link: null
title: 珠峰架构师成长计划
description: src\store\index.tsx
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=43 sentences=96, words=860

1. 生成项目

```
create-react-app zhufeng_connected_router_ts --typescript
cd zhufeng_connected_router_ts
cnpm i react-router-dom @types/react-router-dom   -S
cnpm i redux react-redux @types/react-redux redux-thunk  redux-logger @types/redux-logger connected-react-router -S
```

2.跑通项目

```
import React from 'react';
import ReactDOM from 'react-dom';
import { Route, Link } from 'react-router-dom';
import Home from './components/Home';
import Counter from './components/Counter';
import { ConnectedRouter } from './connected-react-router'
import history from './history';
import store from './store';
import { Provider } from 'react-redux'
ReactDOM.render(

            <>
                Home
                Counter

            </>

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

src\store\index.tsx

```
import { applyMiddleware, createStore } from 'redux'
import { routerMiddleware } from '../connected-react-router'
import history from '../history';
import reducers from './reducers';
const store = applyMiddleware(routerMiddleware(history))(createStore)(reducers);
export default store;
```

src\history.tsx

```
import { createHashHistory } from 'history'
let history = createHashHistory();
export default history;
```

src\store\reducers\index.tsx

```
import { combineReducers, ReducersMapObject, Action, AnyAction, Reducer } from 'redux'
import { connectRouter, RouterState } from '../../connected-react-router'
import counter, { CounterState } from './counter';
import history from '../../history';
interface Reducers {
    router: RouterState,
    counter: CounterState;
}
let reducers: ReducersMapObject = {
    router: connectRouter(history),
    counter
};
export type RootState = {
    [key in keyof typeof reducers]: ReturnType<typeof reducers[key]>
}
let rootReducer: Reducer = combineReducers(reducers);
export default rootReducer;
```

src\store\reducers\counter.tsx

```
import * as types from '../action-types';
import { AnyAction } from 'redux';
export interface CounterState {
    number: number
}
let initialState: CounterState = { number: 0 }
export default function (state: CounterState = initialState, action: AnyAction): CounterState {
    switch (action.type) {
        case types.INCREMENT:
            return { number: state.number + 1 };
        case types.DECREMENT:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
```

src\store\action-types.tsx

```
export const INCREMENT = 'INCREMENT';
export const DECREMENT = 'DECREMENT';
```

src\store\actions\counter.tsx

```
import * as types from '../action-types';
import { push } from '../../connected-react-router';
export default {
    increment() {
        return { type: types.INCREMENT }
    },
    decrement() {
        return { type: types.DECREMENT }
    },
    go(path: string) {
        return push(path);
    }
}
```

src\components\Home.tsx

```
import React, { Component } from 'react';
import { RouteComponentProps } from 'react-router';
interface IParams { }
type RouteProps = RouteComponentProps;
type Props = RouteProps & {
    children?: any
}
export default class Home extends Component<Props> {
    render() {
        return (
            <div>
                <h1>Homeh1>
                <button onClick={() => this.props.history.go(-1)}>返回button>
            div>
        )
    }
}
```

src\components\Counter.tsx

```
import React, { Component } from 'react'
import { connect } from 'react-redux';
import actions from '../store/actions/counter';
import { CounterState } from '../store/reducers/counter';
import { RootState } from '../store/reducers';
type Props = CounterState & typeof actions;
class Counter extends Component<Props> {
    render() {
        return (
            <>
                <p>{this.props.number}p>
                <button onClick={this.props.increment}>+button>
                <button onClick={this.props.decrement}>-button>
                <button onClick={() => this.props.go('/')}>Homebutton>
            </>
        )
    }
}
let mapStateToProps = (state: RootState): CounterState => state.counter;
export default connect(
    mapStateToProps,
    actions
)(Counter);
```

3.实现connected-react-router

src\connected-react-router\index.tsx

```
import push from './push';
import routerMiddleware from './routerMiddleware';
import connectRouter from './connectRouter';
import ConnectedRouter from './ConnectedRouter';
export {
    push, routerMiddleware, connectRouter, ConnectedRouter
}
export * from './types';
```

src\connected-react-router\types.tsx

```
import { LocationState, Location } from 'history';
export const CALL_HISTORY_METHOD: '@@router/CALL_HISTORY_METHOD' = '@@router/CALL_HISTORY_METHOD';
export const LOCATION_CHANGE: '@@router/LOCATION_CHANGE' = '@@router/LOCATION_CHANGE';
export interface LocationActionPayload {
    method: string;
    args?: A;
}

export interface CallHistoryMethodAction {
    type: typeof CALL_HISTORY_METHOD;
    payload: LocationActionPayload;
}

export interface LocationChangeAction {
    type: typeof LOCATION_CHANGE;
    payload: LocationChangePayload;
}
export interface LocationChangePayload extends RouterState {
    isFirstRendering: boolean;
}
export type Action = 'PUSH' | 'POP' | 'REPLACE';
export type RouterActionType = Action;
export interface RouterState {
    location: Location
    action: RouterActionType
}
```

src\connected-react-router\push.tsx

```
import { LocationState, Path, LocationDescriptorObject } from 'history';
import { CALL_HISTORY_METHOD, CallHistoryMethodAction } from './';
export default function push(location: LocationDescriptorObject): CallHistoryMethodAction]>;
export default function push(location: LocationDescriptorObject): CallHistoryMethodAction]> {
    return {
        type: CALL_HISTORY_METHOD,
        payload: {
            method: 'push',
            args: [location]//history.push(location);
        }
    }
}
```

src\connected-react-router\routerMiddleware.tsx

```
import { History } from 'history';
import { CALL_HISTORY_METHOD } from './';
export default function (history: History) {
    return function (api: any) {
        return function (next: any) {
            return function (action: any) {

                if (action.type === CALL_HISTORY_METHOD) {
                    let method: 'push' | 'go' = action.payload.method;
                    history[method](action.payload.args[0]);
                } else {
                    next(action);
                }
            }
        }
    }
}
```

src\connected-react-router\connectRouter.tsx

```
import React from 'react';
import { ReactReduxContext } from 'react-redux';
import { Router } from 'react-router';
import { History, Location, UnregisterCallback } from 'history';
import { LOCATION_CHANGE, Action } from './';

interface Props {
    history: History
}
export default class ConnectedRouter extends React.Component<Props> {
    static contextType = ReactReduxContext;
    unListen: UnregisterCallback
    componentDidMount() {

        this.unListen = this.props.history.listen((location: Location, action: Action) => {
            this.context.store.dispatch({
                type: LOCATION_CHANGE,
                payload: {
                    location,
                    action
                }
            })
        });
    }
    componentWillUnmount() {
        this.unListen();
    }
    render() {
        let { history, children } = this.props;
        return (
            <Router history={history}>
                {children}
            Router>
        )
    }
}
```

src\connected-react-router\connectRouter.tsx

```
import { History, LocationState } from 'history';
import { AnyAction } from 'redux';
import { LocationChangeAction, LOCATION_CHANGE, RouterState } from './';
export default function connectRouter<S = LocationState>(history: History) {
    let initialState: RouterState = {
        action: history.action,
        location: history.location
    }
    return function (state: RouterState = initialState, action: AnyAction) {
        if (action.type === LOCATION_CHANGE) {
            return (action as LocationChangeAction).payload;
        }
        return state;
    }
}
```