## [Thực hành] Redux nâng cao

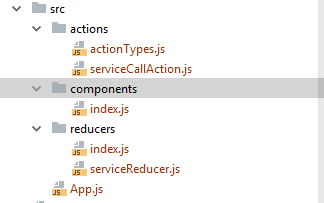
#### **Bài toán**

* Tạo 1 màn hình load về các bài báo mới nhất từ Dân Trí
* Cấu trúc projet theo Redux
* Màn hình sẽ bao gồm 1 listView, mỗi row bao gồm 1 ảnh thumbnail và 1 title của bài báo

#### **Các bước thực hiện**

**Bước 1:** Khởi tạo project mới, giống bài học số 2

**Bước 2:** Cấu trúc dự án theo Redux như sau :



Trong đó :

* App khởi tạo component cha của dự án
* Action : nơi phát ra các action, gọi api
* Reducer : tiếp nhận các action, nhận vào state và trả ra state mới
* Component : thành phần giao diện của dự án, hiển thị kết quả ra màn hình

**Bước 3:**  Cài đặt các thư viện cần thiết trong file Package.json

{  
 **"name"**: **"EmptyProject"**,  
 **"version"**: **"0.0.1"**,  
 **"private"**: **true**,  
 **"scripts"**: {  
 **"start"**: **"node node\_modules/react-native/local-cli/cli.js start"**,  
 **"test"**: **"jest"** },  
 **"dependencies"**: {  
 **"react"**: **"16.3.1"**,  
 **"react-native"**: **"0.55.4"**,  
 **"react-native-axios"**: **"^0.17.1"**,  
 **"react-redux"**: **"^5.0.4"**,  
 **"redux"**: **"^3.6.0"**,  
 **"redux-logger"**: **"^2.6.1"**,  
 **"redux-promise"**: **"^0.5.3"**,  
 **"redux-thunk"**: **"^2.2.0"** },  
 **"devDependencies"**: {  
 **"babel-jest"**: **"23.4.0"**,  
 **"babel-preset-react-native"**: **"4.0.0"**,  
 **"jest"**: **"23.4.1"**,  
 **"react-test-renderer"**: **"16.3.1"** },  
 **"jest"**: {  
 **"preset"**: **"react-native"** }  
}

Các thư viện mới được sử dụng trong dự án là :

* Redux
* Redux-thunk
* React-native-axios

Để cài đặt ta thực hiện : npm install tại thư mục gốc của dự án

**Bước 4:**  Trong App.js chúng ta viết code như sau:

<**Provider store=**{store}>  
 <**ServiceAction** />  
</**Provider**>

Ở đây chúng ta đang gói tất cả các component lại với component Provider. Component Provider được cung cấp bởi react-redux (lý do bạn phải cài đặt nó ở trên là đây) giúp cho chúng ta có thể truy cập store cũng như tất cả những function của nó ở tất các các component con.

**Bước 4:**  Trong ServiceAction.js chúng ta sẽ làm gì ?

**import** \* **as** ActionTypes **from './actionTypes'**;  
**import** { **connect** } **from 'react-redux'**;  
**import axios from 'react-native-axios'**;  
**import** ServiceComponent **from '../components/index'**;  
  
**const** *mapStateToProps* = (state) => ({  
 **isLoading**: state.*serviceReducer*.**isLoading**,  
 **error**: state.*serviceReducer*.**error**,  
 **data**: state.*serviceReducer*.**data**});  
  
**const** *mapDispatchToProps* = (dispatch) => ({  
 callService: () => dispatch(*callWebservice*())  
});  
  
**export const** *callWebservice* = () => {  
 **return** dispatch => {  
 dispatch(*serviceActionPending*());  
 axios.get(**'https://api.rss2json.com/v1/api.json?rss\_url=http%3A%2F%2Fdantri.com.vn%2Ftrangchu.rss'**)  
 .then(response => {  
 ***console***.log(response);  
 dispatch(*serviceActionSuccess*(response.**data**.items))  
 })  
 .catch(error => {  
 dispatch(*serviceActionError*(error))  
 });  
 }  
};  
  
**export const** *serviceActionPending* = () => ({  
 **type**: ActionTypes.SERVICE\_PENDING  
});  
  
**export const** *serviceActionError* = (error) => ({  
 **type**: ActionTypes.SERVICE\_ERROR,  
 **error**: error  
});  
  
**export const** *serviceActionSuccess* = (data) => ({  
 **type**: ActionTypes.SERVICE\_SUCCESS,  
 **data**: data  
});  
  
**export default connect**(*mapStateToProps*, *mapDispatchToProps*)(ServiceComponent);

ở đây :

* Chúng ta chỉ định ServiceComponent sẽ lấy vào state nào. 1 ứng dụng Redux có rất nhiều state, vì thế ta phải chỉ rõ từng state mà component đó muốn lấy. -> *mapStateToProps*
* *callWebservice :* là 1 hàm gọi tới API. Hàm này xử lý bất đồng bộ nên ta sử dụng thư viện axios
* Các hàm *serviceActionSuccess, serviceActionError, serviceActionSuccess như đã học về Redux thì đây là các pure function, chỉ rõ action sẽ xảy ra trong quá trình call API bất đồng bộ : pending, error, success*

**Bước 5:**  Khởi tạo Reducer, đây là nơi xác định state thay đổi như thế nào

**import** \* **as** Actions **from '../actions/actionTypes'  
  
const** *ServiceReducer* = (state = { **isLoading**: **false**, **error**: **undefined**, **data**: {} }, action) => {  
 **switch** (action.**type**) {  
 **case** Actions.SERVICE\_PENDING:  
 **return *Object***.assign({}, state, {  
 **isLoading**: **true**,  
 });  
 **case** Actions.SERVICE\_ERROR:  
 **return *Object***.assign({}, state, {  
 **isLoading**: **false**,  
 **error**: action.**error** });  
 **case** Actions.SERVICE\_SUCCESS:  
 **return *Object***.assign({}, state, {  
 **isLoading**: **false**,  
 **data**: action.**data** });  
 **default**:  
 **return** state;  
 }  
};  
  
**export default** *ServiceReducer*;

Trong đó:

Nó sẽ lắng nghe các action, nhận vào state cũ và trả về state mới, không thay đổi trực tiếp state

**Bước 6:**  Tạo component

* Call Api ở hàm componentDidMount

componentDidMount() {  
 **this**.**props**.callService()  
}

* Sử dụng ListView để hiển thị 1 list bài viết ra màn hình
* Sử dụng ActivityIndicator hiển thị màn hình loading khi dữ liệu chưa trả về

**Bước 7:** Chạy chương trình. Quan sát kết quả.