. 2025/10/30: Loading 시 확장을 하려고 하면 Treecontrol으 생성해야 한다 정도로 일단 이해

네, **Angular Material**의 \*\*mat-tree\*\*와 \*\*TreeControl\*\*을 사용하여 확장/축소(Expand/Collapse) 기능이 있는 트리 구조를 구현하는 샘플 코드를 만들어 드리겠습니다.

이 예시에서는 **Flat Tree**를 사용하며, 데이터를 트리 구조로 변환하고 TreeControl을 이용해 상태를 관리합니다.

**💻 Angular Material Tree 샘플 코드 (Flat Tree)**

이 코드를 사용하려면 먼저 Angular 프로젝트에 **Angular Material**과 **CDK** 모듈이 설치되어 있어야 합니다.

**1. 데이터 구조 및 노드 변환 (component.ts)**

트리 데이터를 정의하고, mat-tree가 필요로 하는 평탄화된(Flat) 노드 형태로 변환하는 로직입니다.

TypeScript

// app.component.ts

import { Component } from '@angular/core';

import { FlatTreeControl } from '@angular/cdk/tree';

import { MatTreeFlatDataSource, MatTreeFlattener } from '@angular/material/tree';

/\*\* 트리의 계층적 데이터 인터페이스 \*/

interface FoodNode {

name: string;

children?: FoodNode[];

}

const TREE\_DATA: FoodNode[] = [

{

name: '과일',

children: [{ name: '사과' }, { name: '바나나' }, { name: '딸기' }],

},

{

name: '채소',

children: [

{

name: '뿌리 채소',

children: [{ name: '당근' }, { name: '감자' }],

},

{

name: '잎 채소',

children: [{ name: '상추' }, { name: '시금치' }],

},

],

},

];

/\*\* Flat Tree를 위한 평탄화된 노드 인터페이스 \*/

interface ExampleFlatNode {

expandable: boolean; // 확장 가능 여부 (자식이 있는지)

name: string;

level: number; // 깊이 (레벨)

}

@Component({

selector: 'app-root',

templateUrl: 'app.component.html',

styleUrls: ['app.component.css'],

})

export class AppComponent {

// 1. MatTreeFlattener: 계층적 노드를 Flat 노드로 변환합니다.

private \_transformer = (node: FoodNode, level: number) => {

return {

expandable: !!node.children && node.children.length > 0,

name: node.name,

level: level,

};

};

// 2. FlatTreeControl: 확장/축소 상태와 레벨을 관리합니다.

treeControl = new FlatTreeControl<ExampleFlatNode>(

(node) => node.level, // 레벨 getter

(node) => node.expandable // 확장 가능 여부 getter

);

// 3. MatTreeFlatDataSource: FlatTreeFlattener와 TreeControl을 사용하여 데이터를 제공합니다.

dataSource = new MatTreeFlatDataSource(

this.treeControl,

new MatTreeFlattener(

this.\_transformer,

(node) => node.level, // getLevel 함수

(node) => node.expandable, // isExpandable 함수

(node) => node.children // getChildren 함수

)

);

constructor() {

this.dataSource.data = TREE\_DATA;

}

// 노드가 확장 가능한지 확인하는 헬퍼 함수

hasChild = (\_: number, node: ExampleFlatNode) => node.expandable;

}

**2. Angular 템플릿 (component.html)**

mat-tree와 treeControl을 사용하여 트리를 렌더링하고 확장/축소 버튼을 바인딩합니다.

HTML

<mat-tree [dataSource]="dataSource" [treeControl]="treeControl">

<mat-tree-node \*matTreeNodeDef="let node" matTreeNodePadding>

<button mat-icon-button disabled></button>

{{node.name}}

</mat-tree-node>

<mat-tree-node \*matTreeNodeDef="let node; when: hasChild" matTreeNodePadding>

<button mat-icon-button matTreeNodeToggle [attr.aria-label]="'Toggle ' + node.name">

<mat-icon class="mat-icon-rtl-mirror">

{{ treeControl.isExpanded(node) ? 'expand\_more' : 'chevron\_right' }}

</mat-icon>

</button>

{{node.name}}

</mat-tree-node>

</mat-tree>

**✨ 핵심 동작 설명**

* **데이터 바인딩:**
  + MatTreeFlattener는 중첩된 TREE\_DATA를 Angular가 렌더링하기 쉬운 \*\*ExampleFlatNode\*\*의 평탄한 배열로 변환합니다. 이때 level과 expandable 정보가 추가됩니다.
  + MatTreeFlatDataSource가 이 평탄화된 데이터를 mat-tree에 제공합니다.
* **확장/축소 (TreeControl):**
  + FlatTreeControl은 각 노드의 level과 expandable 속성을 사용하여 트리의 구조와 상태를 파악합니다.
  + <button matTreeNodeToggle>에 (click)="treeControl.toggle(node)" (Material이 내부적으로 처리)를 바인딩하여 노드를 클릭할 때 TreeControl의 상태를 토글합니다.
  + treeControl.isExpanded(node)를 사용하여 현재 노드의 확장 상태를 확인하고, 이에 따라 아이콘(expand\_more 또는 chevron\_right)을 변경합니다.