DynamicValidationService

August 17, 2026

DynamicValidationService

Kind: Class

Source: `Frontend/src/app/services/configurations/dynamic-validation/dynamic-validation.service.ts` (line 86)

Part of: Frontend

`DynamicValidationService` manages configuration data for dynamic validation in the frontend. It retrieves lookup modules, submodules, and properties, then saves and loads mandatory-field settings for a selected module.

Methods

Method	Signature	Returns
`getlookupModules`	`getlookupModules()`	`void`
`getlookupSubModules`	`getlookupSubModules(id: undefined)`	`void`
`getProperties`	`getProperties(id: undefined)`	`void`
`saveMandatoryFields`	`saveMandatoryFields(list: undefined)`	`void`
`getMandatoryFieldsByModule`	`getMandatoryFieldsByModule(moduleId: number)`	`void`

Diagram

```
graph LR
  UI[Configuration UI] --> Service[DynamicValidationService]
  Service --> Modules[getlookupModules]
  Service --> SubModules[getlookupSubModules]
  Service --> Properties[getProperties]
  Service --> Save[saveMandatoryFields]
  Service --> Load[getMandatoryFieldsByModule]
  Modules --> API[Configuration API]
  SubModules --> API
  Properties --> API
```

```
  Save --> API
  Load --> API
```

Usage

```
import { Component, OnInit } from '@angular/core';
import { DynamicValidationService } from './dynamic-validation.service';

@Component({
  selector: 'app-mandatory-fields',
  template: `...`,
})
export class MandatoryFieldsComponent implements OnInit {
  modules: unknown[] = [];
  mandatoryFields: unknown;

  constructor(
    private readonly dynamicValidationService: DynamicValidationService,
  ) {}

  ngOnInit(): void {
    this.dynamicValidationService.getlookupModules().subscribe((modules) => {
      this.modules = modules;
    });
  }

  loadMandatoryFields(moduleId: string): void {
    this.dynamicValidationService
      .getMandatoryFieldsByModule(moduleId)
      .subscribe((fields) => {
        this.mandatoryFields = fields;
      });
  }

  saveMandatoryFields(fields: unknown): void {
    this.dynamicValidationService.saveMandatoryFields(fields).subscribe(() => {
      this.loadMandatoryFields('selected-module-id');
    });
  }
}
```

AI Coding Instructions

- Inject `DynamicValidationService` through the Angular constructor rather than creating it directly.
- Load modules, submodules, and properties before presenting dependent configuration controls.

- Match the request payload passed to `saveMandatoryFields()` with the backend validation-configuration contract.
- Refresh data with `getMandatoryFieldsByModule()` after saving so the UI reflects persisted settings.
- Handle subscription errors and loading states in the consuming component or state layer.

Used by

9 references from 9 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (9)

- `MrqDetailsComponent` — `Frontend/src/app/components/MRQ/mrq-details/mrq-details.component.ts` :1
- `BillOfMaterialsDetailsComponent` — `Frontend/src/app/components/bill-of-materials/bill-of-materials-details/bill-of-materials-details.component.ts` :1
- `DynamicValidationComponent` — `Frontend/src/app/components/configuration/dynamic-validation/dynamic-validation.component.ts` :1
- `LettersOfGuarantiesDetailsComponent` — `Frontend/src/app/components/letters-of-guaranties/letters-of-guaranties-details/letters-of-guaranties-details.component.ts` :1
- `AccountDetailsComponent` — `Frontend/src/app/components/market/account-details/account-details.component.ts` :1
- `ProcurementDetailsComponent` — `Frontend/src/app/components/procurement/procurement-details/procurement-details.component.ts` :1
- `PurchasingDetailsComponent` — `Frontend/src/app/components/purchasing/purchasing-details/purchasing-details.component.ts` :1
- `NewJobComponent` — `Frontend/src/app/components/sales/new-job/new-job.component.ts` :1

…and 1 more.