Introduction Redux :

1. Introduction

Redux est une bibliothèque de gestion d'état qui fournit un moyen prévisible de stocker et de mettre à jour l'état global de l'application. En utilisant Redux, on va stocker toutes les données dont on a besoin dans un seul « store », qui peut être partagé entre tous les composants de l'application

Nous allons voir ici son utilisation à titre d’introduction.

1. Installation

La première chose à effectuer c’est d’installer les packages utiles pour son utilisation.

npm install @reduxjs/toolkit react-redux

1. Utilisation

* Création d’un store

Avant il convient de créer un store dans lequel on va stocker nos états.

Dans notre exemple on va créer un dossier store avec un fichier store.jsx dans lequel on va déclarer notre store en utilisant Redux.

import { configureStore } from '@reduxjs/toolkit'  
  
export default configureStore({  
 reducer: {},  
})

Pour l’instant la propriété reducer, qui représente le store est vide et nous mettrons par la suite différentes données (Redux State Slice)

* Fournir le store à l’ensemble des composants.

La deuxieme étape est de fournir le store aux composant en englobant un store parent dans le JSX.

Dans notre exemple on va mettre en place cela sur l’ensemble de l’application <App>.

import React from 'react'

import ReactDOM from 'react-dom/client'

import App from './App.jsx'

import './index.css'

import { Provider } from 'react-redux'

import store from './store/store.jsx'

ReactDOM.createRoot(document.getElementById('root')).render(

<React.StrictMode>

<Provider store={store}>

<App />

</Provider>

</React.StrictMode>,

)

Il faut impérativement importer la balise Provider de react-redux et par la suite notre store à mettre à disposition dans l’attribut store en utilisant les {}.

De ce fait, le store est disponible pour l’ensemble de la hiérarchie de composant situant à partir de App.

* Création d’un Redux State Slice

Notre store est actuellement vide, on va le remplir en utilisant un Redux State Slice.

C’est un fichier js (jsx ici) qui respect une construction bien précise.

Dans notre projet nous avons créer dans src, un dossier features, puis counter et le fichier counterSlice.jsx qui represent notre Redux State Slice :

Dans ce fichier on export une variable qui contient le retour de la fonction createSlice (de @reduxjs/toolkit).

Cette méthode prend en argument un objet JS.

Avec :

* Une propriété name représentant le name du slice.
* initialState qui represente l’etat initial pour cette slice.
* reducers : qui represente les méthodes disponibles pour cette slice.
  + Increment
  + Decrement
  + incrementByAmount

un reducer prend en argument le state actuel pour pouvoir le modifier.

Par la suite par déstructuration on va créer des actions de chaque reducers de notre slice. (qu’on pense à exporter)

Et on rajouter une exportation par défaut pour le reducer de notre slice pour finir.

import { createSlice } from '@reduxjs/toolkit'

export const counterSlice = createSlice({

name: 'counter',

initialState: {

value: 0,

},

reducers: {

increment: (state) => {

// Redux Toolkit allows us to write "mutating" logic in reducers. It

// doesn't actually mutate the state because it uses the Immer library,

// which detects changes to a "draft state" and produces a brand new

// immutable state based off those changes.

// Also, no return statement is required from these functions.

state.value += 1

},

decrement: (state) => {

state.value -= 1

},

incrementByAmount: (state, action) => {

state.value += action.payload

},

},

})

// Action creators are generated for each case reducer function

export const { increment, decrement, incrementByAmount } = counterSlice.actions

export default counterSlice.reducer

* ajouter la slice dans le store

Dans cette étape nous allons ajouter cette nouvelle slice dans notre store.

En lui donnant un nom, ici counter (l’export du reducer de notre slice est utile ici pour son référencement.

import { configureStore } from '@reduxjs/toolkit'

import counterReducer from '../features/counter/counterSlice'

/\*Création d'un magasin vide\*/

export default configureStore({

reducer: {

counter: counterReducer

},

})

* lier à un composant :

Une fois notre store complètement en place avec la slice on va la lier à un composant.

Pour lié une valeur contenu dans notre état on utilise le hook useSelector qui prend en parametre une fonction (avec en argument le state et qui retourne la valeur voulue de notre state)

On déclare une variable dispatch en utilisant le hook useDispatch() qui permettra l’appel des différentes fonctions.

On associe par la suite les actions à des event d’éléments de notre html en invoquant la la fonction contenue dans la variable dispach et en y appliquant une action qu’on recupere de notre slice qu’on a exposé.

import React from 'react'

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

import { decrement, increment } from '../features/counter/counterSlice'

export function Counter() {

const count = useSelector((state) => state.counter.value)

const dispatch = useDispatch()

return (

<div>

<div>

<button

aria-label="Increment value"

onClick={() => dispatch(increment())}

>

Increment

</button>

<span>{count}</span>

<button

aria-label="Decrement value"

onClick={() => dispatch(decrement())}

>

Decrement

</button>

</div>

</div>

)

}

L’état du store peut être visible grâce à des outils comme Redux DevTools sur chrome.