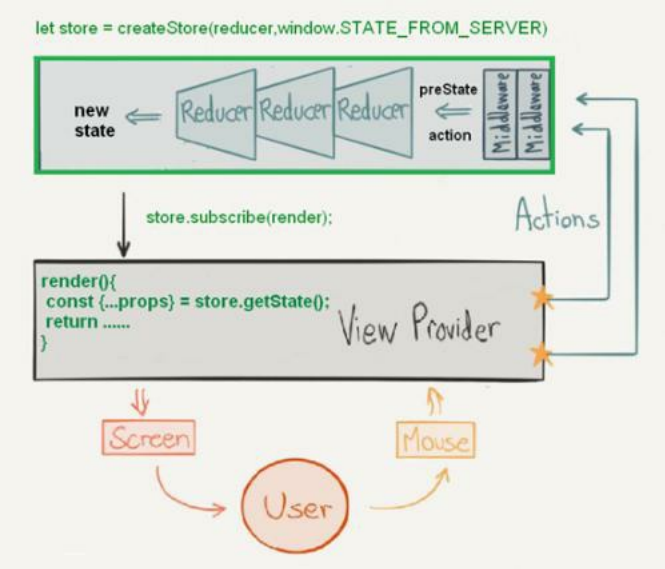
link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=149 sentences=421, words=2244

1. Redux中间件



2. 日志中间件

- 我们改写了，dispatch方法实现了在更改状态时打印前后的状态
- 但是这种方案并不好。所以我们可以采用中间的方式

```
let store = createStore(reducer);
let dispatch = store.dispatch;
store.dispatch = function (action) {
  console.log(store.getState().number);
  dispatch(action);
  console.log(store.getState().number)
};
export default store;
```

2. 实现logger中间件

- 中间件就是一个函数，对store.dispatch方法进行了改造，在发出 Action 和执行 Reducer 这两步之间，添加了其他功能

2.1 store\index.js

src\store\index.js

```
import { createStore,applyMiddleware } from '../redux';
import reducer from './reducers';
let logger = store => dispatch => action=>{
  console.log(store.getState().number);
  dispatch(action);
  console.log(store.getState().number)
};
export default applyMiddleware(logger)(createStore)(reducer);
```

2.2 applyMiddleware.js

src\redux\applyMiddleware.js

- applyMiddleware (https://github.com/reduxjs/redux/blob/master/src/applyMiddleware.js)

```
import compose from './compose'
export default function applyMiddleware(...middlewares) {
 return createStore=>(...args)=>{
     const store = createStore(...args);
     let dispatch = ()=>{
         throw new Error('不允许派发正在构建中的中间件!');
     }
     const middlewareAPI= {
         getState:store.getState,
         dispatch:(...args)=>dispatch(...args)
     }
     const chain = middlewares.map(middleware=>middleware(middlewareAPI));
     dispatch = compose(...chain)(store.dispatch);
     return {
         ...store,
         dispatch
     }
 };
}
```

2.3 compose.js

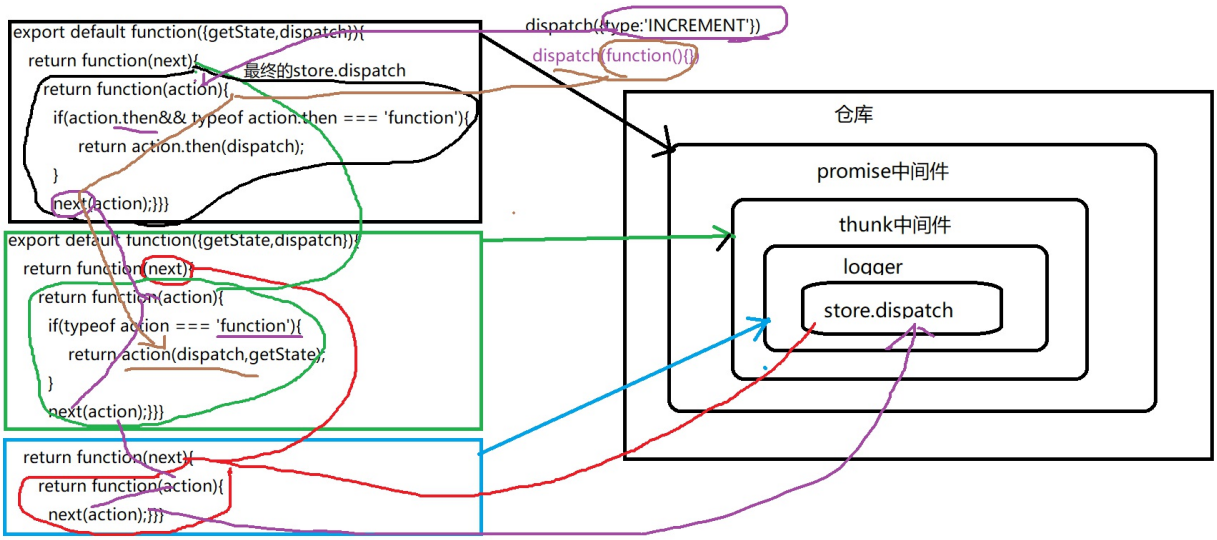
src\redux\compose.js

- compose (https://github.com/reduxjs/redux/blob/master/src/compose.js)

```
function add1(str){
    return '1'+str;
}
function add2(str){
    return '2'+str;
}
function add3(str){
    return '3'+str;
}

function compose(...funcs){
    return funcs.reduce((a,b)=>(...args)=>a(b(...args)));
}

let result = compose(add3,add2,add1)('zfpx');
console.log(result);
```

```
export default function compose(...funcs) {
    if (funcs.length === 0) {
      return arg => arg
    }

    if (funcs.length === 1) {
      return funcs[0]
    }

    return funcs.reduce((a, b) => (...args) => a(b(...args)))
  }
```

3. 级联中间件



3.1 Counter.js

src\components\Counter.js

```
import React, { Component } from 'react';
import actions from '../store/actions/counter';
import {connect} from '../react-redux'
class Counter extends Component {
    render() {
        return (
            <div>
                <p>{this.props.number}p>
                <button onClick={this.props.increment}>+button>
                <button onClick={this.props.incrementAsync}>异步+1button>
                <button onClick={this.props.incrementPromise}>promise异步+1button>
            div>
        )
    }
}

let mapStateToProps = state=>state;
export default connect(
    mapStateToProps,
    actions
)(Counter)
```

3.2 store\index.js

src\store\index.js

```
import { createStore,applyMiddleware } from '../redux';
import reducer from './reducers';
import logger from '../redux-logger';
import thunk from '../redux-thunk';
import promise from '../redux-promise';
export default applyMiddleware(thunk,promise,logger)(createStore)(reducer);
```

3.3 reducers\index.js

src\store\reducers\index.js

```
import counter from './counter';
export default counter;
```

3.4 actions\counter.js

src\store\actions\counter.js

```
import * as types from '../action-types';
export default {
    increment(){
        return {type:types.INCREMENT};
    },
    decrement(){
        return {type:types.DECREMENT};
    },
    incrementAsync(){
        return function(dispatch){
            setTimeout(()=>{
                dispatch({type:types.INCREMENT});
            },1000);
        }
    },
    incrementPromise(){
        return {
            type:types.INCREMENT,
            payload:new Promise((resolve,reject)=>{
                let result = Math.random();
                if(result>.5){
                    resolve(result);
                }else{
                    reject(result);
                }
            },1000)
        }
    }
}
```

3.5 redux-logger.js

src\redux-logger.js redux-logger.js (https://github.com/LogRocket/redux-logger/blob/master/src/index.js)

```
export default  store => dispatch => action=>{
    console.log(store.getState().number);
    dispatch(action);
    console.log(store.getState().number)
};
```

3.6 redux-thunk.js

src\redux-thunk.js redux-thunk (https://github.com/reduxjs/redux-thunk/blob/master/src/index.js)

```
function createThunkMiddleware(extraArgument) {
    return ({dispatch,getState}) => next => action => {
        if (typeof action == 'function') {
            return action(dispatch, getState, extraArgument);
        }
        return next(action);
    }
}
const thunk = createThunkMiddleware();
thunk.withExtraArgument = createThunkMiddleware;
export default thunk;
```

3.7 redux-promise.js

src\redux-promise.js redux-promise (https://github.com/redux-utilities/redux-promise/blob/master/src/index.js)

```
function isPromise(obj) {
    return !!obj && (typeof obj === 'object' || typeof obj === 'function') && typeof obj.then === 'function';
}
export default function promiseMiddleware({ dispatch }) {
    return next => action => {
        return isPromise(action.payload)
            ? action.payload
                .then(result => dispatch({ ...action, payload: result }))
                .catch(error => {
                    dispatch({ ...action, payload: error, error: true });
                    return Promise.reject(error);
                })
            : next(action);
    };
}
```

4. redux-persist

4.1 src\index.js

```
import React,{Component} from 'react';
import ReactDOM from 'react-dom';
import Counter from './components/Counter';
import {Provider} from './react-redux';
+import {store,persistor} from './store';
+import { PersistGate } from './redux-persist/integration/react'

ReactDOM.render(
+

+
,document.getElementById('root'));
```

4.2 store\index.js

src\store\index.js

```
import {createStore} from '../redux';
import reducer from './reducers';
import {applyMiddleware} from '../redux';
import logger from '../redux-logger';
import thunk from '../redux-thunk';
import promise from '../redux-promise';
+import { persistStore, persistReducer } from '../redux-persist'
+import storage from '../redux-persist/lib/storage'
+const persistConfig = {
+    key: 'root',
+    storage,
+}
+const persistedReducer = persistReducer(persistConfig, reducer);
+const store = applyMiddleware(thunk,promise,logger)(createStore)(persistedReducer);
+let persistor = persistStore(store)
+export {persistor,store};
```

4.3 redux-persist\index.js

src\redux-persist\index.js

```
import persistReducer from './persistReducer';
import persistStore from './persistStore';

export {
    persistReducer,
    persistStore
}
```

4.4 persistReducer.js

src\redux-persist\persistReducer.js

```
export default function (persistConfig, reducer) {
    let isInited = false;
    return (state, action) => {
        switch (action.type) {
            case 'PERSIST_INIT':
                isInited = true;
                let value = persistConfig.storage.getItem('persist:'+ persistConfig.key);
                state = value ? JSON.parse(value) : undefined;
                return reducer(state, action);
            default:
                if (isInited) {
                    state = reducer(state, action);
                    persistConfig.storage.setItem('persist:'+ persistConfig.key, JSON.stringify(state));
                    return state;
                }
                return reducer(state, action);
        }

    }
}
```

4.5 persistStore.js

src\redux-persist\persistStore.js

```
export default function (store) {
    let persistor = {
        ...store,
        initState() {
            persistor.dispatch({
                type: 'PERSIST_INIT',
            })
        }
    };
    return persistor;
}
```

4.6 storage.js

src\redux-persist\lib\storage.js

```
let storage = {
    setItem(key,val) {
        localStorage.setItem(key,val);
    },
    getItem(key) {
        return localStorage.getItem(key);
    }
}
export default storage;
```

4.7 react.js

src\redux-persist\integration\react.js

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

class PersistGate extends Component {
    componentDidMount() {
        this.props.persistor.initState();
    }
    render() {
        return this.props.children;
    }
}

export {PersistGate}
```

5. redux-actions

- redux-actions是一个实用的库，让编写redux状态管理变得简单起来。redux-action产生的动作是FSA (https://github.com/redux-utilities/flux-standard-action)标准的

5.1 单个action

5.1.1 actions\counter.js

src\store\actions\counter.js

```
import * as types from '../action-types';

function createAction(type,payloadCreator){
  return function actionCreator(...args){
      return {type,payload:payloadCreator(...args)};
  }
}
const add = createAction(types.ADD,(payload)=>payload*2);
const minus = createAction(types.MINUS,(payload)=>payload*2);
export default {
    add,
    minus
}
```

5.1.2 reducers\counter.js

src\store\reducers\counter.js

```
import *  as types from '../action-types';

import actions from '../actions/counter';
function handleAction(type,reducer ,defaultState){
    return function(state=defaultState,action){
        if(action.type === type){
            return reducer(state,action);
        }
        return state;
    }
}
const initialState = {number:0};
const reducer = handleAction(types.ADD,(state,action)=>{
  return {
      ...state,number:state.number+action.payload
  }
},initialState);
export default reducer;
```

5.2 多个action

5.2.1 actions\counter.js

actions\counter.js

```
import * as types from '../action-types';

export default createActions({
    [types.ADD]:(payload)=>payload*2,
    [types.MINUS]:(payload)=>payload*2
});
function createActions(actions){
    let newActions = {};
    for(let type in actions){
       newActions[type]= function(...args){
           return {type,payload:actions[type](...args)}
       }
    }
    return newActions;
}
```

5.2.2 reducers\counter.js

reducers\counter.js

```
import *  as types from '../action-types';

import actions from '../actions/counter';
const initialState = {number:0};
function handleActions(reducers,initialState){
   return function(state=initialState,action){
        let types = Object.keys(reducers);
        for(let i=0;ilet type = types[i];
            if(type === action.type){
                return reducers[type](state,action);
            }
        }
        return state;
    }
}
export default handleActions({
    [types.ADD]:(state,action)=>{
        return {
            ...state,number:state.number+action.payload
        }
    },
    [types.MINUS]:(state,action)=>{
        return {
            ...state,number:state.number-action.payload
        }
    }
},initialState);
```

6. reselect

- 使用Redux管理React应用状态时, mapStateToProps方法作为从 Redux Store上获取数据过程中的重要一环，它一定不能有性能缺陷，它本身是一个函数，通过计算返回一个对象，这个计算过程通常是基于Redux Store状态树进行的，而很明显的Redux状态树越复杂，这个计算过程可能就越耗时，我们应该要能够尽可能减少这个计算过程，比如重复在相同状态下渲染组件，多次的计算过程显然是多余的，我们是否可以缓存该结果呢？这个问题的解决者就是 reselect，它可以提高应用获取数据的性能
- reselect的原理是，只要相关状态不变，即直接使用上一次的缓存结果

6.1 基本用法

- reselect通过创建选择器（selectors），该函数接受一个state参数，然后返回我们需要在mapStateToProps方法内返回对象的某一个数据项，一个选择器的处理可以分为两个步骤
    - 接受state参数，根据我们提供的映射函数数组分别进行计算，如果返回结果和上次第一步的计算结果一致，说明命中缓存，则不进行第二步计算，直接返回上次第二步的计算结果，否则继续第二步计算。第一步的结果比较，通常仅仅是===相等性检查，性能是足够的
    - 根据第一步返回的结果，计算并返回最终结果
- 需要注意的是，传入createSelector的映射函数返回的状态应该是不可变的，因为默认缓存命中检测函数使用引用检查，如果使用JavaScript对象，仅改变该对象的某一属性，引用检测是无法检测到属性变更的，这将导致组件无法响应更新

```

function createSelector(selector,reducer){
  let lastState;
  let value;
  return function(state){
    let newState = selector(state);
    if(lastState !== newState){
      value = reducer(newState);
      lastState = newState;
    }
    return value;
  }
}
const counterSelector = state => state.counter;

const getCounterSelector = createSelector(
  counterSelector,
  counter => {
    console.log('重新计算number')
    return counter.number;
  }
)

let initialState = {
  counter: {
    number:0
  }
}

console.log(getCounterSelector(initialState));
console.log(getCounterSelector(initialState));
```

```
+console.log(getCounterSelector(initialState));
+initialState.counter.number+=1;
+console.log(getCounterSelector(initialState));
```

```
+ console.log(getCounterSelector(initialState));
+ initialState.counter={number:1}
+ console.log(getCounterSelector(initialState));
```

```
+const immutable = require("immutable");
+let initialState = immutable.Map({counter: {number:0}})
+console.log(getCounterSelector(initialState.toJS()));
+initialState = initialState.setIn(['counter','number'],1);
+console.log(getCounterSelector(initialState.toJS()));
```

6.2 案例

6.2.1 src\index.js

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import Counter1 from './components/Counter1';
import Counter2 from './components/Counter2';
import {Provider} from 'react-redux';
import store from './store';
ReactDOM.render(

  <></>
,document.getElementById('root'));
```

6.2.2 Counter1.js

src\components\Counter1.js

```
import React from 'react';
import {connect} from 'react-redux';
import { createSelector } from 'reselect'
import actions from '../store/actions/counter1';
class Counter extends React.Component{
    render(){
        return (
            <div>
                <p>{this.props.number}p>
                <button onClick={this.props.add}>+button>
                <button onClick={this.props.minus}>-button>
            div>
        )
    }
}

const getCounterSelector = state => state.get('counter1');

const counterSelector = createSelector(
  getCounterSelector,
  counter1 =>{
      console.log('重新计算counter1',counter1);
      return  counter1;
  }
)
export default connect(
    state=>counterSelector(state),
    actions
)(Counter)
```

6.2.3 Counter2.js

src\components\Counter2.js

```
import React from 'react';
import {connect} from 'react-redux';
import { createSelector } from 'reselect'
import actions from '../store/actions/counter2';
class Counter extends React.Component{
    render(){
        return (
            <div>
                <p>{this.props.number}p>
                <button onClick={()=>this.props.add(5)}>+button>
                <button onClick={()=>this.props.minus(5)}>-button>
            div>
        )
    }
}

const getCounterSelector = state => state.get('counter2');

const counterSelector = createSelector(
  getCounterSelector,
  counter2 =>{
      console.log('重新计算counter2',counter2)
      return  counter2;
  }
)
export default connect(
    state=>counterSelector(state),
    actions
)(Counter)
```

6.2.4 src\store\index.js

src\store\index.js

```
import {createStore,applyMiddleware} from 'redux';
import reducer from './reducers';
import logger from 'redux-logger';
import thunk from 'redux-thunk';
import promise from 'redux-promise';
let store = applyMiddleware(promise,thunk,logger)(createStore)(reducer);
export default store;
```

6.2.5 reducers\index.js

src\store\reducers\index.js

```
import {combineReducers} from 'redux-immutable';
import counter1 from './counter1';
import counter2 from './counter2';
export default combineReducers({
    counter1,
    counter2
});
```

6.2.6 reducers\counter1.js

src\store\reducers\counter1.js

```
import * as types from '../action-types';
import actions from '../actions/counter';
const initialState = {number:0};

export default function(state=initialState,action){
  switch(action.type){
      case types.ADD1:
        return {number:state.number+1};
      case types.MINUS1:
        return {number:state.number-1};
      default:
        return state;
  }
};
```

6.2.7 reducers\counter2.js

src\store\reducers\counter2.js

```
import *  as types from '../action-types';
import actions from '../actions/counter';
const initialState = {number:0};

export default function(state=initialState,action){
  switch(action.type){
      case types.ADD2:
        return {number:state.number+1};
      case types.MINUS2:
        return {number:state.number-1};
      default:
        return state;
  }
};
```

6.2.8 counter1.js

src\store\actions\counter1.js

```
import * as types from '../action-types';
export default {
    add(){
        return {type:types.ADD1}
    },
    minus(){
         return {type:types.MINUS1}
    }
}
```

6.2.9 actions\counter2.js

src\store\actions\counter2.js

```
import * as types from '../action-types';
export default {
    add(){
        return {type:types.ADD2}
    },
    minus(){
         return {type:types.MINUS2}
    }
}
```

7.undo

- simple undo/redo functionality for redux state containers
- redux-undo (https://cnpmjs.org/package/redux-undo)
- 官网 (http://redux-undo.js.org/)

```
import React, { Component, lazy, Suspense } from "react";
import ReactDOM from "react-dom";
import PropTypes from 'prop-types';
import {createStore} from 'redux';

const INCREMENT='INCREMENT';
const DECREMENT = 'DECREMENT';
const UNDO_COUNTER = 'UNDO_COUNTER';
const REDO_COUNTER = 'REDO_COUNTER';
function reducer(state=0,action){
    switch(action.type){
        case INCREMENT:
            return state+1;
        case DECREMENT:
            return state-1;
        default:
            return state;
    }
}
function undoable(reducer,config){
    const {undoType="@@redux-unto/UNDO",redoType="@@redux-unto/REDO"}= config;
    const initialState = {
        past:[],
        futer:[],
        present:reducer(undefined,{})
    }
    return function(state=initialState,action){
        const {past,present,future} = state;
        switch(action.type){
            case undoType:
                const previous = past[past.length-1];
                const newPast = past.slice(0,past.length-1);
                return {
                    past:newPast,
                    present:previous,
                    future:[present,...future]
                }
            break;
            case redoType:
                const next = future[0];
                const newFuture = future.slice(1);
                return {
                    past:[...past,present],
                    present:next,
                    future:newFuture
                }
            break;
            default:
              const newPresent = reducer(present,action);
              return {
                  past:[...past,present],
                  present:newPresent,
                  future:[]
              }
        }
    }
}
let undoableReducer  = undoable(reducer,{
    undoType:UNDO_COUNTER,
    redoType:REDO_COUNTER
});
let store=createStore(undoableReducer);
class Counter extends Component{
    constructor(props) {
        super(props);
        this.state={value:store.getState()};
    }
    componentDidMount() {
        this.unsubscribe=store.subscribe(()=>this.setState({value:store.getState()}));
    }
    componentWillUnmount() {
        this.unsubscribe();
    }
    undo(){
      store.dispatch({type:UNDO_COUNTER});
    }
    redo(){
     store.dispatch({type:REDO_COUNTER});
    }
    add = ()=>{
        store.dispatch({type:INCREMENT});
    }
    render() {
        const {value,onInrement,onDecrement}=this.props;

        console.log(JSON.stringify(this.state.value));
        return (
            <div>
                <p>{this.state.value.present}p>
                <button onClick={this.add}>+button>
                <button onClick={()=>store.dispatch({type:DECREMENT})}>-button>
                <button onClick={this.undo}>undobutton>
                <button onClick={this.redo}>redobutton>
            div>
        )
    }
}
ReactDOM.render(<Counter/>, document.querySelector("#root"));
```

附录

- redux (https://github.com/reduxjs/redux)
- redux-logger (https://github.com/LogRocket/redux-logger)

- redux-promise (https://github.com/redux-utilities/redux-promise)
- redux-thunk (https://github.com/reduxjs/redux-thunk)
- redux-persist (https://github.com/rt2zz/redux-persist)
- redux-immutable (https://www.npmjs.com/package/redux-immutable)
- immutable-js (https://immutable-js.github.io/immutable-js)
- reselect (https://github.com/reduxjs/reselect)