**Hướng dẫn sử dụng Redux trong Reactjs**

**Giới thiệu**

Redux là một thư viện JavaScript mã nguồn mở để quản lý trạng thái ứng dụng. Nó được sử dụng phổ biến nhất với các thư viện như React hoặc Angular để xây dựng giao diện người dùng. Nói chung Redux khá là phổ biến. Tuy nhiên, không phải tất cả chúng ta đều biết nó là gì và cách sử dụng nó ra sao. Trong bài này, chúng ta sẽ xem vài lý do tại sao nên sử dụng redux bằng cách phân tích những lợi ích mà nó mang lại và cách hoạt động của nó.

**Lý do Redux ra đời**

Do yêu cầu cho các ứng dụng single-page sử dụng Javascript ngày càng trở lên phức tạp thì code của chúng ta phải quản lý nhiều state hơn. Với Redux, state của ứng dụng được giữ trong một nơi gọi là store và mỗi component đều có thể access bất kỳ state nào mà chúng muốn từ chúng store này.

**Hiểu cách Redux làm việc**

Cái cách mà Redux hoạt động là khá đơn giản. Nó có 1 store lưu trữ toàn bộ state của app. Mỗi component có thể access trực tiếp đến state được lưu trữ thay vì phải send drop down props từ component này đến component khác.

**Action**

Actions đơn giản là các events. Chúng là cách mà chúng ta send data từ app đến Redux store. Những data này có thể là từ sự tương tác của user vs app, API calls hoặc cũng có thể là từ form submission. Actions được gửi bằng cách sử dụng store.dispatch() method, chúng phải có một type property để biểu lộ loại action để thực hiện. Chúng cũng phải có một payload chứa thông tin. Actions được tạo thông qua một action creator. Ví dụ:

**Reducers**

Reducers là các function nguyên thủy chúng lấy state hiện tại của app, thực hiện một action và trả về một state mới. Những states này được lưu như những objects và chúng định rõ cách state của một ứng dụng thay đổi trong việc phản hồi một action được gửi đến store.

**Store**

Store lưu trạng thái ứng dụng và nó là duy nhất trong bất kỳ một ứng dụng Redux nào. Bạn có thể access các state được lưu, update state, và đăng ký or hủy đăng ký các listeners thông qua helper methods. Các actions thực hiện trên một state luôn luôn trả về một state mới. Vì vậy, state này là đơn giản và dễ đoán.

**Nguyên lý vận hành**

**Cài đặt**

npx create-react-app learn-reactjs

npm install --save react-router-dom

npm install @reduxjs/toolkit react-redux

cd learn-reactjs

npm start

**Tạo các thư mục**

learn-reactjs

├─ build

├─ node\_modules

├─ public

└─ src

├─ api

├─ app

│ └─ store.js

├─ components

├─ features

│ └─ Counter

│ ├─ counterSlice.js

│ └─ index.jsx

├─ App.css

├─ App.js

└─ index.js

**App**

**Cung cấp Redux Store cho React**

Chỉnh sửa file index.js

import React from 'react';

import ReactDOM from 'react-dom';

import { BrowserRouter } from 'react-router-dom';

import App from './App';

import './index.css';

import reportWebVitals from './reportWebVitals';

import { Provider } from 'react-redux';

import store from './app/store';

ReactDOM.render(

<React.StrictMode>

<Provider store={store}>

<BrowserRouter>

<App />

</BrowserRouter>

</Provider>

</React.StrictMode>,

document.getElementById('root')

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

Chỉnh sửa file app.js

import { Redirect, Route, Switch } from 'react-router-dom';

import './App.css';

import CounterFeature from './features/Counter';

function App() {

return (

<div className="app">

<Switch>

<Redirect from="home" to="/" exact />

<Route path="/products" component={CounterFeature} />

</Switch>

</div>

);

}

export default App;

**Store**

**Tạo một Redux Store**

Tạo một Redux Store bằng cách chỉnh sửa file store.js trong thư mục store

import couterReducer from '../features/Counter/counterSlice';

const { configureStore } = require("@reduxjs/toolkit");

const rootReducer = {

counter: couterReducer,

}

const store = configureStore({

reducer: rootReducer,

})

export default store

**Counter**

**Tạo một slice state cho Redux**

Tạo một slice state cho Redux bằng cách chỉnh sửa file counterSlice.js trong thư mục Counter. Tạo một slice yêu cầu một tên chuỗi để xác định slice, một giá trị trạng thái ban đầu và một hoặc nhiều hàm giảm thiểu để xác định cách state có thể được cập nhật. Khi slice được tạo, chúng ta có thể export các action Redux đã tạo và hàm giảm thiểu cho toàn bộ slice.

import {createSlice} from '@reduxjs/toolkit';

const counterSlice = createSlice({

name: 'counter', // tên chuỗi xác định slice

initialState: 0, // giá trị khởi tạo ban đầu

reducers: { // tạo các actions

increase(state, action) { //action increase

return state + 1;

},

decrease(state, action) { //action decrease

return state - 1;

},

}

})

const { actions, reducer } = counterSlice

export const {increase, decrease} = actions // export action

export default reducer //ngầm hiểu chúng ta đang export counterSlice

**counter**

Chỉnh sửa index.jsx trong thư mục Counter

import React from 'react';

import { useDispatch, useSelector } from 'react-redux';

import { increase, decrease } from './counterSlice.js';

function CounterFeature(props) {

//lấy counter ở rootReducer

const counter = useSelector(state => state.counter) // useSelector là một hook giúp lấy 1 cái state trong root của mình

const dispatch = useDispatch(); // sử dụng dispatch để gửi action

const handleIncreaseClick = () => {

const action = increase();

dispatch(action);

}

const handleDecreaseClick = () => {

const action = decrease();

dispatch(action);

}

return (

<div>

<h1>Counter</h1>

<p>Counter {counter}</p>

<button onClick={() => handleIncreaseClick()} >increase</button>

<button onClick={() => handleDecreaseClick()}>decrease</button>

</div>

);

}

export default CounterFeature;

Bài viết đến đây là kết thúc. Chúc các bạn thành công