# ThunkProgram

## main.jsx

import ReactDOM from "react-dom/client";

import App from "./App.jsx";

import { Provider } from "react-redux";

import { store } from "./Redux/srore.js";

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

<Provider store={store}>

<App />

</Provider>

);

## App.jsx

import Counter from "./Components/Counter";

import { Product } from "./Components/Product";

function App() {

return (

<>

<Counter />

<Product />

</>

);

}

export default App;

## actiontype.js

//counter

export const ADD = "ADD";

export const SUB = "SUB";

//product

export const GET\_PRODUCT\_REQUEST = "GET\_PRODUCT\_REQUEST";

export const GET\_PRODUCT\_SUCCESS = "GET\_PRODUCT\_SUCCESS";

export const GET\_PRODUCT\_FAIL = "GET\_PRODUCT\_FAIL";

## store.js

import { applyMiddleware, combineReducers, legacy\_createStore } from "redux";

import { reducer as CounterReducer } from "./Counter/reducer";

import { reducer as ProductReducer } from "./Product/reducer";

import { thunk } from "redux-thunk";

const rootReducer = combineReducers({

CounterReducer,

ProductReducer,

});

export const store = legacy\_createStore(rootReducer, applyMiddleware(thunk));

## REDUX Folder

## COUNTER Folder

## action.js

import { ADD, SUB } from "../actiontype";

export const HandleAdd = (value) => {

return { type: ADD, payload: value };

};

export const HandleSub = (value) => {

return { type: SUB, payload: value };

};

## 2. reducer.js

import { ADD, SUB } from "../actiontype";

let initialstate = {

counter: 0,

};

// eslint-disable-next-line no-unused-vars

export const reducer = (state = initialstate, { type, payload }) => {

switch (type) {

case ADD:

return { counter: state.counter + payload };

case SUB:

return { counter: state.counter - payload };

default:

return state;

}

};

## PRODUCT Folder

## action.js

import axios from "axios";

import { GET\_PRODUCT\_FAIL, GET\_PRODUCT\_REQUEST, GET\_PRODUCT\_SUCCESS } from "../actiontype";

export const fetchdata = (dispatch) => {

dispatch({ type: GET\_PRODUCT\_REQUEST, payload: true });

axios

.get("https://fakestoreapi.com/products")

.then((res) => {

console.log(res.data);

dispatch({ type: GET\_PRODUCT\_REQUEST, payload: false });

dispatch({ type: GET\_PRODUCT\_SUCCESS, payload: res.data });

})

.catch((err) => {

console.log(err);

dispatch({ type: GET\_PRODUCT\_REQUEST, payload: false });

dispatch({ type: GET\_PRODUCT\_FAIL, payload: true });

});

};

## reducer.js

import {

GET\_PRODUCT\_FAIL,

GET\_PRODUCT\_REQUEST,

GET\_PRODUCT\_SUCCESS,

} from "../actiontype";

let initialstate = {

isLoading: false,

isError: false,

data: [],

};

// eslint-disable-next-line no-unused-vars

export const reducer = (state = initialstate, { type, payload }) => {

switch (type) {

case GET\_PRODUCT\_REQUEST:

return { ...state, isLoading: payload };

case GET\_PRODUCT\_SUCCESS:

return { ...state, data: payload };

case GET\_PRODUCT\_FAIL:

return { ...state, isError: payload };

default:

return state;

}

};

## COMPONENT

## Counter.jsx

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

import { HandleAdd, HandleSub } from "../Redux/Counter/action";

const Counter = () => {

const countervalue = useSelector((store) => store.CounterReducer.counter);

const dispatch = useDispatch();

console.log(countervalue);

return (

<div>

<h1>Counter : {countervalue}</h1>

<button onClick={() => dispatch(HandleAdd(1))}>+</button>

<button onClick={() => dispatch(HandleSub(1))}>-</button>

</div>

);};

export default Counter;

## 2.Product.jsx

import { useEffect } from "react";

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

import { fetchdata } from "../Redux/Product/action"

export const Product = () => {

// const productvalue = useSelector((store) => store.ProductReducer);

// console.log(productvalue);

const { isLoading, isError, data } = useSelector(

(store) => store.ProductReducer

);

const dispatch = useDispatch();

useEffect(() => {

// fetchdata(dispatch); //not correct dispatch object consider ,function controlling dispatch

dispatch(fetchdata);

}, []);

return isLoading ? (

<h1>Loading...</h1>

) : isError ? (

<h1>Something went to wrong</h1>

) : ( <div>

<h1>Product</h1>

<div style={{ display: "grid", gridTemplateColumns: "repeat(3, 1fr)" }}>

{data.map((el) => (

<div key={el.id}>

<img src={el.image} alt="" height={200} width={200} />

<h3>{el.title}</h3>

</div>))}

</div>

</div>);};