AssetData 및 AssetGroup 컴포넌트 구조 보고서

1. 개요

`AssetData`, `AssetGroup` 컴포넌트와 `asset` 폴더 내 세부 컴포넌트들은 자산군 관리 UI의 핵심 구조를 담당합니다. 자산의 추가/삭제, 자산군/종류/비중/체크박스 등 세부 입력을 모듈화하여, 유지보수성과 확장성을 높인 구조입니다.

2. 주요 컴포넌트 역할

2.1. AssetData

- 자산 단위의 최상위 컴포넌트
- 자산의 제목, 세부 입력(AssetGroup), 삭제/추가 버튼 UI 제공
- zustand store의 addAsset, deleteAsset 액션을 통해 자산 리스트를 동적으로 관리
- 마지막 자산일 때만 "자산 추가" 버튼 노출

2.2. AssetGroup

- 자산의 세부 입력 영역을 묶는 래퍼 컴포넌트
- AssetType(종류), AssetSearchBox(자산군), AssetWeight(비중) 등 세부 입력을 한 줄로 배치
- AssetCheckBox(체크박스) 컴포넌트와 함께 자산별 추가 옵션 제공

2.3. asset 폴더 내 세부 컴포넌트

- AssetType: 자산 종류 선택 UI
- AssetSearchBox: 10만 개 이상의 자산군 검색/선택(react-window windowing 적용)
- AssetWeight: 자산 비중 입력 및 유효성 검사
- AssetCheckBox: 자산별 체크박스 옵션
- AssetManagement: 전체 자산 리스트 관리 및 렌더링

3. 구조적 장점

- **모듈화**: 각 입력/옵션을 독립 컴포넌트로 분리하여, 유지보수와 확장에 용이
- **동적 관리**: zustand store와 연동하여 자산의 추가/삭제/수정이 즉각적으로 반영
- **대용량 대응**: AssetSearchBox에서 windowing 기법으로 대용량 자산군도 빠르게 검색/선택 가능
- **UI/비즈니스 로직 분리**: UI와 상태 관리 로직이 명확히 분리되어 실무적 협업에 유리

4. 결론

AssetData, AssetGroup, asset 폴더 내 세부 컴포넌트들은 자산군 관리 UI를 모듈화·동적·대용량 대응 구조로 설계하여, 실무에서 요구되는 유지보수성, 확장성, 성능을 모두 만족시킵니다. 각 컴포넌트의 역할 분담과 zustand 기반 상태관리로, 자산군 관리의 모든 요구사항을 효과적으로 처리할 수 있습니다.