MrmAccountReceivableComponent

August 18, 2026

MrmAccountReceivableComponent

Kind: Class

Source: `Frontend/src/app/components/dashboard/dashboard-libary/widgets/mrm-account-receivable/mrm-account-receivable.component.ts` (line 77)

Part of: Frontend

`MrmAccountReceivableComponent` manages the accounts receivable dashboard widget. It checks feature access, loads currencies and dashboard data, prepares data for display, calculates year and month totals, and saves dashboard configuration changes.

Extends: `BaseWidgetComponent`

Methods

Method	Signature	Returns
`checkFeatureSubscription`	`checkFeatureSubscription()`	`boolean`
`getCurrencies`	`getCurrencies()`	`void`
`ngOnInit`	`ngOnInit()`	`void`
`ngOnChanges`	`ngOnChanges()`	`void`
`getDashboardData`	`getDashboardData()`	`void`
`initalizingData`	`initalizingData(value: BranchData)`	`void`
`saveDashboardData`	`saveDashboardData()`	`void`
`calculateSumByYear`	`calculateSumByYear()`	`void`
`calculateSumByMonth`	`calculateSumByMonth(monthIndex: number)`	`void`
`AddNewCurrency`	`AddNewCurrency(account: number, SalesCom: number)`	`void`

Method	Signature	Returns
`onCurrencyChanged`	`onCurrencyChanged(e: undefined, account: number, SalesCom: number, currencyIndex: number)`	`void`
`translateMonthName`	`translateMonthName(MonthId: number)`	`string`
`getpriceFormatted`	`getpriceFormatted(x: undefined)`	`string`
`onResize`	`onResize()`	`void`
`setChaching`	`setChaching()`	`void`
`getChaching`	`getChaching()`	`void`
`switchByGroupsHandler`	`switchByGroupsHandler()`	`void`
`switchBasedOnTendencyHandler`	`switchBasedOnTendencyHandler()`	`void`
`comparingWithPrevMonth`	`comparingWithPrevMonth(month: MonthData, accountIndex: number, salesComIndex: number, currencyIndex: number, monthIndex: number, propertyName: string)`	`string`
`comparingSummeryWithPrevMonth`	`comparingSummeryWithPrevMonth(month: MonthData, monthIndex: number, propertyName: string)`	`string`
`isCurrentMonth`	`isCurrentMonth(month: MonthData)`	`boolean`
`getpriceForecastAmount`	`getpriceForecastAmount(month: MonthData)`	`number`
`expectedCurrencyChange`	`expectedCurrencyChange(e: undefined)`	`void`
`selectText`	`selectText(e: undefined)`	`void`
`editSelectedMonthPrepare`	`editSelectedMonthPrepare(month: MonthData, supplierId: number)`	`void`
`mouseoverRowSpan`	`mouseoverRowSpan(e: undefined, bIndex: number)`	`void`
`mouseoutRowSpan`	`mouseoutRowSpan(e: undefined, bIndex: number)`	`void`
`openPopup`	`openPopup(n: string, t: string, i: string, btnTxt: string)`	`void`
`savePopup`	`savePopup()`	`void`
`ClosewithDonotSave`	`ClosewithDonotSave()`	`void`
`OnOpenedPopup`	`OnOpenedPopup()`	`void`

Method	Signature	Returns
`closePopup`	`closePopup()`	`void`

Properties

Property	Type
`sharedPopup`	`SharedPopupComponent`
`_widget`	`Widget`
`_widgetType`	`string`
`yearId`	`number`
`periodId`	`number`
`countOfEditablePrevMonths`	`number`
`withoutOptionBar`	`boolean`
`editMode`	`boolean`
`bookingOrSales`	`number`
`byGroups`	`boolean`
`selectedTeam`	`number`
`forecastRuleId`	`number`
`data`	`BranchData`
`allowedtoEdit`	`boolean`
`allowedtoEnableEdting`	`boolean`
`widgetEditRequest`	`any`
`widgetEditEnd`	`any`
`forecastRules`	`any[]`
`currencies`	`CompanyCurrency[]`
`selectedYear`	`number`
`currentMonthIndex`	`number`
`numYearsDatasource`	`any[]`
`todayWidgetsEnum`	`any`
`cachingWidgetData`	`CachingWidgetData`

Property	Type
`originalData`	`BranchData`
`selectedData`	`BranchData`
`singleMonths`	`MonthData`
`monthsSummery`	`MonthsSum[]`
`nonGroupAccountReceivables`	`MonthData[]`
`branchCurrency`	`any`
`basedOnTendency`	`boolean`
`SeriseRange`	`any[]`
`echartsIntance`	`any`
`totalTargets`	`number`
`yearBooking`	`number`
`totalForecasts`	`number`
`tempTotalTarget`	`TotalTargets`
`totalProfitAndLossAmount`	`number`
`totalOthers`	`number`
`totalTotalSGAndA`	`number`
`totalFORX`	`number`
`totalTotal`	`number`
`totalCumulative`	`number`
`totalPercentageUsedBudget`	`number`
`popup`	`any`
`searchText`	`string`
`urrenFeature`	`number`

Where it refuses work

- `MrmAccountReceivableComponent` stops the work with an early return when `this.dashboardLibraryService.checkFeatureSubscription(this.widget.chartEnumId)`.
- `MrmAccountReceivableComponent` stops the work with an early return when `MonthId == 12`.

- `MrmAccountReceivableComponent` stops the work with an early return when `month[propertyName] < this.data.Products.Currencies[currencyIndex].Details[monthIndex - 1…`.
- `MrmAccountReceivableComponent` stops the work with an early return when `month[propertyName] < this.data.Scrap.Currencies[currencyIndex].Details[monthIndex - 1][p…`.
- `MrmAccountReceivableComponent` stops the work with an early return when `month[propertyName] < this.monthsSummery[monthIndex - 1][propertyName]`.
- `MrmAccountReceivableComponent` stops the work with an early return when `AppSettings.calculateMonthsBeforeToday(month.YearId, month.MonthId -1) == 0 || AppSetting…`.

Diagram

```
graph LR
  Init[ngOnInit] --> Subscription[checkFeatureSubscription]
  Subscription --> Currencies[getCurrencies]
  Subscription --> Dashboard[getDashboardData]
  Dashboard --> Initialize[initalizingData]
  Initialize --> YearTotals[calculateSumByYear]
  Initialize --> MonthTotals[calculateSumByMonth]
  Changes[ngOnChanges] --> Dashboard
  CurrencyAction[AddNewCurrency] --> Currencies
  Save[saveDashboardData] --> Dashboard
```

Usage

```
import { MrmAccountReceivableComponent } from './mrm-account-receivable.component';

function refreshReceivableWidget(
  widget: MrmAccountReceivableComponent,
): void {
  if (!widget.checkFeatureSubscription()) {
    return;
  }

  widget.getCurrencies();
  widget.getDashboardData();
  widget.calculateSumByYear();
  widget.calculateSumByMonth();
}

// Call from an Angular host component after obtaining the widget instance.
function saveWidgetSettings(widget: MrmAccountReceivableComponent): void {
```

```
    widget.saveDashboardData();
  }
```

AI Coding Instructions

- Keep dashboard loading behind `checkFeatureSubscription()` so unavailable features do not request or display receivable data.
- Preserve the Angular lifecycle flow: use `ngOnInit()` for initial loading and `ngOnChanges()` when input-driven dashboard state changes.
- Run `initalizingData()` before calculating totals when replacing dashboard data, so month and year calculations operate on current state.
- Refresh currency-related state after `AddNewCurrency()` by calling `getCurrencies()` when the UI needs the updated list.
- Keep `saveDashboardData()` aligned with the same dashboard state shape returned by `getDashboardData()`.

Used by

3 references from 3 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (3)

- `DashboardTabComponent` — `Frontend/src/app/components/dashboard/dashboard-libary/dashboard-tab/dashboard-tab.component.ts` :1
- `DashboardWidgetsModule` — `Frontend/src/app/components/dashboard/dashboard-widgets.module.ts` :1
- `ReportsTabComponent` — `Frontend/src/app/components/reports/reports-library/reports-tab/reports-tab.component.ts` :1