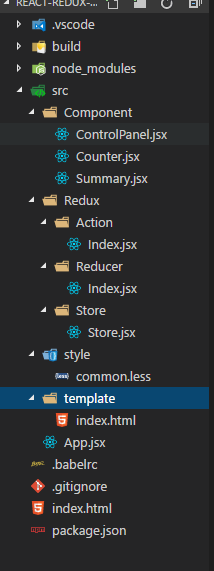
2017-08-25 发布

# [React-Redux 入门教程](https://segmentfault.com/a/1190000010851224)

# ****项目目录****

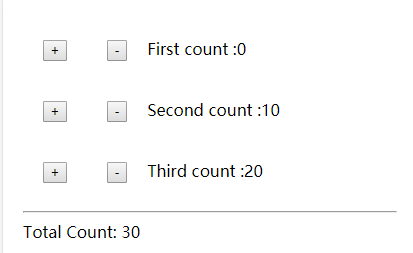


整个项目目录分为图中所示：

Redux分为{**Action**,Reducer,Store}

入口文件为App.jsx

# ****项目效果****



从图中可以看出整个组件可以分为3个组件，内部Counter组件，计算Count的Summary的组件，以及整个容器组件ControlPanel

# Content

React Redux 事实上是两个独立的产品， 应用可以使用 React 而不使用Redux ，也可以使用 Redux 而不使用 React ，但是，如果两者结合使用，没有理由不使用**一个名叫** react-redux 的库这个库能够大大简化代码的书写;

**react-redux** 的两个最主要功能：connect ：连接数据处理组件和内部UI组件；Provider ：提供包含 store的context；通过Content实现传递Store的目的首先定义好

Action/index.jsx

**export** **const** Increment='increment'**export** **const** Decrement='decrement'

**export** **const** increment=(counterCaption)=>({

**type**:Increment,

counterCaption

}

)**export** **const** decrement=(counterCaption)=>({

**type**:Decrement,

counterCaption

})

Reducer/index.jsx

**import** {Increment,Decrement} **from** '../Action'**export** **default**(state,action)=>{

**const** {counterCaption}=action

**switch** (action.type){

**case** Increment:

**return** {...state,[counterCaption]:state[counterCaption]+1}

**case** Decrement:

**return** {...state,[counterCaption]:state[counterCaption]-1}

**default**:

**return** state

}

}

Store/store.jsx

**import** {createStore} **from** 'redux'**import** reducer **from** '../Reducer' **const** initValue={

'First':0,

'Second':10,

'Third':20

}**const** store=createStore(reducer,initValue)**export** **default** store

在**action**中我们会发现定义了两个常量，一个控制增加，一个控制减少，然后暴露出增加减少的函数。这两个函

数可以在Couter组件中调用

Counter.jsx

**import** React, { Component } **from** 'react'**import** {increment,decrement} **from** '../Redux/Action'**import** {connect} **from** 'react-redux';**const** buttonStyle = {margin: "20px"

}

**function** **Counter**({caption, Increment, Decrement, value}){**return** (

<div>

<button style={buttonStyle} onClick={Increment}>+</button>

<button style={buttonStyle} onClick={Decrement}>-</button>

<span>{caption} count :{value}</span>

</div>

)

}**function** **mapState**(state,ownProps){**return**{

value:state[ownProps.caption]

}

}**function** **mapDispatch**(dispatch,ownProps){**return** {

Increment:()=>{

dispatch(increment(ownProps.caption))

},

Decrement:()=>{

dispatch(decrement(ownProps.caption))

}

}

}

**export** **default** connect(mapState,mapDispatch)(Counter)

1.在counter组件中我们会发现引入了增加和减少这两个函数，然后在mapDispatch函数中进行调用，暴露出增

加和减少合并的一个对象，然后通过解构在Counter函数组件中获得传递过来的经过mapDispath包装过后的增

加和减少组件。mapDispatch函数的作用就是把内层函数组件的增加和减少的动作派发给Store

然后我们转过来看Reducer/index.jsx

reducer是专门处理数据逻辑的，通过传入（state,**action**），针对不同的**action**返回一个不同的store对象

Store/store.js

是专门对store进行的一个封装，通过createStore方法传入reducer和初始化state（initValue）来暴露

store对象，此对象非原始的store对象，该对象是对原始store进行注册，增加了若干方法。具体了解此方法可以**\*\*请戳这里\*\***

[https://github.com/reactjs/redux/blob/master/src/createStore.js][1]

最后把**store**对象暴露给**App**.jsx在主入口进行调用

**import** React, {Component, PropTypes} **from** 'react';**import** ReactDOM, {render} **from** 'react-dom';**import** store **from** './Redux/Store/Store.jsx'**import** {Provider} **from** 'react-redux';**import** ControlPanel **from** './Component/ControlPanel.jsx'**import** './style/common.less'

render(

<Provider store={store}>

<ControlPanel />

</Provider>,

document.body.appendChild(document.createElement('div'))

);

我们通过react-redux提供的顶层组件Provider传入store然后把要展示的ControlPanel写入顶层组件就行了，

Provider提供了整个全局的store供所有的子组件进行调用

具体代码实现请git clone   
[https://github.com/jeromehan/...](https://github.com/jeromehan/react-redux-test.git" \t "https://segmentfault.com/a/_blank)