DashboardWidget

August 17, 2026

DashboardWidget

Kind: Class

Source: `Pams/Web/Pams.WebApp/Client/app/models/dashboard/dashboard-widget.model.ts` (line 11)

Part of: Pams

`DashboardWidget` manages the state and data transformations for a dashboard widget. It synchronizes labels, responds to data-source and color changes, builds schemas, combines selections, and refreshes chart and pie data.

Methods

Method	Signature	Returns
`syncLabels`	`syncLabels(source: Array<string>)`	`void`
`DataSourceChanged`	`DataSourceChanged()`	`void`
`ChangeColor`	`ChangeColor(serial: string, value: string)`	`void`
`BuildSchema`	`BuildSchema()`	`void`
`CombaineSelection`	`CombaineSelection()`	`void`
`updateCharts`	`updateCharts()`	`void`
`createPieData`	`createPieData(dataSource: undefined, chart: undefined)`	`void`

Properties

Property	Type
`ID`	`number`
`PrimaryID`	`number`
`DisplayID`	`string`

Property	Type
`Caption`	`string`
`DatasourceInit`	`any[]`
`Datasource`	`any`
`OrginalDatasource`	`any`
`MasterFilter`	`boolean`
`AcceptFilter`	`any[]`
`charts`	`any`
`FilterString`	`string`
`FilterSelectionType`	`string`
`ChartSeriseSelectionType`	`string`
`CurrentData`	`any`
`table`	`Table`
`view`	`View`
`isTable`	`boolean`
`Operations`	`any[]`
`DataOperations`	`DashboardDataFields[]`
`IsSelected`	`boolean`
`isNested`	`boolean`
`Stacked`	`boolean`
`ExpBar`	`boolean`
`selectedArgument`	`Array<DimensionField>`
`selectedSerise`	`Array<DimensionField>`
`AutoColoredMeasure`	`boolean`
`ColorschemaList`	`Array<schema>`
`colorCounter`	`number`
`colorPlatte`	`any`
`Item`	`GridStackItem`
`WidgetType`	`DashboardWidgetTypeEnum`

Property	Type
`GridStackMainWidth`	`number`
`ColorContext`	`any`
`ColorBehavior`	`boolean`
`widgetLabels`	`Array<{label:string ,priority:number}>`

Where it refuses work

- `DashboardWidget` stops the work with an early return when `inx == all_Fields.length`.

Diagram

```
graph LR
  DataSource[Data Source] --> Changed[DataSourceChanged]
  Changed --> Schema[BuildSchema]
  Schema --> Selection[CombaineSelection]
  Selection --> Charts[updateCharts]
  Charts --> PieData[createPieData]
  Labels[syncLabels] --> Charts
  Color[ChangeColor] --> Charts
```

Usage

```
import { DashboardWidget } from './models/dashboard/dashboard-widget.model';

const widget = new DashboardWidget();

widget.DataSourceChanged();
widget.syncLabels();
widget.ChangeColor();
widget.BuildSchema();
widget.CombaineSelection();
widget.updateCharts();
widget.createPieData();
```

AI Coding Instructions

- Keep chart refresh logic within `updateCharts()` so data updates follow a single path.
- Call `BuildSchema()` after a data-source change when widget fields or chart configuration must be rebuilt.

- Keep label updates in `syncLabels()` rather than changing chart labels directly in calling code.
- Preserve the existing `CombaineSelection()` method name when calling or modifying the class.

Relationships

- IMPORTS → `DashboardWidgetTypeEnum`
- IMPORTS → `Table`
- IMPORTS → `View`
- IMPORTS → `platte`
- IMPORTS → `Query`
- IMPORTS → `DashboardDataFields`
- IMPORTS → `DataTypeEnum`
- IMPORTS → `OperationTypeEnum`
- IMPORTS → `QueryTypeEnum`
- IMPORTS → `chartType`
- IMPORTS → `DimensionField`
- IMPORTS → `DateGroupEnum`

Used by

1 reference from 1 file. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (1)

- `DashboardGroup` — `Pams/Web/Pams.WebApp/Client/app/models/dashboard/dashboard-group.model.ts`:2