React

## Redux

### Redux的基本使用(例子)

#### Index.js

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

import { createStore, applyMiddleware, compose } from 'redux'

import thunk from 'redux-thunk'

import { counter, addGun, removeGun, addGunAsync } from './index.redux'

// compose把多个中间件连接在一块

const store = createStore(counter, compose(

applyMiddleware(thunk),

window.devToolsExtension? window.devToolsExtension(): f=>f // 使用redux-devtools

))

function render() {

ReactDOM.render(<App store={ store } addGun={addGun} addGunAsync={addGunAsync} removeGun={removeGun} />, document.getElementById('root'));

}

render()

// subscribe订阅render函数，当函数内部状态发生变化时会重新执行render函数

store.subscribe(render) // 状态改变后手动执行render

#### Index.redux.js

const ADD\_GUN = '加机关枪'

const REMOVE\_GUN = '减机关枪'

export function counter(state=0, action) {

switch(action.type) {

case ADD\_GUN:

return state + 1

case REMOVE\_GUN:

return state -1

default:

return 10

}

}

// action creator

export function addGun() {

return { type: ADD\_GUN }

}

export function removeGun() {

return { type: REMOVE\_GUN }

}

// 处理异步

export function addGunAsync() {

return dispatch => {

setTimeout(()=>{

dispatch(addGun())

}, 2000)

}

}

#### App.js

import React from 'react'

class App extends React.Component{

// constructor(props) {

// super(props)

// }

render() {

const store = this.props.store

const num = store.getState()

const addGun = this.props.addGun

const removeGun = this.props.removeGun

const addGunAsync =this.props.addGunAsync

return (

<div className="App">

<h3>现在有机枪{ num }把</h3>

<button onClick={()=>store.dispatch(addGun())}>申请武器</button>

<button onClick={()=>store.dispatch(removeGun())}>上交武器</button>

<button onClick={()=>store.dispatch(addGunAsync())}>拖两天给武器</button>

</div>

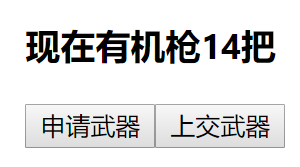
);

}

}

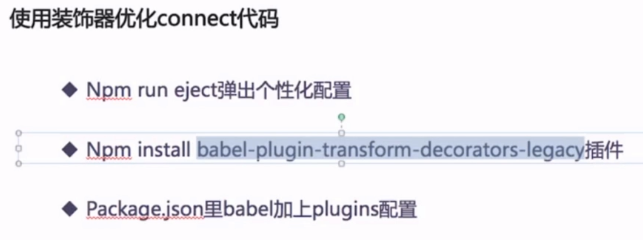
export default App

#### 效果



### Reatc-redux的基本使用(例子)





#### 安装

Yarn add react-redux

#### Index.js

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

import { Provider } from 'react-redux'

import { createStore, applyMiddleware, compose } from 'redux'

import thunk from 'redux-thunk'

import { counter } from './index.redux'

// compose把多个中间件连接在一块

const store = createStore(counter, compose(

applyMiddleware(thunk),

window.devToolsExtension? window.devToolsExtension(): f=>f // 使用redux-devtools

))

ReactDOM.render(

<Provider store={ store }>

<App />

</Provider>,

document.getElementById('root')

);

#### Index.redux.js

const ADD\_GUN = '加机关枪'

const REMOVE\_GUN = '减机关枪'

export function counter(state=0, action) {

switch(action.type) {

case ADD\_GUN:

return state + 1

case REMOVE\_GUN:

return state -1

default:

return 10

}

}

// action creator

export function addGun() {

return { type: ADD\_GUN }

}

export function removeGun() {

return { type: REMOVE\_GUN }

}

// 异步数据

export function addGunAsync() {

return dispatch => {

setTimeout(()=>{

dispatch(addGun())

}, 2000)

}

}

#### App.js (不使用装饰器)

import React from 'react'

import { connect } from 'react-redux'

import { addGun, removeGun, addGunAsync } from './index.redux'

class App extends React.Component{

render() {

return (

<div className="App">

<h3>现在有机枪{ this.props.num }把</h3>

<button onClick={this.props.addGun}>申请武器</button>

<button onClick={this.props.removeGun}>上交武器</button>

<button onClick={this.props.addGunAsync}>拖两天给武器</button>

</div>

);

}

}

// connect 的四个参数

const mapStatetoProps = (state)=>{

return {num:state}

}

const actionCreators = { addGun, removeGun, addGunAsync }

// 未使用装饰器connect

App = connect(mapStatetoProps, actionCreators)(App)

export default App

#### App.js (使用装饰器)

##### 安装：

yarn add babel-plugin-transform-decorators-legacy

##### package.json

修改babel项

"babel": {

"presets": [

"react-app"

],

"plugins": ["transform-decorators-legacy"]

}

测试后发现以上方法报错，修正为

##### 安装：

yarn add @babel/plugin-proposal-decorators

##### package.json

"babel": {

"presets": [

"react-app"

],

"plugins": [

["@babel/plugin-proposal-decorators", { "legacy": true }]

]

}

##### App.js

import React from 'react'

import { connect } from 'react-redux'

import { addGun, removeGun, addGunAsync } from './index.redux'

// 使用装饰器

@connect(

// 你要state什么属性放到props里

state=>({num:state}),

// 你要什么方法 放到props里 自动dispatch

{ addGun, removeGun, addGunAsync }

)

class App extends React.Component{

render() {

return (

<div className="App">

<h3>现在有机枪{ this.props.num }把</h3>

<button onClick={this.props.addGun}>申请武器</button>

<button onClick={this.props.removeGun}>上交武器</button>

<button onClick={this.props.addGunAsync}>拖两天给武器</button>

</div>

);

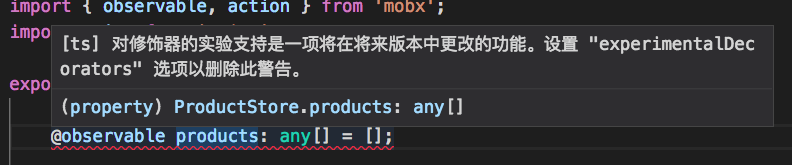
}

}

export default App

#### 使用修饰器问题及解决方案

##### 问题显示



##### 2. 解决办法

###### 方法一：打开tsconfig.js，在compilerOptions选项里面添加/修改为

* "experimentalDecorators": true,

###### 方法二：打开VSCode的全局设置

文件 => 首选项 => 设置

搜索experimentalDecorators,ranh

设置"javascript.implicitProjectConfig.experimentalDecorators": true，该选项默认为false，需要改为true

## Redux-router4

### 安装

yarn add react-router-dom

### 简单使用

#### Index.js

import React from 'react';

import ReactDOM from 'react-dom';

import { Provider } from 'react-redux'

import { createStore, applyMiddleware, compose } from 'redux'

import thunk from 'redux-thunk'

import {

BrowserRouter,

Route,

Link,

Redirect,

Switch

} from 'react-router-dom'

import { counter } from './index.redux'

import App from './App';

// compose把多个中间件连接在一块

const store = createStore(counter, compose(

applyMiddleware(thunk),

window.devToolsExtension? window.devToolsExtension(): f=>f // 使用redux-devtools

))

function Erying() {

return <h2>二营</h2>

}

function Qibinglian() {

return <h2>骑兵连</h2>

}

class Test1 extends React.Component{

render() {

console.log(this.props)

return <h2>{this.props.match.params.location}</h2>

}

}

ReactDOM.render(

<Provider store={ store }>

<BrowserRouter>

<div>

<ul>

<li><Link to='/'>一营</Link></li>

<li><Link to='/erying'>二营</Link></li>

<li><Link to='/qibinglian'>骑兵连</Link></li>

<li><Link to='/dongtai/qibinglian'>动态骑兵连</Link></li>

<li><Link to='/dongtai/qibing'>动态骑兵</Link></li>

</ul>

<Switch> {/\*只渲染命中的第一个Route\*/}

<Route path='/' exact component={App}></Route>

<Route path='/erying' component={Erying}></Route>

<Route path='/qibinglian' component={Qibinglian}></Route>

<Route path='/dongtai/:location' component={Test1}></Route> {/\* 动态路由 \*/}

<Route path='/:location' component={Test1}></Route>

<Redirect to='/qibinglian'></Redirect> {/\* 默认加载的路由 \*/}

</Switch>

</div>

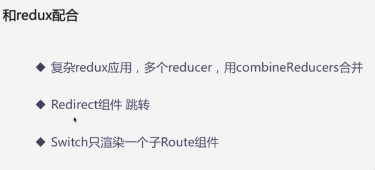
</BrowserRouter>

</Provider>,

document.getElementById('root')

);

#### React-router4和redux配合



##### Index.redux.js

const ADD\_GUN = '加机关枪'

const REMOVE\_GUN = '减机关枪'

export function counter(state=0, action) {

switch(action.type) {

case ADD\_GUN:

return state + 1

case REMOVE\_GUN:

return state - 1

default:

return state

}

}

// action creator

export function addGun() {

return { type: ADD\_GUN }

}

export function removeGun() {

return { type: REMOVE\_GUN }

}

export function addGunAsync() {

return dispatch => {

setTimeout(()=>{

dispatch(addGun())

}, 2000)

}

}

##### Auth.redux.js

const LOGIN = 'LOGIN'

const LOGOUT = 'LOGOUT'

export function auth(state={ isAuth: false, user: '李云龙'}, action) {

switch(action.type) {

case LOGIN:

return {...state, isAuth: true}

case LOGOUT:

return {...state, isAuth: false}

default:

return state

}

}

// action

export function login() {

return { type: LOGIN }

}

export function logout() {

return { type: LOGOUT }

}

##### reducer.js（合并所有reducer并返回）

// 合并所有reducer 并返回

import { combineReducers } from 'redux'

import { counter } from './index.redux'

import { auth } from './Auth.redux'

export default combineReducers({counter, auth})

##### index.js

import React from 'react';

import ReactDOM from 'react-dom';

import { Provider } from 'react-redux'

import { createStore, applyMiddleware, compose } from 'redux'

import thunk from 'redux-thunk'

import {

BrowserRouter,

Route,

Redirect,

Switch

} from 'react-router-dom'

// import { counter } from './index.redux'

import reducers from './reducer'

import Auth from './Auth'

import Dashboard from './Dashboard'

// compose把多个中间件连接在一块

const store = createStore(reducers, compose(

applyMiddleware(thunk),

window.devToolsExtension? window.devToolsExtension(): f=>f // 使用redux-devtools

))

console.log(store.getState())

ReactDOM.render(

(<Provider store={ store }>

<BrowserRouter>

<Switch> {/\*只渲染命中的第一个Route\*/}

<Route path='/login' exact component={Auth}></Route>

<Route path='/dashboard' component={Dashboard}></Route>

<Redirect to='/dashboard'></Redirect>

</Switch>

</BrowserRouter>

</Provider>),

document.getElementById('root')

);

##### Auth.js

import React from 'react'

import { connect } from 'react-redux'

import { Redirect } from 'react-router-dom'

import { login } from './Auth.redux'

// 有两个reducers 每个reducers都有一个state

// 合并reducer

@connect(

state=>state.auth,

{ login }

)

class Auth extends React.Component{

render() {

return (

<div>

{ this.props.isAuth? <Redirect to='/dashboard'></Redirect>: null }

<h2>你没有权限， 需要登陆才能查看</h2>

<button onClick={this.props.login}>登录</button>

</div>

)

}

}

export default Auth

##### Dashboard.js

import React from 'react'

import { Link, Route, Redirect } from 'react-router-dom'

import { connect } from 'react-redux'

import App from './App';

import { logout } from './Auth.redux'

function Erying() {

return <h2>二营</h2>

}

function Qibinglian() {

return <h2>骑兵连</h2>

}

@connect(

state => state.auth,

{ logout }

)

class Dashboard extends React.Component{

render() {

const match = this.props.match

const redirectToLogin = <Redirect to='/login'></Redirect>

const app = (

<div>

<h1>独立团</h1>

{ this.props.isAuth? <button onClick={this.props.logout}>注销</button>: null }

<ul>

<li><Link to={`${match.url}/`}>一营</Link></li>

<li><Link to={`${match.url}/erying`}>二营</Link></li>

<li><Link to={`${match.url}/qibinglian`}>骑兵连</Link></li>

</ul>

<Route path={`${match.url}/`} exact component={App}></Route>

<Route path={`${match.url}/erying`} component={Erying}></Route>

<Route path={`${match.url}/qibinglian`} component={Qibinglian}></Route>

</div>

)

return this.props.isAuth? app : redirectToLogin

}

}

export default Dashboard

##### App.js

import React from 'react'

import { connect } from 'react-redux'

import { addGun, removeGun, addGunAsync } from './index.redux'

// 使用装饰器

@connect(

// 你要state什么属性放到props里

state=>({num:state.counter}),

// 你要什么方法 放到props里 自动dispatch

{ addGun, removeGun, addGunAsync }

)

class App extends React.Component{

render() {

return (

<div className="App">

<h3>现在有机枪{ this.props.num }把</h3>

<button onClick={this.props.addGun}>申请武器</button>

<button onClick={this.props.removeGun}>上交武器</button>

<button onClick={this.props.addGunAsync}>拖两天给武器</button>

</div>

);

}

}

export default App

## 前后端联调

### Server.js

const app = require('express')()

const mongoose = require('mongoose')

const http = require('http')

// 链接mongodb

const DB\_URL = 'mongodb://129.28.187.206:27017/boss'

mongoose.connect(DB\_URL)

mongoose.connection.on('connected', function() {

console.log('mongo connect success !')

})

const User = mongoose.model('user', new mongoose.Schema({

user: { type: String, require: true },

age: { type: Number, require: true }

}))

// 新增数据

// User.create({

// user: 'xiaohua',

// age: 12

// }, function(err, doc) {

// if (!err) {

// console.log(doc)

// } else {

// console.log(err)

// }

// })

// 新建app

// User.remove({ age: 18 }, function(err, doc) {

// console.log(doc)

// })

// User.update({ 'user': 'xiaoming' }, { '$set': { age: 26 } }, function(err, doc){

// console.log(doc)

// })

app.get('/', function(req, res) {

res.send('<h1>Hello world</h1>')

})

app.get('/data', function(req, res){

User.findOne({'user': 'xiaoming'}, function(err, doc) {

res.json(doc)

})

})

app.get('/delete', function(req, res) {

res.json({

code: 0

})

})

http.createServer(app).listen(3002);

### conf.js (axios拦截器)

import axios from 'axios'

import { Toast } from 'antd-mobile'

// 拦截请求

axios.interceptors.request.use(function(config) {

Toast.loading('加载中', 0)

return config

})

// 拦截响应

axios.interceptors.response.use(function(config) {

setTimeout(() => {

Toast.hide()

}, 2000)

return config

})

### Auth.redux.js

import axios from 'axios'

const LOGIN = 'LOGIN'

const LOGOUT = 'LOGOUT'

const USER\_DATA = "USER\_DATA"

const initState = {

isAuth: false,

user: '李云龙',

age: 18

}

export function auth(state=initState, action) {

switch(action.type) {

case LOGIN:

return {...state, isAuth: true}

case LOGOUT:

return {...state, isAuth: false}

case USER\_DATA:

return {...state, user: action.payload.user, age: action.payload.age}

default:

return state

}

}

// action

export function getUserData() {

// dispatch 用来通知数据修改

return dispatch => {

axios.get('/data').then(res => {

if(res.status === 200) {

dispatch(userData(res.data))

}

})

}

}

export function userData(data) {

return {type: USER\_DATA, payload: data}

}

export function login() {

return { type: LOGIN }

}

export function logout() {

return { type: LOGOUT }

}

### Auth.js

import React from 'react'

import axios from 'axios'

import { connect } from 'react-redux'

import { Redirect } from 'react-router-dom'

import { login, getUserData } from './Auth.redux'

// 有两个reducers 每个reducers都有一个state

// 合并reducer

@connect(

state=>state.auth,

{ login, getUserData }

)

class Auth extends React.Component{

constructor(props) {

super(props)

this.state = {

data: {}

}

}

// componentDidMount() {

// this.props.getUserData()

// axios.get('/data').then(res => {

// if(res.status === 200) {

// this.setState({data: res.data})

// }

// })

// }

render() {

return (

<div>

<h2>我的名字是： { this.props.user }, 年龄： {this.props.age}</h2>

{ this.props.isAuth? <Redirect to='/dashboard'></Redirect>: null }

<h2>你没有权限， 需要登陆才能查看</h2>

<button onClick={this.props.login}>登录</button>

</div>

)

}

}

export default Auth

## 招聘WebApp项目

### 准备

#### 安装

基于前面项目令需安装

cookie相关 yarn add cookie-parser

package.json里面引入

// antd-mobile按需加载

"@babel/plugin-import": "^1.6.2",

"babel": {

"presets": [

"react-app"

],

"plugins": [

[

"import",

{

"libraryName": "antd-mobile",

"style": "css"

}

],

[

"@babel/plugin-proposal-decorators",

{

"legacy": true

}

]

]

},

### 登录注册模块

#### user.redux.js

import axios from 'axios'

import { getRedirectPath } from '../util'

const REGISTER\_SUCCESS = 'REGISTER\_SUCCESS'

const LOGIN\_SUCCESS = 'LOGIN\_SUCCESS'

const ERROR\_MSG = 'ERROR\_MSG'

const LOAD\_DATA = 'LOAD\_DATA'

const initState = {

redirectTo: '',

msg: '',

isAuth: false,

user: '',

type: ''

}

// reducer

export function user(state = initState, action){

switch(action.type) {

case REGISTER\_SUCCESS:

return { ...state, msg: '', redirectTo: getRedirectPath(action.payload), isAuth: true, ...action.payload }

case LOGIN\_SUCCESS:

return { ...state, msg: '', redirectTo: getRedirectPath(action.payload), isAuth: true, ...action.payload }

case ERROR\_MSG:

return { ...state, isAuth: false, msg: action.msg }

case LOAD\_DATA:

return {...state, ...action.payload}

default:

return state

}

}

function loginSuccess(data) {

return { type: LOGIN\_SUCCESS, payload: data }

}

function registerSuccess(data) {

return { type: REGISTER\_SUCCESS, payload: data }

}

function errorMsg(msg) {

return { msg, type: ERROR\_MSG}

}

// 用户信息

export function loadData(userinfo) {

return { type: LOAD\_DATA, payload: userinfo }

}

// 登录

export function login({user, pwd}){

if (!user || !pwd) {

return errorMsg('用户名密码必须输入')

}

return dispatch => {

axios.post('/user/login', { user, pwd })

.then(res => {

if (res.status === 200 && res.data.code === 0) {

dispatch(loginSuccess(res.data.data))

} else {

dispatch(errorMsg(res.data.msg))

}

})

}

}

// 注册

export function register({user, pwd, repeatpwd, type}) {

if (!user||!pwd||!type) {

return errorMsg('用户名密码必须输入')

}

if (pwd!==repeatpwd) {

return errorMsg('两次密码输入不一致')

}

return dispatch => {

axios.post('/user/register', { user, pwd, type })

.then(res => {

if (res.status === 200 && res.data.code === 0) {

dispatch(registerSuccess({ user, pwd, type }))

} else {

dispatch(errorMsg(res.data.msg))

}

})

}

}

#### index.js

import React from 'react';

import ReactDOM from 'react-dom';

import { Provider } from 'react-redux'

import { createStore, applyMiddleware, compose } from 'redux'

import thunk from 'redux-thunk'

import {

BrowserRouter,

Route,

} from 'react-router-dom'

import reducers from './reducer'

import Login from './container/login/login'

import Register from './container/register/register'

import BossInfo from './container/bossinfo/bossinfo'

import AuthRoute from './component/authroute/authroute'

import './config'

import './index.css'

import 'antd-mobile/dist/antd-mobile.css'

// compose把多个中间件连接在一块

const store = createStore(reducers, compose(

applyMiddleware(thunk),

window.devToolsExtension? window.devToolsExtension(): f=>f // 使用redux-devtools

))

ReactDOM.render(

(<Provider store={ store }>

<BrowserRouter>

<div>

<AuthRoute></AuthRoute>

<Route path='/bossinfo' component={ BossInfo }></Route>

<Route path='/login' component={ Login }></Route>

<Route path='/register' component={ Register }></Route>

</div>

</BrowserRouter>

</Provider>),

document.getElementById('root')

);

#### authroute.js

import React from 'react'

import axios from 'axios'

import { withRouter } from 'react-router-dom'

import { loadData } from '../../redux/user.redux'

import { connect } from 'react-redux'

@withRouter // 注意若想在此获取router的相关信息,需要引入react-router4的withRouter并进行包裹

@connect(

state => state.user,

{ loadData }

)

class AuthRoute extends React.Component{

async componentDidMount() {

const publicList = ['/login', '/register']

const pathname = this.props.location.pathname

if (publicList.indexOf(pathname) > -1) {

return null

}

// 获取用户信息

let res = await axios.get('/user/info')

if (res.status === 200) {

if (res.data.code === 0) {

// 有登陆信息

this.props.loadData(res.data.data)

} else {

// 无登陆信息

// 注意若想在此获取router的相关信息,需要引入react-router4的withRouter并进行包裹

this.props.history.push('/login')

}

}

}

render() {

return null

}

}

export default AuthRoute

#### logo.js

import React from 'react'

import logoImg from './job.png'

import './logo.css'

class Logo extends React.Component{

    render(){

        return (

            <div className="logo-container">

                <img src={logoImg} alt=""/>

            </div>

        )

    }

}

export default Logo

#### login.js

import React from 'react'

import Logo from '../../component/logo/logo'

import { List, InputItem, WingBlank, WhiteSpace, Button } from 'antd-mobile'

import { Redirect } from 'react-router-dom'

import { connect } from 'react-redux'

import { login } from '../../redux/user.redux'

@connect(

state => state.user,

{ login }

)

class Login extends React.Component{

constructor(props) {

super(props)

this.state = {

user: '',

pwd: ''

}

this.handleLogin = this.handleLogin.bind(this)

this.register = this.register.bind(this)

}

register() {

this.props.history.push('/register')

}

handleChange(key, val) {

this.setState({

[key]: val

})

}

handleLogin() {

this.props.login(this.state)

}

render() {

return (

<div>

{ this.props.redirectTo?<Redirect to={ this.props.redirectTo } />: null }

<Logo></Logo>

<WingBlank>

<List>

<InputItem

onChange={v => this.handleChange('user', v)}

>用户名</InputItem>

<WhiteSpace />

<InputItem

onChange={v => this.handleChange('pwd', v)}

type='password'

>密码</InputItem>

</List>

<Button onClick={ this.handleLogin } type='primary'>登陆</Button>

<WhiteSpace />

<Button onClick={ this.register } type='primary'>注册</Button>

</WingBlank>

</div>

)

}

}

export default Login

#### register.js

import React from 'react'

import Logo from '../../component/logo/logo'

import { List, InputItem, Radio, WhiteSpace, Button } from 'antd-mobile'

import { Redirect } from 'react-router-dom'

import { connect } from 'react-redux'

import { register } from '../../redux/user.redux'

@connect(

state => state.user,

{ register }

)

class Register extends React.Component{

constructor(props) {

super(props)

this.state = {

user: '',

pwd: '',

repeatpwd: '',

type: 'genius' //或者叫boss

}

this.handleRegister = this.handleRegister.bind(this) // 第一种方式（性能好）

}

handleChange(key, val) {

this.setState({

[key]: val

})

}

handleRegister() {

console.log(this.state)

this.props.register(this.state)

}

render() {

const RadioItem = Radio.RadioItem

return (

<div>

{ this.props.redirectTo?<Redirect to={ this.props.redirectTo } />: null }

<Logo></Logo>

<List>

{ this.props.msg? <p className='error-msg'>{this.props.msg}</p>: '' }

<InputItem

onChange={v => this.handleChange('user', v)}

>用户名</InputItem> {/\* 第二种方式（更简单）\*/}

<WhiteSpace />

<InputItem

onChange={v => this.handleChange('pwd', v)}

type='password'

>密码</InputItem>

<WhiteSpace />

<InputItem

onChange={v => this.handleChange('repeatpwd', v)}

type='password'

>确认密码</InputItem>

<WhiteSpace />

<RadioItem

checked={ this.state.type === 'genius' }

onChange={() => this.handleChange('type', 'genius')}

>牛人

</RadioItem>

<WhiteSpace />

<RadioItem

checked={ this.state.type === 'boss' }

onChange={() => this.handleChange('type', 'boss')}

>BOSS

</RadioItem>

<WhiteSpace />

<Button type="primary" onClick={this.handleRegister}>注册</Button>

</List>

</div>

)

}

}

export default Register

### boss信息完善模块

#### 安装

##### yarn add prop-types 用于数据类型校验

#### avatar-select.js

import React from 'react'

import { Grid, List } from 'antd-mobile'

import PropTypes from 'prop-types'

class AvatarSelector extends React.Component{

// 数据内容检测

static propTypes = {

selectAvatar: PropTypes.func.isRequired

}

constructor(pros) {

super(pros)

this.state = {}

}

render() {

const AvatarList = 'boy,girl,man,woman,bull,chick,crab,hedgehog,hippopotamus,koala,lemur,pig,tiger,whale,zebra'

.split(',')

.map(v=>({

icon:require(`../img/${v}.png`),

text:v

}))

const girdHeader = this.state.icon?

(<div>

<span>已选选择头像</span>

<img style={{width:20,marginLeft:8}} src={this.state.icon} alt='' />

</div>):

'请选择头像'

return (

<div>

<List renderHeader={() => girdHeader}>

<Grid

data={AvatarList}

activeStyle={false}

columnNum={5}

onClick={ele=>{

this.setState(ele)

this.props.selectAvatar(ele.text)

}}

/>

</List>

</div>

)

}

}

export default AvatarSelector

#### bossinfo.js

import React from 'react'

import { Redirect } from 'react-router-dom'

import { NavBar, TextareaItem, InputItem, Button } from 'antd-mobile'

import AvatarSelector from '../../component/avatar-select/avatar-select'

import { connect } from 'react-redux'

import { update } from '../../redux/user.redux'

@connect(

state => state.user,

{ update }

)

class BossInfo extends React.Component{

constructor(pros) {

super(pros)

this.state = {

title: ''

}

}

onChange(key, val) {

this.setState({

[key]: val

})

}

render() {

const path = this.props.location.pathname

const redirect = this.props.redirectTo

return (

<div>

{ this.props.redirectTo && path !== redirect?

<Redirect to={ this.props.redirectTo }></Redirect> :

null

}

<NavBar mode="dark">BOSS完善信息页面</NavBar>

<AvatarSelector

selectAvatar={(imgname) => {

console.log(imgname)

this.setState({

avatar: imgname

})

}}

></AvatarSelector>

<InputItem onChange={(v) => this.onChange('title',v)}>

招聘职位

</InputItem>

<InputItem onChange={(v) => this.onChange('company',v)}>

公司名称

</InputItem>

<InputItem onChange={(v) => this.onChange('money',v)}>

职位薪资

</InputItem>

<TextareaItem

rows={3}

autoHeight

title='职位要求'

onChange={(v) => this.onChange('desc',v)}>

</TextareaItem>

<Button

type='primary'

onClick={() => {

this.props.update(this.state)

}}

>保存</Button>

</div>

)

}

}

export default BossInfo

#### user.redux.js完善信息相关

import axios from 'axios'

import { getRedirectPath } from '../util'

const AUTH\_SUCCESS = 'AUTH\_SUCCESS'

const ERROR\_MSG = 'ERROR\_MSG'

const LOAD\_DATA = 'LOAD\_DATA'

const initState = {

redirectTo: '',

msg: '',

user: '',

type: ''

}

// reducer

export function user(state = initState, action){

switch(action.type) {

case AUTH\_SUCCESS:

return { ...state, msg: '', redirectTo: getRedirectPath(action.payload), ...action.payload }

case ERROR\_MSG:

return { ...state, isAuth: false, msg: action.msg }

case LOAD\_DATA:

return {...state, ...action.payload}

default:

return state

}

}

function authSuccess(obj) {

const { pwd, ...data } = obj // 小技巧 过滤掉pwd字段

return { type: AUTH\_SUCCESS, payload: data }

}

function errorMsg(msg) {

return { msg, type: ERROR\_MSG}

}

// 用户信息

export function loadData(userinfo) {

return { type: LOAD\_DATA, payload: userinfo }

}

export function update(data) {

return async dispatch => {

let res = await axios.post('/user/update', data)

if (res.status === 200 && res.data.code === 0) {

dispatch(authSuccess(res.data.data))

} else {

dispatch(errorMsg(res.data.msg))

}

}

}

#### 后端 user.js更新信息模块

Router.post('/update', function(req, res) {

const userid = req.cookies.userid

if (!userid) {

return json.dumps({code: 1})

}

const body = req.body

User.findByIdAndUpdate(userid, body, (err, doc) => {

console.log(doc)

const data = Object.assign({}, {

user: doc.user,

type: doc.type

}, body)

return res.json({ code: 0, data })

})

})

### 牛人信息完善模块

#### geniusinfo.js

import React from 'react'

import { Redirect } from 'react-router-dom'

import { NavBar, TextareaItem, InputItem, Button } from 'antd-mobile'

import AvatarSelector from '../../component/avatar-select/avatar-select'

import { connect } from 'react-redux'

import { update } from '../../redux/user.redux'

@connect(

state => state.user,

{ update }

)

class GeniusInfo extends React.Component{

constructor(pros) {

super(pros)

this.state = {

title: '',

desc: ''

}

}

onChange(key, val) {

this.setState({

[key]: val

})

}

render() {

const path = this.props.location.pathname

const redirect = this.props.redirectTo

return (

<div>

{ this.props.redirectTo && path !== redirect?

<Redirect to={ this.props.redirectTo }></Redirect> :

null

}

<NavBar mode="dark">牛人完善信息页面</NavBar>

<AvatarSelector

selectAvatar={(imgname) => {

console.log(imgname)

this.setState({

avatar: imgname

})

}}

></AvatarSelector>

<InputItem onChange={(v) => this.onChange('title',v)}>

求职岗位

</InputItem>

<TextareaItem

rows={3}

autoHeight

title='个人简介'

onChange={(v) => this.onChange('desc',v)}>

</TextareaItem>

<Button

type='primary'

onClick={() => {

this.props.update(this.state)

}}

>保存</Button>

</div>

)

}

}

export default GeniusInfo

### 应用骨架的建立

#### Dashboard.js(主体骨架)

import React from 'react'

import { Switch, Route } from 'react-router-dom'

import { connect } from 'react-redux'

import { NavBar } from 'antd-mobile'

import NavLinkBar from '../../component/navlink/navlink'

import Boss from '../../component/boss/boss'

function Genius() {

return <h2>Genius首页</h2>

}

function Msg() {

return <h2>消息列表页</h2>

}

function User() {

return <h2>个人中心</h2>

}

@connect(

state => state

)

class Dashboard extends React.Component{

render() {

const { pathname } = this.props.location

const user = this.props.user

const navList = [

{

path: '/boss',

text: '牛人',

icon: 'boss',

title: '牛人列表',

component: Boss,

hide: user.type === 'genius'

},

{

path: '/genius',

text: 'boss',

icon: 'job',

title: 'BOSS列表',

component: Genius,

hide: user.type === 'boss'

},

{

path: '/msg',

text: '消息',

icon: 'msg',

title: '消息列表',

component: Msg,

},

{

path: '/me',

text: '我',

icon: 'user',

title: '个人中心',

component: User,

}

]

return (

<div>

<NavBar className='fixd-header' mode='dard'>

{navList.find(v=> v.path === pathname).title}

</NavBar>

<div style={{marginTop:45}}>

<Switch>

{ navList.map(v => (

<Route

key={ v.path }

path={ v.path }

component={ v.component }

/>

))}

</Switch>

</div>

<NavLinkBar data={ navList }></NavLinkBar>

</div>

)

}

}

export default Dashboard

#### navlink.js(底部菜单)

import React from 'react'

import PropTypes from 'prop-types'

import { TabBar } from 'antd-mobile';

import { withRouter } from 'react-router-dom'

@withRouter

class NavLink extends React.Component{

static propTypes = {

data: PropTypes.array.isRequired

}

render() {

const navList = this.props.data.filter(v => !v.hide)

const { pathname } = this.props.location

return (

<TabBar>

{

navList.map(v => (

<TabBar.Item

key= { v.path }

title = { v.text }

icon = {{ uri: require(`./img/${v.icon}.png`) }}

selectedIcon = {{ uri: require(`./img/${v.icon}-active.png`) }}

selected = { pathname === v.path }

onPress={() => {

this.props.history.push(v.path)

}}

>

</TabBar.Item>

))

}

</TabBar>

)

}

}

export default NavLink

#### boss.js页

import React from 'react'

import axios from 'axios'

import { connect } from 'react-redux'

import { Card, WhiteSpace, WingBlank } from 'antd-mobile'

import { getUserList } from '../../redux/chatuser.redux'

@connect(

state => state.chatuser,

{ getUserList }

)

class Boss extends React.Component{

constructor(props) {

super(props)

this.state = {

data: []

}

}

async componentDidMount() {

this.props.getUserList('genius')

}

render() {

const Header = Card.Header

const Body = Card.Body

return (

<WingBlank>

<WhiteSpace></WhiteSpace>

{ this.props.userlist.map(v => (

v.avatar? (<Card key={v.\_id}>

<Header

title = { v.user }

thumb = { require(`../img/${v.avatar}.png`)}

extra = { <span>{ v.title }</span> }

/>

<Body>{ v.desc.split('\n').map(item => (

<div key={ item }>{ item }</div>

)) }</Body>

</Card>) : null

)) }

</WingBlank>

)

}

}

export default Boss

#### chatuser.js(用户列表)

import axios from 'axios'

const USER\_LIST = 'USER\_LIST'

const initState = {

userlist: []

}

export function chatuser(state=initState, action) {

switch(action.type) {

case USER\_LIST:

return { ...state, userlist: action.payload }

default:

return state

}

}

function userList(data) {

return { type: USER\_LIST, payload: data }

}

export function getUserList(type) {

return async dispatch => {

let res = await axios.get('/user/list?type=genius')

if (res && res.data.code === 0) {

dispatch(userList(res.data.data))

}

}

}

### Boss列表和牛人列表的组件化

#### usercard.js（公共组件）

import React from 'react'

import PropTypes from 'prop-types'

import { Card, WhiteSpace, WingBlank } from 'antd-mobile'

class UserCard extends React.Component{

static propTypes = {

userlist: PropTypes.array.isRequired

}

render() {

const Header = Card.Header

const Body = Card.Body

console.log(this.props.userlist)

return (

<WingBlank>

<WhiteSpace></WhiteSpace>

{ this.props.userlist.map(v => (

v.avatar? (<Card key={v.\_id}>

<Header

title = { v.user }

thumb = { require(`../img/${v.avatar}.png`)}

extra = { <span>{ v.title }</span> }

/>

<Body>

{ v.type === 'boss'? <div>公司：{ v.company }</div>: null }

{ v.desc.split('\n').map(item => (

<div key={ item }>{ item }</div>

)) }

{ v.type === 'boss'? <div>薪资：{ v.money }</div>: null }

</Body>

</Card>) : null

)) }

</WingBlank>

)

}

}

export default UserCard

#### boss列表组件化（boss.js）

import React from 'react'

import { connect } from 'react-redux'

import { Card, WhiteSpace, WingBlank } from 'antd-mobile'

import { getUserList } from '../../redux/chatuser.redux'

import UserCard from '../usercard/usercard'

@connect(

state => state.chatuser,

{ getUserList }

)

class Boss extends React.Component{

async componentDidMount() {

this.props.getUserList('genius')

}

render() {

return (

<UserCard userlist={ this.props.userlist }></UserCard>

)

}

}

export default Boss

#### 牛人列表组件化

import React from 'react'

import { connect } from 'react-redux'

import { Card, WhiteSpace, WingBlank } from 'antd-mobile'

import { getUserList } from '../../redux/chatuser.redux'

import UserCard from '../usercard/usercard'

@connect(

state => state.chatuser,

{ getUserList }

)

class Genius extends React.Component{

async componentDidMount() {

this.props.getUserList('boss')

}

render() {

return <UserCard userlist={ this.props.userlist }></UserCard>

}

}

export default Genius

### 用户个人中心页

#### cookie的插件（browser-cookies）

yarn add browser-cookies

使用方法

var cookies **=** require('browser-cookies');

cookies.set('firstName', 'Lisa');

cookies.set('firstName', 'Lisa', {expires**:** 365}); *// Expires after 1 year*

cookies.set('firstName', 'Lisa', {secure**:** true, domain**:** 'www.example.org'});

cookies.get('firstName'); *// Returns cookie value (or null)*

cookies.erase('firstName'); *// Removes cookie*

#### user.js(用户个人主页)

import React from 'react'

import { connect } from 'react-redux'

import { Result, List, WhiteSpace, Modal } from 'antd-mobile'

import browserCookies from 'browser-cookies'

import { logoutSubmit } from '../../redux/user.redux'

import { Redirect } from 'react-router-dom'

@connect(

state => state.user,

{ logoutSubmit }

)

class User extends React.Component{

constructor(props) {

super(props)

this.logout = this.logout.bind(this)

}

logout() {

const alert = Modal.alert

alert('注销', '确认退出吗？', [

{ text: '取消', onPress: () => console.log('cancel') },

{ text: '确认', onPress: () => {

browserCookies.erase('userid')

// window.location.reload()

this.props.logoutSubmit()

} }

])

}

render() {

const props = this.props

const Item = List.Item

const Brief = Item.Brief

return props.user ? (

<div>

<Result

img={ <img src={require(`../img/${props.avatar}.png`)} style={{width: 50}} alt='' /> }

title={ props.user }

message={ props.type === 'boss' ? props.company: '' }

/>

<List

renderHeader = {() => '简介' }>

<Item

multipleLine = { true }

>

{ props.title }

{ this.props.desc.split('\n').map(v=><Brief key={v}>{ v }</Brief>) }

{ props.money ? <Brief>薪资:{ props.money }</Brief> : null }

</Item>

</List>

<WhiteSpace></WhiteSpace>

<List>

<Item onClick={ this.logout }>退出登录</Item>

</List>

</div>

) : <Redirect to={ props.redirectTo } /> // 到登录页

}

}

export default User

#### user.redux.js(相关代码)

const LOGOUT = 'LOGOUT'

// reducer

export function user(state = initState, action){

switch(action.type) {

case AUTH\_SUCCESS:

return { ...state, msg: '', redirectTo: getRedirectPath(action.payload) }

case ERROR\_MSG:

return { ...state, isAuth: false, msg: action.msg }

case LOAD\_DATA:

return {...state, ...action.payload}

case LOGOUT:

return { ...initState, redirectTo: './login' }

default:

return state

}

}

// 退出登录

export function logoutSubmit() {

return { type: LOGOUT }

}

### 函数式编程

function hello() {

console.log('hello world')

}

Hello()

function WrapperHello(fn) {

return function() {

console.log('before say hello')

fn()

console.log('after say hello')

}

}

hello = WrapperHello(hello)

hello()

这段代码执行的虽然依旧是hello(),但返回的已经不只是原hello返回到额内容了，这是一种类似装饰器的HOC（高阶组件）的写法

### HOC高阶组件

高阶函数有两个主要功能

1. 属性代理
2. 反向继承

#### 属性代理

class Hello extends React.Component{

render() {

return (

<h2>hello HOC</h2>

)

}

}

function WrapperHello(Comp) {

class WaroComp extends React.Component{

render() {

return (

<div>

<p>这是HOC高阶组件特有元素</p>

<Comp {...this.props}></Comp>

</div>

)

}

}

return WaroComp

}

Hello = WrapperHello(Hello)

##### 使用装饰器的效果

function WrapperHello(Comp) {

class WaroComp extends React.Component{

render() {

return (

<div>

<p>这是HOC高阶组件特有元素</p>

<Comp {...this.props}></Comp>

</div>

)

}

}

return WaroComp

}

@WrapperHello

class Hello extends React.Component{

render() {

return (

<h2>hello HOC</h2>

)

}

}

给一个组件返回另外一个组件

定义一个函数，传进来一个组件，参数是一个组件，返回另外一个组件，另外一个组件包裹了之前的组件

#### 反向继承

// 反向继承

function WrapperHello(Comp) {

class WrapComp extends Comp {

componentDidMount() {

console.log('高阶组件新增的生命周期，加载完成')

}

render() {

return <Comp></Comp>

}

}

return WrapComp

}

@WrapperHello

class Hello extends React.Component{

render() {

return (

<h2>hello HOC</h2>

)

}

}

#### 简单高阶组件的封装

Immoc-form.js (封装增加hadleChange函数)

import React from 'react'

export default function imoocForm (Comp) {

return class WrapperComp extends React.Component{

constructor(props) {

super(props)

this.state = {}

this.handleChange = this.handleChange.bind(this)

}

handleChange(key, val) {

console.log(key, val)

this.setState({

[key]: val

})

}

render() {

return <Comp handleChange={this.handleChange} state={this.state} {...this.props}></Comp>

}

}

}

此时，login.js修改为

import React from 'react'

import Logo from '../../component/logo/logo'

import { List, InputItem, WingBlank, WhiteSpace, Button } from 'antd-mobile'

import { Redirect } from 'react-router-dom'

import { connect } from 'react-redux'

import { login } from '../../redux/user.redux'

import immocFrom from '../../component/imooc-form/imooc-form'

@connect(

state => state.user,

{ login }

)

@immocFrom // 引入并包裹组件 此时state应为this.props.state

class Login extends React.Component{

constructor(props) {

super(props)

this.handleLogin = this.handleLogin.bind(this)

this.register = this.register.bind(this)

}

register() {

this.props.history.push('/register')

}

handleLogin() {

this.props.login(this.props.state)

}

render() {

return (

<div>

{ this.props.redirectTo&&this.props.redirectTo!=='/login'?<Redirect to={ this.props.redirectTo } />: null }

<Logo></Logo>

<WingBlank>

<List>

<InputItem

onChange={v => this.props.handleChange('user', v)}

>用户名</InputItem>

<WhiteSpace />

<InputItem

onChange={v => this.props.handleChange('pwd', v)}

type='password'

>密码</InputItem>

</List>

<Button onClick={ this.handleLogin } type='primary'>登陆</Button>

<WhiteSpace />

<Button onClick={ this.register } type='primary'>注册</Button>

</WingBlank>

</div>

)

}

}

export default Login

### 聊天功能实现

#### socket.io

安装

yarn add socket.io （服务端）

yarn add socket.io-client （客户端）

#### model.js（服务端）

const mongoose = require('mongoose')

mongoose.set('useFindAndModify', false)

// 链接mongodb

const DB\_URL = 'mongodb://129.28.187.206:27017/boss'

mongoose.connect(DB\_URL, {useNewUrlParser: true})

const models = {

user: {

'user': { type: String, require: true },

'pwd': { type: String, require: true },

'type': { type: String, require: true },

'avatar': { type: String }, // 头像

'desc': { type: String }, // 个人简介/职位简介

'title': { type: String }, // 职位名称

// 如果是boss则还有两个字段

'company': { type: String }, // 公司

'money': { type: String } // 薪资

},

chat: {

'chatid': { type: String, require: true},

'from': { type: String, require: true },

'to': { type: String, require: true },

'read': { type: Boolean, default: false },

'content': { type: String, require: true, default: '' },

'create\_time': { type: Number, default: new Date().getTime() }

}

}

for (let m in models) {

mongoose.model(m, new mongoose.Schema(models[m]))

}

module.exports = {

getModel: function(name) {

return mongoose.model(name)

}

}

#### Sever.js（服务端）

const app = require('express')()

const userRoute = require('./user')

const bodyParser = require('body-parser')

const cookieParser = require('cookie-parser')

const model = require('./model')

// const User = model.getModel('user')

const Chat = model.getModel('chat')

// work with express

const server = require('http').Server(app)

const io = require('socket.io')(server)

io.on('connection', function(socket) {

socket.on('sendmsg', function(data){

const { from, to, msg } = data

const chatid = [from, to].sort().join('\_')

Chat.create({ chatid, from, to, content: msg }, function(err, doc) {

io.emit('recvmsg', Object.assign({}, doc.\_doc))

})

})

})

app.use(cookieParser())

app.use(bodyParser.json())

app.use('/user', userRoute)

server.listen(3002, function() {

console.log('node server listen 3002...')

})

#### User.js（服务端）

const express = require('express')

const Router = express.Router()

const model = require('./model')

const User = model.getModel('user')

const Chat = model.getModel('chat')

const utils = require('utility')

const \_filter = { 'pwd': 0, '\_\_v': 0 }

// 用户列表

Router.get('/list', function(req, res) {

const { type } = req.query

User.find({ type }, function(err, doc) {

return res.json({code: 0, data: doc})

})

})

// 消息列表

Router.get('/getmsglist', (req, res) => {

const user = req.cookies.userid

User.find({}, function(err, userdoc) {

let users = {}

userdoc.forEach(v => {

users[v.\_id] = { name: v.user, avatar: v.avatar }

})

Chat.find({'$or': [{from: user}, {to: user}]}, function(err,doc) {

if(!err) {

return res.json({code: 0, msgs: doc, users: users})

}

})

})

// {'$or':[{ from: user, to: user }]}

})

// 登录功能

Router.post('/login', (req, res) => {

const { user, pwd } = req.body

User.findOne({user, pwd: md5Pwd(pwd)}, \_filter, function(err, doc) {

if (!doc) {

return res.json({code: 1, msg: '用户名不存在或者密码错误'})

}

res.cookie('userid', doc.\_id)

return res.json({ code: 0, data: doc })

})

})

// 注册

Router.post('/register', function(req, res) {

const { user, pwd, type } = req.body

User.findOne({ user }, (err, doc)=>{

if(doc) {

return res.json({code: 1, msg: '用户名重复'})

}

// User.create({user, pwd: md5Pwd(pwd), type}, function(err, doc) {

// if (err) {

// return res.json({code: 1, msg: '注册失败!'})

// }

// return res.json({code: 0, msg: '注册成功!'})

// })

const userModel = new User({ user, pwd: md5Pwd(pwd), type })

userModel.save((err, doc) => {

if (err) {

return res.json({ code: 1, msg: '注册失败' })

} else {

const { user, type, \_id } = doc

res.cookie('userid', \_id)

return res.json({ code: 0, data: { user, type, \_id } })

}

})

})

})

Router.post('/update', function(req, res) {

const userid = req.cookies.userid

if (!userid) {

return res.json.dumps({code: 1})

}

const body = req.body

User.findByIdAndUpdate(userid, body, (err, doc) => {

const data = Object.assign({}, {

user: doc.user,

type: doc.type

}, body)

return res.json({ code: 0, data })

})

})

// 用户信息

Router.get('/info', function(req, res) {

const { userid } = req.cookies

if (!userid) {

return res.json({code: 1})

}

User.findOne({\_id: userid}, \_filter, function(err, doc) {

if (err) {

return res.json({code: 1, msg: '出错了'})

} else {

res.json({code: 0, data: doc})

}

})

})

// md5 加密

function md5Pwd(pwd) {

const salt = 'yqzdsa'

return utils.md5(utils.md5(pwd+salt))

}

module.exports = Router

#### util.js

export function getChatId(userId, targetId) {

return [ userId, targetId ].sort().join('\_')

}

#### chat-redux.js

import axios from 'axios'

import io from 'socket.io-client'

const socket = io('ws://localhost:3002')

// 获取聊天列表

const MSG\_LIST = 'MSG\_LIST'

// 读信息

const MSG\_RECV = 'MSG\_RECV'

// 标识已读

const MSG\_READ = 'MSG\_READ'

const initState = {

chatmsg: [],

users: {},

unread: 0

}

export function chat(state=initState, action) {

switch(action.type) {

case MSG\_LIST:

return { ...state, users: action.payload.users, chatmsg: action.payload.msgs, unread: action.payload.msgs.filter(v=>!v.read&&v.to===action.payload.userid).length }

case MSG\_RECV:

const n = action.payload.to === action.userid ? 1 : 0

return { ...state, chatmsg: [...state.chatmsg, action.payload], unread: state.unread + n }

// case MSG\_READ:

default:

return state

}

}

function msgList(msgs, users, userid) {

return { type: MSG\_LIST, payload: {msgs, users, userid} }

}

function msgRecv(msg, userid) {

return { userid, type: MSG\_RECV, payload: msg }

}

export function recvMsg() {

return (dispatch, getState) => {

socket.on('recvmsg', function(data) {

const userid = getState().user.\_id

dispatch(msgRecv(data, userid))

})

}

}

// 发送聊天信息

export function sendMsg({from, to, msg}) {

return dispatch => {

socket.emit('sendmsg', {from, to, msg})

}

}

// 获取信息列表

export function getMsglist() {

return async (dispatch, getState) => {

let res = await axios.get('/user/getmsglist')

if(res.status === 200 && res.data.code === 0) {

const userid = getState().user.\_id

dispatch(msgList(res.data.msgs, res.data.users, userid))

}

}

}

#### chat.js

import React from 'react'

import { List, InputItem, NavBar, Icon, Grid } from 'antd-mobile'

// import io from 'socket.io-client'

import { connect } from 'react-redux'

import { getMsglist, sendMsg, recvMsg } from '../../redux/chat.redux'

import { getChatId } from '../../util'

// const socket = io('ws://localhost:3002')

@connect(

state => state,

{ getMsglist, sendMsg, recvMsg }

)

class Chat extends React.Component{

constructor(props){

super(props)

this.state = {

text: '',

msg: [],

showEmoji: false

}

}

// 解决Grid图标第一次渲染只能加载一次的bug

fixCarousel() {

setTimeout(function(){

window.dispatchEvent(new Event('resize'))

}, 0)

}

componentDidMount() {

if(!this.props.chat.chatmsg.length) {

this.props.getMsglist()

this.props.recvMsg()

}

this.fixCarousel()

}

handleSubmit() {

// socket.emit('sendmsg', {text: this.state.text})

const from = this.props.user.\_id

const to = this.props.match.params.user

const msg = this.state.text

this.props.sendMsg({ from, to, msg })

this.setState({text: ''})

}

render() {

const emoji = `😀 😃 😄 😂 🙂 😊 😇 🤩 😍 😙 😝 🤮 😟 😡 ❤ 🤔 👀 🐻 🌇 ⚽ 🙃 🐺 🐵 🤽‍ ♂️ 🏆 🏀 💗 🕑 🚫 ☢ 🍉 🍏 🍅 🥒 🍄 🍗 🍖 🌭 🥘 🥘 🍺 👩 🌱 ☀ 🕎`

.split(' ')

.filter(v => v)

.map(v => ({ text: v }))

const userid = this.props.match.params.user

const Item = List.Item

const users = this.props.chat.users

if(!users[userid]) {

return null

}

const chatid = getChatId(userid, this.props.user.\_id)

const chatmsgs = this.props.chat.chatmsg.filter(v => v.chatid === chatid)

return (

<div id='chat-page'>

<NavBar

mode='dark'

icon={<Icon type="left" />}

onLeftClick={() => {

this.props.history.goBack()

}}

>

{ users[userid].name }

</NavBar>

{ chatmsgs.map(v => {

const avatar = require(`../img/${users[v.from].avatar}.png`)

return v.from === userid?(

<List key={ v.\_id }>

<Item

thumb={ avatar }

>{ v.content }</Item>

</List>

):(

<List key={ v.\_id }>

<Item

// eslint-disable-next-line jsx-a11y/alt-text

extra={ <img src={avatar} /> }

className='chat-me'

>{ v.content }</Item>

</List>

)

}) }

<div className='stick-footer'>

<List>

<InputItem

placeholder='请输入'

value={ this.state.text }

onChange={v=>{

this.setState({text:v})

}}

extra={

<div>

{/\* eslint-disable-next-line jsx-a11y/accessible-emoji\*/}

<span

className='emoji\_span'

onClick = {() => {

this.setState({

showEmoji: !this.state.showEmoji

})

this.fixCarousel()

}}

>😃</span>

<span onClick={() => this.handleSubmit()}>发送</span>

</div>

}

></InputItem>

</List>

{ this.state.showEmoji ?

<Grid

data={ emoji }

columnNum={ 9 }

carouselMaxRow={ 4 }

isCarousel={ true }

onClick={el => {

this.setState({

text: this.state.text+el.text

})

}}

/> :

null}

</div>

</div>

)

}

}

export default Chat

### 消息列表实现

#### Msg.js

import React from 'react'

import { connect } from 'react-redux'

import { List, Badge } from 'antd-mobile'

@connect(

state => state

)

class Msg extends React.Component{

getlast(arr) {

return arr[arr.length-1]

}

render() {

const Item = List.Item

const Brief = Item.Brief

const userid = this.props.user.\_id

const userinfo = this.props.chat.users

// 按照聊天用户分组，根据chatid

const msgGroup = {}

this.props.chat.chatmsg.forEach(v => {

msgGroup[v.chatid] = msgGroup[v.chatid] || []

msgGroup[v.chatid].push(v)

})

const chatList = Object.values(msgGroup).sort((a, b) => {

const a\_last = this.getlast(a).create\_time

const b\_last = this.getlast(b).create\_time

return b\_last-a\_last

})

return (

<div>

{ chatList.map(v => {

const lastItem = this.getlast(v)

const targetId = v[0].from === userid? v[0].to: v[0].from

const unreadNum = v.filter(v=>!v.read && v.to===userid).length

if(!userinfo[targetId]) {

return null

}

return (

<List key={ lastItem.\_id }>

<Item

extra={ <Badge text={unreadNum}></Badge> }

thumb={ require(`../img/${userinfo[targetId].avatar}.png`) }

arrow="horizontal"

onClick={() => {

this.props.history.push(`/chat/${targetId}`)

}}

>

{ lastItem.content }

<Brief>{ userinfo[targetId].name }</Brief>

</Item>

</List>

)

}) }

</div>

)

}

}

export default Msg

### 未读消息修正

#### User.js（后端相关逻辑）

// 标志为已读

Router.post('/readmsg', function(req, res) {

const userid = req.cookies.userid

const { from } = req.body

Chat.updateMany(

{ from, to: userid },

{ '$set': { read: true } },

{ multi: true },

function(err, doc) {

if(!err) {

return res.json({ code: 0, num: doc.nModified })

} else {

return res.json({ code: 1, msg: '修改失败' })

}

})

})

#### Chat.redux.js

import axios from 'axios'

import io from 'socket.io-client'

const socket = io('ws://localhost:3002')

// 标识已读

const MSG\_READ = 'MSG\_READ'

const initState = {

chatmsg: [],

users: {},

unread: 0

}

export function chat(state=initState, action) {

switch(action.type) {

case MSG\_LIST:

return {

...state,

users: action.payload.users,

chatmsg: action.payload.msgs,

unread: action.payload.msgs.filter(v=>!v.read&&v.to===action.payload.userid).length

}

case MSG\_RECV:

const n = action.payload.to === action.userid ? 1 : 0

return { ...state, chatmsg: [...state.chatmsg, action.payload], unread: state.unread + n }

case MSG\_READ:

const { from, num } = action.payload

return {

...state,

chatmsg: state.chatmsg.map(v =>({ ...v, read: from === v.from ? true: v.read })), // 注意这里map的小技巧

unread: state.unread - num

}

default:

return state

}

}

function msgRead({from, userid , num}) {

return { type: MSG\_READ, payload: {from, userid, num} }

}

export function readMsg(from, to, msg) {

return async (dispatch, getState) => {

let res = await axios.post('/user/readmsg', { from })

const userid = getState().user.\_id

console.log(res)

if(res.status === 200 && res.data.code === 0) {

console.log(345678)

dispatch(msgRead({ userid, from, num: res.data.num }))

}

}

}

#### Chat.js

import React from 'react'

import { List, InputItem, NavBar, Icon, Grid } from 'antd-mobile'

// import io from 'socket.io-client'

import { connect } from 'react-redux'

import { getMsglist, sendMsg, recvMsg, readMsg } from '../../redux/chat.redux'

import { getChatId } from '../../util';

// const socket = io('ws://localhost:3002')

@connect(

state => state,

{ getMsglist, sendMsg, recvMsg, readMsg }

)

class Chat extends React.Component{

constructor(props){

super(props)

this.state = {

text: '',

msg: [],

showEmoji: false

}

}

// 解决Grid图标第一次渲染只能加载一次的bug

fixCarousel() {

setTimeout(function(){

window.dispatchEvent(new Event('resize'))

}, 0)

}

componentDidMount() {

if(!this.props.chat.chatmsg.length) {

this.props.getMsglist() // 获取消息列表

this.props.recvMsg() // 请求监听webscoket信息

}

const to = this.props.match.params.user

this.props.readMsg(to) // 读消息将read标志为已读

}

componentWillUnmount() {

const to = this.props.match.params.user

this.props.readMsg(to) // 读消息将read标志为已读

}

handleSubmit() {

// socket.emit('sendmsg', {text: this.state.text})

const from = this.props.user.\_id

const to = this.props.match.params.user

const msg = this.state.text

this.props.sendMsg({ from, to, msg })

this.setState({text: ''})

}

render() {

const emoji = `😀 😃 😄 😂 🙂 😊 😇 🤩 😍 😙 😝 🤮 😟 😡 ❤ 🤔 👀 🐻 🌇 ⚽ 🙃 🐺 🐵 🤽‍ ♂️ 🏆 🏀 💗 🕑 🚫 ☢ 🍉 🍏 🍅 🥒 🍄 🍗 🍖 🌭 🥘 🥘 🍺 👩 🌱 ☀ 🕎`

.split(' ')

.filter(v => v)

.map(v => ({ text: v }))

const userid = this.props.match.params.user

const Item = List.Item

const users = this.props.chat.users

if(!users[userid]) {

return null

}

const chatid = getChatId(userid, this.props.user.\_id)

const chatmsgs = this.props.chat.chatmsg.filter(v => v.chatid === chatid)

return (

<div id='chat-page'>

<NavBar

mode='dark'

icon={<Icon type="left" />}

onLeftClick={() => {

this.props.history.goBack()

}}

>

{ users[userid].name }

</NavBar>

{ chatmsgs.map(v => {

const avatar = require(`../img/${users[v.from].avatar}.png`)

return v.from === userid?(

<List key={ v.\_id }>

<Item

thumb={ avatar }

>{ v.content }</Item>

</List>

):(

<List key={ v.\_id }>

<Item

// eslint-disable-next-line jsx-a11y/alt-text

extra={ <img src={avatar} /> }

className='chat-me'

>{ v.content }</Item>

</List>

)

}) }

<div className='stick-footer'>

<List>

<InputItem

placeholder='请输入'

value={ this.state.text }

onChange={v=>{

this.setState({text:v})

}}

extra={

<div>

{/\* eslint-disable-next-line jsx-a11y/accessible-emoji\*/}

<span

className='emoji\_span'

onClick = {() => {

this.setState({

showEmoji: !this.state.showEmoji

})

this.fixCarousel()

}}

>😃</span>

<span onClick={() => this.handleSubmit()}>发送</span>

</div>

}

></InputItem>

</List>

{ this.state.showEmoji ?

<Grid

data={ emoji }

columnNum={ 9 }

carouselMaxRow={ 4 }

isCarousel={ true }

onClick={el => {

this.setState({

text: this.state.text+el.text

})

}}

/> :

null}

</div>

</div>

)

}

}

export default Chat

## React AntD 搭建单车后台管理

### yarn 更换为淘宝镜像 ：

yarn config set registry https://registry.npm.taobao.org

### 两种绑定事件的方法

import React from 'react'

export default class Life extends React.Component{

constructor(props) {

super(props)

this.state = {

count: 0

}

}

// 第一种绑事件的方法

handleAdd = () => {

this.setState({

count: this.state.count + 1

})

}

// 第二种绑事件的方法

handleClick() {

this.setState({

count: this.state.count - 1

})

}

render() {

let style = {

padding: '50px'

}

return (

<div style={style}>

<p>react生命周期</p>

<button onClick={this.handleAdd}>点击一下</button>

<button onClick={this.handleClick.bind(this)}>点击一下</button>

<p>{ this.state.count }</p>

</div>

)

}

}