Redux Toolkit

Redux는 SPA(Single Page Application)을 위하 Front Framework에서 가장 많이 사용하는 글로벌 상태 관리 라이브러이다. 그러나 복잡한 구조 및 외부 라이브러리와의 연동, 보일러플레이트 코드 (action, reducer 등) 반복으로 Redux 공식 Homepage에서도 redux toolkit을 사용하기를 권장한다.

redux toolkit은 복잡한 Redux를 해결하기 위한 Redux를 위한 공식 툴킷이다.

- configureStore()
  - Redux 스토어를 설정을 위한 함수
  - Redux 스토어를 설정할 때 반복되는 설정을 자동으로 처리해줘서, 더 적은 코드로 스토어를 구성할 수 있다.
- createSlice()
  - Slice 를 생성하는 함수
  - Redux 스토어를 위한 액션 타입, 액션 생성자, 리듀서를 선언한다.
  - 리듀서만 선언하면 리듀서를 기반으로 action 함수를 자동으로 제공한다.

1. 설치하기

```
npm install @reduxjs/toolkit react-redux
```

2. Store 정의하기

store> index나 store 파일을 한 개만 작성한다.

2.1 slice 구현하기

createSlice({name, initialState, reducers}) 함수를 이용해서 구현한다.

```
const slice = createSlice({
  name: slice이름,
  initialState: state초기값,
  reducers: {
    // 리듀서구현
  },
});
```

구현한 reducer와 자동 생성해주는 action함수를 export한다.

2.2 store 만들기

configureStore({상태정의})

```
const store = configureStore({
  reducer: {리듀서 map},
});
```

```
export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

- RootState : Redux 스토어의 state를 나타내는 타입
- AppDispath : Redux 액션을 dispatch하는 함수의 타입

store/index.ts

```
import { configureStore } from '@reduxjs/toolkit';

const store = configureStore({
  reducer: {},
});

export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;

export default store;
```

slice를 통해 구현한 Reducer를 import 해서 정의한다.

2.3 useSelector()로 state 사용

2.4 useDispatch()로 action 실행

2.5 useSelector(), useDispatch()를 편히 사용하기 위해 사용자 hook 만들어 재사용하기

```
import { TypedUseSelectorHook, useDispatch, useSelector } from "react-redux";
import type { RootState, AppDispatch } from "./index";

export const useAppDispatch = () => useDispatch<AppDispatch>();
export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector;
```

2.7 hook 을 이용해서 사용하기

const 변수명 = useAppSelector((state) => state.slicename.상태이름)

const dispatch = useAppDispatch();

dispatch(액션함수명())

3. Provider 적용하기

react-redux api에서 제공하는 Provider를 redux를 공유할 위치에 작성한다.

```
import { Provider } from "react-redux";
import { store } from "@/store/index";
```

```
<Provider store={store}>
  <div>
    <SelectColors />
    <ColorBox />
  </div>
</Provider>
```

4. custom hook을 만들어서 사용하기.

상태를 자주 사용할 경우 좀 더 쉽게 사용하기 위해 2.6 코드만 작성한 custom hook을 만든다.

```
const useColorHooks = () => {
  const color = useAppSelector((state) => state.color.color);
  const subColor = useAppSelector((state) => state.color.subcolor);
  const dispatch = useDispatch();
```

5. redux-persist 적용하기

Redux의 상태(state)를 브라우저의 저장소(localStorage나 sessionStorage 등)에 저장했다가, 페이지를 새로 고침하거나 앱을 다시 시작해도 이전 상태를 복원해주는 라이브러리이다.

■ persistReducer

Redux의 리듀서를 "저장 가능한" 리듀서로 변경하는 요소

```
const persistedReducer = persistReducer(persistConfig, rootReducer);
```

◆ persistConfig { key, storage, whitelist }

key : storage에 저장될 key

storage : redux의 상태를 저장할 storage. Default는 localStorage이다.

whitelist: storage에 저장될

```
const rootPersistConfig = {
  key: "root",
  storage: storageSession,
  whitelist: ["member", "bookmark"],
};
```

◆ rootReducer

persistReducer로 관리할 대상

```
const rootReducer = combineReducers({
  member: memberReducer,
  bookmark: bookmarkReducer,
});
```

■ persistStore 스토어를 persist(지속) 가능한 상태

```
export const store = configureStore({
  reducer: persistedReducer,
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware({
      serializableCheck: {
        ignoredActions: [FLUSH, REHYDRATE, PAUSE, PERSIST, PURGE, REGISTER],
      },
    }),
});
```

```
const persistor = persistStore(store);
```

◆ 미들웨어

- 액션과 리듀서 사이에서 동작하는 중간 처리기
- Redux의 미들웨어는 액션이 리듀서에 도달하기 전에 가로채서,
  그 액션을 기록하거나, 수정하거나, 비동기 작업을 처리
- serializableCheck
  - ➢ 상태와 액션이 JSON처럼 직렬화 가능한지 검사하는 기능
  - ➢ redux-persist가 내부에서 사용하는 액션(REHYDRATE, PERSIST 등)은 직렬화되지 않기 때문에 serializableCheck 설정을 하지 않으면 실행시 경고가 발생하므로 ignoredActions 옵션을 설정한다.

■ PersistGate 설정하기

PersistGate는 저장된 상태가 복원되기 전까지 로딩 화면을 보여주고, 복원되면 자식 컴포넌트를 렌더링 한다.

```
<Provider store={store}>
  <PersistGate loading={<p>로딩 중...</p>} persistor={persistor}>
    <App />
  </PersistGate>
</Provider>
```