目录

[api 1](#_Toc485580372)

[routerReducer 1](#_Toc485580373)

[syncHistoryWithStore(history, store, [options]) 1](#_Toc485580374)

[listen 1](#_Toc485580375)

[push(location),replace(location),go(number),goBack(),goForward() 2](#_Toc485580376)

[routerMiddleware(history) 2](#_Toc485580377)

[LOCATION\_CHANGE 2](#_Toc485580378)

# api

都是顶层api，除了最后一个字符串。

## routerReducer

一个让store中loaction从history更新的reducer函数、

必须将此reducer添加到您的store，以便同步工作。

如果使用 combineReducers,它应该被嵌套在 routing这个key下。

import { syncHistoryWithStore, routerReducer } from 'react-router-redux'

combineReducers({

...reducers,

routing: routerReducer

})

## syncHistoryWithStore(history, store, [options])

返回创建增强型history。

这个history 修改了history.listen，使其传入所有location首先更新提供的store。这样确保了store是最新状态，无论它是来自导航事件或者时光机的action（例如replay），增强历史记录的监听器将保持同步。

options对象接受以下可选键：

* selectLocationState-(默认state=>state.routing)一个选择器函数，从store中获取history状态。当不使用提供的routerReducer来存储历史记录状态的时候，它就很有用。允许您使用包装器，如Immutable.js。
* adjustUrlOnReplay-(默认true),当使用false时，用时光机时URL不会保持同步。当使用来自Redux Dev的persistState而且在恢复state时不想维持URL状态，这个属性就很有用。

### listen

history.listen(location => analyticsService.track(location.pathname))

## push(location),replace(location),go(number),goBack(),goForward()

必须使用routerMiddleware才能用。

 push- 将一个新的location push到历史记录中，使之成为当前location。

 replace -替换历史记录中的当前位置。

 go-在历史记录中向后或向前移动相对数量的位置。

 goForward-向前移动一个位置。相当于go（1）。

 goBack-向后移动一个位置。相当于go（-1）

push和replace都使用location 描述符，可以是描述URL的对象或者纯URL 字符串。

这些action creator 也可以在一个对象中作为routerActions使用,当使用 Redux 的bindActionCreators()时提供便利。

## routerMiddleware(history)

一个中间件，你可以应用到你的Redux Store中来捕获派发action creator创建的acion。它会将这些操作重定向到提供的history实例。

## LOCATION\_CHANGE

你可以在reducer中侦听的action type。在对历史记录进行任何更改后触发。

# 最新

最新的react-router-redux已经融入react-router(还是要单独安装。)，通过ConnectedRouter替换syncHistoryWithStore.

import React from 'react'

import ReactDOM from 'react-dom'

import { createStore, combineReducers, applyMiddleware } from 'redux'

import { Provider } from 'react-redux'

import createHistory from 'history/createBrowserHistory'

import { Route } from 'react-router'

import { ConnectedRouter, routerReducer, routerMiddleware, push } from 'react-router-redux'

// Create a history of your choosing (we're using a browser history in this case)

const history = createHistory()

// Build the middleware for intercepting and dispatching navigation actions

const middleware = routerMiddleware(history)

// Add the reducer to your store on the `router` key

// Also apply our middleware for navigating

const store = createStore(

combineReducers({

...reducers,

router: routerReducer

}),

applyMiddleware(middleware)

)

// Now you can dispatch navigation actions from anywhere!

// store.dispatch(push('/foo'))

ReactDOM.render(

<Provider store={store}>

{ /\* ConnectedRouter will use the store from Provider automatically \*/ }

<ConnectedRouter history={history}>

<div>

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

<Route path="/about" component={About}/>

<Route path="/topics" component={Topics}/>

</div>

</ConnectedRouter>

</Provider>,

document.getElementById('root')

)