Understanding structural directives - Demo

Highlights:

* To change the visibility of controls using ngIf directive
* To create template for repetitive data using ngFor directive

**Demosteps:**

In order to learn how to use structural directives, try out the following demo steps:

**Step 1:**

Open QuickKartApp project in Visual studio. You will now try to change the visibility of the controls and create a template for representing repetitive data using ngIf and ngFor directives.

**Step 2:**

Open the view-products.component.ts file present in QuickKartApp --> src --> app --> view-products folder.

**Step 3:**

Declare a **products**array of **any**type inside ViewProductsComponent class as shown below in line 10:

1. import { Component, OnInit } from '@angular/core';
2. @Component({
3. selector: 'app-view-products',
4. templateUrl: './view-products.component.html',
5. styleUrls: ['./view-products.component.css']
6. })
7. export class ViewProductsComponent implements OnInit {
8. products: any[];
9. constructor() { }
10. ngOnInit() {
11. }
12. }

This array will be used later to populate the products information.

**Step 4:**

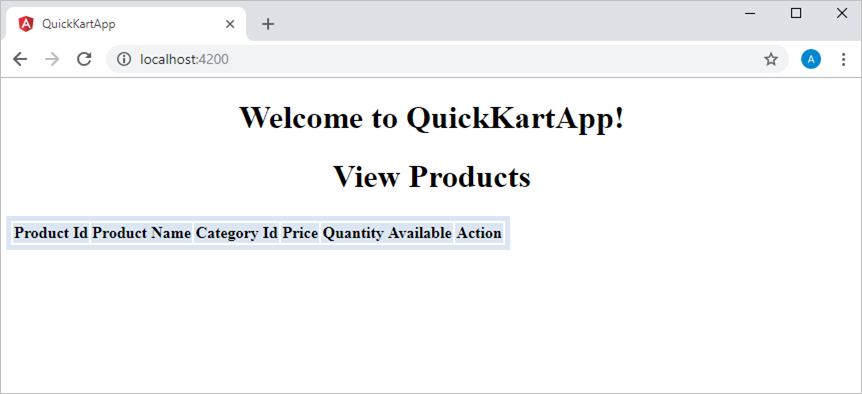
Open view-products.component.html and replace the existing code with the following lines of code to create a table, for displaying products details.

1. <div style="text-align:center;">
2. <h1>View Products</h1>
3. <div class="table-responsive">
4. <table class="table" style="border:5px solid rgba(220,230,242,1);">
5. <tr style="background-color:rgba(220,230,242,1); font-size:12pt">
6. <th style="text-align:center">Product Id</th>
7. <th style="text-align:center">Product Name</th>
8. <th style="text-align:center">Category Id</th>
9. <th style="text-align:center">Price</th>
10. <th style="text-align:center">Quantity Available</th>
11. <th style="text-align:center">Action</th>
12. </tr>
13. <tr \*ngFor="let product of products" style="background-color:white">
14. </tr>
15. </table>
16. </div>
17. </div>

Note that here the structural directive **\*ngFor**is used to iterate over the products array declared in the ViewProductsComponent class in the previous step.

Currently, the products array is empty. The code for displaying the details of the products will be written in the upcoming demos.

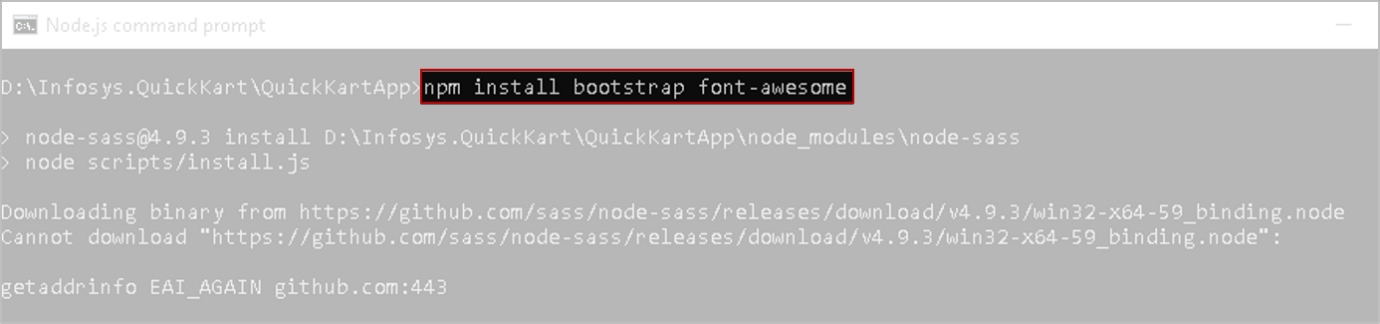
Execute your application and check if the page is loaded as shown below.



**Step 5:**

To customize your web page and the font you can install Bootstrap and FontAwesome by giving the following command in command prompt. Skip this step of installing bootstrap,if you already have bootstrap installed as a part of node modules.

1. npm install bootstrap font-awesome



To make use of above installed styles sheets, you should import bootstrap and font-awesome in your styles.css in QuickKartApp --> src folder. Include the following lines of code into styles.css file.

1. @import "~bootstrap/dist/css/bootstrap.css";
2. @import "~font-awesome/css/font-awesome.css";

Now run the application by executing ng serve -o command and observe the output.



**Step 6:**

Since the products array is empty, the table added in the view-products.component.html should not be visible to the end users.

It should made visible to users only when product details are present in products array.

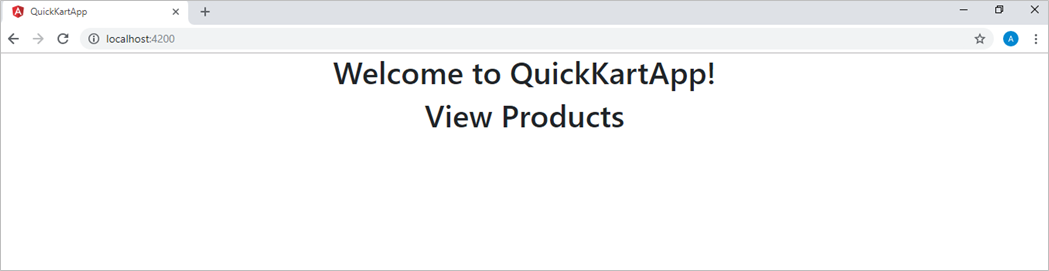
You can use **\*ngIf**to achieve this.

Modify the existing table present in view-products.component.html as shown below by adding line 4:

1. <div style="text-align:center;">
2. <h1>View Products</h1>
3. <div class="table-responsive">
4. <table class="table" style="border:5px solid rgba(220,230,242,1);" \*ngIf="products">
5. <tr style="background-color:rgba(220,230,242,1); font-size:12pt">
6. <th style="text-align:center">Product Id</th>
7. <th style="text-align:center">Product Name</th>
8. <th style="text-align:center">Category Id</th>
9. <th style="text-align:center">Price</th>
10. <th style="text-align:center">Quantity Available</th>
11. <th style="text-align:center">Action</th>
12. </tr>
13. <tr \*ngFor="let product of products" style="background-color:white">
14. </tr>
15. </table>
16. </div>
17. </div>

Here \*ngIf="products" checks if the products array is empty. If it is empty then the table will not be displayed.

Execute the application and observe that the table is not displayed.



**Step 7:**

When the table is not displayed, appropriate message should be displayed to the end users.

Here you can make use of **else**statement.

Add line 4 to view-products.component.html file.

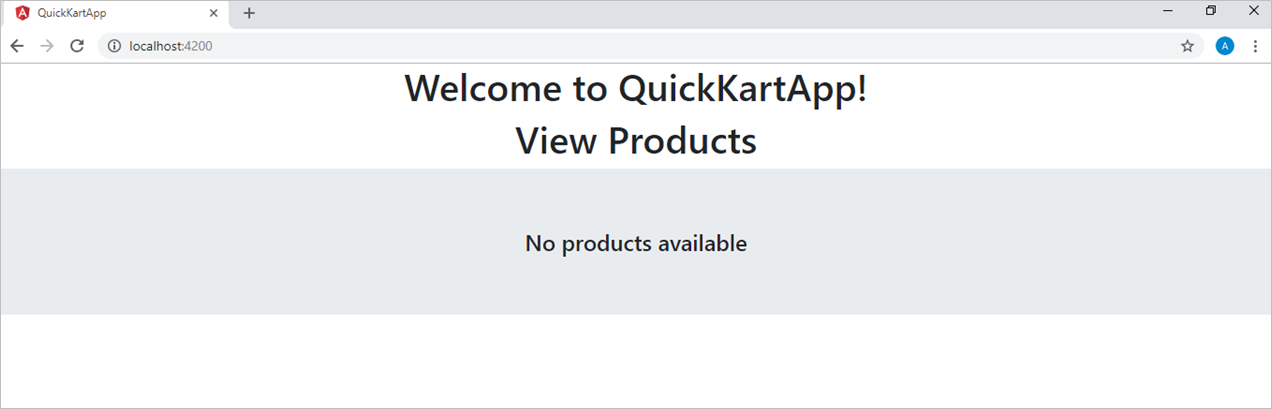
1. <div style="text-align:center;">
2. <h1>View Products</h1>
3. <div class="table-responsive">
4. <table class="table" style="border:5px solid rgba(220,230,242,1);" \*ngIf="products; else elseBlock">
5. <tr style="background-color:rgba(220,230,242,1); font-size:12pt">
6. <th>Product Id</th>
7. <th>Product Name</th>
8. <th>Category Id</th>
9. <th>Price</th>
10. <th>Quantity Available</th>
11. <th>Action</th>
12. </tr>
13. <tr \*ngFor="let product of products" style="background-color:white"></tr>
14. </table>
15. </div>
16. <ng-template #elseBlock>
17. <div>
18. <h4 class="jumbotron" style="text-align:center;">No products available</h4>
19. </div>
20. </ng-template>
21. </div>

Here a reference variable **#elseBlock** is declared.

 If the first \*ngIf statement is false then the else part is executed i.e. the template with the reference as #elseBlock will be displayed.

**Step 8:**

Execute your application and check if the page is loaded as shown below.



**Step 9:**

Another way to display the appropriate message is to add another **\*ngIf**in place of the else part.

In order to do this, add a variable showMsgDiv which is a boolean value to indicate if the products array has elements or not.

Modify the view-products.component.ts by adding line 12 as shown below:

1. import { Component, OnInit } from '@angular/core';
2. @Component({
3. selector: 'app-view-products',
4. templateUrl: './view-products.component.html',
5. styleUrls: ['./view-products.component.css']
6. })
7. export class ViewProductsComponent implements OnInit {
8. products: any[];
9. showMsgDiv: boolean = false;
10. constructor() { }
11. ngOnInit() {
12. }
13. }

Here showMsgDiv is initialized with false. Its value should be changed based on whether products array contains any data. This check should be performed every time the component is loaded.

**Step 10:**

In order to check the presence of data in products array whenever the component is loaded you can make use of ngOnInit() method of OnInit.

Implement the ngOnInit() method from line 17 - 20 as shown below.

1. import { Component, OnInit } from '@angular/core';
2. @Component({
3. selector: 'app-view-products',
4. templateUrl: './view-products.component.html',
5. styleUrls: ['./view-products.component.css']
6. })
7. export class ViewProductsComponent implements OnInit {
8. products: any[];
9. showMsgDiv: boolean = false;
10. constructor() { }
11. ngOnInit() {
12. if (this.products == null)
13. {
14. this.showMsgDiv = true;
15. }
16. }
17. }

Here **OnInit**is a life cycle hook imported from the core module, which is implemented by the ViewProductsComponent class in order to recognize ngOnInit() method. This method checks if the the products array is null. If it is null, showMsgDiv property is set to true.

**Lifecycle hooks:**

Component instances have a lifecycle as angular creates, updates, and destroys them. Developers can tap into key moments in that lifecycle by implementing one or more of the lifecycle hook interfaces in the angular core library.

Each interface has a single hook method whose name is the interface name prefixed with ng. For example, the OnInit interface has a hook method named **ngOnInit()**

**ngOnInit()**will initialize the component after angular displays the data-bound properties and sets the component's input properties. It is invoked only once when the component is instantiated.

Similarly there are many Lifecycle hooks in angular but in this demo only **OnInit**is being used.

You can learn about [life cycle hooks](https://angular.io/guide/lifecycle-hooks) in detail.

**Step 11:**

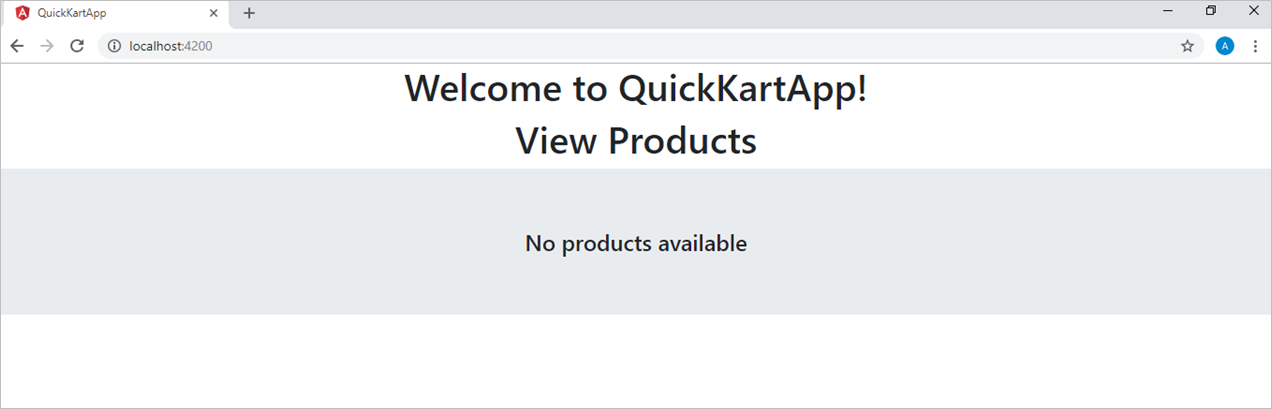
Add the following code from line 17 - 19 in view-products.component.html to display message based on the value of showDivMsg property.

1. <div style="text-align:center;">
2. <h1>View Products</h1>
3. <div class="table-responsive">
4. <table class="table" style="border:5px solid rgba(220,230,242,1);" \*ngIf="products">
5. <tr style="background-color:rgba(220,230,242,1); font-size:12pt">
6. <th>Product Id</th>
7. <th>Product Name</th>
8. <th>Category Id</th>
9. <th>Price</th>
10. <th>Quantity Available</th>
11. <th>Action</th>
12. </tr>
13. <tr \*ngFor="let product of products" style="background-color:white">
14. </tr>
15. </table>
16. </div>
17. <div \*ngIf="showMsgDiv" style="">
18. <h4 class="jumbotron" style="text-align:center;">No products available</h4>
19. </div>
20. </div>

You can alternatively do this using \*ngIf="!products". The message will be displayed if there are no products.

**Step 12:**

Execute your application and check if the page is loaded as shown below.



# Understanding attribute directives - Do it Yourself

**Highlights:**

* To change the style of elements using ngClass and ngStyle directives

**Demosteps:**

In order to learn how to use attribute directives, try out the following steps:

Make changes to the ViewProductsComponent that you created in the previous demo.

**ngClass**

ngClass directive allows you to add or remove CSS classes to an HTML element.

**Step 1:**

Add the following classes to view-products.component.css

1. .red{color:red;}
2. .size30{font-size:30px;}

**Step 2:**

Add the below code shown in Line 2 to view-products.component.html

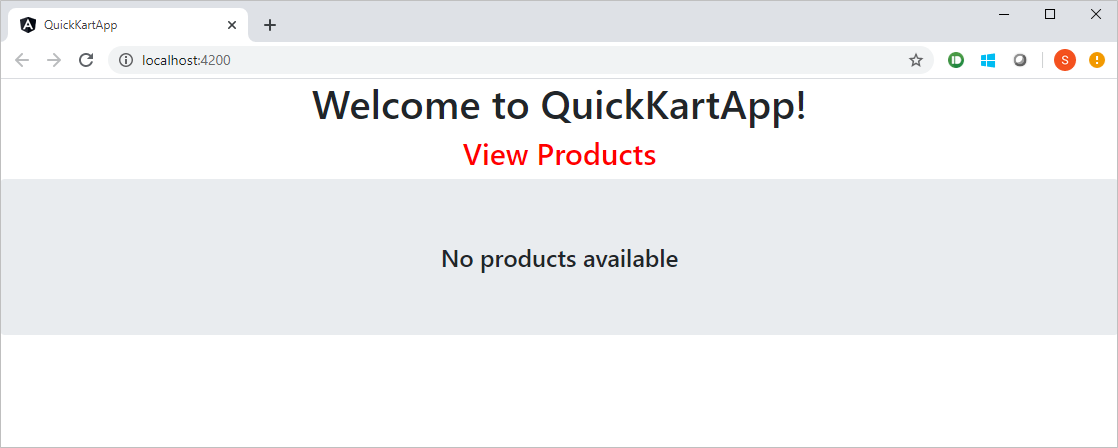
1. <div style="text-align:center;">
2. <h1 ngClass='red size30'>View Products</h1>
3. <div class="table-responsive">
4. <table class="table" style="border:5px solid rgba(220,230,242,1);" \*ngIf="products">
5. <tr style="background-color:rgba(220,230,242,1); font-size:12pt">
6. <th>Product Id</th>
7. <th>Product Name</th>
8. <th>Category Id</th>
9. <th>Price</th>
10. <th>Quantity Available</th>
11. <th>Action</th>
12. </tr>
13. <tr \*ngFor="let product of products" style="background-color:white">
14. </tr>
15. </table>
16. </div>
17. <div \*ngIf="showMsgDiv" style="">
18. <h4 class="jumbotron" style="text-align:center;">No products available</h4>
19. </div>
20. </div>

You can also use ngClass as shown below.

1. <h1 [ngClass]="'red size30'">View Products</h1>

**Step 3:**

Execute the application and observe the output.



**ngStyle**

ngStyle directive allows you to set the inline style of an HTML element using an expression.

**Step 1:**

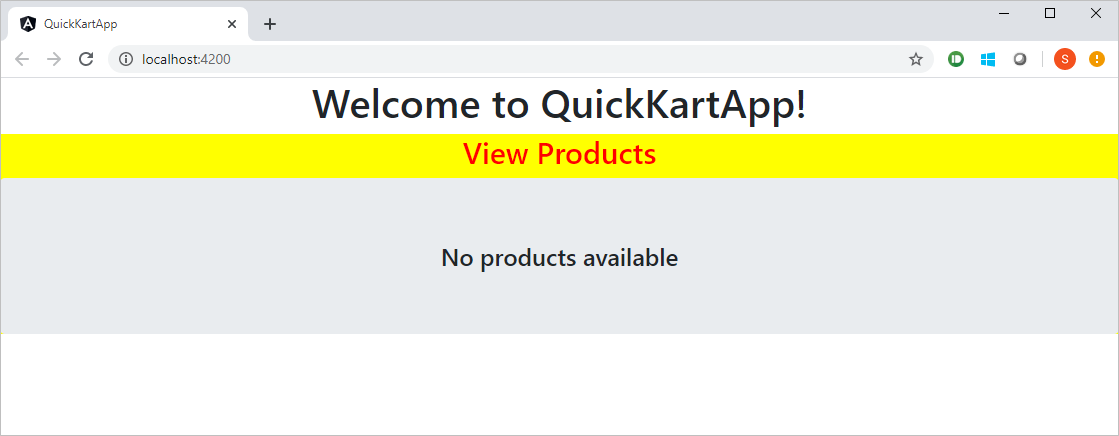
Add the below code in Line 1 to view-products.component.html.

1. <div [ngStyle]="{'background-color':'yellow'}" style="text-align:center;">
2. <h1 ngClass='red size30'>View Products</h1>
3. <div class="table-responsive">
4. <table class="table" style="border:5px solid rgba(220,230,242,1);" \*ngIf="products">
5. <tr style="background-color:rgba(220,230,242,1); font-size:12pt">
6. <th>Product Id</th>
7. <th>Product Name</th>
8. <th>Category Id</th>
9. <th>Price</th>
10. <th>Quantity Available</th>
11. <th>Action</th>
12. </tr>
13. <tr \*ngFor="let product of products" style="background-color:white">
14. </tr>
15. </table>
16. </div>
17. <div \*ngIf="showMsgDiv" style="">
18. <h4 class="jumbotron" style="text-align:center;">No products available</h4>
19. </div>
20. </div>

Here, if you observe [ngStyle] the [] represent property binding which you will be learning in detail in the upcoming demos.

**Step 2:**

Execute the application and observe the changes.



Discard all the changes you made in this demo and revert to the previous state of the demo before you proceed with the next demo.