React

json data server

```
npm install -g json-server
# 또는
yarn global add json-server
```

db.json

```
{
   "users": [
     { "id": 1, "name": "Alice", "age": 30, "email": "alice@example.com", "city": "New
York" },
     { "id": 2, "name": "Bob", "age": 25, "email": "bob@example.com", "city": "Los Angeles"
},
     { "id": 3, "name": "Charlie", "age": 35, "email": "charlie@example.com", "city":
"Chicago" },
     { "id": 4, "name": "David", "age": 28, "email": "david@example.com", "city": "Houston"
},
     { "id": 5, "name": "Eve", "age": 22, "email": "eve@example.com", "city": "Phoenix" },
     { "id": 6, "name": "Jona", "age": 40, "email": "jona@example.com", "city":
"Philadelphia" },
     { "id": 7, "name": "Kim", "age": 31, "email": "kim@example.com", "city": "San Antonio"
}
   ],
}
```

```
"products": [
     { "id": "A1", "name": "Laptop", "price": 1200, "category": "Electronics", "inStock": true },
     { "id": "B2", "name": "Mouse", "price": 25, "category": "Electronics", "inStock": true },
     { "id": "C3", "name": "Keyboard", "price": 75, "category": "Electronics", "inStock": false },
     { "id": "D4", "name": "Monitor", "price": 300, "category": "Electronics", "inStock": true },
     { "id": "E5", "name": "Tablet", "price": 450, "category": "Electronics", "inStock": true },
     { "id": "F6", "name": "Smartphone", "price": 900, "category": "Electronics", "inStock": true },
     { "id": "G7", "name": "Headphones", "price": 150, "category": "Electronics", "inStock": false },
     { "id": "H8", "name": "T-Shirt", "price": 20, "category": "Apparel", "inStock": true },
     { "id": "I9", "name": "Jeans", "price": 60, "category": "Apparel", "inStock": true },
     { "id": "J10", "name": "Book", "price": 15, "category": "Books", "inStock": true }
   ]
 }
```

package.json

```
{
   "name": "my-json-server-project",
   "version": "1.0.0",
   "description": "",
   "main": "index.js",
   "scripts": {
     "start": "json-server --watch db.json --port 3001"
   },
   "keywords": [],
   "author": "",
   "license": "ISC",
   "dependencies": {
     "json-server": "^0.17.4" // 예시 버전1
   }
 }
```

실행 명령어

json-server --watch db.json --port 3001

라이브러리

npm install @reduxjs/toolkit react-redux
or
yarn add @reduxjs/toolkit react-redux

redux(action, reducer)

=>

redux(slice, asyncThunk)

redux thunk(Count)

npm install redux react-redux redux-thunk

redux/actions.js

```
export const increase = () => (dispatch) => dispatch({type:
"INCREASE"})
export const decrease = () => (dispatch) => dispatch({type:
"DECREASE"})
```

redux/reducers.js

```
const initialState = {
    value: 0
}

export const countReducer = (state=initialState, action) => {
    switch(action.type){
        case "INCREASE":
            return {...state, value: state.value+1}
        case "DECREASE":
            return {...state, value: state.value-1}
        default:
            return state;
    }
}
```

redux/countSlice.js

```
// features/countSlice.js
import { createSlice } from '@reduxjs/toolkit';
const countSlice = createSlice({
 name: 'count',
 initialState: {
   value: 0,
 },
 reducers: {
   increase: (state) => {
     state.value += 1;
   },
   decrease: (state) => {
     state.value -= 1;
   },},});
export const { increase, decrease } = countSlice.actions;
export default countSlice.reducer;
```

redux/index.js

```
import { combineReducers } from "redux";
import count from "./countSlice";


const rootReducers = combineReducers({
   count,
})

export default rootReducers;
```

redux/store.js

```
import { applyMiddleware, legacy_createStore } from "redux";
import rootReducers from ".";
import { thunk } from "redux-thunk";

const store = legacy_createStore(rootReducers, applyMiddleware(thunk));

export default store;
```

index.js

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

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

components/Count.jsx

```
import React from 'react'
import { useDispatch, useSelector } from 'react-redux';
import { increase, decrease } from '../redux/countSlice';

const Count = () => {
   const state = useSelector((state) => state.count)
   const dispatch = useDispatch();
 return (
   <>
       <p>
           Count: <b>{state.value}</b>
       </p>
       <button onClick={()=>dispatch(increase())}>+</button >
       <button onClick={()=>dispatch(decrease())}>-</button>
   </>)}
export default Count
```

redux thunk (name, nickname)

redux/actions.js

```
export const changeInput = (name, value) => {
 return (dispatch) => {
   dispatch({
     type: 'CHANGE_INPUT',
     payload: { name, value }
   });
 };
};
```

redux/reducers.js

```
// reducers/nicknameReducer.js
const nicknameInitialState = {
 name: '',
 nickname: ''
};

export const nicknameReducer = (state = nicknameInitialState,
action) => {
 switch (action.type) {
   case 'CHANGE_INPUT':
     return {...state,
       [action.payload.name]: action.payload.value
     };
   default:
     return state;}};
```

redux/nicknameSlice.js

```
// features/nicknameSlice.js
import { createSlice } from '@reduxjs/toolkit';
const nicknameSlice = createSlice({
 name: 'nickname',
 initialState: {
   name: '',
   nickname: ''
 },
 reducers: {
   changeInput: (state, action) => {
     const { name, value } = action.payload;
     state[name] = value;
   }}});
export const { changeInput } = nicknameSlice.actions;
export default nicknameSlice.reducer;
```

redux/index.js

```
import { combineReducers } from "redux";
import nickname from "./nicknameSlice";

const rootReducers = combineReducers({
   nickname,
})
export default rootReducers;
```

redux/store.js

```
import { applyMiddleware, legacy_createStore } from "redux";
import rootReducers from ".";
import { thunk } from "redux-thunk";

const store = legacy_createStore(rootReducers, applyMiddleware(thunk));

export default store;
```

index.js

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

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

components/Nickname.jsx

```
// components/Nickname.js
import React, { useCallback } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { changeInput } from '../redux/nicknameSlice';

const Nickname = () => {
 const { name, nickname} = useSelector((state) => state.nickname);
 const dispatch = useDispatch();

 const handleOnChange = useCallback((e) => {
   const { name, value } = e.target;
   dispatch(changeInput({name, value}));
 }, [dispatch]);
```

```
return (
    <>
      <div>
        <input type="text" name="name" value={name} onChange={handleOnChange} />
      </div>
      <div>
        <input type="text" name="nickname" value={nickname} onChange={handleOnChange} />
      </div>
      <div>
        <b>이름: </b> {name}
      </div>
      <div>
        <b>닉네임: </b> {nickname}
      </div>
    </>);};
export default Nickname;
```

redux thunk (userList)

redux/actions.js

```
import axios from 'axios';
export const fetchUsers = () => async (dispatch) => {
 dispatch({ type: 'FETCH_USERS_REQUEST' });

 try {
   const response = await axios.get('http://localhost:3001/users');
   dispatch({ type: 'FETCH_USERS_SUCCESS', payload: response.data });
 } catch (error) {
   dispatch({
     type: 'FETCH_USERS_FAILURE',
     error: error.message || 'Something went wrong',
   });
 }
};
```

redux/reducers.js

```
const userInitialState = {
  loading: false,
  data: [],
  error: null,
};
 export const userReducer = (state = userInitialState, action) => {
  switch (action.type) {
    case 'FETCH_USERS_REQUEST':
      return { ...state, loading: true, error: null };
    case 'FETCH_USERS_SUCCESS':
      return { ...state, loading: false, data: action.payload };
    case 'FETCH_USERS_FAILURE':
      return { ...state, loading: false, error: action.error };
    default:
      return state;
  }};
```

redux/userSlice.js

```
import axios from 'axios';
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';

export const fetchUsers = createAsyncThunk(
 'users/fetchUsers',
 async (_, thunkAPI) => {
   try {
     const response = await axios.get('http://localhost:3001/users');
     return response.data;
   } catch (error) {
     return thunkAPI.rejectWithValue(error.message);
   }
 }
);
```

```
const userSlice = createSlice({
 name: 'users',
 initialState: {
   loading: false,
   data: [],
   error: null,
 },
 reducers: {},
 extraReducers: (builder) => {
   builder
     .addCase(fetchUsers.pending, (state) => {
       state.loading = true;
       state.error = null;
     })
```

```
      .addCase(fetchUsers.fulfilled, (state, action) => {
        state.loading = false;
        state.data = action.payload;
      })
      .addCase(fetchUsers.rejected, (state, action) => {
        state.loading = false;
        state.error = action.payload || 'Something went wrong';
      });
  },
});

export default userSlice.reducer;
```

redux/index.js

```
import { combineReducers } from "redux";
import users from "./userSlice";

const rootReducers = combineReducers({
   users,
})
export default rootReducers;
```

redux/store.js

```
import { applyMiddleware, legacy_createStore } from "redux";
import rootReducers from ".";
import { thunk } from "redux-thunk";

const store = legacy_createStore(rootReducers, applyMiddleware(thunk));

export default store;
```

index.js

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

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

UserList.jsx

```
import React, { useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { fetchUsers } from '../redux/actions/userActions';

const UserList = () => {
 const dispatch = useDispatch();
 const { loading, data: users, error } = useSelector((state) => state.users);

 useEffect(() => {
   dispatch(fetchUsers());
 }, [dispatch]);
```

```
return (
   <div>
     <h1>사용자 목록</h1>
     {loading && <p>사용자 데이터를 불러오는 중...</p>}
     {error && <p>에러 발생: {error}</p>}
     {!loading && users.length === 0 && <p>사용자 데이터가 없습니다.</p>}
     <ul>
       {users.map((user) => (
         <li key={user.id}>
           <strong>{user.name}</strong> ({user.age}세, {user.city}) - {user.email}
         </li>
       ))}
     </ul>
   </div>);};
export default UserList;
```