**完整的一套React-Redux**

**首先最外层**

**import** finalCreateStore **from './src/store/configureStore'**

*//引入store配置***import** DevTools **from './src/containers/DevTools'**

*// 引入Redux调试工具DevTools***import** reducer **from './src/reducers'** *// 引入reducers集合***import** routes **from './src/routes'** *// 引入路由配置*

**const** store = finalCreateStore(reducer);  
  
**const** history = **syncHistoryWithStore**(browserHistory,store)  
  
render(  
 <**Provider store=**{store}>  
 <**div**>  
 <**Router history=**{history} **routes=**{routes}/>  
 <**DevTools** />  
 </**div**>  
 </**Provider**>,  
 **document**.getElementById(**'mount'**)  
)

**Provider包裹**

**命名action名称**

**export const** INCREASE = **'INCREASE'  
export const** DECREASE = **'DECREASE'**

**action内引用名称**

**import** { INCREASE, DECREASE, 。。。。 } **from '../constants'**

*// 引入action类型名常量***import 'whatwg-fetch'** *// 可以引入fetch来进行Ajax  
  
// 这里的方法返回一个action对象***export const** *increase* = n => {  
 **return** {  
 **type**: INCREASE,  
 **amount**: n  
 }  
}  
  
**export const** *decrease* = n => {  
 **return** {  
 **type**: DECREASE,  
 **amount**: n  
 }  
}

**返回一个action对象**

**元件内**

**import** { *increase*, *decrease*,*multiThree* } **from '../actions/count'**;

render() {  
 **const** { number, increase, decrease, multiThree } = **this**.props

<**button onClick=**{()=>increase(1)}>+1按钮</**button**>  
{ **' '** }  
<**button onClick=**{()=>decrease(1)}>-1按钮</**button**>

**元件外**

**const** *getNumber* = state => {  
 **return** {  
 **number**: state.**update**.**number** }  
}  
*//mapStateToProps***export default connect**(  
 *getNumber*,  
 {*increase*,*decrease*,*multiThree*}  
)(Home)

**connect连接函数与纯UI元件**

**然后就是reducer**

**import** { INCREASE, DECREASE, GETSUCCESS, REFRESHDATA, MULTITHREE } **from '../constants'** *// 引入action类型常量名*

**export default function** update(state=initialState,action) {  
 **switch**(action.**type**) {  
 **case** INCREASE:  
 **return** Object.assign({},state,{**number**:state.**number**+action.amount});  
 **break**;  
 **case** DECREASE:  
 **return** Object.assign({},state,{**number**:state.**number**-action.amount});  
 **break**;

**。。。。。。**

**/////////////////////////////////////////////////////////////////**

*// reducers/index.js***import** { *combineReducers* } **from 'redux'**

*// 利用combineReducers 合并reducers***import** { **routerReducer** } **from 'react-router-redux'**

*// 将routerReducer一起合并管理***import** *update* **from './count'** *// 引入update这个reducer***export default** *combineReducers*({  
 *update*,  
 **routing**: **routerReducer**})

**合并reducer**

**然后是store**

**import** thunk **from 'redux-thunk'**

*// redux-thunk 支持 dispatch function，并且可以异步调用它***import** createLogger **from 'redux-logger'**

*// 利用redux-logger打印日志***import** { createStore, *applyMiddleware*, *compose* } **from 'redux'**

*// 引入redux createStore、中间件及compose***import** DevTools **from '../containers/DevTools'** *// 引入DevTools调试组件***const** loggerMiddleware = createLogger();  
  
**const** middleware = [thunk,loggerMiddleware];  
  
**const** finalCreateStore = *compose*(  
 *applyMiddleware*(...middleware),  
 DevTools.*instrument*(),  
)(createStore);  
  
**export default** finalCreateStore

**const** config = {  
 **mongoURL**: process.**env**.MONGO\_URL || **'mongodb://localhost:27017/mern-starter'**,  
 **port**: process.**env**.PORT || 8000,  
};  
  
**export default** config;

///////////////////////////////////////////////////////////////////////////////////////////

**const** app = **new** Express();

**if** (process.**env**.**NODE\_ENV** === **'development'**) {  
 **const** compiler = webpack(config);  
 app.use(webpackDevMiddleware(compiler, { **noInfo**: **true**, **publicPath**: config.**output**.**publicPath** }));  
 app.use(webpackHotMiddleware(compiler));  
}

mongoose.connect(serverConfig.**mongoURL**, (error) => {  
 **if** (error) {  
 **console**.**error**(**'Please make sure Mongodb is installed and running!'**); *// eslint-disable-line no-console* **throw** error;  
 }  
  
 *// feed some dummy data in DB.  
 dummyData*();  
});

app.use(compression());  
app.use(bodyParser.**json**({ **limit**: **'20mb'** }));  
app.use(bodyParser.**urlencoded**({ **limit**: **'20mb'**, **extended**: **false** }));  
app.use(**'/static'**, Express.static(path.**resolve**(\_\_dirname, **'../dist/static'**)));  
app.use(**'/api'**, posts);

…………………………

**return `  
 <!doctype html>  
 <html>  
 <head>** ${head.**base**.toString()}

**</body>  
 </html>  
`**;