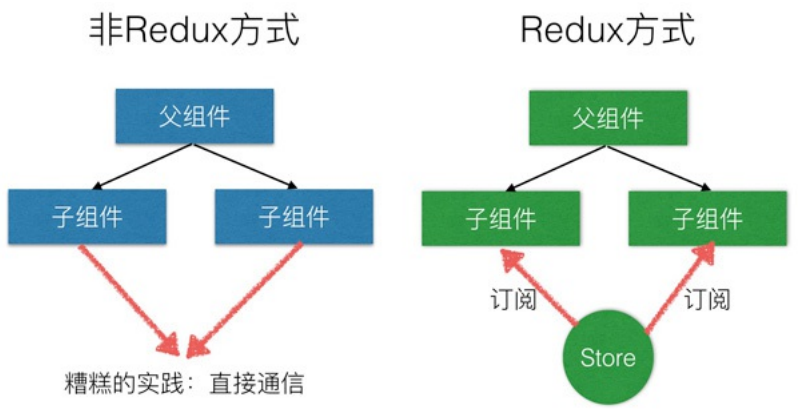
link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=139 sentences=360, words=2662

1. Redux应用场景

- 随着 JavaScript 单页应用开发日趋复杂,管理不断变化的 state 非常困难
- Redux的出现就是为了解决state里的数据问题
- 在React中，数据在组件中是单向流动的
- 数据从一个方向父组件流向子组件(通过props)，由于这个特征，两个非父子关系的组件（或者称作兄弟组件）之间的通信就比较麻烦



2. Redux设计思想

- Redux是将整个应用状态存储到到一个地方，称为store
- 里面保存一棵状态树state tree
- 组件可以派发dispatch行为action给store,而不是直接通知其它组件
- 其它组件可以通过订阅store中的状态(state)来刷新自己的视图

3. Redux三大原则

- 整个应用的 state 被储存在一棵 object tree 中，并且这个 object tree 只存在于唯一一个 store 中
- State 是只读的，惟一改变 state 的方法就是触发 action，action 是一个用于描述已发生事件的普通对象 使用纯函数来执行修改，为了描述action如何改变state tree ，你需要编写 reducers
- 单一数据源的设计让React的组件之间的通信更加方便，同时也便于状态的统一管理

4. 原生计数器

redux (https://github.com/reduxjs/redux)

```
create-react-app zhufeng_redux_prepare --typescript
cd zhufeng_redux_prepare
cnpm install redux @types/redux -S
yarn start
```

4.1 public\index.html

public\index.html

```
<html lang="en">

<head>
  <meta charset="utf-8" />
  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <title>React Apptitle>
head>

<body>
  <div id="root">div>
  <div id="counter">
    <p id="counter-value">0p>
    <button id="increment-btn">+button>
    <button id="decrement-btn">-button>
  div>
body>

html>
```

4.2 src\index.tsx

src\index.tsx

```
import { createStore, Action, Reducer, Store } from './redux';
let counterValue: HTMLElement | null = document.getElementById('counter-value');
let incrementBtn: HTMLElement | null = document.getElementById('increment-btn');
let decrementBtn: HTMLElement | null = document.getElementById('decrement-btn');

const INCREMENT: string = 'INCREMENT';
const DECREMENT: string = 'DECREMENT';
interface State {
    number: number
}
let initState: State = { number: 0 };
type CounterAction = Action;
const reducer: Reducer = (state: State = initState, action: CounterAction) => {
    switch (action.type) {
        case INCREMENT:
            return { number: state.number + 1 };
        case DECREMENT:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
let store: Store = createStore(reducer);
function render(): void {
    counterValue!.innerHTML = store.getState().number + '';
}
store.subscribe(render);
render();
incrementBtn!.addEventListener('click', function () {
    store.dispatch({ type: INCREMENT });
});
decrementBtn!.addEventListener('click', function () {
    store.dispatch({ type: DECREMENT });
});
```

4.3 redux\index.tsx

src\redux\index.tsx

```
import createStore from './createStore'
export {
    createStore
}
export * from './types';
```

4.4 redux\types.tsx

src\redux\types.tsx

```
export interface Action {
    type: T
}

export interface AnyAction extends Action {
    [extraProps: string]: any
}

export type Reducer = (
    state: S | undefined,
    action: A
) => S

export interface Dispatch {
    (action: A): A
}
export interface Unsubscribe {
    (): void
}
export interface Store {
    dispatch: Dispatch;
    getState(): S;
    subscribe(listener: () => void): Unsubscribe;
}

export interface StoreCreator {
    ,Ext, StateExt>(
        reducer: Reducer,
        preloadedState?: S
    ): Store
}
```

4.5 redux\createStore.tsx

src\redux\createStore.tsx

```
import { Reducer, StoreCreator, Action, Dispatch, Store } from './';
const createStore: StoreCreator = (reducer: Reducer, preloadedState?: S): Store => {
    let currentState: S = preloadedState as S;
    let currentListeners: (() => void)[] = [];

    function getState(): S {
        return currentState;
    }

    function subscribe(listener: () => void) {
        currentListeners.push(listener);
        return function unsubscribe() {
            const index: number = currentListeners.indexOf(listener);
            currentListeners.splice(index, 1);
        };
    }
    const dispatch: Dispatch = (action: T): T => {
        currentState = reducer(currentState, action);
        for (let i = 0; i < currentListeners.length; i++) {
            const listener = currentListeners[i];
            listener();
        }
        return action;
    }
    dispatch({ type: '@@redux/INIT' } as A);
    const store: Store = {
        dispatch,
        subscribe,
        getState
    };
    return store;
}
export default createStore;
```

5. React计数器

5.1 src\index.tsx

src\index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom';
import Counter1 from './components/Counter1';

ReactDOM.render(<Counter1 />, document.getElementById('root'));
```

5.2 Counter1.tsx

src\components\Counter1.tsx

```
import React, { Component } from 'react';
import { createStore, Action, Reducer, Store, Unsubscribe } from '../redux';
interface State {
    number: number
}

type CounterAction = Action;
const INCREMENT: string = 'INCREMENT';
const DECREMENT: string = 'DECREMENT';
const reducer: Reducer = (state: State = initState, action: CounterAction) => {
    switch (action.type) {
        case INCREMENT:
            return { number: state.number + 1 };
        case DECREMENT:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
let initState: State = { number: 0 };
const store: Store = createStore(reducer, initState);
interface Props { }
interface State { number: number }
export default class Counter extends Component<Props, State> {
    unsubscribe: Unsubscribe;
    constructor(props: Props) {
        super(props);
        this.state = { number: 0 };
    }
    componentDidMount() {
        this.unsubscribe = store.subscribe(() => this.setState({ number: store.getState().number }));
    }
    componentWillUnmount() {
        this.unsubscribe();
    }
    render() {
        return (
            <div>
                <p>{this.state.number}p>
                <button onClick={() => store.dispatch({ type: 'INCREMENT' })}>+button>
                <button onClick={() => store.dispatch({ type: 'DECREMENT' })}>-button>
                <button onClick={
                    () => {
                        setTimeout(() => {
                            store.dispatch({ type: 'INCREMENT' })
                        }, 1000);
                    }
                }>1秒后加1button>
            div>
        )
    }
}
```

6.bindActionCreators

6.1 Counter1.tsx

src\components\Counter1.tsx

```
import React, { Component } from 'react';
+import {
+    createStore, bindActionCreators, Action,
+    Reducer, Store, Unsubscribe
+} from '../redux';
interface State {
    number: number
}

type CounterAction = Action;
const INCREMENT: string = 'INCREMENT';
const DECREMENT: string = 'DECREMENT';
const reducer: Reducer = (state: State = initState, action: CounterAction) => {
    switch (action.type) {
        case INCREMENT:
            return { number: state.number + 1 };
        case DECREMENT:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
let initState: State = { number: 0 };
+const store: Store = createStore(reducer, initState);
+function increment() {
+    return { type: 'INCREMENT' };
+}
+function decrement() {
+    return { type: 'DECREMENT' };
+}
+const actions = { increment, decrement };
+const boundActions = bindActionCreators(actions, store.dispatch);
interface Props { }
interface State { number: number }
export default class Counter extends Component {
    unsubscribe: Unsubscribe;
    constructor(props: Props) {
        super(props);
        this.state = { number: 0 };
    }
    componentDidMount() {
        this.unsubscribe = store.subscribe(() => this.setState({ number: store.getState().number }));
    }
    componentWillUnmount() {
        this.unsubscribe();
    }
    render() {
        return (

                {this.state.number}
+                +
+                -
                 {
                        setTimeout(() => {
+                            boundActions.increment();
                        }, 1000);
                    }
                }>1秒后加1

        )
    }
}
```

6.2 bindActionCreators.tsx

src\redux\bindActionCreators.tsx

```
import { Dispatch, AnyAction } from './';
export interface ActionCreator {
    (...args: any[]): A
}
export interface ActionCreatorsMapObject {
    [key: string]: ActionCreator
}
export default function bindActionCreators<A extends AnyAction, M extends ActionCreatorsMapObject<A>>(
    actionCreators: M,
    dispatch: Dispatch
): M
export default function bindActionCreators<A extends AnyAction, M extends ActionCreatorsMapObject<A>>(
    actionCreators: M,
    dispatch: Dispatch
): M {
    const boundActionCreators: ActionCreatorsMapObject = {};
    for (const key in actionCreators) {
        const actionCreator = actionCreators[key]
        if (typeof actionCreator === 'function') {
            boundActionCreators[key] = bindActionCreator(actionCreator, dispatch)
        }
    }
    return boundActionCreators as M;
}
function bindActionCreator<A extends AnyAction = AnyAction>(
    actionCreator: ActionCreator,
    dispatch: Dispatch
) {
    return function (this: any, ...args: any[]) {
        return dispatch(actionCreator.apply(this, args));
    }
}
```

7.combineReducers

7.1 src\index.tsx

src\index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom';
import Counter1 from './components/Counter1';
import Counter2 from './components/Counter2';
+ReactDOM.render(<></>, document.getElementById('root'));
```

7.2 actions\counter1.tsx

src\store\actions\counter1.tsx

```
import * as types from '../action-types';
import { AnyAction } from '../../redux';
const actions = {
    increment1(): AnyAction {
        return { type: types.INCREMENT1 };
    },
    decrement1(): AnyAction {
        return { type: types.DECREMENT1 };
    }
}
export default actions;
```

7.3 reducers\counter1.tsx

src\store\reducers\counter1.tsx

```
import * as types from '../action-types';
import { AnyAction } from '../../redux';
export interface Counter1State {
    number: number
}
let initialState: Counter1State = { number: 0 }

export default function (state: Counter1State = initialState, action: AnyAction): Counter1State {
    switch (action.type) {
        case types.INCREMENT1:
            return { number: state.number + 1 };
        case types.DECREMENT1:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
```

7.4 actions\counter2.tsx

src\store\actions\counter2.tsx

```
import * as types from '../action-types';
import { AnyAction } from '../../redux';
export default {
    increment2(): AnyAction {
        return { type: types.INCREMENT2 };
    },
    decrement2(): AnyAction {
        return { type: types.DECREMENT2 };
    }
}
```

7.5 reducers\counter2.tsx

src\store\reducers\counter2.tsx

```
import * as types from '../action-types';
import { AnyAction } from '../../redux';
export interface Counter2State {
    number: number
}
let initialState: Counter2State = { number: 0 };

export default function (state: Counter2State = initialState, action: AnyAction): Counter2State {
    switch (action.type) {
        case types.INCREMENT2:
            return { number: state.number + 1 };
        case types.DECREMENT2:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
```

7.6 reducers\index.tsx

src\store\reducers\index.tsx

```
import { combineReducers, ReducersMapObject, Reducer, AnyAction } from 'redux';
import counter1, { Counter1State } from './counter1';
import counter2, { Counter2State } from './counter2';
interface Reducers {
    counter1: Counter1State;
    counter2: Counter2State;
}
let reducers: ReducersMapObject = {
    counter1,
    counter2
};
export type RootState = {
    [key in keyof typeof reducers]: ReturnType<typeof reducers[key]>
}
let rootReducer: Reducer = combineReducers(reducers);
export default rootReducer;
```

7.7 combineReducers.tsx

src\redux\combineReducers.tsx

```
import { Action, Reducer } from 'redux';
export type ReducersMapObject = {
    [K in keyof S]: Reducer
}
export default function combineReducers<S, A extends Action = Action>(
    reducers: ReducersMapObject
): Reducer<S>
export default function combineReducers<S, A extends Action = Action>(
    reducers: ReducersMapObject) {
    return function (state: S = {} as S, action: A) {
        const nextState: S = {} as S;
        let key: keyof S;
        for (key in reducers) {
            const reducer = reducers[key];
            const previousStateForKey = state[key];
            const nextStateForKey = reducer(previousStateForKey, action);
            nextState[key] = nextStateForKey;
        }
        return nextState;
    }
}
```

7.8 store\index.tsx

src\store\index.tsx

```
+import { createStore, Store, AnyAction } from '../redux';
+import reducer from './reducers';
+import { RootState } from './reducers';
+const store: Store = createStore(reducer, { counter1: { number: 0 }, counter2: { number: 0 } });
+export default store;
```

7.9 action-types.tsx

src\store\action-types.tsx

```
export const INCREMENT1 = 'INCREMENT1';
export const DECREMENT1 = 'DECREMENT1';

export const INCREMENT2 = 'INCREMENT2';
export const DECREMENT2 = 'DECREMENT2';
```

7.10 src\redux\index.tsx

src\redux\index.tsx

```
import createStore from './createStore';
import bindActionCreators from './bindActionCreators';
+import combineReducers from './combineReducers'
export {
    createStore,
    bindActionCreators,
+    combineReducers
}
export * from './types';
+export * from './bindActionCreators';
+export * from './combineReducers';
```

7.11 Counter1.tsx

src\components\Counter1.tsx

```
import React, { Component } from 'react';
import actions from '../store/actions/counter1';
import {
    bindActionCreators, Unsubscribe, ActionCreatorsMapObject, AnyAction
} from '../redux';
import store from '../store';

const boundActions = bindActionCreators>(actions, store.dispatch);
interface Props { }
interface State { number: number }
export default class extends Component<Props, State> {
    unsubscribe: Unsubscribe;
    constructor(props: Props) {
        super(props);
        this.state = { number: 0 };
    }
    componentDidMount() {
        this.unsubscribe = store.subscribe(() => this.setState({ number: store.getState().counter1.number }));
    }
    componentWillUnmount() {
        this.unsubscribe();
    }
    render() {
        return (
            <div>
                <p>{this.state.number}p>
                <button onClick={boundActions.increment1}>+button>
                <button onClick={boundActions.decrement1}>-button>
            div>
        )
    }
}
```

7.12 Counter2.tsx

src\components\Counter2.tsx

```
import React, { Component } from 'react';
import actions from '../store/actions/counter2';
import {
    bindActionCreators, Unsubscribe, ActionCreatorsMapObject, AnyAction
} from '../redux';
import store from '../store';

const boundActions = bindActionCreators>(actions, store.dispatch);
interface Props { }
interface State { number: number }
export default class extends Component<Props, State> {
    unsubscribe: Unsubscribe;
    constructor(props: Props) {
        super(props);
        this.state = { number: 0 };
    }
    componentDidMount() {
        this.unsubscribe = store.subscribe(() => this.setState({ number: store.getState().counter2.number }));
    }
    componentWillUnmount() {
        this.unsubscribe();
    }
    render() {
        return (
            <div>
                <p>{this.state.number}p>
                <button onClick={boundActions.increment2}>+button>
                <button onClick={boundActions.decrement2}>-button>
            div>
        )
    }
}
```

8. react-redux

8.1 src\index.tsx

src\index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom';
import Counter1 from './components/Counter1';
import Counter2 from './components/Counter2';
+import store from './store';
+import { Provider } from './react-redux';
+ReactDOM.render(
+
+
+
+
+    , document.getElementById('root'));
```

8.2 Counter1.tsx

src\components\Counter1.tsx

```
import React, { Component } from 'react';
import actions from '../store/actions/counter1';
+import { RootState } from '../store/reducers';
+import { Counter1State } from '../store/reducers/counter1';
+import { connect } from '../react-redux';
+type Props = Counter1State & typeof actions;
+class Counter1 extends Component {
+    render() {
+        let { number, increment1, decrement1 } = this.props;
+        return (
+
+                {number}
+                +
+                -
+
+        )
+    }
+}
+let mapStateToProps = (state: RootState): Counter1State => state.counter1;
+export default connect(
+    mapStateToProps,
+    actions
+)(Counter1)
```

8.3 Counter2.tsx

src\components\Counter2.tsx

```
import React, { Component } from 'react';
import actions from '../store/actions/counter2';
+import { RootState } from '../store/reducers';
+import { Counter2State } from '../store/reducers/counter2';
+import { connect } from '../react-redux';
+type Props = Counter2State & typeof actions;
+class Counter2 extends Component {
+    render() {
+        let { number, increment2, decrement2 } = this.props;
+        return (
+
+                {number}
+                +
+                -
+
+        )
+    }
+}
+let mapStateToProps = (state: RootState): Counter2State => state.counter2;
+export default connect(
+    mapStateToProps,
+    actions
+)(Counter2)
```

8.4 reducers\counter1.tsx

src\store\reducers\counter1.tsx

```
import * as types from '../action-types';
import { AnyAction } from '../../redux';
export interface Counter1State {
    number: number
}
let initialState: Counter1State = { number: 0 }

export default function (state: Counter1State = initialState, action: AnyAction): Counter1State {
    switch (action.type) {
        case types.INCREMENT1:
            return { number: state.number + 1 };
        case types.DECREMENT1:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
```

8.5 reducers\counter2.tsx

src\store\reducers\counter2.tsx

```
import * as types from '../action-types';
import { AnyAction } from '../../redux';
export interface Counter2State {
    number: number
}
let initialState: Counter2State = { number: 0 };

export default function (state: Counter2State = initialState, action: AnyAction): Counter2State {
    switch (action.type) {
        case types.INCREMENT2:
            return { number: state.number + 1 };
        case types.DECREMENT2:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
```

8.6 reducers\index.tsx

src\store\reducers\index.tsx

```ts
import { combineReducers, ReducersMapObject, Reducer, AnyAction } from '../../redux';
import counter1, { Counter1State } from './counter1';
import counter2, { Counter2State } from './counter2';
interface Reducers {
    counter1: Counter1State;
    counter2: Counter2State;
}
let reducers: ReducersMapObject = {
    counter1,
    counter2
};
export type RootState = {
    [key in keyof typeof reducers]: ReturnType<typeof reducers[key]>
}
let rootReducer: Reducer = combineReducers(reducers);
export default rootReducer;
```

8.7 react-redux\index.tsx

src\react-redux\index.tsx

```ts
import Provider from './Provider';
import connect from './connect';
export {
    Provider,
    connect
}
```

8.8 react-redux\context.tsx

src\react-redux\context.tsx

```ts
import React from 'react';
import { ContextValue } from './types';
export const ReactReduxContext = React.createContextnull>(null)
export default ReactReduxContext;
```

8.9 react-redux\types.tsx

src\react-redux\types.tsx

```ts
import { Store } from '../redux';

export interface ContextValue {
    store: Store
}
```

8.10 Provider.tsx

src\react-redux\Provider.tsx

```ts
import React, { Component } from 'react'
import { Store } from '../redux';
import ReactReduxContext from './context';
interface Props {
  store: Store
}
export default class Provider extends Component<Props> {
  render() {
    return (
      <ReactReduxContext.Provider value={{ store: this.props.store }}>
        {this.props.children}
      ReactReduxContext.Provider>
    )
  }
}
```

8.11 connect.tsx

src\react-redux\connect.tsx

```
import React from "react";
import { bindActionCreators, Unsubscribe, AnyAction, ActionCreatorsMapObject } from "../redux";
import ReactReduxContext from "./context";
import { RootState } from '../store/reducers';
import { ContextValue } from './types';
interface MapStateToProps {
    (state: RootState): any;
}

export default function (mapStateToProps: MapStateToProps, mapDispatchToProps: ActionCreatorsMapObject) {
    return function wrapWithConnect(WrappedComponent: React.ComponentType) {
        return class extends React.Component {
            static contextType = ReactReduxContext;
            unsubscribe: Unsubscribe;
            constructor(props: any, context: ContextValue) {
                super(props);
                this.state = mapStateToProps(context.store.getState());
            }
            componentDidMount() {
                this.unsubscribe = this.context.store.subscribe(() =>
                    this.setState(mapStateToProps(this.context.store.getState()))
                );
            }
            componentWillUnmount() {
                this.unsubscribe();
            }
            render() {
                let actions = bindActionCreators(
                    mapDispatchToProps,
                    this.context.store.dispatch
                );
                return <WrappedComponent {...this.state} {...actions} />;
            }
        };
    };
}
```

hooks版

```
import React, { useContext, useEffect, useState } from 'react';
import ReactReduxContext from './Context';
import { bindActionCreators } from '../redux';
export default function (mapStateToProps, mapDispatchToProps) {
    return function (OldComponent) {
        return function (props) {
            let context = useContext(ReactReduxContext);
            let [state, setState] = useState(mapStateToProps(context.store.getState()));
            useEffect(() => {
                return context.store.subscribe(() =>
                    setState(mapStateToProps(context.store.getState()))
                );
            }, []);
            let actions = bindActionCreators(
                mapDispatchToProps,
                context.store.dispatch
            );
            return <OldComponent {...state} {...actions} />;
        }
    }
}
```

参考

- zhufeng_redux_prepare (https://gitee.com/zhufengpeixun/zhufeng_redux_prepare)