React Redux

> - Single Page 개발
>
> > Single Page 를 app 또는 component 형태로 분리하여 접근하면, 효과적으로 개발가능
>
> - Single Page 개발시 발생한는 문제
>
> > 단, 각 component 와 app 사이의 연동되는 변경이 있을때, html 이 정적인 rendering 방식이므로 상호 반영이 안되는 문제 발생
>
> - 1 차적 해결방식
>
> > 각 component 및 app 이 call 하는 형태 (page include) 에서는 call 하는 page 에서 useState 사용해 <Link > 방식으로 전달 가능했다.
>
> - 1 차적 해결방식의 문제점.
>
> > 문제는 불러지는 자식 component 및 app 또는 형제 관계에 있는 페이지에서는 부모 및 형제 관계의 component 에 state 반영을 하기 어렵다.
>
> - 2 차적 해결방식
>
> > Redux 는 임의의 provider 로 적용된 범위의 component 및 app 들이 상호 반영 할 수 있는 state(store) 를 제공하는 것이다. state 를 역활에 따라 분류한 임의의 slice 를 생성하고, 이때의 Slice 는 reducer 들을 가지고 할당된 state 에 저장된 data 를 manipulation 한다.
>
> - 연습용 예제 작성 요령
>
> > 1. 화면에 보여질 간단한 simple page 를 제작한다.
> >
> > - - 이때 page 를 여러개의 component 및 app 으로 작성한다.
> >
> > 2. 서로 공유할 data 에 대해 slice 작성
> > 3. 임의의 slice 들이 정의된 store 작성
> > 4. store 에 저장된 state 값을 사용할 페이지에서는 useSelector 를 통해 store 에 등록된 각 slice 된 state value 에 접근
> > 5. slice 된 state 에 data 를 저장하기 위한 reducer 에 접근하기 위한 임의의 slice reducer 들을 사용하여, useDispatch 객체를 통해 slice 된 state 에 반영
> > 6. 적용할 component 및 app 에 적용될 범위를 설정하기 위한 Provider 와 store 정의
>
> - dispatch 를 통해 state 값 변경을 등록하는 이유?
>
> > html 화면 rendering 을 다시 하기 위해서 이다.

react app 만들기

```
npx ceate-react-app myApp
```

redux, redux-toolkit 설치

```
cd myApp
# useDispatch, useSelector, Provider 사용하기 위해
npm install react-redux
# createSlice, configureStore 사용하기 위해
npm install @reduxjs/toolkit
```

myApp start

```
npm start
```

사용자 데이터 처리 예제

 # 데이터를 담을 간단한 form page 만들기 (User_form.js)

```
// useState 를 사용하기 위해 import
import {useState} from 'react'
// css import
import '../css/Ex1.css'
// 이후에 구성해볼 userSlice.js import
import { addUser } from '../redux/userSlice';
// index 에서 사용할 Provider 를 위한 useDispatch
import { useDispatch } from 'react-redux'

function Ex1_Form() {

    // 각 데이터를 다시 rendering 할 수 있도록 useState 객체로 저장
    const [name, setName] = useState( "" );
    const [phone, setPhone] = useState( "" );
    const [email, setEmail] = useState( "" );

    // console.log( name, phone, email );

    // 이후에 index.js 에서 다룰 Provider 와 연동하기 위해 useDispatch 객체 생성
    const dispatch = useDispatch();

    return (
        <div className="Ex1_Form">
            <form className='form'>
                <div>
                    {/*
                        값이 변경될때마다 변경된 값을 state 에 저장
                    */}
                    <label >Name</label>
                    <input type={'text'} name='name' id='name' placeholder='Enter
your name'
                            onChange={ (e)=> setName(e.target.value) }
                            />
```

```
                    <label >Phone</label>
                    <input type={'tel'} name='phone' id='phone' placeholder='Enter
your name'
                            onChange={ (e)=> setPhone(e.target.value) }
                            />
                    <label >Email</label>
                    <input type={'email'} name='email' id='email'
placeholder='Enter your name'
                            onChange={ (e)=> setEmail(e.target.value) }
                            />
                    {/*
                        현재 지정된 각 state 및 추가적으로 key 역활을 해줄 num 까지
                        addUser Producer 를 이용해 해당  Slice 로 저장 그리고
dispatch로
                        store 에 저장함.
                    */}
                    <button type='button' onClick={(e) => {
                        dispatch(addUser( {
                            num: 1, name, phone, email
                        }));
                    }}>Add</button>
                </div>
            </form>

        </div>
    );
}

export default Ex1_Form;
```

 # 데이터를 담을 간단한 slice 만들기 (userSlice.js)

```
// redux-tookit 에서 createSlice import
import { createSlice } from '@reduxjs/toolkit'

// userSice 라는 createSlice 객체 생성
const userSlice = createSlice({
    // 일반적인 클래스의 이름정도 생각하면 됨.
    name : 'users',
    // 일반적인 클래스의 맴버 변수로 생각하면 됨.
    // 여러개의 vo 객체들이 들어올수 있으므로 비어있는 array 로 초기와 선언
    initialState : {
        users:[]
    },
    // 일반적인 클래스의 메소드를 생각하면 됨.
    // reducer 안에 각각의 메소드가 추가 되는 방식
    reducers : {
        // addUser 라는 이름의 메소드 생성
        addUser: function( state, action ) {
            console.log( action );
```

```
            // 넘겨받은 각 값들 push( 저장함. ) payload 상태 ( json )로 넘어옴.
            state.users.push( action.payload );
        }
    }
});

// 정의한 메소드 import 가능하게 setting
export const{ addUser }  = userSlice.actions;

export default userSlice.reducer;
```

 # Provider 범위의 컴포넌트들에 지정된 store 사용가능하게 작성 (index.js)

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { Provider } from 'react-redux';
import store from './redux/store'

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    {/*
        # 하단의 Provider Element 를 통해 store 가 지정된 범위내에서 영향을 주도록 한
다.
    */}
    <Provider store={store} >
      <App />
    </Provider>
  </React.StrictMode>
);

// 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();
```

 # 각 slice 를 저장할 store 객체 만들기 (store.js)

```
// redux-tookit 에서 configureStore import
import { configureStore } from "@reduxjs/toolkit"
// 위에서 정의한 slice 및 정의한 reducer import
import userReducer from './userSlice'
```

```
// redux-toolkit 에서 가져온 configureStore 객체 만듦.
const configStore = configureStore({
    // store 에 저장될 reducer 정의
    reducer : {
        user: userReducer
    }
});

export default configStore;
```

 # 상기 정의된 내용들을 토대로 reaction 할 User_res.js 작성

```
import {useState} from 'react'
import '../css/Ex1.css'
// 아래 import 내용 관련해서 메인 주석 참고
import { useSelector } from 'react-redux';

function Ex1_Res()  {
    /*
        # state 는 저장소! 즉, store.js 를 의미하고,
        state.user 는 store.js 에 등록된 리듀서들 중 user 라는 이름으로
        등록된 userSlice.js 를 의미한다.
        그리고 그 뒤의 users 가 바로
        userSlice 에 있는 initialState 안의 명칭중 하나다.
    */
    const list = useSelector( (state) => state.user.users );
    return (
        <div className='Ex1_Res'>
            {/*
                삼항 연산자를 활용하여 list 의 길이가  0 보다 클 때만 반복함.
             */}
            <div className='res' >
             {list.length > 0 ? list.map( data =>
                <div key={data.num}>
                    <h2>User Name : {data.name}</h2>
                    <h2>User Phone : {data.phone}</h2>
                    <h2>User email : {data.email}</h2>
                    <button type='button'> Edit </button>
                    <button type='button'> Delete </button>
                </div>

                ): 'This is empty' }
            </div>

        </div>

    );
}

export default Ex1_Res;
```

 # 마지막 App.js 의 작성내용 (단출함.)

```
import './App.css';
import Ex1_Form from './user/Ex1_Form';
import Ex1_Res from './user/Ex1_Res';

function App() {
  return (
    <div className="App">
      <Ex1_Form />
      <Ex1_Res />
    </div>
  );
}

export default App;
```

숫자 증감 예제

 # 1. 화면 page 만들기 (Inc.js, Dec.js, Number.js)

Dec.js 숫자 감소 버튼 component

```
// ( Dec.js )
// 숫자 감소 버튼 만들기
import {decreaseNumber} from '../redux/NumberSlice'
import { useDispatch} from 'react-redux'

function Dec()  {
    const dispatch = useDispatch();
    return (
        <div className='Dec'>
            <button type='button' onClick={ (e) => {
                dispatch( decreaseNumber() );
            }}>[ - ]</button>
        </div>
    );
}

export default Dec;
```

Inc.js 숫자 증가 버튼 component

```
// ( Inc.js )
// 숫자 증가 버튼 만들기
import {increaseNumber} from '../redux/NumberSlice'
import { useDispatch } from 'react-redux'

function Inc()  {
    const dispatch = useDispatch();
    return (
        <div className='Inc'>
            <button type='button' onClick={ (e) => {
                dispatch( increaseNumber() );
            } }>[ + ]</button>
        </div>

    );
}

export default Inc;
```

Number.js 변경된 숫자 표시 화면 component

```
// ( Number.js )
// 변경된 숫자 표시 화면 만들기
import {useSelector} from 'react-redux';

function Number() {
    const viewValue = useSelector( (state) => state.Number.num );
    return (
        <div className='Number'>
            <input type='text' disabled value={viewValue} />
        </div>
    );
}

export default Number;
```

2. slice 만들기 (NumberSlice.js)

NumberSlice.js

```
// ( NumberSlice.js )
// slice 만들기
import { createSlice } from '@reduxjs/toolkit'
```

```
const numberSlice = createSlice( {
    name : 'numberCounter',
    initialState : {
        num : 0
    },
    reducers:{
        increaseNumber: function( state, action ) {
            state.num += 1;
        },
        decreaseNumber: function( state, action ) {
            state.num -= 1;
        }
    }
});

export const{ increaseNumber, decreaseNumber } = numberSlice.actions;

export default numberSlice.reducer;
```

 # 3. store 만들기 (Store.js)

Store.js

```
// ( Store.js )
// Store 만들기
import { configureStore } from '@reduxjs/toolkit'
import useReducer from './NumberSlice'

const configStore = configureStore({
    reducer : {
        Number : useReducer
    }
});

export default configStore;
```

 # 4. 각 component 에 생성한 slice 및 store 적용

 # 5. index.js 에 반영하기 (index.js)

---

index.js

```
// ( index.js )
// index 만들기
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

import { Provider } from 'react-redux';
import store from './redux/Store';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Provider store={store} >
      <App />
    </Provider>
  </React.StrictMode>
);

// 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();
```

 # 마지막 App.js 의 작성내용 (단촐함.)

---

index.js

```
// ( index.js )
// 각 컴포넌트들이 sibling 으로 나열됨.
// sibling 으로 나열되었음에도,
// index.js 에서 같은 Provider 와 store 의 영향권을 받아 같은 state 를 사용가능함.
import './App.css';
import Dec from './pages/Dec';
import Inc from './pages/Inc';
import Number from './pages/Number';
import './css/number.css';

function App() {
  return (
    <div className="App">
      <Inc />
      <Number />
      <Dec />
    </div>
  );
}
```

```
export default App;
```