceptors.request.use(function (config) {
  const { method, data } = config;
  if (method.toLocaleLowerCase() === 'post' && typeof data === 'object') {
    config.data = qs.stringify(data);
  }
  return config;
}, function (error) {
  // Do something with request error
  return Promise.reject(error);
});
/*
响应拦截器，响应数据之前做工作
*/
axios.interceptors.response.use(function (response) {
  return response.data;
}, function (error) {
  message.error('请求失败了:'+error.message);
  // return Promise.reject(error);
  // 让错误处于pending状态，不再往下执行，
  //返回一个pending的promise，中断promise链
  return new Promise(()=>{})
});

export default axios;
```

新建api/index.js

```
/*
包含应用中所有请求接口的函数：接口请求函数
*/
import ajax from './ajax'
const BASE = '';
// 请求登录
export const reqLogin = (username, password) => {
  return ajax({
    method: 'post',
    url: BASE + '/login',
    data: {
      // 默认使用json格式的请求体携带参数数据
      username,
      password
    }
  })
}

```

配置代理

package.json

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

登录功能

Login/Login.js

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

```

```
import { Form, Icon, Input, Button, message } from 'antd';
import { Redirect } from "react-router-dom";
import './Login.less'
import logo from '../../assets/images/logo.svg'
import { reqLogin } from '../../api';
class Login extends Component {
  handleSubmit = e => {
    e.preventDefault();
    this.props.form.validateFields(async (err, values) => {
      if (!err) {

        const { username, password } = values;
        const res = await reqLogin(username, password);
        if (res.status === 0) {
          // 将用户信息保存到local中
          const user = res.data;
          // 保存用户信息
          localStorage.setItem('user_key', JSON.stringify(user))
          // 跳转到管理页面
          this.props.history.replace('/')
          message.success('登录成功')
        } else {
          message.error(res.msg);
        }
      } else {

      }
    });
  };
  /**
   * 自定义校验规则
   */
```

```
  validatorPwd = (rule, value, callback) => {
    ....
  }
  render() {
    // 读取保存的user 如果存在，直接跳转到管理界面
    const user = JSON.parse(localStorage.getItem('user_key')) || {};
    if (user._id) {
      // 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
      // this.props.history.replace('/login')
      return <Redirect to='/' />
    }
    const { getFieldDecorator } = this.props.form;
    return (
      <div className='login'>
        ...
      </div>
    )
  }
}

export default Form.create({ name: 'normal_login' })(Login);;
```

Admin/Admin.js

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

```
    if (!user._id) {
      // 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
      // this.props.history.replace('/login')
      return <Redirect to='/login' />
    }
    return (
      <div>
        hello,Admin组件
      </div>
    )
  }
}

```

封装storageUtils模块

新建utils/storageUtils.js

```
export default {
  saveUser(user){

 localStorage.setItem('user_key',JSON.stringify(user))
  },
  getUser(){
    return JSON.parse(localStorage.getItem('user_key')) || {}
  },
  removeUser(){
    localStorage.removeItem('user_key')
  }

}
```

修改Admin.js

```
import React, { Component } from 'react'
import { Redirect } from "react-router-dom";
import storageUtils from '../../utils/storageUtils.js';
export default class Admin extends Component {
  render() {
    // 读取保存的user 如果不存在，直接跳转到登录及诶面
    const user = storageUtils.getUser();
    if (!user._id) {
      // 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
      // this.props.history.replace('/login')
      return <Redirect to='/login' />
    }
    return (
      <div>
        Admin组件
      </div>
    )

  }
}

```

修改Login/Login.js

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

```

```
class Login extends Component {
  handleSubmit = e => {
    e.preventDefault();
    this.props.form.validateFields(async (err, values)
=> {
      if (!err) {
        const { username, password } = values;
        const res = await reqLogin(username, password);
        if (res.status === 0) {
          // 将用户信息保存到local中
          const user = res.data;
          // 保存用户信息
          storageUtils.saveUser(user)
          // 跳转到管理页面
          this.props.history.replace('/')
          message.success('登录成功')
        } else {
          message.error(res.msg);
        }
      } else {

      }
    });
  };
  /**
   * 自定义校验规则
   */
  validatorPwd = (rule, value, callback) => {
    ....
  }
  render() {
    // 读取保存的user 如果存在，直接跳转到管理界面
     const user = storageUtils.getUser();
    if (user._id) {
```

```
      // 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
      // this.props.history.replace('/login')
      return <Redirect to='/' />
    }
    const { getFieldDecorator } = this.props.form;
    return (
      <div className='login'>
        ...
      </div>
    )
  }
}

export default Form.create({ name: 'normal_login' })(Login);;
```

使用store插件封装存储模块

安装

```
npm i store -S
```

修改storageUtils.js

```
/*
 操作local数据的工具函数模块
*/
import store from 'store'
const USER_KEY = 'user_key'
export default {
  saveUser(user){
    store.set(USER_KEY,user);
  },
```

```
  getUser(){
    return store.get(USER_KEY) || {}
  },
  removeUser(){
    store.remove(USER_KEY)
  }

}
```

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

```
import storageUtils from "./storageUtils"

// 初始时取一次并保存为user
const user = storageUtils.getUser()
export default {
  user, // 用来存储登陆用户的信息，初始值为local中读取的user
}
```

修改Amdin.js

```
import React, { Component } from 'react'
import { Redirect } from "react-router-dom";
import storageUtils from '../../utils/storageUtils.js';
import memoryUtils from '../../utils/memoryUtils'

export default class Admin extends Component {
```

```
  render() {
    // 读取保存的user 如果不存在，直接跳转到登录及诶面
    // const user = storageUtils.getUser();
    const user = memoryUtils.user;
    if (!user._id) {
      // 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
      // this.props.history.replace('/login')
      return <Redirect to='/login' />
    }
    return (
      <div>
        Admin组件
      </div>
    )

  }
}

```

修改Login.js

```
import React, { Component } from 'react'
import { Form, Icon, Input, Button, message } from 'antd';
import { Redirect } from "react-router-dom";
import './Login.less'
import logo from '../../assets/images/logo.svg'
import { reqLogin } from '../../api';
import storageUtils from '../../utils/storageUtils.js';
import memoryUtils from '../../utils/memoryUtils';

class Login extends Component {
  handleSubmit = e => {

```

```
    e.preventDefault();
    this.props.form.validateFields(async (err, values) => {
      if (!err) {
        // console.log('Received values of form: ', values);
        const { username, password } = values;
        const res = await reqLogin(username, password);
        if (res.status === 0) {
          // 将用户信息保存到local中
          const user = res.data;
          // localStorage.setItem('user_key', JSON.stringify(user))
          storageUtils.saveUser(user)
          // 不要忘记将用户信息保存到内存中
          memoryUtils.user = user;
          // 跳转到管理页面
          this.props.history.replace('/')
          message.success('登录成功')
        } else {
          message.error(res.msg);
        }
      } else {

      }
    });
  };
  /**
   * 自定义校验规则
   */
  validatorPwd = (rule, value, callback) => {
    value = value.trim();
    if (!value) {
      callback('密码必须输入')
    } else if (value.length < 4) {
```

```
      callback('密码不能小于4位')
    } else if (value.length > 12) {
      callback('密码不能大于12位')
    } else if (!/^[a-zA-Z0-9_]+$/.test(value)) {
      callback('密码必须是英文、数字或下划线组成')
    } else {
      // 验证通过一定要callback,如果callback,验证无法响应
      callback();
    }
  }
  render() {
    // 读取保存的user 如果存在，直接跳转到管理界面
    // const user = JSON.parse(localStorage.getItem('user_key')) || {};
    // const user = storageUtils.getUser();
    const user = memoryUtils.user
    if (user._id) {
      // 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
      // this.props.history.replace('/login')
      return <Redirect to='/' />
    }
    const { getFieldDecorator } = this.props.form;
    return (
      <div className='login'>
        ....
      </div>
    )
  }
}

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

```
npm i antd -S
```

按需加载引入

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

package.json修改

```
{
    "scripts": {
    "start": "react-app-rewired start",
    "build": "react-app-rewired build",
    "test": "react-app-rewired test",
    "eject": "react-app-rewired eject"
  },
}
```

根目录下新建config-overrides.js用于修改默认配置

支持装饰器

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

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

```
  addBabelPlugins //babel插件配置函数
} = require('customize-cra');
module.exports = override(
  fixBabelImports('import', {
    libraryName: 'antd',
    libraryDirectory: 'es',
    style: 'css',
  }),
  addBabelPlugins( // 支持装饰器
    [
      '@babel/plugin-proposal-decorators',
      {
        legacy: true
      }
    ]
  )
);
```

Admin/Admin.jsx

```
import React, { Component } from 'react'
import { Redirect } from "react-router-dom";
import storageUtils from '../../utils/storageUtils.js';
import memoryUtils from '../../utils/memoryUtils'
import './Admin.less'
import LeftNav from '../../components/LeftNav'
import MHeader from '../../components/MHeader'
import { Switch,Route } from "react-router-dom";


import Home from '../Home/Home';
import Category from '../Category/Category';
import Product from '../Product/Product';
import Role from '../Role/Role';
```

```
import User from '../User/User';
import Bar from '../Charts/Bar';
import Line from '../Charts/Line';
import Pie from '../Charts/Pie';

import { Layout } from 'antd';

const { Content, Footer } = Layout;

export default class Admin extends Component {
  state = {
    collapsed: false,
  };

  toggle = () => {
    this.setState({
      collapsed: !this.state.collapsed,
    });
  };
  render() {
    // 读取保存的user 如果不存在，直接跳转到登录及诶面
    // const user = storageUtils.getUser();
    const user = memoryUtils.user;
    if (!user._id) {
      // 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
      // this.props.history.replace('/login')
      return <Redirect to='/login' />
    }
    return (
      <Layout style={{height:'100%'}}>
        <LeftNav collapsed={this.state.collapsed}></LeftNav>

        <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  }
```

```
77
```

LeftNav组件

```
import React, { Component } from 'react'
import { Link } from "react-router-dom";
import { Layout, Menu, Icon } from "antd";
import Logo from '../../assets/images/logo192.png';
import './index.less'
const { Sider } = Layout;
const { SubMenu } = Menu;
export default class LeftNav extends Component {
  render() {
    return (
      <Sider collapsible collapsed={this.props.collapsed}>
        <div className='leftnav'>
          <div className="logo">
            <Link to='/home' className='left-nav-link'>
              <img src={Logo} alt="" />
              <h1>小马哥后台</h1>
            </Link>
            <Menu theme="dark" defaultSelectedKeys={['1']} mode="inline">
              <Menu.Item key="/home">
                <Link to='/home'>
                  <Icon type="home" />
                  <span>首页</span>
                </Link>
              </Menu.Item>
              <SubMenu
                key="sub1"
```

```
              title={
                <span>
                  <Icon type="project" />
                  <span>
                    商品
                  </span>
                </span>
              }
            >
              <Menu.Item key="/category">
              <Link to='/category'>
                <Icon type="menu-fold" />
                <span>
                  品类管理
                </span>
              </Link>

              </Menu.Item>
              <Menu.Item key="/product">
               <Link to='/product'>
                <Icon type="pic-right" />
                <span>
                  商品管理
                </span>
               </Link>

              </Menu.Item>
            </SubMenu>

            <Menu.Item key="/user">
            <Link to='/user'>
            <Icon type="user" />
              <span>
                用户管理
              </span>
```

```
            </Link>

            </Menu.Item>
            <Menu.Item key="/role">
            <Link to='/role'>
              <Icon type="ant-design" />
              <span>
                角色管理
              </span>
            </Link>

            </Menu.Item>
            <SubMenu
              key="sub3"
              title={
                <span>
                  <Icon type="line-chart" />
                  <span>
                    图形图标
                  </span>
                </span>
              }
            >
              <Menu.Item key="/charts/bar">
                <Link to='/charts/bar'>
                  <Icon type="bar-chart" />
                  <span>
                    柱形图
                  </span>
                </Link>
              </Menu.Item>
              <Menu.Item key="/charts/line">
                <Link to='/charts/line'>
                  <Icon type="line-chart" />
                  <span>
```

```
                        折线图
                      </span>
                    </Link>
                  </Menu.Item>
                  <Menu.Item key="/charts/pie">
                  <Link to='/charts/pie'>
                     <Icon type="pie-chart" />
                    <span>
                      饼形图
                    </span>
                  </Link>
                  </Menu.Item>
                </SubMenu>
              </Menu>
            </div>
          </div>
        </Sider>

      )
    }
}
```

LeftNav/index.less

```
.logo {
  height: 32px;
  margin: 16px;

  .left-nav-link {
    display: flex;
    align-items: center;
    height: 80px;

    img {
```

```
      width: 40px;
      height: 40px;
      margin: 0 10px;
    }

    h1 {
      color: #fff;
      font-size: 18px;
      margin-bottom: -5px;
    }
  }
}

a {
  color: rgba(255, 255, 255, 0.65);

}
```

侧边栏配置数据

config/menConfig.js

```
const menuList = [
  {
    title: '首页', // 菜单标题名称
    key: '/home', // 对应的path
    icon: 'home', // 图标名称
    public: true, // 公开的
  },
  {
    title: '商品',
    key: '/products',
    icon: 'appstore',
    children: [ // 子菜单列表
```

```
      {
        title: '品类管理',
        key: '/category',
        icon: 'bars'
      },
      {
        title: '商品管理',
        key: '/product',
        icon: 'tool'
      },
    ]
  },

  {
    title: '用户管理',
    key: '/user',
    icon: 'user'
  },
  {
    title: '角色管理',
    key: '/role',
    icon: 'safety',
  },

  {
    title: '图形图表',
    key: '/charts',
    icon: 'area-chart',
    children: [
      {
        title: '柱形图',
        key: '/charts/bar',
        icon: 'bar-chart'
      },
      {
```

```
        title: '折线图',
        key: '/charts/line',
        icon: 'line-chart'
      },
      {
        title: '饼图',
        key: '/charts/pie',
        icon: 'pie-chart'
      },
    ]
  },
]

export default menuList
```

封装侧边栏动态菜单

```
import React, { Component } from 'react'
import { Link } from "react-router-dom";
import { Layout, Menu, Icon } from "antd";
import Logo from '../../assets/images/logo192.png';
import './index.less'
import menuList from '../../config/menuConfig';

const { Sider } = Layout;
const { SubMenu } = Menu;
export default class LeftNav extends Component {
  /*
  根据指定菜单数据列表产生<Menu>的子节点数组
  使用 reduce() + 递归
  */
  getMenuNodes2 = (menulist) => {
```

```
    return menulist.reduce((pre, item) => {
      if (!item.children) {
        pre.push((
          <Menu.Item key={item.key}>
            <Link to={item.key}>
              <Icon type={item.icon} />
              <span>{item.title}</span>
            </Link>
          </Menu.Item>
        ))
      } else {
        pre.push((
          <SubMenu
            key={item.key}
            title={
              <span>
                <Icon type={item.icon} />
                <span>
                  {item.title}
                </span>
              </span>
            }
          >
            {this.getMenuNodes(item.children)}
          </SubMenu>
        ))
      }
      return pre;
    }, [])
  }

  /*
    根据指定菜单数据列表产生<Menu>的子节点数组
    使用map()+递归
  */
```

```
getMenuNodes = (menuList) => {
  return menuList.map(item => {
    if (!item.children) {
      return (
        <Menu.Item key={item.key}>
          <Link to={item.key}>
            <Icon type={item.icon} />
            <span>{item.title}</span>
          </Link>
        </Menu.Item>
      )
    } else {
      return (
        <SubMenu
          key={item.key}
          title={
            <span>
              <Icon type={item.icon} />
              <span>
                {item.title}
              </span>
            </span>
          }
        >
          {this.getMenuNodes(item.children)}
        </SubMenu>
      )
    }
  })
}

componentWillMount() {
  this.menuNodes = this.getMenuNodes2(menuList);
}

```

```
  render() {
    return (
      <Sider collapsible collapsed={this.props.collapsed}>
        <div className='leftnav'>
          <div className="logo">
            <Link to='/home' className='left-nav-link'>
              <img src={Logo} alt="" />
              <h1>小马哥后台</h1>
            </Link>
            <Menu theme="dark" defaultSelectedKeys={['1']} mode="inline">
              {this.menuNodes}
            </Menu>
          </div>
        </div>
      </Sider>

    )
  }
}

```

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`

```
import React, { Component } from 'react'
import { Link, withRouter } from "react-router-dom";
//....
class LeftNav extends Component{
    //....
    render(){
        let defaultKey = this.props.location.pathname;
        console.log(defaultKey);
    }

}
export default withRouter(LeftNav);
```

自动打开二级菜单

```
import React, { Component } from 'react'
import { Link, withRouter } from "react-router-dom";
import { Layout, Menu, Icon } from "antd";
import Logo from '../../assets/images/logo192.png';
import './index.less'
import menuList from '../../config/menuConfig';

const { Sider } = Layout;
const { SubMenu } = Menu;
class LeftNav extends Component {
  /*
  根据指定菜单数据列表产生<Menu>的子节点数组
  使用 reduce() + 递归
  */
  getMenuNodes2 = (menulist) => {
    const path = this.props.location.pathname;
    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匹配，将菜单的key保存在openKey
30          /*
31            判断当前item的key是否是我需要的key
32            查找item的所有children中cItem的key,k按是否有一个跟请求的path匹配
33          */
34          const cItem = item.children.find(cItem => cItem.key === path)
35          if (cItem) {
36            this.openKey = item.key;
37          }
38
39          pre.push((
40            <SubMenu
41              key={item.key}
42              title={
43                <span>
44                  <Icon type={item.icon} />
45                  <span>
46                    {item.title}
47                  </span>
48                </span>
49              }
```

```
          >
            {this.getMenuNodes(item.children)}
          </SubMenu>
        ))
      }
      return pre;
    }, [])
  }

  render() {
    let defaultKey = this.props.location.pathname;
    return (
      <Sider collapsible collapsed={this.props.collapsed}>
        <div className='leftnav'>
          <div className="logo">
            <Link to='/home' className='left-nav-link'>
              <img src={Logo} alt="" />
              <h1>小马哥后台</h1>
            </Link>
            <Menu
              theme="dark"
              defaultSelectedKeys={[defaultKey]}
              mode="inline"
              defaultOpenKeys={[this.openKey]}
            >
              {this.menuNodes}
            </Menu>

          </div>
        </div>
      </Sider>

    )
  }
```

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

```

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

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

Header组件

MHeader/index.js

```
import React, { Component } from 'react'
import { Layout,Button } from 'antd';
import './index.less'
const { Header } = Layout;
export default class MHeader extends Component {
  logout = ()=>{
    alert('退出了')
  }
  render() {
    return (
      <Header style={{ background: '#fff', padding: 0
}}>
        <div className="header">
          <h2 className='header-title'>首页</h2>
```

```
          <div className='header-user'>
            <div className='currentTime'>2020-02-30 12:00:08</div>
            <div className='weather'>天气晴</div>
            <div class='userInfo'>
              欢迎,admin
              <Button style={{ marginLeft: '20px' }} onClick={this.logout}>退出</Button>
            </div>
          </div>
        </div>
      </Header>
    )
  }
}

```

MHeader/index.less

```
.header{
  display: flex;
  justify-content: space-between;
  .header-title{
    margin-left: 100px;
    font-size: 30px;
    color: #666;
  }
  .header-user{
    display: flex;
    margin-right: 100px;
    .currentTime{
      margin-right: 10px;
    }
    .weather{

```

```
      margin-right: 10px;
    }
  }
}
```

退出登录

MHeader/index.js

```
import React, { Component } from 'react'
import { Layout, Button, Modal } from 'antd';
import { withRouter } from "react-router-dom";
import './index.less'
import memoryUtils from '../../utils/memoryUtils';
import storageUtils from '../../utils/storageUtils';
const { Header } = Layout;
const { confirm } = Modal;
class MHeader extends Component {
  logout = () => {
    confirm({
      title: '确定要退出登录吗? ',
      content: '',
      onOk: () => {
        storageUtils.removeUser();
        memoryUtils.user = {};
        this.props.history.replace('/home');
      },
      onCancel: () => {
        console.log('Cancel');

      },
    });
  }
  render() {
```

```
    const user = memoryUtils.user
    return (
      <Header style={{ background: '#fff', padding: 0 }}>
        <div className="header">
          <h2 className='header-title'>首页</h2>
          <div className='header-user'>
            <div className='currentTime'>2020-02-30 12:00:08</div>
            <div className='weather'>天气晴</div>
            <div class='userInfo'>
              欢迎,{user.username}
              <Button style={{ marginLeft: '20px' }} onClick={this.logout}>退出</Button>
            </div>
          </div>
        </div>
      </Header>
    )
  }
}
export default withRouter(MHeader)
```

动态显示标题

MHeader/index.js

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

```
        } else if (item.children) {
            const cItem = item.children.find(cItem=>path===cItem.key);
            if(cItem){
                title = cItem.title;
            }
        }
    })
    return title;
}
```

动态显示当前时间

```
//...
class MHeader extends Component {
  constructor(props) {
    super(props);
    this.state = {
      currentTime: new Date().toLocaleString()
    }
  }

  // 动态显示当前时间
  componentDidMount(){
    // 开启定时器
    this.timer = setInterval(() => {
      this.setState({
        currentTime: new Date().toLocaleString()
      })
    }, 1000);
  }
  componentWillMount(){
    clearInterval(this.timer)
  }
```

```
  render() {
    const user = memoryUtils.user
    return (
      <Header style={{ background: '#fff', padding: 0 }}>
        <div className="header">
          <h2 className='header-title'>{this.getTitle()}</h2>
          <div className='header-user'>
            <div className='currentTime'>{this.state.currentTime}</div>
            ....
          </div>
        </div>
      </Header>
    )
  }
}
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

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

```
    const url = `http://api.map.baidu.com/telematics/v3/weather?location=${city}&output=json&ak=3p49MVra6urFRGOT9s8UBWr2`
    jsonp(url, {}, (error, data) => {
      if (!error && data.error === 0) {
        const { dayPictureUrl, weather } = data.results[0].weather_data[0]
        // 成功的
        resolve({ dayPictureUrl,weather});
      } else {
        message.error('获取天气信息失败')
      }
    })
  })

}
```

MHeader/index.js

```
import React, { Component } from 'react'
import { reqWeather } from '../../api';
//....
class MHeader extends Component {
  constructor(props) {
    super(props);
    this.state = {
      dayPictureUrl: '', // 图片url
      weather: '', // 天气文本
    }

  }
  // 动态显示当前时间
  componentDidMount(){
```

```
    // 发jsonp请求获取天气信息显示
    this.getWeather();
  }

  // 获取天气信息显示
  getWeather = async ()=>{
    const { dayPictureUrl, weather } = await
reqWeather('北京');
    // 更新状态
    this.setState({
      dayPictureUrl,
      weather
    })

  }
  render() {
    return (
      <Header style={{ background: '#fff', padding: 0
}}>
        <div className="header">
          <div className='header-user'>
            <div className='weather'>天气
{this.state.weather}</div>
            <div className='weatherPic'>
              <img src={this.state.dayPictureUrl}
alt=""/>
            </div>
        </div>
      </Header>
    )
  }
}
export default withRouter(MHeader)
```

分类组件

api/index.js

```
// 获取分类列表
export const reqCategorys = () => {
  return ajax(BASE +'/manage/category/list')
}
// 修改分类
export const reqUpdateCategory =
({categoryId,categoryName})=>{
  return ajax.post(BASE + '/manage/category/update',{
    categoryId,
    categoryName
  })
}
// 添加分类
export const reqAddCategory = ({ categoryName }) => {
  return ajax.post(BASE + '/manage/category/add', {
    categoryName
  })
}
```

Category/category.js

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

```
    showStatus: 0,//0:不显示 1:显示添加 2:显示修改
  }
  getCategorys = async () => {
    const res = await reqCategorys();
    if (res.status === 0) {
      // 成功了
      this.setState({
        categorys: res.data,

      })
    } else {
      message.error('获取分类列表失败了')
    }

  }
  initColumns = () => {
    this.columns = [
      {
        title: '分类的名称',
        dataIndex: 'name',
      },
      {
        title: '操作',
        width: 300,
        render: (category) => <Button type='primary' onClick={() => {
          console.log(category);

          this.category = category; //保存当前分类，其它地方都可以读取到
          this.setState({
            showStatus: 2
          })
        }}>
          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({
categoryId, categoryName })
70          }
71
72          // 重置输入的数据(变成初始值)
73          this.form.resetFields();
```

```
        this.setState({ showStatus: 0 })
        const action = showStatus === 1 ? '添加' : '修改';
        // 根据响应结果，做不同处理
        if (result.status === 0) {
          // 重新获取分类列表显示
          this.getCategorys();
          message.success(action + '分类成功');
        } else {
          message.error(action + '分类失败')
        }
      }
    })
  };

  handleCancel = e => {

    // 重置表单
    this.form.resetFields();
    this.setState({
      showStatus: 0
    });
  };
  render() {
    // Cart右上角的结构
    const extra = (
      <Button type='primary' onClick={() => {
        this.setState({
          showStatus: 1
        })
      }}>
        <Icon type="plus" />
        添加
      </Button>
    )
```

```
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.categorys}
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

```jsx
import React, { Component } from 'react'
import PropTypes from 'prop-types';
import { Form, Input } from "antd";
class AddUpdateForm extends Component {
  static propTypes = {
    setForm: PropTypes.func.isRequired,
    categoryName: PropTypes.string
  }
  componentWillMount() {
    this.props.setForm(this.props.form)
  }
  render() {
    const { getFieldDecorator } = this.props.form;
    const { categoryName } = this.props;
    return (
      <Form>
        <Form.Item>
          {
            getFieldDecorator('categoryName',{
              initialValue:categoryName || '',
              rules:[
                {
                  required:true,
                  message:'分类名称必须输入'
                }
              ]
            })(
              <Input type='text' placeholder='请输入分类名称' />
            )
          }
        </Form.Item>
      </Form>
    )
```

```
  }
}
export default Form.create()(AddUpdateForm);

```

商品分页组件

获取商品分页显示

接口

```
export const reqProducts = (pageNum, pageSize) => {
  return ajax(BASE + '/manage/product/list', {
    params: {
      pageNum,
      pageSize
    }
  })
}
```

Product/product.js

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

```
    loading: false,
    searchType: 'productName', //默认是按商品名称搜索
    searchName: '', //搜索的关键字
    products: [], //商品列表
    total: 0,//商品的总数量
  }
  /*
  异步获取指定页码商品分页(可能带搜索)列表显示
  */
  getProducts = async (pageNum) => {
    // 保存当前请求的页码
    this.pageNum = pageNum;
    const { searchName, searchType } = this.state;
    let result = await reqProducts(pageNum, PAGE_SIZE)
    if (result.status === 0) {
      const { list, total } = result.data;

      this.setState({
        total,
        products: list
      })
    }
  }
  updateStatus = (id, status) => {
    console.log(id, status);

  }
  initColumns = () => {
    this.columns = [
      {
        title: '商品名称',
        dataIndex: 'name'
      },
      {
        title: '商品描述',
```

```
        dataIndex: 'desc'
      },
      {
        title: '价格',
        width: 100,
        dataIndex: 'price',
        render: price => '¥' + price
      },
      {
        title: '状态',
        width: 100,
        // dataIndex: 'status'
        render: ({ _id, status }) => {
          let btnText = '下架', text = '在售';
          if (status === 2) {
            btnText = '上架';
            text = '已下架';
          }
          return (
            <span>
              <Button onClick={this.updateStatus(_id, status)}>{btnText}</Button>
              <span className='status'>{text}</span>
            </span>
          )

        }
      },
      {
        title: '操作',
        width: 100,
        dataIndex: 'done',
        render: (product) => {
          return <span>
            <Button
```

```
              type='link'
              onClick={() => {
                // 跳转商品详情页面
              }}
            >详情</Button>
            <Button
              type='link'
              onClick={() => {
                // 跳转商品修改页面
              }}
            >修改</Button>
          </span>
        }
      }
    ]
  }
  componentWillMount() {
    this.initColumns()
  }
  componentDidMount() {
    // 获取第一页显示
    this.getProducts(1);
  }
  render() {
    const { total, searchType, searchName, products, loading } = this.state;
    const title = (
      <span>
        <Select value={searchType} style={{ width: 200 }} onChange={value => this.setState({ searchType: value })}>
          <Option value="productName">按名称搜索</Option>
          <Option value="productDesc">按描述搜索</Option>
```

```
        </Select>
        <Input
          placeholder="关键字"
          style={{ width: 200, margin: '0 10px' }}
          value={searchName}
          onChange={e => this.setState({ searchName: e.target.value })}
        />
        <Button type='primary' onClick={() => {
            //搜索功能
        }}>搜索</Button>
      </span>
    )
    const extra = (
      <Button type='primary' onClick={() => {
          //添加商品操作
      }}>
        <Icon type="plus" />
        添加商品
      </Button>
    )
    return (
      <Card title={title} extra={extra}>
        <Table
          bordered
          loading={loading}
          rowKey='_id'
          columns={this.columns}
          dataSource={products}
          pagination={{
            total,
            defaultPageSize:PAGE_SIZE,
            showQuickJumper: true,
            onChange: this.getProducts,
            current: this.pageNum
```

```
            }}
          >
          </Table>
        </Card>
      )
    }
}
```

按描述/名称搜索

接口

```
// 根据搜索关键字获取商品分页列表
export const reqSearchProducts = ({ pageNum, pageSize, searchName, searchType }) => {
  return ajax(BASE + '/manage/product/search', {
    params: {
      pageNum,
      pageSize,
      [searchType]: searchName
    }
  })
}
```

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

```
  /*
  异步获取指定页码商品分页(可能带搜索)列表显示
  */
  getProducts = async (pageNum) => {
    // 保存当前请求的页码
    this.pageNum = pageNum;
    const { searchName, searchType } = this.state;
    let result;
    // 发请求获取数据
    if (!this.isSearch) {
      result = await reqProducts(pageNum, PAGE_SIZE)
    } else {
      //搜索功能
      result = await reqSearchProducts({
        pageNum,
        pageSize: PAGE_SIZE,
        searchName,
        searchType
      })
    }
    if (result.status === 0) {
      const { list, total } = result.data;
      this.setState({
        total,
        products: list
      })
    }
  }

  render() {
    const { total, searchType, searchName, products,
loading } = this.state;

    const title = (
      <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

```
import React, { Component } from 'react'
import { Switch,Route,Redirect } from "react-router-dom";
import './Product.less'
import ProductHome from './home';
import ProductAddUpdate from './add-update'
import ProductDetail from './detail'

export default class Product extends Component {
  render() {
    return (
      <Switch>
        <Route path='/product' exact component={ProductHome}></Route>
        <Route path='/product/addupdate' component={ProductAddUpdate}></Route>
        <Route path='/product/detail/:id' component={ProductDetail}></Route>
        <Redirect to='/product'/>
      </Switch>
    )
  }
}
```

add-update.js

```
import React from 'react'
class ProductAddUpdate extends React.Component {
  render() {
    return (
        <div>商品修改</div>
    );
  }
}

export default ProductAddUpdate;
```

detail.js

```
import React from 'react'
class ProductDetail extends React.Component {
  render() {
    return (
        <div>商品详情</div>
    );
  }
}

export default ProductAddUpdate;
```

home.js

```
//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
<span>
    <Button
        type='link'
        onClick={() => {
            // 保存当前商品的数据
            memoryUtils.product = product
```

```
            // 跳转商品详情页面
            this.props.history.push('/product/detail/'
+ product._id)
        }}
        >详情</Button>
    <Button
        type='link'
        onClick={() => {
            // 保存当前商品的数据
            memoryUtils.product = product
            // 跳转商品修改页面

 this.props.history.push('/product/addUpdate')
        }}
        >修改</Button>
</span>
//添加商品
<Button type='primary' onClick={() => {
        //添加商品,不需要获取当前商品的数据,先把商品数据置
空
        memoryUtils.product = {}
        //跳转商品添加页面
        this.props.history.push('/product/addupdate');
    }}>
    <Icon type="plus" />
    添加商品
</Button>
```

查看商品详情

接口

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

```
  return ajax(BASE + '/manage/product/info', {
    params: {
      productId
    }
  })
}
// 根据分类ID获取该商品分类
export const reqCategory = (categoryId) => {
  return ajax(BASE +'/manage/category/info',{
    params:{
      categoryId
    }
  })
}
```

detail.js

```
import React, { Component } from 'react'
import { Card, Icon, Button, List } from "antd";
// import memoryUtils from '../../utils/memoryUtils'
import { reqProduct, reqCategory } from "../../api";
import './detail.less'
export default class detail extends Component {
  state = {
    product: {},
    categoryName: ''
  }
  getCategory = async (categoryId) => {
    const res = await reqCategory(categoryId);
    console.log(res);

    if (res.status === 0) {
      this.setState({
        categoryName: res.data.name
```

```
      })
    }
  }
  async componentDidMount() {
    let product = this.state.product;
    if (product._id) {
      // 如果商品中有数据,获取对应的分类
    } else {
      // 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
      const id = this.props.match.params.id;
      const res = await reqProduct(id);
      if (res.status === 0) {
        product = res.data;
        this.setState({ product })
        //获取对应的分类
        this.getCategory(product.categoryId);
      }
    }
  }
  render() {
    const title = (
      <span>
        <Button type='link' onClick={() => this.props.history.goBack()}>
          <Icon type="arrow-left" />
        </Button>
        <span>商品详情</span>
      </span>
    )
    const { product, categoryName} = this.state;

    return (
      <Card title={title}>
        <List>
```

```
          <List.Item>
            <p>
              <span>商品名称:</span>
              <span>{product.name}</span>
            </p>
          </List.Item>

          <List.Item>
            <p>
              <span>商品描述:</span>
              <span>{product.desc}</span>
            </p>
          </List.Item>

          <List.Item>
            <p>
              <span>商品价格:</span>
              <span>{product.price}</span>
            </p>
          </List.Item>

          <List.Item>
            <p>
              <span>所属分类:</span>
              <span>{categoryName}</span>
            </p>
          </List.Item>

          <List.Item>
            <p>
              <span>商品分类:</span>
              <span>
                {
```

```
                    product.imgs && product.imgs.map(img => <img className="detail-img" key={img} src={'http://localhost:5000/upload/' + img} alt="img" />)
                }

              </span>
            </p>
          </List.Item>
          <List.Item>
              <span>商品详情:</span>
              <div dangerouslySetInnerHTML={{__html:product.detail}}></div>
          </List.Item>

        </List>
      </Card>
    )
  }
}

```

修改/添加商品

接口 api/index.js

```
// 请求所有商品
export const reqProducts = (pageNum, pageSize) => {
  return ajax(BASE + '/manage/product/list', {
    params: {
      pageNum,
      pageSize
    }
  })
}
```

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

Product/add-update.js

```
import React from 'react'
import {
  Form,
  Input,
  Icon,
  Select,
  Button,
  Card,
  message
} from 'antd';
import PicturesWall from './pictures-wall';
import RichTextEditor from './rich-text-editor';
import memoryUtils from '../../utils/memoryUtils';
import { reqCategorys, reqUpdateProduct } from "../../api";
const { Option } = Select;


class ProductAddUpdate extends React.Component {
  state = {
    categorys: []
  };
  constructor(props) {
    super(props)
```

```
    // 创建ref容器，并保存到组件对象
    this.pwRef = React.createRef()
    this.editorRef = React.createRef()
  }

  handleSubmit = e => {
    e.preventDefault();
    this.props.form.validateFields(async (err, values) => {
      if (!err) {
        const { name, desc, price, categoryId } = values;
        // 收集上传的图片文件名的数组
        const imgs = this.pwRef.current.getImgs();
        const detail = this.editorRef.current.getDetail();
        // 封装product对象
        const product = {name,desc,price,categoryId,imgs,detail}
        if(this.isUpdate){
          // 如果是更新，获取商品的id
          product._id = this.product._id;
        }
        console.log(product);

        // 发请求添加或修改
        const res = await reqUpdateProduct(product);
        if(res.status === 0){
          message.success(`${this.isUpdate ? '修改' : '添加'}商品成功`)
          this.props.history.replace('/product')
        }else{
          message.error(res.msg)
        }

```

```
      }
    });
  };
  componentWillMount() {
    // 组件将要加载的时候将商品的数据取出 存起来
    this.product = memoryUtils.product;
    this.isUpdate = !!this.product._id
  }
  componentDidMount() {
    this.getCategorys()
  }
  getCategorys = async () => {
    const res = await reqCategorys();
    if (res.status === 0) {
      this.setState({
        categorys: res.data
      })
    }

  }
  render() {
    const { getFieldDecorator } = this.props.form;

    const formItemLayout = {
      labelCol: { span: 2 },
      wrapperCol: { span: 8 }
    };
    const { categorys } = this.state
    const { isUpdate, product } = this
    const title = (
      <span>
        <Button type='link' onClick={() =>
this.props.history.goBack()}>
          <Icon type="arrow-left" />
        </Button>
```

```
        <span>{isUpdate ? '修改商品' : '添加商品'}</span>
      </span>
    )

    return (
      <Card title={title}>
        <Form {...formItemLayout} onSubmit={this.handleSubmit}>
          <Form.Item label="商品名称" >
            {getFieldDecorator('name', {
              initialValue: product.name,
              rules: [
                {
                  required: true,
                  message: '必须输入商品名称',
                },
              ],
            })(<Input placeholder='商品名称' />)}
          </Form.Item>
          <Form.Item label="商品描述">
            {getFieldDecorator('desc', {
              initialValue: product.desc,
              rules: [
                {
                  required: true,
                  message: '必须输入商品描述',
                }
              ],
            })(<Input placeholder='商品描述' />)}
          </Form.Item>
          <Form.Item label="商品价格">
            {getFieldDecorator('price', {
              initialValue: product.price,
              rules: [
```

```
                {
                  required: true,
                  message: '必须输入价格',
                },
                {
                  validator: this.validatePrice,
                },
              ],
            })(<Input type="number" placeholder="商品价格" addonAfter="元" />)}
          </Form.Item>
          <Form.Item label='商品分类'>
            {getFieldDecorator('categoryId', {
              initialValue: product.categoryId || '',
              rules: [{ required: true, message: '必须输入商品分类' }],
            })(
              <Select>
                <Option value=''>未选择</Option>
                {
                  categorys.map(c => <Option value={c._id} key={c._id}>{c.name}</Option>)
                }
              </Select>
            )}
          </Form.Item>
          <Form.Item label="商品图片">
            <PicturesWall ref={this.pwRef} imgs={product.imgs} />
          </Form.Item>
          <Form.Item label="商品详情" wrapperCol={{ col: 20 }}>
            <br />
            <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: [],
```

```
  };

  handleCancel = () => this.setState({ previewVisible: false });

  handlePreview = async file => {
    if (!file.url && !file.preview) {
      file.preview = await getBase64(file.originFileObj);
    }
    this.setState({
      previewImage: file.url || file.preview,
      previewVisible: true,
    });
  };
  // 获取所有已上传图片文件名的数组
  getImgs = () => this.state.fileList.map(file => file.name)

     handleChange = async ({ file, fileList }) => {
        // file与fileList中最后一个file代表同个图片的不同对象
        console.log('handleChange()', file.status, file===fileList[fileList.length-1])
        // 如果上传成功
        if (file.status==='done') {
          // 将数组最后一个file保存到file变量
          file = fileList[fileList.length - 1]
          // 取出响应数据中的图片文件名和url
          const {name, url} = file.response.data
          // 保存到上传的file对象
          file.name = name
          file.url = url
        } else if (file.status==='removed') { // 删除
          const result = await reqDeleteImg(file.name)
```

```
          if (result.status===0) {
            message.success('删除图片成功')
          } else {
            message.error('删除图片失败')
          }
        }

        // 更新状态
        this.setState({ fileList })
      }
  //组件将要加载时
  componentWillMount() {
    // 根据传入的imgs生成fileList并更新
    const imgs = this.props.imgs;
    if (imgs && imgs.length) {
      const fileList = imgs.map((img, i) => ({
        uid: -i,//唯一标识
        name: img, //文件名
        status: 'done',//状态有:uploading done error removed
        url: 'http://localhost:5000/upload/' + img
      }))
      this.setState({ fileList })
    }
  }

  render() {
    const { previewVisible, previewImage, fileList } = this.state;
    const uploadButton = (
      <div>
        <Icon type="plus" />
        <div className="ant-upload-text">Upload</div>
      </div>
    );
```

```
    return (
      <div className="clearfix">
        <Upload
          action="/manage/img/upload" //上传图片的url
          name="image" // 图片文件对应参数名
          listType="picture-card" //显示图片风格
          fileList={fileList} //已上传的所有图片文件信息对象的数组
          onPreview={this.handlePreview}
          onChange={this.handleChange}
        >
          {fileList.length >= 3 ? null : uploadButton}
        </Upload>
        <Modal visible={previewVisible} footer={null} onCancel={this.handleCancel}>
          <img alt="example" style={{ width: '100%' }} src={previewImage} />
        </Modal>
      </div>
    );
  }
}

export default PicturesWall;
```

富文本编辑器

[文档链接]

Product/rich-text-editor.js

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

```
  constructor(props) {
    super(props);
    this.state = {
      editorContent: '',
      editor:null
    };
  }
  getDetail = ()=> this.state.editorContent;

  componentDidMount() {
    const elemMenu = this.refs.editorElemMenu;
    const elemBody = this.refs.editorElemBody;
    const editor = new E(elemMenu, elemBody)
    editor.create()
    const detail = this.props.detail;
    if(detail){
      editor.txt.html(detail)
    }
    // 使用 onchange 函数监听内容的变化，并实时更新到 state 中
    editor.customConfig.onchange = html => {
      console.log(editor);

      console.log(editor.txt.html())
      this.setState({
        // editorContent: editor.txt.text()
        editorContent: editor.txt.html()
      })
    }
    editor.customConfig.menus = [
      'head',  // 标题
      'bold',  // 粗体
      'fontSize',  // 字号
      'fontName',  // 字体
      'italic',  // 斜体
```

```
      'underline',  // 下划线
      'strikeThrough',  // 删除线
      'foreColor',  // 文字颜色
      'backColor',  // 背景颜色
      'link',  // 插入链接
      'list',  // 列表
      'justify',  // 对齐方式
      'quote',  // 引用
      'emoticon',  // 表情
      'image',  // 插入图片
      'table',  // 表格
      'video',  // 插入视频
      'code',  // 插入代码
      'undo',  // 撤销
      'redo'  // 重复
    ]
    editor.customConfig.uploadImgShowBase64 = true
    editor.create()

  };
  render() {
    return (
      <div className="shop">
        <div className="text-area" >
          <div ref="editorElemMenu"
            style={{ backgroundColor: '#f1f1f1',
border: "1px solid #ccc" }}
            className="editorElem-menu">

          </div>
          <div
            style={{
              padding: "0 10px",
              overflowY: "scroll",
              height: 300,
```

```
              border: "1px solid #ccc",
              borderTop: "none"
            }}
            ref="editorElemBody" className="editorElem-
body">
            {this.state.editorContent}
          </div>
        </div>
      </div>
    );
  }
}
```

角色管理组件

api/index.js

```
// 获取所有角色列表
export const reqRoles = () => {
  return ajax(BASE + '/manage/role/list')
}

// 添加角色
export const reqAddRole = (roleName) => {
  return ajax.post(BASE + '/manage/role/add', {
    roleName
  })
}

// 修改权限
export const reqAddAuth = (role) => {
  return ajax.post(BASE + '/manage/role/update', role)
}
```

Role/role.js

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

```
        } else {
          message.error(result.msg);
        }

      }
    })
  }
  // 增加用户权限操作
  addAuth = async () => {
    // 隐藏确认框
    this.setState({
      isShowAuth: false
    })

    // 更新role对象相关属性
    const { role } = this;
    role.menus = this.refs.authRef.getMenus();
    role.auth_time = Date.now();
    role.auth_name = memoryUtils.user.username;
    // 请求更新角色
    const result = await reqAddAuth(role);
    if(result.status === 0){
      message.success('角色授权成功');
      // 重新获取所有角色
      this.getRoles();
    }else{
      message.error(result.msg);
    }


  }
  // 获取所有的角色
  getRoles = async () => {
    const result = await reqRoles();
    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) {
```

```
            return new Date(auth_time).toLocaleString()
          } else {
            return;
          }

        }
      },
      {
        title: '授权人',
        dataIndex: 'auth_name'
      },
      {
        title: '操作',
        render: (role) => <Button type='primary' onClick={() => this.showAuth(role)}>设置权限</Button>
      },
    ]
  }
  showAuth = (role) => {
    console.log(role);
    // 将当前需要设置的角色保存到组件对象上
    this.role = role;
    this.setState({
      isShowAuth: true
    })
  }
  componentWillMount() {
    this.initColumn()
  }
  componentDidMount() {
    this.getRoles();
  }

  render() {
```

```
    const { isShowAdd, roles, isShowAuth } = this.state;

    const title = (
      <Button type='primary' onClick={() => this.setState({ isShowAdd: true })}>创建角色</Button>
    )
    return (
      <Card title={title}>
        <Table
          dataSource={roles}
          columns={this.columns}
          bordered
          rowKey='_id'
          pagination={{ defaultPageSize: 2 }}
        ></Table>
        <Modal
          title="添加角色"
          visible={isShowAdd}
          onOk={this.addRole}
          onCancel={() => {
            this.setState({ isShowAdd: false })
          }}
        >
          <AddRole setForm={form => this.form = form}></AddRole>

        </Modal>

        <Modal
          title="设置角色权限"
          visible={isShowAuth}
          onOk={this.addAuth}
          onCancel={() => {
            this.setState({ isShowAuth: false })
```

```
          }}
        >
          <AddAuth ref='authRef' setForm={form => this.form = form} role={this.role}></AddAuth>

        </Modal>
      </Card>
    )
  }
}

```

添加角色

Role/add-role.js

```
import React, { Component } from 'react'
import {
  Form,
  Input,
} from 'antd';

class AddRole extends Component {
  state = {
  };

  componentWillMount(){
    this.props.setForm(this.props.form);
  }
  render() {
    const { getFieldDecorator } = this.props.form;
    const formItemLayout = {
      labelCol: { span: 4 },
      wrapperCol: { span: 15 }
```

```
    };
    return (
      <Form {...formItemLayout} onSubmit={this.handleSubmit}>
        <Form.Item label="角色名称" >
          {getFieldDecorator('roleName', {
            rules: [
              {
                required: true,
                message: '必须输入角色名称',
              },
            ],
          })(<Input placeholder='请输入角色名称'/>)}
        </Form.Item>

      </Form>
    );
  }
}

export default Form.create({ name: 'role' })(AddRole);
```

修改权限

```
import React, { Component, PureComponent } from "react";
import { Tree, Form, Input } from 'antd';
import menuConfig from "../../config/menuConfig";
const { TreeNode } = Tree;
const Item = Form.Item


class AddAuth extends PureComponent {
  state = {
```

```
    checkedKeys: [],
  };

  onCheck = checkedKeys => {
    console.log('onCheck', checkedKeys);
    this.setState({ checkedKeys });
  };

  // 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
  getMenus = () => this.state.checkedKeys;
  renderTreeNodes = data =>
    data.map(item => {
      if (item.children) {
        return (
          <TreeNode title={item.title} key={item.key} dataRef={item}>
            {this.renderTreeNodes(item.children)}
          </TreeNode>
        );
      }
      return <TreeNode key={item.key} {...item} />;
    });
  componentWillMount() {
    // 根据传入角色的menus来更新默认选中checkedKeys状态
    console.log(this.props.role);
    const menus = this.props.role.menus;
    this.setState({
      checkedKeys: menus
    })

  }
  /*
  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)
  nextProps: 接收到的包含新的属性的对象
```

```
  */
  componentWillReceiveProps(nextProps) {
    const menus = nextProps.role.menus;
    this.setState({
      checkedKeys: menus
    })
  }
  render() {
    const { role } = this.props;
    const { checkedKeys } = this.state;
    // 指定Item布局的配置对象
    const formItemLayout = {
      labelCol: { span: 4 },  // 左侧label的宽度
      wrapperCol: { span: 15 }, // 右侧包裹的宽度
    }
    return (
      <div>
        <Item label='角色名称' {...formItemLayout}>
          <Input value={role.name} disabled />
        </Item>
        <Tree
          checkable
          onCheck={this.onCheck}
          checkedKeys={checkedKeys}
          defaultExpandAll
        >
          <TreeNode title='平台权限' key='all'>
            {this.renderTreeNodes(menuConfig)}
          </TreeNode>
        </Tree>
      </div>
    );
  }
}

```

```
78  export default AddAuth;
```

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',
```

```
        dataIndex: 'phone'
      },
      {
        title: '注册时间',
        dataIndex: 'create_time',
        render: (create_time) => {
          if (create_time) {
            return new
Date(create_time).toLocaleString()
          } else {
            return;
          }
        }
      },
      {
        title: '所属角色',
        dataIndex: 'role_id',
        render: role_id => this.state.roles.find(role
=> role._id === role_id).name
      },
      {
        title: '操作',
        render: user => (
          <span>
            <Button onClick={() =>
this.showUpdate(user)}>修改</Button>
            <Button onClick={() =>
this.deleteUser(user)}>删除</Button>

          </span>
        )
      }

    ]
  }
```

```
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
```

```
        }
        const res = await reqAddOrUpdateUser(values);
        if (res.status === 0) {
          message.success('添加/更新用户成功!')
          this.getUsers();
        } else {
          message.error(res.msg);
        }

      }
    })
  };
  // 获取所有用户
  getUsers = async () => {
    const res = await reqUsers();
    console.log(res);
    if (res.status === 0) {
      const { users, roles } = res.data;
      this.setState({
        users,
        roles
      })
    }

  }
  componentWillMount() {
    this.initColumns();
  }
  componentDidMount() {
    this.getUsers();
  }


  // 取消之后的操作
  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
```

```
          title={user._id ? '修改用户' : '添加用户'}
          visible={this.state.visible}
          onOk={this.addOrUpdateUser}
          onCancel={this.handleCancel}
        >
          <UserForm
            setForm={form => this.form = form}
            roles={roles}
            user={user}
          ></UserForm>
        </Modal>
      </Card>
    )
  }
}

```

User/user-form.js

```
import React, { Component } from 'react'
import {
  Form,
  Input,
  Select,
} from 'antd';

const { Option } = Select;
class UserForm extends Component {
  state = {
    confirmDirty: false,
    autoCompleteResult: [],
  };

  componentWillMount() {

```

```
    this.props.setForm(this.props.form)
  }
  render() {
    const { getFieldDecorator } = this.props.form;
    const { roles, user } = this.props;
    const formItemLayout = {
      labelCol: { span: 4 },
      wrapperCol: { span: 15 }
    };
    const prefixSelector = getFieldDecorator('prefix',
{
      initialValue: '86',
    })(
      <Select style={{ width: 70 }}>
        <Option value="86">+86</Option>
        <Option value="87">+87</Option>
      </Select>,
    );

    return (

      <Form {...formItemLayout}>
        <Form.Item label="用户名">
          {getFieldDecorator('username', {
            initialValue:user.username,
            rules: [
              {
                required: true,
                message: '请输入用户名',
              },
            ],
          })(<Input placeholder='请输入用户名' />)}
        </Form.Item>
        {
          user._id ? null : (
```

```
            <Form.Item label="密码" hasFeedback>
              {getFieldDecorator('password', {
                initialValue: user.password,
                rules: [
                  {
                    required: true,
                    message: '请输入密码',
                  },

                ],
              })(<Input.Password placeholder='请输入密码' />)}
            </Form.Item>
          )
        }
        <Form.Item label="手机号">
          {getFieldDecorator('phone', {
            initialValue: user.phone,
            rules: [{ required: true, message: '请输入正确的手机号码' }],
          })(<Input addonBefore={prefixSelector} style={{ width: '100%' }} placeholder='请输入手机号' />)}
        </Form.Item>

        <Form.Item label="邮箱">
          {getFieldDecorator('email', {
            initialValue: user.email,
            rules: [
              {
                pattern: /^([a-zA-Z]|[0-9])(\w|\-)+@[a-zA-Z0-9]+\.([a-zA-Z]{2,4})$/,
                // required: true,
                message: '邮箱格式不正确',
              },
```

```
            ],
          })(<Input placeholder='请输入邮箱' />)}
        </Form.Item>

        <Form.Item label="角色">
          {getFieldDecorator('role_id', {
            initialValue: user.role_id,
            rules: [
              {
                required: true,
                message: '必须制定角色',
              },
            ],
          })(
            <Select>
              {
                roles.map(role=><Option key={role._id}
value={role._id}>{role.name}</Option>)
              }
            </Select>
          )}
        </Form.Item>
      </Form>
    );
  }
}

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

```
import React, { Component } from 'react'
import { Card, Button } from "antd";
import ReactEcharts from 'echarts-for-react';
export default class Charts extends Component {
  state = {
    sales: [30, 20, 36, 10, 10, 20], // 销量的数组
    stores: [20, 10, 25, 20, 15, 10], // 库存的数组
  }
  update = ()=>{
    this.setState(state=>({
      sales:state.sales.map(sale=>sale+1),
      stores:state.stores.reduce((pre,store)=>{
        pre.push(store-1)
        return pre;
      },[])
    }))
  }
  getOption = (sales, stores) => {
    return {
      title: {
        text: 'ECharts 入门示例'
      },
      tooltip: {},
      legend: {
        data: ['销量','库存'],
      },
      xAxis: {
```

```
        data: ["Vue", "Rect", "Angular", "Nodejs", "微信小程序", "自动化测试"]
      },
      yAxis: {},
      series: [{
        name: '销量',
        type: 'bar',
        data: sales
      }, {
        name: "库存",
        type: 'bar',
        data: stores
      }]
    }
  }
  render() {
    const { sales, stores } = this.state;
    return (
      <div>
        <Card>
          <Button type='primary' onClick={this.update}>更新</Button>
        </Card>
        <ReactEcharts option={this.getOption(sales, stores)}></ReactEcharts>
      </div>
    )
  }
}

```

折线组件

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

```
import { Card, Button } from 'antd'
import ReactEcharts from 'echarts-for-react'
export default class Line extends Component {
  state = {
    sales: [30, 20, 36, 10, 10, 20], // 销量的数组
    stores: [20, 10, 25, 20, 15, 10], // 库存的数组
  }
  getOption = (sales,stores) => {
    return {
      xAxis: {
        type: 'category',
        data: ["Vue", "Rect", "Angular", "Nodejs", "微信小程序", "自动化测试"]
      },
      legend: {
        data: ['销量', '库存']
      },
      yAxis: {
        type: 'value'
      },
      series: [{
        name: '销量',
        type: 'line',
        data: sales
      }, {
          name: '库存',
          type: 'line',
          data: stores
        }]
    };
  }
  update = () => {
    this.setState(state => ({
      sales: state.sales.map(sale => sale + 1),
      stores: state.stores.reduce((pre, store) => {
```

```
        pre.push(store - 1)
        return pre
      }, []),
    }))
  }
  render() {
    const {sales,stores} = this.state;
    return (
      <div>
        <Card>
          <Button type='primary' onClick={this.update}>更新</Button>
        </Card>
        <ReactEcharts option={this.getOption(sales,stores)}></ReactEcharts>
      </div>
    )
  }
}

```

饼形组件

```
import React, { Component } from 'react'
import { Card } from "antd";
import ReactEcharts from 'echarts-for-react'
export default class Pie extends Component {
  getOption = ()=>{
    return {
      title: {
        text: '某站点用户访问来源',
        subtext: '纯属虚构',
        left: 'center'
      },
```

```
      tooltip: {
        trigger: 'item',
        formatter: '{a} <br/>{b} : {c} ({d}%)'
      },
      legend: {
        orient: 'vertical',
        left: 'left',
        data: ['直接访问', '邮件营销', '联盟广告', '视频广告', '搜索引擎']
      },
      series: [
        {
          name: '访问来源',
          type: 'pie',
          radius: '55%',
          center: ['50%', '60%'],
          data: [
            { value: 335, name: '直接访问' },
            { value: 310, name: '邮件营销' },
            { value: 234, name: '联盟广告' },
            { value: 135, name: '视频广告' },
            { value: 1548, name: '搜索引擎' }
          ],
          emphasis: {
            itemStyle: {
              shadowBlur: 10,
              shadowOffsetX: 0,
              shadowColor: 'rgba(0, 0, 0, 0.5)'
            }
          }
        }
      ]
    };
  }
  getOption2 = ()=>{
```

```
    return {
      backgroundColor: '#2c343c',

      title: {
        text: 'Customized Pie',
        left: 'center',
        top: 20,
        textStyle: {
          color: '#ccc'
        }
      },

      tooltip: {
        trigger: 'item',
        formatter: '{a} <br/>{b} : {c} ({d}%)'
      },

      visualMap: {
        show: false,
        min: 80,
        max: 600,
        inRange: {
          colorLightness: [0, 1]
        }
      },
      series: [
        {
          name: '访问来源',
          type: 'pie',
          radius: '55%',
          center: ['50%', '50%'],
          data: [
            { value: 335, name: '直接访问' },
            { value: 310, name: '邮件营销' },
            { value: 274, name: '联盟广告' },
```

```
            { value: 235, name: '视频广告' },
            { value: 400, name: '搜索引擎' }
          ].sort(function (a, b) { return a.value - b.value; }),
          roseType: 'radius',
          label: {
            color: 'rgba(255, 255, 255, 0.3)'
          },
          labelLine: {
            lineStyle: {
              color: 'rgba(255, 255, 255, 0.3)'
            },
            smooth: 0.2,
            length: 10,
            length2: 20
          },
          itemStyle: {
            color: '#c23531',
            shadowBlur: 200,
            shadowColor: 'rgba(0, 0, 0, 0.5)'
          },

          animationType: 'scale',
          animationEasing: 'elasticOut',
          animationDelay: function (idx) {
            return Math.random() * 200;
          }
        }
      ]
    };
  }
  render() {
    return (
      <div>
        <Card title='饼图一'>
```

```
          <ReactEcharts option={this.getOption()}
style={{ height: 300 }} />
        </Card>
        <Card title='饼图二'>
          <ReactEcharts option={this.getOption2()}
style={{ height: 300 }} />
        </Card>
      </div>
    )
  }
}

```

使用React-redux

下载安装：

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

新建store/index.js

```
import { createStore } from "redux";
import userer from './user.reducer'

// 创建store  有state和reducer的store
const store = createStore(userer);
export default store;
```

新建store/user.reducer.js

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

```
      user = action.user
      return user;
    case 'REMOVEUSER':
      user = {};
      return user;
    default:
      return user;
  }
}
export const mapStateToProps = (state /*, ownProps*/)
=> {
  return {
    user: state
  }
}
export const mapDispatchToProps = dispatch => {
  return {
    setUser: (user) => {
      console.log(user);

      dispatch({ type: 'SETUSER', user: user })
    },
    removeUser: () => {
      dispatch({ type: 'REMOVEUSER' })
    }
  }
}
export default userReducer;
```

index.js中连接使用

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import './api'
import { Provider } from "react-redux";
import store from './store'
ReactDOM.render((
  <Provider store = {store}>
    <App />
  </Provider>
), document.getElementById('root'));
```

修改Login/login.jsx

```
import React, { Component } from 'react'
import { Form, Icon, Input, Button, message } from
'antd';
import { Redirect } from "react-router-dom";
import './Login.less'
import logo from '../../assets/images/logo.svg'
import { reqLogin } from '../../api';
import storageUtils from
'../../utils/storageUtils.js';
import memoryUtils from '../../utils/memoryUtils';
import { connect } from "react-redux";
import { mapStateToProps, mapDispatchToProps } from
'../../store/user.reducer';
@connect(mapStateToProps,mapDispatchToProps)
class Login extends Component {
  handleSubmit = e => {
    e.preventDefault();
    this.props.form.validateFields(async (err, values)
=> {
```

```
      if (!err) {
        // console.log('Received values of form: ', values);
        const { username, password } = values;
        const res = await reqLogin(username, password);
        if (res.status === 0) {
          // 将用户信息保存到local中
          const user = res.data;
          // localStorage.setItem('user_key', JSON.stringify(user))
          storageUtils.saveUser(user)

          // 保存到内存中
          // memoryUtils.user = user;
          // 修改
          this.props.setUser(user);
          // 跳转到管理页面
          this.props.history.replace('/')
          message.success('登录成功')
        } else {
          message.error(res.msg);
        }
      } else {

      }
    });
  };
  /**
   * 自定义校验规则
   */
  validatorPwd = (rule, value, callback) => {
    value = value.trim();
    if (!value) {
      callback('密码必须输入')
```

```
    } else if (value.length < 4) {
      callback('密码不能小于4位')
    } else if (value.length > 12) {
      callback('密码不能大于12位')
    } else if (!/^[a-zA-Z0-9_]+$/.test(value)) {
      callback('密码必须是英文、数字或下划线组成')
    } else {
      // 验证通过一定要callback,如果callback,验证无法响应
      callback();
    }
  }
  render() {
    // 读取保存的user 如果存在，直接跳转到管理界面
    // const user = JSON.parse(localStorage.getItem('user_key')) || {};
    // const user = storageUtils.getUser();
    //const user = memoryUtils.user
      //修改
    const user = this.props.user
    if (user._id) {
      // 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
      // this.props.history.replace('/login')
      return <Redirect to='/' />
    }
    const { getFieldDecorator } = this.props.form;
    return (
      <div className='login'>
        <div className='login-header'>
          <img src={logo} alt="" />
          <h1>React后台管理系统</h1>
        </div>
        <div className="login-content">
          <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"
```

```
                />,
              )}
            </Form.Item>
            <Form.Item>
              <Button type="primary" htmlType="submit" className="login-form-button" style={{ width: '100%' }}>
                登录
              </Button>
            </Form.Item>
          </Form>
        </div>
      </div>
    )
  }
}

export default Form.create({ name: 'normal_login' })(Login);
```

Admin/admin.jsx

```
import {connect} from 'react-redux';
import { mapStateToProps,mapDispatchToProps } from "../../store/user.reducer";

@connect(mapStateToProps, mapDispatchToProps)
class Admin extends Component {
  state = {
    collapsed: false,
  };

  toggle = () => {
    this.setState({
```

```
      collapsed: !this.state.collapsed,
    });
  };
  render() {
    // 读取保存的user 如果不存在，直接跳转到登录界面
    // const user = storageUtils.getUser();
    // const user = memoryUtils.user;
    //修改
    const user = this.props.user;
    if (!user._id) {
      // 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
      // this.props.history.replace('/login')
      return <Redirect to='/login' />
    }
    return (
      <Layout style={{height:'100%'}}>
        <LeftNav collapsed={this.state.collapsed}></LeftNav>

        <Layout>
          <MHeader collapsed = {this.state.collapsed} toggle={this.toggle}/>
          <Content
            style={{
              margin: '24px 16px',
              padding: 24,
              background: '#fff',
              minHeight: 280,
            }}
          >
            {/* 路由配置 */}
            <Switch>
              <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';
```

```
import { reqWeather } from '../../api';
import {connect} from 'react-redux'
import {mapStateToProps,mapDispatchToProps} from
'../../store/user.reducer'
const { Header } = Layout;
const { confirm } = Modal;
@connect(mapStateToProps,mapDispatchToProps)
class MHeader extends Component {
  constructor(props) {
    super(props);
    this.state = {
      currentTime: new Date().toLocaleString(),
      dayPictureUrl: '', // 图片url
      weather: '', // 天气文本
    }

  }

  logout = () => {
    confirm({
      title: '确定要退出登录吗? ',
      content: '',
      onOk: () => {
        storageUtils.removeUser();
        // memoryUtils.user = {};
        // 修改
        this.props.removeUser()
        this.props.history.replace('/home');
      },
      onCancel: () => {
        console.log('Cancel');

      },
    });
  }
```

```
  //....
  // 获取天气信息显示
  getWeather = async ()=>{
    const { dayPictureUrl, weather } = await reqWeather('北京');
    // 更新状态
    this.setState({
      dayPictureUrl,
      weather
    })

  }

  render() {
    // const user = memoryUtils.user
    // 修改
    const user = this.props.user;
    return (
      <Header style={{ background: '#fff', padding: 0 }}>
        <div className="header">
          <h2 className='header-title'>{this.getTitle()}</h2>
          <div className='header-user'>
            <div className='currentTime'>{this.state.currentTime}</div>
            <div className='weather'>天气{this.state.weather}</div>
            <div className='weatherPic'>
              <img src={this.state.dayPictureUrl} alt=""/>
            </div>
            <div className='userInfo'>
              欢迎,{user.username}
```

```
              <Button style={{ marginLeft: '20px' }} onClick={this.logout}>退出</Button>
            </div>
          </div>
        </div>
      </Header>
    )
  }
}
export default withRouter(MHeader)
```

保证刷新时状态不变，在App.js组件中更新user

```
import React, { Component } from 'react'
import { BrowserRouter,Switch,Route } from "react-router-dom";
import Login from './pages/Login/login';
import Admin from './pages/Admin/admin';
import { connect } from "react-redux";
import { mapStateToProps, mapDispatchToProps } from "./store/user.reducer";
import storageUtils from './utils/storageUtils'
@connect(mapStateToProps,mapDispatchToProps)
class App extends Component {

  componentWillMount() {
    const user = storageUtils.getUser();
    this.props.setUser(user);
  }

  render() {
    return (
      <BrowserRouter>
        <Switch>
```

```
          <Route path='/login' component={Login}></Route>
          <Route path='/' component={Admin}></Route>
        </Switch>
      </BrowserRouter>
    )
  }
}
export default App;

```

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

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

安装pm2进程管理工具

```
npm install pm2 -g
```

安装mongodb

```
yum install mongodb-server mongodb -y 安装客户端和服务端

systemctl stop mongod 关闭mongodb服务

systemctl start mongod 启动mongodb服务

netstat -tunlp |grep mongod 查询应用的端口号

t：表示查看tcp

u：表示查看udp

n：表示端口以数字形式表示，没有n直接显示服务名。

l：表示显示所监听的端口

p：表示占用端口的进程

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：

```
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

```
//以我的服务器为例：nginx目录
cd /opt/ngx112/conf
vim nginx.conf

//修改配置文件如下
 server {
        listen       80; //端口号
        server_name  www.pythonav.cn; //域名
        location / {
                    try_files $uri $uri/ /index.html;  #匹配所有的路由
                    root /tmp/dist; //填写前端的根目录
                index  index.html index.htm;
        }
```

```
14      }
```

输入nginx的启动命令

```
1 /opt/ngx112/sbin/nginx 第一次输入是启动
2 /opt/ngx112/sbin/nginx -s reload  #平滑重启，重新读取配置文件，不重启进程
3 /opt/ngx112/sbin/nginx -s stop
```

访问http://www.pythonav.cn 查看xmall商城项目