2017年6月16日 星期五

redux使用

1.定义常量

./app/constants/store.js

```
export const STORE_UPDATE = 'STORE_UPDATE'
export const STORE_ADD = 'STORE_ADD'
export const STORE_REMOVE = ‘STORE_REMOVE’
```

./app/constants/userinfo.js

```
export const USERINFO_UPDATE = ‘USERINFO_UPDATE'
```

2.创建 Action

./app/actions/store.js

```
import * as actionTypes from '../constants/store'

export function update(data) {
    return {
        type: actionTypes.STORE_UPDATE,
        data
    }
}

export function add(data) {
    return {
        type: actionTypes.STORE_ADD,
        data
    }
}

export function remove(data) {
    return {
        type: actionTypes.STORE_REMOVE,
        data
    }
}
```

./app/actions/userinfo.js

```
import * as actionTypes from '../constants/userinfo'
```

```
export function update(data) {
    return {
        type: actionTypes.USERINFO_UPDATE,
        data
    }
}
```

3.定义规则

./app/reducers/store.js

```
import * as actionTypes from '../constants/store'

const initialState = []

export default function store (state = initialState, action) {
     switch (action.type) {
          case actionTypes.STORE_UPDATE:
               return action.data
          case actionTypes.STORE_ADD:
               state.unshift(action.data)
               return state
          case actionTypes.STORE_REMOVE:
               return state.filter(item => {
                    if (item.id !== action.data.id) {
                         return item
                    }
               })
          default:
               return state
     }
}
```

./app/reducers/userinfo.js

```
import * as actionTypes from '../constants/userinfo'

const initialState = {}

export default function userinfo (state = initialState, action) {
    switch (action.type) {
        case actionTypes.USERINFO_UPDATE:
            return action.data
        default:
            return state
    }
}
```

4.导出规则 ./app/reducers/index.js

```
import { combineReducers } from 'redux'
import userinfo from './userinfo'
import store from './store'

// Redux 提供了一个combineReducers方法，用于 Reducer 的拆分。
// 只要定义各个子 Reducer 函数，然后用这个方法，将它们合成一个大的
Reducer。
// combineReducers()做的就是产生一个整体的 Reducer 函数。
// 该函数根据 State 的 key 去执行相应的子 Reducer，并将返回结果合并成
一个大的 State 对象。
export default combineReducers({
    userinfo,
    store
})
```

5.创建store

./app/store/configureStore.js

```
import { createStore } from 'redux'
import rootReducer from '../reducers'

export default function configureStore(initialState) {
    const store = createStore(rootReducer, initialState,
        // 触发 redux-devtools
        window.devToolsExtension ? window.devToolsExtension() :
undefined
    )
    return store
}
```

./app/index.js

```
import React from 'react'
import { render } from 'react-dom'
import { Provider } from 'react-redux'
import { hashHistory } from 'react-router'
import configureStore from './store/configureStore'

import './static/css/common.less'
import './static/css/font.css'
```

```
// 创建 Redux 的 store 对象
const store = configureStore()

import RouteMap from './router/routeMap'

render(
    <Provider store={store}>
        <RouteMap history={hashHistory}/>
    </Provider>,
    document.getElementById('root')
)
```

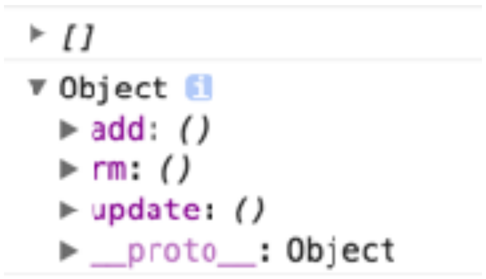
6.连接redux

```
import { connect } from 'react-redux'
import { bindActionCreators } from 'redux'
import * as userInfoActionsFromOtherFile from '../../../actions/
userinfo'
import * as storeActionsFromFile from '../../../actions/store'

function mapStateToProps(state) {
    return {
        userinfo: state.userinfo,
        store: state.store
    }
}

function mapDispatchToProps(dispatch) {
    return {
          userInfoActions:
bindActionCreators(userInfoActionsFromOtherFile, dispatch)
        storeActions:
bindActionCreators(storeActionsFromFile, dispatch)
    }
}
export default connect(
    mapStateToProps,
    mapDispatchToProps
)(Buy)
```

7.测试是否连接成功

```
componentDidMount() {
     this.checkStoreState()
     console.log(this.props.store)
```

```
	console.log(this.props.storeActions)
}
```



```
▸ []
▾ Object
  ▸ add: ()
  ▸ rm: ()
  ▸ update: ()
  ▸ __proto__: Object
```

8.使用

———userinfo 对象 中存储用户名和城市

更新用户名

```
const actions = this.props.userInfoActions
let userinfo = this.props.userinfo
userinfo.username = username
actions.update(userinfo)
```

更新城市

```
this.props.userInfoActions.update({
	cityName: cityName
})
```

———store 数组 中存储不同对象

```
const storeActions = this.props.storeActions
storeActions.add({id:id})
storeActions.remove({id:id})
```