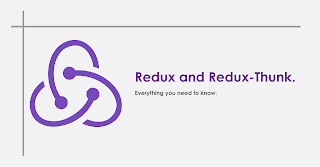
**Xử lý Asynchronous Redux action với Redux-thunk**

Julian Dong viết ngày 23/03/2020

Xin chào, hôm nay mình xin giới thiệu đến mọi người cách xử lý những action bất đồng bộ trong Redux bằng việc sử dụng Redux thunk. Bên cạnh đó cũng sẽ làm 1 ví dụ để các bạn có thể nắm rõ hơn về cách sử dụng **redux-thunk**

[[](https://1.bp.blogspot.com/-2jke1GrSKB0/XnbVKt4XzqI/AAAAAAAANlY/7nZ2d0TpCHUVSTDdTHbpe6U1v8_6VeTdACLcBGAsYHQ/s320/1_SyVnOjQZF-FVaPPYdNjkTQ.png)](https://1.bp.blogspot.com/-2jke1GrSKB0/XnbVKt4XzqI/AAAAAAAANlY/7nZ2d0TpCHUVSTDdTHbpe6U1v8_6VeTdACLcBGAsYHQ/s320/1_SyVnOjQZF-FVaPPYdNjkTQ.png)

*Bài viết gốc tại đây:*[*https://www.juliandong.com/2020/03/xu-ly-asynchronous-redux-action-voi.html*](https://www.juliandong.com/2020/03/xu-ly-asynchronous-redux-action-voi.html)

**1. Giới thiệu**

Trong bài viết lần trước của mình tại [đây](https://www.juliandong.com/2019/09/introduction-of-redux-and-react-redux.html) và [đây](https://www.juliandong.com/2019/11/integrate-redux-in-react-app.html), mình đã giới thiệu về Redux và các áp dụng nó vào trong một project React. Trong đó, mọi action trong redux đều đồng bộ, tức là state sẽ được update ngay lập tức khi action được dispatch.  
Vậy sẽ ra sao nếu action của chúng ta là bất đồng bộ, có nghĩa là action cần gọi đến 1 API bên ngoài để lấy dữ liệu hoặc thực hiện 1 side-effect khiến cho kết quả không thể trả về ngay lập tức được?

Rất may mắn Redux có hỗ trợ các middleware để xử lý vấn đề với asynchronous action và side effect. Nổi tiếng nhất là redux-thunk và redux-saga. Trong phạm vi bài viết này mình sẽ giới thiệu với các bạn về redux-thunk.

**2. Redux-thunk và asynchronous actions**

**Asynchronous actions**

Khi ta gọi 1 API bất đồng bộ, có 2 thời điểm ta cần quan tâm:

1. Thời điểm bắt đầu gọi
2. Thời điểm nhận được kết quả

Tại mỗi thời điểm trên ta đều cần thay đổi state của ứng dụng, và để làm điều đó, ta cần dispatch những action mà sẽ được reducer xử lý một cách đồng bộ. Thông thường với mỗi API call ta cần dispatch 3 loại action

* **Action thông báo cho reducer là bắt đầu thực hiện API call**: Reducer sẽ xử lý action này bằng việc thay đổi cờ loading hoặc isFetching trong state. Khi đó UI sẽ hiển 1 spinner thể hiện dữ liệu đang được xử lý
* **Action thông báo cho reducer là việc gọi API thành công**: Reducer sẽ xử lý action này bằng việc cập nhật kết quả trả về từ API và đồng thời tắt cờ loading. UI khi đó sẽ ẩn spinner và hiển thị kết quả
* **Action thông báo cho reducer là gọi thất bạ**i : Reducer sẽ reset lại cờ loading , lưu lại error vào state và hiển thị error message ở UI

**redux-thunk**

* *thunk*: là một cách gọi khác của function, nhưng nó có 1 điểm đặc biệt là nó là một hàm được trả về từ một hàm khác.

Như chúng ta đã biết về action trong redux chỉ đơn thuần là những plain object có chứa 1 field là type và bất kì dữ liệu nào ta muốn thêm vào

{

"type": "ACTION\_TYPE",

"payload" :"Anything you want"

}

Và action creator là một hàm trả về về một action (plain object)

const actionCreator = (data) => ({

type: "ACTION\_TYPE",

payload: data

})

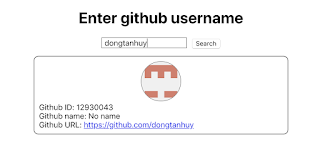
Đối với redux-thunk, nó là 1 middleware cho phép action creator trả về một function (thunk) thay vì trả về plain object. Function này sẽ nhận tham số là hàm dispatch của store, và nó sẽ dispatch các action một cách đồng bộ bên trong thunk khi mà asynchronous call được gọi.  
Nói các khác, use-case thông thường nhất của redux-thunk là khi lấy dữ liệu từ external API, redux-thunk cho phép dispatch các action theo lifecycle của request đến API ngoài.

Ví dụ: ta cần fetch dữ liệu của 1 API, đầu tiên ta sẽ dispatch 1 action để báo rằng dữ liệu đang được fetch, rồi tiếp đó nếu kết quả trả về thành công, ta sẽ dispatch 1 action để báo rằng việc fetch dữ liệu đã kết thúc và nhận được kết quả. Nếu việc fetch thất bại, ta sẽ dispatch 1 action để báo rằng việc fetch dữ liệu kết thúc và nhận về lỗi.

Để nắm rõ hơn về redux-thunk được sử dụng như thế nào trong thực tế, chúng ta sẽ đi qua 1 demo ở phần tiếp theo.

**3. Demo**

Trong ví dụ này, mình sẽ tạo ra 1 react app có nhiệm vụ search user name từ github bằng việc sử dụng react, redux và redux-thunk

[[](https://1.bp.blogspot.com/-p7v7D-OLPtw/XnbmAG9b8NI/AAAAAAAANlk/JcdsC5OYvLs8mAAVDHrJ1MncyYTazcbBACLcBGAsYHQ/s320/Screen+Shot+2020-03-22+at+11.13.37+AM.png)](https://1.bp.blogspot.com/-p7v7D-OLPtw/XnbmAG9b8NI/AAAAAAAANlk/JcdsC5OYvLs8mAAVDHrJ1MncyYTazcbBACLcBGAsYHQ/s320/Screen+Shot+2020-03-22+at+11.13.37+AM.png)

**Khởi tạo project và cài đặt các package cần thiết**

Mình sẽ dùng create react app để khởi tạo project

$ npm init react-app react-thunk *# init project*

$ cd react-thunk

$ npm i redux redux-thunk axios *# install packages*

Sau khi khởi tạo xong, chúng ta tiến hành cấu trúc lại thư mục src như sau:

* src/components: chứa các component dùng trong ứng dụng
* src/actions: chứa action của redux
* src/reducer: chứa store và reducers
* src/service: các service để gọi API
* src/style: chứa file css

**Setup redux**

Trong src/index.js

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './components/App';

import { Provider } from 'react-redux';

import {store} from './reducer/store';

import \* as serviceWorker from './serviceWorker';

ReactDOM.render(

<Provider store={store}>

<React.StrictMode>

<App />

</React.StrictMode>

</Provider>,

document.getElementById('root')

);

serviceWorker.unregister();

Thêm redux thunk vào src/reducer/store.js

import { createStore, applyMiddleware } from 'redux';

import thunk from 'redux-thunk';

import {rootReducer} from './reducers';

export const store = createStore(rootReducer, applyMiddleware(thunk));

**Viết service gọi API**

Trong src/services/index.js

import axios from 'axios';

const fetchUserService = username => {

return new Promise((resolve, reject) => {

axios.get(`https:*//api.github.com/users/${username}`)*

.then(response => resolve(response.data))

.catch(error => reject(error))

})

}

export default fetchUserService;

**Tạo action**

Trong src/actions/fetchUser.js

import {

FETCH\_USER,

FETCH\_USER\_FAILED,

FETCH\_USER\_SUCCESS

} from './constants';

import fetchUserSerivce from '../services';

export default username => {

return dispatch => {

dispatch(fetchUser());

fetchUserSerivce(username)

.then(user => dispatch(fetchUserSuccess(user)))

.catch(error => dispatch(fetchUserFailed(error)))

}

}

const fetchUser = () => ({

type: FETCH\_USER

});

const fetchUserSuccess = user => ({

type: FETCH\_USER\_SUCCESS,

payload: {

user

},

})

const fetchUserFailed = error => ({

type: FETCH\_USER\_FAILED,

payload: { error }

})

**Tạo reducer**

Trong src/reducer/reducers.js

import {

FETCH\_USER,

FETCH\_USER\_SUCCESS,

FETCH\_USER\_FAILED

} from '../actions/constants';

const initialState = {

loading: false,

error: null,

user: null

}

export const rootReducer = (state = initialState, action) => {

switch (action.type) {

case FETCH\_USER:

return {

loading: true,

user: null,

error: null,

};

case FETCH\_USER\_SUCCESS: {

return {

loading: false,

user: action.payload.user,

error: null,

};

}

case FETCH\_USER\_FAILED: {

return {

loading: false,

user: null,

error: action.payload.error

}

}

default:

return state;

}

}

Như vậy chúng ta đã setup redux xong, giờ sẽ hiện thực UI components

**Search bar**

Trong src/components/SearchBar.js

import React from 'react';

import fetchUser from '../actions/fetchUser';

import { connect } from 'react-redux';

import '../style/SearchBar.css'

class SearchBar extends React.Component {

constructor(props) {

super(props);

this.state = {

username: ''

}

this.\_onChange = this.\_onChange.bind(this);

this.\_onSubmit = this.\_onSubmit.bind(this);

}

\_onChange(event) {

const value = event.target.value;

this.setState({

username: value

})

}

\_onSubmit(event) {

event.preventDefault();

this.props.fetchUser(this.state.username)

}

render() {

return (

<div className="form-wrapper">

<h1>Enter github username</h1>

<form onSubmit={this.\_onSubmit}>

<input className="input" type="text" placeholder="User name" onChange={this.\_onChange} required />

<input className="button" type="submit" value={this.props.loading ? "Searching..." : "Search"} disabled={this.props.loading} />

</form>

</div>

)

}

}

const mapState = state => ({

loading: state.loading

})

const mapDispatch = dispatch => ({

fetchUser: username => dispatch(fetchUser(username))

});

export default connect(mapState,mapDispatch)(SearchBar);

**UserInfomation**

Trong src/components/UserInformation.js

import React from 'react';

import { connect } from 'react-redux';

import '../style/UserInformation.css'

const UserInformation = (props) => {

const { user, error, loading } = props;

return (

<>

{loading && (<h3 className="loading">Searching... </h3>)}

{error && (<h3 className="error">{error.message}</h3>)}

{user && (

<div className="main">

<img src={user.avatar\_url} alt="avatar" />

<DataField

label="Github ID"

value={user.id}

/>

<DataField

label="Github name"

value={user.name}

/>

<DataField

label="Github URL"

value={user.html\_url}

isURL

/>

</div>

)}

</>

)

}

const DataField = ({

label,

value,

isURL

}) => {

return (

<div className="data">

<label>{label}: </label>

{isURL ? (<a href={value}>{value}</a>) : (<span>{value || "No name"}</span>)}

</div>

)

}

const mapState = state => ({

user: state.user,

error: state.error,

loading: state.loading

});

export default connect(mapState, null)(UserInformation);

**Import into App.js**

Trong src/components/App.js

import React from 'react';

import SearchBar from './SearchBar';

import UserInformation from './UserInformation';

import '../style/App.css';

const App = () => {

return(

<div className="App">

<SearchBar />

<UserInformation />

</div>

)

}

export default App;

**Cuối cùng chúng ta style cho các component**

*/\* src/style/App.css\*/*

.App {

text-align: center;

}

.App-logo {

height: 40vmin;

pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

.App-logo {

animation: App-logo-spin infinite 20s linear;

}

}

.App-header {

background-color: #282c34;

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

.App-link {

color: #61dafb;

}

@keyframes App-logo-spin {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

*/\* src/style/SearchBar.css\*/*

.form-wrapper {

margin: 15px;

}

.input {

outline: none;

font-size: 16px;

padding-left: 5px;

border: 1px solid gray;

}

.button {

margin-left: 10px;

font-size: 16px;

}

*/\* src/style/UserInformation.css\*/*

.main {

text-align: center;

max-width: 500px;

margin: auto;

border: 1px solid black;

padding: 10px;

border-radius: 10px;

}

.loading {

color: greenyellow;

}

.error {

color: red;

}

.data {

text-align: justify;

}

img {

width: 80px;

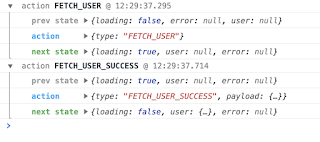
height: 80px;

border-radius: 40px;

border: 1px solid gray;

}

Chạy thử bằng npm run start và mở trình duyệt http://localhost:3000 và thử nhập tên github user của bạn.  
Bạn có thể thử sử dụng middleware logger và kiểm tra kết quả ở console.log. Khi thực hiện tìm username thành, sẽ có lần lượt 2 action được dispatch

[[](https://1.bp.blogspot.com/-sHAPtvpeomM/Xnb4L4zYB7I/AAAAAAAANlw/C754OcLFFgMpehlFa6NZdwBuvONMn1fbgCLcBGAsYHQ/s320/Screen+Shot+2020-03-22+at+12.31.17+PM.png)](https://1.bp.blogspot.com/-sHAPtvpeomM/Xnb4L4zYB7I/AAAAAAAANlw/C754OcLFFgMpehlFa6NZdwBuvONMn1fbgCLcBGAsYHQ/s320/Screen+Shot+2020-03-22+at+12.31.17+PM.png)

**Kết**

Hi vọng qua bài viết trên các bạn có thể nắm được cách sử dụng redux-thunk và có thể áp dụng nó trong dự án thực tế.  
Mình có đính kém link source code demo ở [đây](https://github.com/dongtanhuy/react-thunk) để mọi người tham khảo.  
Nếu thấy bài viết hay hãy chia sẻ cho mọi người. Nếu bạn có ý kiến đóng góp đừng ngần ngại để lại comment nhé.  
Thanks

**Reference**

* Redux asynchronous action: <https://redux.js.org/advanced/async-actions>
* Redux-thunk: <https://github.com/reduxjs/redux-thunk>