WidgetBacklogOverTheMonthsComponent

August 18, 2026

WidgetBacklogOverTheMonthsComponent

Kind: Class

Source: `Pams/Web/Pams.Web/Client/app/components/dashboard/dashboard-libary/widgets/widget-backlog-over-the-months/widget-backlog-over-the-months.component.ts` (line 45)

Part of: Pams

`WidgetBacklogOverTheMonthsComponent` is an Angular dashboard widget that prepares and displays backlog data across monthly ranges. It loads dashboard and team lookup data, manages lifecycle updates, calculates performance percentages, and translates month labels for presentation.

Implements: `OnInit`

Methods

Method	Signature	Returns
`ngOnInit`	`ngOnInit()`	`void`
`ngDoCheck`	`ngDoCheck()`	`void`
`ngOnChanges`	`ngOnChanges()`	`void`
`ngOnDestroy`	`ngOnDestroy()`	`void`
`getDashboardData`	`getDashboardData()`	`void`
`initalizingData`	`initalizingData(value: BranchData[])`	`void`
`getTeamLookup`	`getTeamLookup()`	`void`
`setSeriseRange`	`setSeriseRange()`	`any[]`
`calculatePerformancePer`	`calculatePerformancePer(month: BacklogData)`	`number`
`translateMonthName`	`translateMonthName(month: BacklogData)`	`string`
`getpriceFormatted`	`getpriceFormatted(x: undefined)`	`string`

Method	Signature	Returns
`onResize`	`onResize()`	`void`
`setChaching`	`setChaching()`	`void`
`getChaching`	`getChaching()`	`void`
`onValueChanged`	`onValueChanged(e: undefined)`	`void`
`onChartInit`	`onChartInit(e: any, index: number)`	`void`
`switchChartTypeHandler`	`switchChartTypeHandler()`	`void`
`switchBasedOnTendencyHandler`	`switchBasedOnTendencyHandler()`	`void`
`setValue`	`setValue(value: number)`	`void`
`setForecastValue`	`setForecastValue(value: number)`	`void`
`setMarginAvarage`	`setMarginAvarage(value: number)`	`void`
`setMarginAvarageColor`	`setMarginAvarageColor(value: number)`	`void`
`isForecast`	`isForecast(month: BacklogData)`	`boolean`
`isAchievedOrCurrent`	`isAchievedOrCurrent(month: BacklogData)`	`boolean`
`getpriceForecastAmount`	`getpriceForecastAmount(month: BacklogData)`	`number`
`expectedCurrencyChange`	`expectedCurrencyChange(e: undefined)`	`void`

Properties

Property	Type
`triangleNeedle`	`DxCircularGaugeComponent`
`originalWidget`	`Widget`
`_widget`	`Widget`
`_widgetType`	`string`
`withoutOptionBar`	`boolean`
`bookingOrSales`	`number`
`withNominal`	`boolean`
`selectedTeam`	`number`
`forecastRuleId`	`number`
`forecastRules`	`any[]`

Property	Type
`selectedYear`	`number`
`numYearsDatasource`	`any[]`
`todayWidgetsEnum`	`any`
`cachingWidgetData`	`CachingWidgetData`
`data`	`BranchData[]`
`originalData`	`BranchData[]`
`singlePrincipal`	`BacklogData`
`singleBacklogMonth`	`MonthData`
`singleBranch`	`BranchData`
`branchCurrency`	`any`
`basedOnTendency`	`boolean`
`subscription`	`Subscription`
`SeriseRange`	`any[]`
`initOpts`	`any`
`headerData`	`any`
`echartsIntance`	`any`
`TargetBooking`	`string
`AchievedBooking`	`string
`teamsLookup`	`any[]`
`value`	`BranchData[]`

Where it refuses work

- `WidgetBacklogOverTheMonthsComponent` stops the work with an early return when `month.MonthId == 12`.
- `WidgetBacklogOverTheMonthsComponent` stops the work with an early return when `month.MonthId >= new Date().getMonth() + 1`.
- `WidgetBacklogOverTheMonthsComponent` stops the work with an early return when `month.MonthId <= new Date().getMonth() + 1`.

Diagram

```
graph LR
  A[Angular lifecycle] --> B[WidgetBacklogOverTheMonthsComponent]
  B --> C[getDashboardData]
  B --> D[getTeamLookup]
  C --> E[initalizingData]
  E --> F[setSeriseRange]
  E --> G[calculatePerformancePer]
  F --> H[translateMonthName]
  G --> I[Dashboard chart data]
  H --> I
```

Usage

```
import { WidgetBacklogOverTheMonthsComponent } from './widget-backlog-over-the-months.component

function refreshBacklogWidget(
  widget: WidgetBacklogOverTheMonthsComponent,
): void {
  widget.getDashboardData();

  const seriesRange = widget.setSeriseRange();
  const performancePercentage = widget.calculatePerformancePer();
  const monthLabel = widget.translateMonthName();

  console.log({ seriesRange, performancePercentage, monthLabel });
}

// Angular calls ngOnInit, ngOnChanges, ngDoCheck, and ngOnDestroy
// as part of the component lifecycle.
```

AI Coding Instructions

- Keep data initialization inside `initalizingData()` so dashboard values are prepared consistently after data retrieval.
- Call `getDashboardData()` when widget inputs or dashboard context changes rather than duplicating data-loading logic.
- Preserve the existing `setSeriseRange()` return shape because chart bindings may depend on its series structure.
- Use `translateMonthName()` for displayed month labels instead of formatting month values directly in templates.

- Release subscriptions, timers, or other resources in `ngOnDestroy()` when adding asynchronous behavior.

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)

- `TranslateHandler` — `Pams/Web/Pams.Web/Client/app/app.module.ts`:1