

Join GitHub today


GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

[Table] Add example with dynamic columns #5927

New issue

 Open

rafaelss95 opened this issue on Jul 21, 2017 · 49 comments

rafaelss95 commented on Jul 21, 2017

Contributor

Bug, feature request, or proposal:

Proposal

What is the expected behavior?

It'd be nice to have example(s) on how to use `md-table` with dynamic columns.

What is the current behavior?

Currently all the examples are with hard coded columns, like this:

```
<!-- ID Column -->
<ng-container cdkColumnDef="userId">
  <md-header-cell *cdkHeaderCellDef md-sort-header> ID </md-header-cell>
  <md-cell *cdkCellDef="let row"> {{row.id}} </md-cell>
```


Assignees

 andrewseguin

Labels

P4
docs

Projects

 Table
Docs

Milestone

<ng-container *cdkColumnDef="progress">

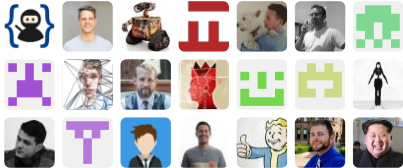
```
<md-header-cell *cdkHeaderCellDef md-sort-header> Progress </md-header-cell>
<md-cell *cdkCellDef="let row"> {{row.progress}}% </md-cell>
</ng-container>
```

@andrewseguin

34

1

27 participants



and others



willshowell commented on Jul 21, 2017 • edited

Contributor

For those interested, here was my approach

<https://plnkr.co/edit/UntMipJO7lQVFCCSq3sZ?p=preview>

```
<!-- Generic column definition -->
<ng-container *ngFor="let column of columns" [cdkColumnDef]="column.columnDef">
  <md-header-cell *cdkHeaderCellDef>{{ column.header }}</md-header-cell>
  <md-cell *cdkCellDef="let row">{{ column.cell(row) }}</md-cell>
</ng-container>

/** Table columns */
columns = [
  { columnDef: 'userId',    header: 'ID',      cell: (row: UserData) => `${row.id}`      },
  { columnDef: 'userName', header: 'Name',    cell: (row: UserData) => `${row.name}`    },
  { columnDef: 'progress', header: 'Progress', cell: (row: UserData) => `${row.progress}%` },
];

/** Column definitions in order */
displayedColumns = this.columns.map(x => x.columnDef);
```

EDIT: Here is a (hopefully) evergreen stackblitz with the exact same approach

--	--	--	--	--

andrewseguin self-assigned this on Jul 22, 2017

andrewseguin added the `md-table` label on Jul 22, 2017



alexrun commented on Jul 31, 2017

Is it possible to internationalize these dynamic columns implementation?

4



pueaau commented on Aug 9, 2017

I wonder what you refer to when you say "dynamic"? Many things regarding columns can be dynamic. If your goal is to dynamically change which columns are shown then I think the simplest approach is to change `displayedColumns`.

willshowell referenced this issue on Aug 30, 2017

Table with irregular headers #6732

Closed



sophistyx commented on Sep 4, 2017

I think he outlined it pretty well here:

//-----

What he (and myself) are looking for is the ability to render an arbitrary number of columns with heading names taken from some underlying datasource, e.g. columns: ColumnDefinition[];

It has to be said that the current mechanism for binding md-table to a dataSource is dreadful. It should be consistent with how we *ngFor over collections to render results, only in this instance, stamp out column definitions. Same story for rows.



willshowell referenced this issue on Sep 12, 2017

Dynamic Columns in md table #6159

Closed



eltimmo commented on Oct 5, 2017

Hi @willshowell. Thanks, just looking at how you approached this. I'm not sure why but the Plunker you created has stopped working.



willshowell commented on Oct 5, 2017

Contributor

@eltimmo updated the original comment!

1



andrewseguin added this to **Features in Table** on Oct 19, 2017




andrewseguin moved this from **Features** to **Docs in Table** on Oct 19, 2017

★  donroyco referenced this issue on Oct 20, 2017

mat-table implementation is confusing #7919

 Closed

🏷️  andrewseguin removed the `mat-table` label on Oct 24, 2017



jscharett commented on Oct 30, 2017

Can't seem to get this working in beta 12. Always get error that cdk-table can't find column with id.

Also, if I am using the MatTableModule, do I need to pull in the CdkTableModule?



jscharett commented on Oct 30, 2017

It seems that there is an issue with using dynamic columns inside the ng-bootstrap modal. As neither the ngx-bootstrap modal and the material2 modal seem to work when lazy loaded, I'm using ng-bootstrap. It seems that the ContentChildren don't include column defs when using *ngFor.



jscharett commented on Oct 31, 2017

Well, looks like I am wrong. I was able to create [this](#) Stackblitz and it works, so not sure why its failing for me locally. Ugh



jscharett commented on Oct 31, 2017



1

**eltimmo** commented on Oct 31, 2017

From Will's example, I created this. I'm looking forward to simple tables being implemented :-)

<https://stackblitz.com/edit/angular-dynamic-tables>



1

**Bsujeet** commented on Nov 6, 2017

Hi,

I was looking for how to write generic code to implement the pagination,filtering and sorting.

Thanks



1

**donmccurdy** commented on Dec 9, 2017 • edited ▼

Has this support for dynamic columns made it into a stable release? I'm trying to implement this and I'm not sure whether I'm doing something wrong, or whether my use case just isn't supported. I'd like to load arbitrary data from a CSV, detect column headers at runtime, and show that in a table.

```
<mat-table *ngIf="data" [dataSource]="data">
  <ng-container *ngFor="let column of columns" [matColumnDef]="column">
    <mat-header-cell *matHeaderCellDef>{{ column }}</mat-header-cell>
    <mat-cell *matCellDef="let row">{{ row[column] }}</mat-cell>
  </ng-container>
</mat-table>
```

```
{ "id": "1", "name": "Macaw, scarlet", "lat": "31.215291", "lng": "118.931012"},  
{ "id": "2", "name": "Armadillo, nine-banded", "lat": "35.663752", "lng": "103.389346"},  
{ "id": "3", "name": "Greater roadrunner", "lat": "13.17535", "lng": "44.27461"},  
{ "id": "4", "name": "Goanna lizard", "lat": "22.671042", "lng": "113.823131"},  
{ "id": "5", "name": "Cape starling", "lat": "16.0213558", "lng": "100.417181"}  
];  
const columns = Array<string> (Object.keys(rows[0]));  
const data = new MatTableDataSource<Object>(rows);
```

Result:

AppComponent.html:31 ERROR Error: Missing definitions for header and row, cannot determine which columns should be rendered.

```
at getTableMissingRowDefsError (table.es5.js:363)  
at MatTable.CdkTable.ngAfterContentInit (table.es5.js:512)  
at callProviderLifecycles (core.js:12422)  
at callElementProvidersLifecycles (core.js:12399)  
at callLifecycleHooksChildrenFirst (core.js:12383)  
at checkAndUpdateView (core.js:13511)  
at callViewAction (core.js:13858)  
at execEmbeddedViewsAction (core.js:13816)  
at checkAndUpdateView (core.js:13509)  
at callViewAction (core.js:13858)
```

EDIT: The problem was that I needed to include the following, inside of `<mat-table/>` :

```
<mat-header-row *matHeaderRowDef="columns"></mat-header-row>  
<mat-row *matRowDef="let row; columns: columns;"></mat-row>
```



10



4

@donmccurdy looks like you're an bot except you are missing definitions for `mat-row` and `mat-header-row` where you will define which columns to display



donmccurdy commented on Dec 9, 2017

@andrewseguin thanks! That was exactly it. 😊



boombard commented on Dec 28, 2017

Hi

Has anyone tried this with sorting enabled?

```
columnSetup = [  
  {  
    def: 'name',  
    title: 'Name',  
    displayFcn: (item: Item) => item.name,  
    sort: true  
  }  
];
```

```
<mat-table #table [dataSource]="tableData" matSort>  
  <ng-container *ngFor="let column of columnSetup" matColumnDef="{{ column.def }}">  
    <mat-header-cell *matHeaderCellDef [attr.mat-sort-header]="column.sort ? column.def : null">  
      <mat-cell *matCellDef="let item"> {{ column.displayFcn(item) }} </mat-cell>  
    </ng-container>  
    <mat-header-row *matHeaderRowDef="displayedColumns()"></mat-header-row>  
    <mat-row *matRowDef="let row; columns: displayedColumns();"></mat-row>  
</mat-table>
```


Any suggestions appreciated!



anywayTsao commented on Feb 7, 2018 • edited ▼

@boombard , I've tried you template code.

```
[attr.mat-sort-header]="column.sort ? column.def : null" => not work
```

```
[mat-sort-header]="column.sort ? column.def : null"
```

It looks the binding you use => [attr.mat-sort-header] is bind to the DOM, not a property binding.

<https://angular.io/guide/template-syntax#binding-targets>



nothingisnecessary commented on Mar 9, 2018 • edited ▼

@boombard I tried different property bindings and I couldn't get it to work. I found that if you add "disabled" attribute it disables it, but other than using *ngIf I didn't find a way to conditionally emit the disabled attribute.

Conditionally sortable columns (works fine in a dynamic column loop too)

```
<div *ngIf="sortable">
  <mat-header-cell *matHeaderCellDef mat-sort-header> No. </mat-header-cell>
</div>
<div *ngIf="!sortable">
  <mat-header-cell *matHeaderCellDef mat-sort-header disabled> No. </mat-header-cell>
</div>
```



boombard commented on Mar 9, 2018

@nothingisnecessary Thanks for the suggestion, in the end I also went with *ngIf - seems like the only option



jimmykane commented on Apr 11, 2018

Jeee! This is not added to the docs yet?



HDaghash commented on Apr 18, 2018

any update on that guys , the above code not working with the latest version



jimmykane commented on Apr 18, 2018

@HDaghash Works still for me.



HDaghash commented on Apr 19, 2018

@jimmykane i handled that in different way , it seems ur not using the latest version of angular material
this guy "cdkHeaderCellDef" not there anymore .
anyway it's sorted



jimmykane commented on Apr 19, 2018 • edited ▼

I was refering to @donmccurdy s answer.

the `cdkHeaderCellDef` of course it's there

<https://material.angular.io/components/table/api#MatHeaderCellDef>



paco76 commented on May 21, 2018 • edited ▼

and if I want a checkbox or a button to be added for each row? how is it possible using @willshowell approach?



willshowell commented on May 21, 2018

Contributor

@paco76

Note I just copy-pasted syntax from my original comment. It's from an outdated beta version of Material and should be updated to the current syntax

```
<!-- Button column -->  
<ng-container ... ></>
```

```
<!-- Checkbox column -->  
<ng-container ... ></>
```

```
<!-- Generic column definition for dynamic columns -->  
<ng-container *ngFor="let column of columns" [cdkColumnDef]="column.columnDef">  
  <md-header-cell *cdkHeaderCellDef>{{ column.header }}</md-header-cell>
```

```
/** Static columns */
staticColumns = ['button', 'checkbox'];

/** Dynamically generated columns */
dynamicColumns = [
  { columnDef: 'userId', header: 'ID', cell: (row: UserData) => `${row.id}` },
  { columnDef: 'userName', header: 'Name', cell: (row: UserData) => `${row.name}` },
  { columnDef: 'progress', header: 'Progress', cell: (row: UserData) => `${row.progress}%` }
];

/** Column definitions in order */
displayedColumns = [...this.staticColumns, ...this.columns.map(x => x.columnDef)];
```



paco76 commented on May 21, 2018 • edited ▼

Hi @willshowell, I tried your approach and it works great but I was wondering how to add a checkbox or a button for each row of the table? thank you!



vgrados2 commented on May 22, 2018

and how to add buttons?



benjamincharity commented on May 22, 2018

@willshowell had some pseudo-code showing that you would just define the checkbox or button template as a column definition:

```
<!-- Checkbox column -->
<ng-container ... ></>
```

Which, with a checkbox would look something like:

```
<ng-container matColumnDef="myColumn">
  <md-header-cell *cdkHeaderCellDef>{{ column.header }}</md-header-cell>
  <md-cell *cdkCellDef="let row">
    <mat-checkbox [(ngModel)]="row.checked"></mat-checkbox>
  </md-cell>
</ng-container>
```

People may be able to provide better feedback if you explain what you have tried and what is not working for you.



vgrados2 commented on May 22, 2018 • edited ▼

I want to go through the object discarding the column actions with an `*ngIf` but it does not work, it shows me the following error.

Error: Duplicate column definition name provided: "actions".

```
columns = ['id', 'name', 'actions'];
```

```
<ng-container matColumnDef="{{column}}" *ngFor="let column of displayedColumns">
  <div *ngIf="column!='actions'">
    <mat-header-cell *matHeaderCellDef> {{column}} </mat-header-cell>
    <mat-cell *matCellDef="let element">
      {{element[column]}}
    </mat-cell>
```

```
</mat-header-cell> </actions> </mat-header-cell>
<mat-cell *cdkCellDef="let element" >
  <button md-raised-button >Edit</button>
</mat-cell>
</ng-container>
```



DaDave commented on May 22, 2018 • edited ▼

This should work:

```
<div *ngFor="let displayedColumn of displayedColumns; let columnIndex = index"> <ng-container
*ngIf="displayedColumn != 'actions'" matColumnDef="{{displayedColumn}}"> <mat-header-cell
*matHeaderCellDef mat-sort-header>{{displayedColumn}}</mat-header-cell> <mat-cell *matCellDef="let
element "> {{element[displayedColumn]}}</mat-cell> </ng-container> </div> <ng-container
matColumnDef="actions"> <mat-header-cell *matHeaderCellDef >Actions</mat-header-cell> <mat-cell
*matCellDef="let element "> <button mat-raised-button> Edit </button> </mat-cell> </ng-container>
```

5

3

3

4



vgrados2 commented on May 22, 2018 • edited ▼

that worked, thanks @DaDave



paco76 commented on May 25, 2018

Hi @DaDave,
can you please connect all parts together and show one working example?

And here is a more complicated one with sorting, checkboxes and filtering

FYI

```
<mat-card class="mat-elevation-z4">
  <mat-form-field>
    <input matInput (keyup)="applyFilter($event.target.value)" placeholder="Filter">
  </mat-form-field>
  <mat-table *ngIf="data" [dataSource]="data" matSort>
    <ng-container *ngFor="let column of columns; first as isFirst; last as isLast"
[matColumnDef]="column">
      <mat-header-cell *matHeaderCellDef mat-sort-header [disabled]="isFirst || isLast">
        <mat-checkbox *ngIf="isFirst" (change)="$event ? masterToggle() : null"
          [checked]="selection.hasValue() && isAllSelected()"
          [indeterminate]="selection.hasValue() && !isAllSelected()">
        </mat-checkbox>
        <span *ngIf="!isFirst && !isLast">
          <mat-icon *ngIf="getColumnIcon(column)">{{ getColumnIcon(column) }}</mat-icon>
        </span>
      </mat-header-cell>
      <mat-cell *matCellDef="let row">
        <mat-checkbox *ngIf="column === 'Checkbox'" (click)="$event.stopPropagation()"
          (change)="$event ? checkBoxClick(row) : null"
          [checked]="selection.isSelected(row)">
        </mat-checkbox>
        <span *ngIf="column === 'Actions'">
          <app-event-card-actions-menu [event]="row[column]"></app-event-card-actions-menu>
        </span>
        <span *ngIf="column === 'Name'" matTooltip="{{ row[column] }}">
          {{ row[column] | slice:0:10 }}
        </span>
        <span *ngIf="column !== 'Checkbox' && column !== 'Actions' && column !== 'Name'">
          {{ row[column] }}
        </span>
      </mat-cell>
    </ng-container>
  <mat-header-row *matHeaderRowDef="columns"></mat-header-row>
  <mat-row *matRowDef="let row; columns: columns;" [routerLink]="['/eventDetails']">
```



3



1



paco76 commented on May 25, 2018 • edited ▼

Hi **jimmykane**, thank you!
do you mind also sharing your component logic? for us to have a real full example.



2



natecowen commented on Jun 27, 2018

@paco76 Here is an example of my code, both the view and the typescript logic. It's based on the example that **@jimmykane**. I am not using sort, but I do have filtering and pagination working.

In my example, I am using the mat-table as a reusable child component. The parent component passes in certain items. Below is my **code that I use to call implement the child component in the parent**.

```
<div *ngIf="results">
  <app-cust-data-table *ngIf="showTableResults"
    [(receivedData)] = "results"
    [(columns)] = "columns"
    tableTitle = "Search Results"
    (clickedItem) = "viewItem($event)"
    (pageEvent) = "updatePagination($event)"
  >
</app-cust-data-table>
</div>
```

This is my child component HTML view


```

<h4>{{tableTitle}}</h4>
</mat-card-title>
</mat-card-header>

<mat-card-content>
  <mat-form-field class="full-width-filter">
    <input matInput (keyup)="applyFilter($event.target.value)" placeholder="Filter" autoc
  </mat-form-field>

  <table mat-table #table [dataSource]="dataSource" style="width:100%">

    <ng-container *ngFor="let column of columns" [matColumnDef]="column.columnDef">
      <div *ngIf="column.columnDef !== 'detailBtn'">
        <th mat-header-cell *matHeaderCellDef > {{column.header}} </th>
        <td td mat-cell *matCellDef="let row">
          <strong> &nbsp;{{ column.dataName(row) }}</strong>
        </td>
      </div>

      <div *ngIf="column.columnDef === 'detailBtn'">
        <th mat-header-cell *matHeaderCellDef > {{column.header}} </th>
        <td td mat-cell *matCellDef="let row">
          <button mat-raised-button color="primary" id="column.dataName(row)" (click)="view

        </td>
      </div>
    </ng-container>

    <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
    <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
  </table>

  <mat-paginator [pageSizeOptions]="[25, 50, 100]" showFirstLastButtons [length]="length" (pa
</mat-card-content>
</mat-card>

```

```
import { Component, ViewChild, Input, OnChanges, Output, EventEmitter, OnInit } from '@angular/core';
import { MatPaginator, MatTableDataSource, PageEvent } from '@angular/material';

@Component({
  selector: 'app-cust-data-table',
  templateUrl: 'data-table.component.html',
  styleUrls: ['./data-table.component.scss']
})
export class CustDataTableComponent implements OnChanges, OnInit {
  @Input() displayedColumns: string[];
  @Input() receivedData;
  @Input() tableTitle: string;
  @Input() columns: any[] = [];
  @Input() metaCount: number;

  @Output() clickedItem = new EventEmitter();
  @Output() pageEvent = new EventEmitter<PageEvent>();

  dataSource: MatTableDataSource<any>;
  @ViewChild(MatPaginator) paginator: MatPaginator;

  pageIndex = 0;
  pageSize = 25;
  length;

  ngOnInit() {}

  ngOnChanges() {
    if (this.columns !== undefined || this.columns !== null) {
      if (this.metaCount) {
        this.dataSource = new MatTableDataSource(this.receivedData);
        this.displayedColumns = this.columns.map(x => x.columnDef);

        this.length = this.metaCount;
        this.paginator.length = this.metaCount;

        // IMPORTANT! Do NOT use below if you are getting pagination from API Call. This is only
        // this.dataSource.paginator = this.paginator;
      } else {
        this.dataSource = new MatTableDataSource(this.receivedData);
      }
    }
  }
}
```

```
    this.dataSource.paginator.pageSize = this.pageSize;
    this.dataSource.paginator.pageIndex = this.pageIndex;

    this.dataSource.paginator.length = this.receivedData.length;
  }
}

applyFilter(filterValue: string) {
  filterValue = filterValue.trim(); // Remove whitespace
  filterValue = filterValue.toLowerCase(); // Datasource defaults to lowercase matches
  this.dataSource.filter = filterValue;
  if (this.dataSource.paginator) {
    this.dataSource.paginator.firstPage();
  }
}

updateProductsTable(event: PageEvent) {
  this.pageSize = event.pageSize;
  this.pageIndex = event.pageIndex + 1; // API starts 1, Mat-Table starts at 0

  this.pageEvent.emit(event);
}

viewItem(guid) {
  this.clickedItem.emit(guid);
}
}
```

A few items to note: I am using the component twice in my parent component. Using mat-tabs I have a search tab that uses can enter in text to search for. Using the other tab, they can view all 3000 items.

The search uses materials built in pagination by setting `this.dataSource.paginator = this.paginator`. The all products uses pagination set via the server. In my `getAllProducts` server I pass in a default number of items to return and a default page number to start with. The mat-paginator then fires an event back to the parent component for the call of the next page of items.



dgreatorex commented on Jun 27, 2018

Hi @natecowen,

Any chance you can share a stackblitz?

Thanks! :)

Dave



fxck commented on Jul 3, 2018

Contributor

Is it possible to somehow pass columndefs down from a parent?



natecowen commented on Jul 3, 2018 • edited ▼

@fxck Yes, that is what I am doing in my example. In my parents typescript I declare columns near the top of the parent class.

```
columns: any[] = [  
  { columnDef: 'productName', header: 'Product Name', dataName: row => `${row.name}` },  
  { columnDef: 'productDescription', header: 'Description', dataName: row => `${row.itemDescription}` },  
  { columnDef: 'detailBtn', header: 'View/Edit', dataName: row => `${row.guid}` }  
];``
```

That is then passed into the parents html via [(columns)] = "columns". See example

```
````javascript  
<app-cust-data-table *ngIf="showTableResults"
 [(receivedData)] = "results"
```

```
(pageEvent) => updatePagination(pageEvent)
>
</app-cust-data-table>
```

◀ ▶

@dgreatorex I started working on a stackblitz, but haven't been able to finish it. It's currently not fully working as I need to find an online api for search functionality. However, you can see it working for all products. Click the all products tab to see it in action. Hopefully it helps. [Here is the stackblitz](#)



fxck commented on Jul 3, 2018

Contributor

What I mean is to pass down whole templates, imagine you want to use a component inside the cell.



willshowell commented on Jul 3, 2018

Contributor

@fxck CdkTable has `addColumnDef` and `removeColumnDef` methods. It's a little clunky to use, but I think it does what you need

<https://stackblitz.com/edit/angular-i4dftq?file=app/cdk-table.ts>



fxck commented on Jul 3, 2018 • edited ▼

Contributor

@willshowell I threw together this <https://stackblitz.com/edit/angular-i4dftq-kxqbk4?file=app%2Ftable.ts> as what I really need is to be able to control the inside of the cell, depending on the data from the column... it works, but not quite, for example I couldn't think of a way to hide the original text in case a special template is passed down for this particular column..

basically something like this (pseudo code)

```
<vsh-universal-list-table
 [data]="data"
 [columns]="columns"
 [activeColumns]="activeColumns">
 <ng-container
 #data="vshUniversallistTableCell"
 *vshUniversallistTableCell="email">
 {{ data.element[data.column?.name] }}
 </ng-container>
</vsh-universal-list-table>
```

in which case the original thing wouldn't show up

```
<mat-cell *matCellDef="let element">
 <!-- doesn't have special template -->
 <ng-container *ngIf="!templates[column.name]">
 {{ element[column.name] }}
 </ng-container>

 <!-- has special template -->
 <ng-container *ngIf="templates[column.name]">
 <ng-template
 [ngTemplateOutletContext]="{
 column: column,
 element: element
 }"
 [ngTemplateOutlet]="templates[column.name]">
 </ng-template>
 </ng-container>
</mat-cell>
```

ugh..

FYI what I wanted to achieve is quite possible, same principle as shown in this article - <https://alligator.io/angular/reusable-components-ngtemplateoutlet/>



daiyis commented on Sep 14, 2018

FYI what I wanted to achieve is quite possible, same principle as shown in this article - <https://alligator.io/angular/reusable-components-ngtemplateoutlet/>

hey @fxck, I am doing something similar to create dynamic column with different templates. how can you pass the templates together to the base component and reference it with something like `templates[column.name]`?



relair commented on Oct 1, 2018

Is it possible to somehow pass `columndefs` down from a parent?  
What I mean is to pass down whole templates, imagine you want to use a component inside the cell.

Yes, I have created a library that does that: <https://www.npmjs.com/package/material-dynamic-table>  
Feel free to take a look - or use the library itself. It works by defining what components do you want to used for your column types and then just providing a definition of your table as an array of your column configurations. It also allows to define individual column filters in the similar way.



shripalshah commented on Oct 31, 2018 • edited ▼

**Bug, feature request, or proposal:**

It'd be nice to have example(s) on how to use `md-table` with dynamic columns.

### What is the current behavior?

Currently all the examples are with hard coded columns, like this:

```
<!-- ID Column -->
<ng-container cdkColumnDef="userId">
 <md-header-cell *cdkHeaderCellDef md-sort-header> ID </md-header-cell>
 <md-cell *cdkCellDef="let row"> {{row.id}} </md-cell>
</ng-container>

<!-- Progress Column -->
<ng-container cdkColumnDef="progress">
 <md-header-cell *cdkHeaderCellDef md-sort-header> Progress </md-header-cell>
 <md-cell *cdkCellDef="let row"> {{row.progress}}% </md-cell>
</ng-container>
```

@andrewseguin

Try doing `<md-cell *cdkCellDef="let row"> {{row[progress]}}`

This is not a bug. This was related to concatenating two literals in Interpolation.

Good Luck!!



1



nasreddineskandr... commented on May 18 • edited ▼

Hi!

@daiyis @fxck

With this solution the columns are dynamic and the cell content also.



extract a wrapper of `mat-table` for now :p).

Example:

the column 4 in the example is `symbol` with a custom component to render in the cell.

Github:

<https://github.com/nasreddineskandrani/angular-dynamic-cell-mat-table>

Stackblitz:

<https://stackblitz.com/edit/angular-dynamic-cell-mat-table?file=app%2Ftable-basic.component.ts>

**Members of material repo: @andrewseguin**

I was going to PR a proper solution based on this inside material repo.

A solution so to allow passing a component to a cell.

I want to know first if this is wanted? or better to wrap `mat-table` outside material if we want it and keep the `mat-table` lean.

Thank you



2



**andrewseguin** commented on May 21

Contributor

Hey @nasreddineskandrani - looks like a neat way of doing dynamic components. I think if we had such a feature, it would likely go the route of using `CdkPortal`. That said, we don't currently have plans on bringing something like this into the repository right now



1



**plixxer** commented on Aug 8

For those interested, here was my approach

```
<!-- Generic column definition -->
<ng-container *ngFor="let column of columns" [cdkColumnDef]="column.columnDef">
 <md-header-cell *cdkHeaderCellDef>{{ column.header }}</md-header-cell>
 <md-cell *cdkCellDef="let row">{{ column.cell(row) }}</md-cell>
</ng-container>

/** Table columns */
columns = [
 { columnDef: 'userId', header: 'ID', cell: (row: UserData) => `${row.id}`
 { columnDef: 'userName', header: 'Name', cell: (row: UserData) => `${row.name}`
 { columnDef: 'progress', header: 'Progress', cell: (row: UserData) => `${row.progress}%`
];

/** Column definitions in order */
displayedColumns = this.columns.map(x => x.columnDef);
```

EDIT: Here is a (hopefully) evergreen stackblitz with the exact same approach

<https://stackblitz.com/edit/material2-beta11-vmwjpe>

Thank you good Sir, I am converting a bootstrap UI to a material UI and i was stuck on this for quite some time.