## **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**

### **2.1 src\index.tsx**

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(

<Provider store={store}>

<ConnectedRouter history={history}>

<>

<Link to="/">Home</Link>

<Link to="/counter">Counter</Link>

<Route exact={true} path="/" component={Home} />

<Route path="/counter" component={Counter} />

</>

</ConnectedRouter>

</Provider>

, document.getElementById('root'));

### **2.2 store\index.tsx**

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;

### **2.3 history.tsx**

src\history.tsx

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

### **2.4 reducers\index.tsx**

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<Reducers, any> = {

router: connectRouter(history),

counter

};**export** type RootState = {

[key **in** keyof **typeof** reducers]: ReturnType<**typeof** reducers[key]>

}**let** rootReducer: Reducer<RootState, AnyAction> = combineReducers<RootState, AnyAction>(reducers);**export** **default** rootReducer;

### **2.5 reducers\counter.tsx**

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;

}

}

### **2.6 action-types.tsx**

src\store\action-types.tsx

**export** **const** INCREMENT = 'INCREMENT';**export** **const** DECREMENT = 'DECREMENT';

### **2.7 actions\counter.tsx**

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);

}

}

### **2.8 components\Home.tsx**

src\components\Home.tsx

**import** React, { Component } **from** 'react';**import** { RouteComponentProps } **from** 'react-router';

interface IParams { }

type RouteProps = RouteComponentProps<IParams>;

type Props = RouteProps & {

children?: any

}**export** **default** **class** **Home** **extends** **Component**<**Props**> {

render() {

**return** (

<div>

<h1>Home</h1>

<button onClick={() => this.props.history.go(-1)}>返回</button>

</div>

)

}

}

### **2.9 Counter.tsx**

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('/')}>Home</button>

</>

)

}

}**let** mapStateToProps = (state: RootState): CounterState => state.counter;**export** **default** connect(

mapStateToProps,

actions

)(Counter);

## **3.实现connected-react-router**

### **3.1 index.tsx**

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';

### **3.2 types.tsx**

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<A = any[]> {

method: string;

args?: A;

}//这是一个调用历史对象方法的action//告诉 中间件，我要调用history对象的方法 method args history[method](...args);**export** interface CallHistoryMethodAction<A = any[]> {

type: **typeof** CALL\_HISTORY\_METHOD;

payload: LocationActionPayload<A>;

}

**export** interface LocationChangeAction<S = LocationState> {

type: **typeof** LOCATION\_CHANGE;

payload: LocationChangePayload<S>;

}**export** interface LocationChangePayload<S = LocationState> extends RouterState<S> {

isFirstRendering: boolean;

}**export** type Action = 'PUSH' | 'POP' | 'REPLACE';**export** type RouterActionType = Action;**export** interface RouterState<S = LocationState> {

location: Location<S>

action: RouterActionType

}

### **3.3 push.tsx**

src\connected-react-router\push.tsx

import { LocationState, Path, LocationDescriptorObject } from 'history';

import { CALL\_HISTORY\_METHOD, CallHistoryMethodAction } from './';

export default function push<S = LocationState>(location: LocationDescriptorObject<S>): CallHistoryMethodAction<[LocationDescriptorObject<S>]>;

export default function push<S = LocationState>(location: LocationDescriptorObject<S>): CallHistoryMethodAction<[LocationDescriptorObject<S>]> {

return {

type: CALL\_HISTORY\_METHOD,

payload: {

method: 'push',

args: [location]//history.push(location);

}

}

}

### **3.4 routerMiddleware.tsx**

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) {

//如果相等说明派发的这个动作就是 store.dispatch(push({pathname:'/'}));

**if** (action.type === CALL\_HISTORY\_METHOD) {

**let** method: 'push' | 'go' = action.payload.method;

history[method](action.payload.args[0]);

} **else** {

next(action);

}

}

}

}

}

### **3.5 connectRouter.tsx**

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** './';//react-router-dom/\*\*

\* 这个组件的主要工作就是订阅路径变化事件，当路径发生变化后向仓库派发动作

\* 改变仓库状态对象的router属性

\*/

interface Props {

history: History

}**export** **default** **class** **ConnectedRouter** **extends** **React**.**Component**<**Props**> {

**static** contextType = ReactReduxContext;

unListen: UnregisterCallback

componentDidMount() {

//调用history的listen方法进行监听，监听路径的变化 ，当路径发生变化之后会执行此监听函数，

**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>

)

}

}

### **3.6 connectRouter.tsx**

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<S> = {

action: history.action,

location: history.location

}

**return** **function** (state: RouterState<S> = initialState, action: AnyAction) {

**if** (action.type === LOCATION\_CHANGE) {

**return** (action **as** LocationChangeAction).payload;

}

**return** state;

}

}//saga=>dva=>umi=>antdesignpro