**5 Ways to Connect Redux Actions**

Bài đăng này đã không được cập nhật trong 3 năm

Với function connect của react-redux, ta có thể kết nối redux store của chugns ta đến một component

connect(mapStateToProps, mapDispatchToProps)(MyComponent)

Bài viết này sẽ cung cấp 5 cách để xác định mapDispatchToProps

**Bài toán**

Đoạn code dưới đây sẽ là bài toán mà ta sẽ tìm hiểu lần này. Đây là một ví dụ đơn giản với 2 action sendMessage & deleteMessage

import React, { Component } from 'react'

import { connect } from 'react-redux'

import { sendMessage, deleteMessage } from './actions'

class ChatComponent extends Component {

handleSend = message => {

this.props.sendMessage(message)

}

handleDelete = id => {

this.props.deleteMessage(id)

}

render() {

// ...

}

}

const mapStateToProps = state => state

// ▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽

const mapDispatchToProps = /\* this part is to be discussed \*/

// △ △ △ △ △ △ △ △ △ △

export default connect(

mapStateToProps,

mapDispatchToProps,

)(ChatComponent)

**1. Wrap into Dispatch Manually**

const mapDispatchToProps = dispatch => ({

sendMessage: message => dispatch(sendMessage(message)),

deleteMessage: id => dispatch(deleteMessage(id)),

})

| **Ưu điểm** | **Nhược điểm** |
| --- | --- |
| Có full quyền điều khiển & dễ hiểu | Phải viết sẵn, dễ gấy rối hoặc quên đóng ngoặc, Cú pháp lộn xộn |

**2. Wrap into Dispatch Automatically with Connect**

Trích từ document của hàm connect

mapDispatchToProps [...] Nếu một object được truyền, mỗi functio bên trong nó sẽ được coi như là một Redux action creator. Một object với cùng tên function nhưng với mỗi action creator được gói vào bên trong 1 dispatch call. Vì thế nên chúng có thể gọi trực tiếp & sẽ được merge vào bên trong props của component

Nó có nghĩa là chúng ta không phỉa gửi những action của mình, chỉ cần truyền một object vào connect và chugns ta có thể gọi những action đã được đóng gói đó từ props

Example:

const mapDispatchToProps = {

sendMessage, // will be wrapped into a dispatch call

deleteMessage, // will be wrapped into a dispatch call

};

| **Ưu điểm** | **Nhược điểm** |
| --- | --- |
| Đóng gói ngầm, không cần phải viết 1 bản mẫu | Tất cả thuộc tính sẽ bị đóng gói bên trong 1 dispatch call, Vì vậy không có cách nào để có thể inject functions non-action vào trong props ở đây |

**3. Use bindActionCreators**

Trích từ document của [redux](https://redux.js.org/api/bindactioncreators)

bindActionCreators: Biến một object có values thành action creators thành một đối tượng có cùng khóa, nhưng với mỗi action creator sẽ được đóng gói vào một dispatch call để chúng có thể gọi trực tiếp

import { bindActionCreators } from 'redux'

// ...

const mapDispatchToProps = dispatch => bindActionCreators(

{

sendMessage,

deleteMessage,

},

dispatch,

)

| **Ưu điểm** | **Nhược điểm** |
| --- | --- |
| Ta có thể điều khiển việc đóng gói nhưng không cần thiết phải viết một bản mẫu cho toàn bộ action creators | Phải import thêm và cần phải gửi dispatch rõ ràng |

**4. Use bindActionCreators and Extend the Props**

import { bindActionCreators } from 'redux'

// ...

const mapDispatchToProps = dispatch => ({

...bindActionCreators(

{

sendMessage,

deleteMessage,

},

dispatch,

),

otherService, // this is not to be wrapped into dispatch

})

| **Ưu điểm** | **Nhược điểm** |
| --- | --- |
| Bạn có thể mở rộng object được trả về với các hàm không gọi dispatch | Cú pháp lộn xộn |

**5. Wrap into Dispatch Automatically**

Phần này là khá giống với #2

import \* as messageActions from './messageActions'

import \* as userActions from './userActions'

// ...

const mapDispatchToProps = {

...messageActions,

...userActions,

};

| **Ưu điểm** | **Nhược điểm** |
| --- | --- |
| Ngắn gọi. Tất cả các action có thể được kết nối mà không cần phải liệt kê ra toàn bộ. Những action đã được kết nối sẽ không che những import khi mà dùng hàm destructure | Cách này sẽ import & connect tất cả các action |