Meet The Overflow, a newsletter by developers, for developers. Fascinating questions, illuminating answers, and entertaining links from around the web. **Learn more**

Angular 7 mat-table each row reusable component

Asked 9 months ago Active 9 months ago Viewed 738 times

0

How can each row in mat-table be reusable component ?

In regular html table i use this approach

```
<table class="table table-hover table-bordered">
    <thead>
        <tr>
            <th class="text-left width-50" ></th>
            <th class="text-left width-85">Id</th>
            <th class="text-left">Price</th>
            <th class="text-left width-160">City</th>
            <th class="text-left width-160">State</th>
            <th class="text-left width-160">Qty</th>
            <th class="text-left width-160">Action</th>
        </tr>
    </thead>
    <tbody>
      <tr pdp-adjustment-list-item *ngFor="let currentItem of pagedResponse?.content"
         (idCheckedOutput)="addItemIdToCheckedArray($event)"
         [item]="currentItem" >
      </tr>
    </tbody>
</table>
```

pdp-adjustment-list-item is selector of AdjustmentListItemComponent. This is convenient because each row is one same instance of AdjustmentListItemComponent with reactive form and one @Input() item i pass object in in a loop.

This is clean and intuitive.

Now, in Angular7 material table examples i could find everything is placed in one uber component.

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
  <td mat-cell *matCellDef="let element"> {{element.position}} </td>
</ng-container>


<ng-container matColumnDef="name">
  <th mat-header-cell *matHeaderCellDef> Name </th>
  <td mat-cell *matCellDef="let element"> {{element.name}} </td>
</ng-container>

<ng-container matColumnDef="weight">
  <th mat-header-cell *matHeaderCellDef> Weight </th>
  <td mat-cell *matCellDef="let element"> {{element.weight}} </td>
</ng-container>

<ng-container matColumnDef="symbol">
  <th mat-header-cell *matHeaderCellDef> Symbol </th>
  <td mat-cell *matCellDef="let element"> {{element.symbol}} </td>
</ng-container>

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

If this is true, and you actually must keep everything in one uber component, this will not only be poor design choice when we talk about reusability, but also having to keep everything in one component creates huge mess of spaghetti code.

So way around this would be to dynamically create one reactive form for each row in uber component, and then utilize form array, but what is the point? Actually i did that and decided to delete it because code would be totally unmaintainable.

angular | html-table | angular-material

asked Dec 27 '18 at 22:08

SeaBiscuit

1,397 3 15 27

1 Answer

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

1

```
<td *ngFor="let ....
```

but is exactly the same here, you define the columns and the rows are generated dynamically based on the "datasource".

```
<table mat-table [dataSource]="dataSource" class="mat-elevation-z8">
```

you even can make the columns definitions been generated dynamically

```
<ng-container [matColumnDef]="column" *ngFor="let column of displayedColumns">
            <mat-cell *matCellDef="let element" >
                        {{ element[column] }}
            </mat-cell>
        </ng-container>
```

Inspect this example from the material docs

Example of material with reactive forms

edited Dec 27 '18 at 23:33

answered Dec 27 '18 at 22:37

Sinuee Hernández

82 5

How would i achieve "one reactive form per one row" using this approach? How would this help me when i need each row to carry certain logic? I have no problem what so ever simply displaying data. – SeaBiscuit Dec 27 '18 at 22:42

`rows: FormArray = this.fb.array([]); form: FormGroup = this.fb.group({ 'streams': this.rows, });` rows will be populated when you get your datasource, you will create one formGoup per row and you can access them by index. `<td mat-cell *matCellDef="let element; let index = index" [formGroupName]="index">` – Sinuee Hernández Dec 27 '18 at 22:59

Example with reactive forms – Sinuee Hernández Dec 27 '18 at 23:30

this is actually really good. Much simpler than what i originally wrote. thnks – SeaBiscuit Dec 27 '18 at 23:35

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.