**บทที่ 10 พื้นฐานการใช้งาน Redux**

**การใช้งาน Redux ใน Node.js**

**ขั้นตอนที่ 1 สร้าง store สำหรับเก็บข้อมูลของ state ไว้ที่ศูนย์กลาง**

const redux = require('redux');

const createStore = redux.createStore;

const initialState = {

counter: 1

}

const reducer = (state = initialState, action) => {

switch(action.type) {

case 'INC\_COUNTER':

return {

...initialState,

counter: initialState.counter + 1

};

case 'DEC\_COUNTER':

return {

...initialState,

counter: initialState.counter + 10

}

default:

break;

}

return state;

}

const store = createStore(reducer);

console.log(store.getState());

**ขั้นตอนที่ 2 ส่ง Action ไปยัง Reducer เพื่อสั่งให้อัพเดท State**

store.dispatch({type: 'INC\_COUNTER'});

store.dispatch({type: 'DEC\_COUNTER'});

**ขั้นตอนที่ 3 กำหนดฟังก์ชันใน subscribe()**

store.subscribe(() => {

console.log('State Updated !!!', store.getState());

});

store.dispatch({type: 'INC\_COUNTER'});

store.dispatch({type: 'INC\_COUNTER'});

store.dispatch({type: 'DEC\_COUNTER'});

const redux = require('redux');

const createStore = redux.createStore;

const initialState = {

counter: 1

}

const reducer = (state = initialState, action) => {

switch(action.type) {

case 'INC\_COUNTER':

return {

...initialState,

counter: initialState.counter + 1

};

case 'DEC\_COUNTER':

return {

...initialState,

counter: initialState.counter - 1

}

default:

break;

}

return state;

}

const store = createStore(reducer);

console.log(store.getState());

store.subscribe(() => {

console.log('State Updated !!!', store.getState());

});

store.dispatch({type: 'INC\_COUNTER'});

store.dispatch({type: 'INC\_COUNTER'});

store.dispatch({type: 'DEC\_COUNTER'});

**การผ่านค่า playload ไปพร้อมกับ action**

const data = {

id: 1,

name: 'Jeerawuth',

score: 89

}

store.dispatch({type:'ADD', playload: data});

const redux = require('redux');

const createStore = redux.createStore;

const initialState = {

data: [

{ id: 1, name: 'Jeerawuth', score: 98 },

{ id: 3, name: 'Sombat', score: 87}

]

}

const reducer = (state = initialState, action) => {

switch(action.type) {

case 'ADD':

const newState = {

...state,

data: [...state.data, action.playload]

}

return newState;

case 'DEL':

const delState = {

...state,

data: state.data.filter(student => student.id !== action.playload)

}

return delState;

default:

break;

}

return state;

}

const store = createStore(reducer);

store.subscribe(() => {

console.log('State Updated!! ', store.getState().data);

})

const data = {

id: 8,

name: 'Anna',

score: 77

}

store.dispatch({type:'ADD', playload: data});

store.dispatch({type:'DEL', playload: 3});

**ตัวอย่างการใช้งาน HOC**

**ขั้นตอนที่ 1 สร้างคอมโปเน้นท์ Box ซึ่งจะทำหน้าที่เป็น HOC**

import React from 'react';

const Box = (props) => {

return(

<div>

<h2>This is from box</h2>

{props.children}

</div>

)

}

export default Box;

**ขั้นตอนที่ 2 กำหนด คอมโปเน้นท์ย่อยลงใน HOC**

import React from 'react';

import Box from './Box';

const Posts = () => {

const data = [

{id:1, name:'Jeerawuth'},

{id:2, name:'Worrawan'},

{id:3, name:'Thammasak'},

{id:4, name:'Sombat'}

]

const lists = data.map((item) => {

return (

<div key={item.id}>

<span>id: {item.id}</span>

<span>name: {item.name}</span>

</div>

)

});

return (

<Box>

{lists}

</Box>

)

}

export default Posts;

**ขั้นตอนที่ 3 กำหนดแท็ก <Posts /> ไปไว้ที่ App คอมโปเน้นท์**

import React from 'react';

import './App.css';

import Posts from './Posts';

function App() {

return (

<div>

<Posts />

</div>

);

}

export default App;

**วิธีกำหนด HOC ฟังก์ชัน**

**ขั้นตอนที่ 1 สร้างคอมโปเน้นท์ Box ซึ่งจะทำหน้าที่เป็น HOC**

import React from 'react';

const Box = (WrappedComponent, param) => {

return (props) => {

return(

<div>

{param}

<WrappedComponent />

</div>

)

}

}

export default Box;

**ขั้นตอนที่ 2 กำหนด คอมโปเน้นท์ย่อยลงใน HOC**

import React from 'react';

import Box from './Box';

const Posts = () => {

const data = [

{id:1, name:'Jeerawuth'},

{id:2, name:'Worrawan'},

{id:3, name:'Thammasak'},

{id:4, name:'Sombat'}

]

return data.map((item) => {

return (

<div key={item.id}>

<span>id: {item.id}</span> <span>name: {item.name}</span>

</div>

)

});

}

export default Box(Posts, 'My param');