WidgetTeamsTendancyComponent

August 18, 2026

WidgetTeamsTendancyComponent

Kind: Class

Source: `Frontend/src/app/components/dashboard/dashboard-libary/widgets/widget-teams-tendancy/widget-teams-tendancy.component.ts` (line 18)

Part of: Frontend

`WidgetTeamsTendancyComponent` is an Angular dashboard widget that retrieves and displays team tendency data. It manages feature-subscription checks, chart initialization and type switching, cache handling, value updates, and resize behavior within the dashboard widget lifecycle.

Extends: `BaseWidgetComponent`

Implements: `OnInit` , `DoCheck`

Methods

Method	Signature	Returns
`getDashboardData`	`getDashboardData()`	`void`
`checkFeatureSubscription`	`checkFeatureSubscription()`	`boolean`
`onResize`	`onResize()`	`void`
`setChaching`	`setChaching()`	`void`
`getChaching`	`getChaching()`	`void`
`onValueChanged`	`onValueChanged(e: undefined)`	`void`
`ngOnInit`	`ngOnInit()`	`void`
`ngDoCheck`	`ngDoCheck()`	`void`
`onChartInit`	`onChartInit(e: any, index: number)`	`void`
`switchChartTypeHandler`	`switchChartTypeHandler()`	`void`

Method	Signature	Returns
`switchBasedOnTendencyHandler`	`switchBasedOnTendencyHandler()`	`void`
`setValue`	`setValue(value: number)`	`void`

Properties

Property	Type
`triangleNeedle`	`DxCircularGaugeComponent`
`originalWidget`	`Widget`
`_widget`	`Widget`
`numYears`	`number`
`numYearsDatasource`	`any[]`
`todayWidgetsEnum`	`any`
`cachingWidgetData`	`CachingWidgetData`
`teams`	`TeamsTendancy[]`
`withNominal`	`boolean`
`basedOnTendency`	`boolean`
`initOpts`	`any`
`series`	`any[]`
`headerData`	`any`
`echartsIntance`	`any`
`options`	`any`
`TargetBooking`	`string
`AchievedBooking`	`string
`currenFeature`	`number`

Where it refuses work

- `WidgetTeamsTendancyComponent` stops the work with an early return when `this.dashboardLibraryService.checkFeatureSubscription(this.widget.chartEnumId)`.

- `WidgetTeamsTendancyComponent` stops the work with an early return when `!Number.isNaN(value)`.

Diagram

```
graph LR
  Dashboard[Dashboard Container] --> Widget[WidgetTeamsTendancyComponent]
  Widget --> Subscription[checkFeatureSubscription]
  Subscription --> Data[getDashboardData]
  Widget --> Cache[getChaching / setChaching]
  Data --> Chart[onChartInit]
  Widget --> Values[onValueChanged]
  Widget --> ChartType[switchChartTypeHandler]
  Widget --> Resize[onResize]
  Widget --> Lifecycle[ngOnInit / ngDoCheck]
```

Usage

```
import { Component, ViewChild } from '@angular/core';
import { WidgetTeamsTendancyComponent } from './widget-teams-tendancy.component';

@Component({
  selector: 'app-dashboard-page',
  template: `
    <app-widget-teams-tendancy #teamsTendancyWidget />
    <button type="button" (click)="refreshTeamsTendency()">
      Refresh
    </button>
  `
})
export class DashboardPageComponent {
  @ViewChild('teamsTendancyWidget')
  teamsTendancyWidget?: WidgetTeamsTendancyComponent;

  refreshTeamsTendency(): void {
    const widget = this.teamsTendancyWidget;

    if (widget?.checkFeatureSubscription()) {
      widget.getDashboardData();
    }
  }

  changeChartType(): void {
    this.teamsTendancyWidget?.switchChartTypeHandler();
  }
}
```

AI Coding Instructions

- Keep Angular lifecycle handling in `ngOnInit` and `ngDoCheck`; do not call lifecycle hooks directly from application code.
- Check `checkFeatureSubscription()` before triggering data retrieval or exposing widget actions that require a subscribed feature.
- Keep cache reads and writes paired through `getChaching()` and `setChaching()` rather than adding separate storage behavior.
- Route chart-related updates through `onChartInit()`, `onValueChanged()`, and `switchChartTypeHandler()` so chart state remains in the component.
- Preserve resize handling through `onResize()` when changing the widget container or chart integration.

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)

- `DashboardWidgetsModule` — `Frontend/src/app/components/dashboard/dashboard-widgets.module.ts`:1