**Angular Container and Nested Components**

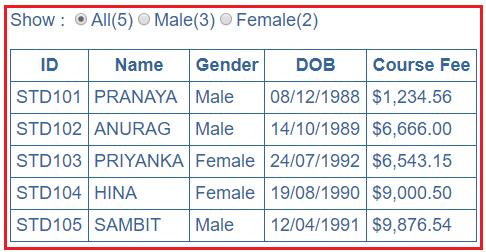
1. **What is a nested component?**
2. **What is a container component?**
3. **Example to understand container and nested components in angular.**

we will discuss the following things

1. **How to pass data from the nested component to container component?**
2. **How to pass data from the container component to the nested component?**
3. **What are component input and output properties?**
4. **How to create custom events using Event Emitter class?**

##### **Example:**

We are going to create the following user interface to understand all the above concepts.



As shown in the above image, we have a table that displays a list of students.  You can also see that, above the table, we have 3 radio buttons i.e. All, Male and Female and next to each radio button we also display the count of students as shown below.

Angular Nested Components

All(5) radio button has the total count of students i.e. both male and female students. Male(3) radio button has the total count of male students and similarly, the Female(2) radio button has the total count of female students.

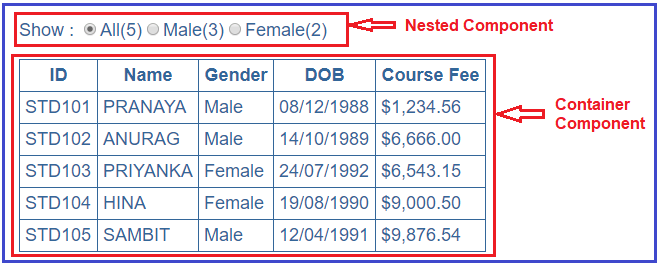
At the moment the All(5) radio button is selected, so all the students (both male and female) are displayed in the table. But, if we select the Male(3) radio button, then only the 3 male students to be displayed in the table. Similarly, if we select the Female(2) radio button then only the female students to be displayed in the table.

##### **How to achieve the above requirement?**

To achieve the above requirement, we need to create two components. One component will display the list of students in a table while the other component will display the radio buttons and the count of students.

To display the list of students, we will create a component called StudentListComponent and to display the radio buttons along with their total count of students, we will create a component with the name StudentCountComponet.

Then we will nest the StudentCountComponet within the StudentListComponet and when we do so, then the StudentCountComponet becomes the nested component or the child component and StudentListComponet becomes the container component or parent component. For better understanding please have a look at the following image.



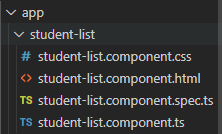
**Let us implement this step by step:**

##### **Step1: Creating StudentList Component**

To create the StudentList component, open visual studio code terminal and then type **ng g c StudentList** and press enter as shown below.

Creating StudentList Component

Once you press enter key, then it will create a folder with the name student-list within the app folder with four files as shown in the below image.



##### **Modify student-list.component.ts file:**

Open student-list.component.ts file and then copy and paste the following code in it. Here, we simply create an student array.

**import** **{** Component, OnInit **}** from '@angular/core';

@Component**({**

selector: 'app-student-list',

templateUrl: './student-list.component.html',

styleUrls: **[**'./student-list.component.css'**]**

**})**

**export** **class** StudentListComponent **{**

students: **any[]** = **[**

**{**

ID: 'std101', FirstName: 'Pranaya', LastName: 'Rout',

DOB: '12/8/1988', Gender: 'Male', CourseFee: 1234.56

**}**,

**{**

ID: 'std102', FirstName: 'Anurag', LastName: 'Mohanty',

DOB: '10/14/1989', Gender: 'Male', CourseFee: 6666.00

**}**,

**{**

ID: 'std103', FirstName: 'Priyanka', LastName: 'Dewangan',

DOB: '7/24/1992', Gender: 'Female', CourseFee: 6543.15

**}**,

**{**

ID: 'std104', FirstName: 'Hina', LastName: 'Sharma',

DOB: '8/19/1990', Gender: 'Female', CourseFee: 9000.50

**}**,

**{**

ID: 'std105', FirstName: 'Sambit', LastName: 'Satapathy',

DOB: '4/12/1991', Gender: 'Male', CourseFee: 9876.54

**}**

**]**;

**}**

##### **Modify student-list.component.css file**

Open **student-list.component.css** file and then copy and paste the following code in it.

table **{**

color: *#369*;

font-family: Arial, Helvetica, sans-serif;

font-size: large;

border-collapse: collapse;

**}**

td **{**

border: 1px solid *#369*;

padding:5px;

**}**

th**{**

border: 1px solid *#369*;

padding:5px;

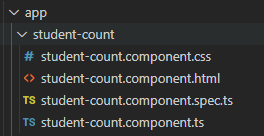
**}**

##### **Creating StudentCountComponent:**

To create the StudentCount component, type **ng g c StudentCount**in the visual studio code terminal and then press the enter key as shown in the below image.

Creating StudentCountComponent:

Once you press the enter button, then it will create a folder with the name student-count within the app folder with four files as shown in the below image.



##### **Modifying student-count.component.ts file:**

Open the **student-count.component.ts** file and then copy and paste the following code in it. As you can see, here we created three variables to hold the count.

**import** **{** Component, OnInit **}** from '@angular/core';

@Component**({**

selector: 'app-student-count',

templateUrl: './student-count.component.html',

styleUrls: **[**'./student-count.component.css'**]**

**})**

**export** **class** StudentCountComponent **{**

all: **number** = 5;

male: **number** = 3;

female: **number** = 2;

**}**

Here, we have set the **selector=’app-student-countt’**. So, we can use this selector as a directive where we want to use this component. We are going to use this **StudentCountComponent**within the **StudentListComponent**using the app-student-countt selector as a directive.

Within the **StudentCountComponent**, we declare 3 properties (all, male and Female) and at the moment we have hard-coded the values for these properties.

we will discuss how to pass the values to these properties from the container component i.e. from the StudnetListComponent.

##### **Modifying student-count.component.css file:**

Open student-count.component.css file and then copy and paste the following code in it.

**.radioClass** **{**

color: *#369*;

font-family: Arial, Helvetica, sans-serif;

font-size: large;

**}**

##### **Modifying app.module.ts file:**

In order to use the StudentCountComponent and StudnetListComponent component, you need to register it in the app.module.ts file. If you are creating the component using Angular CLI as we did, then the angular framework automatically register these components within the app.module.ts file. So, the complete code in app.module.ts file is shown below. But you need to change **bootstrap: [StudentListComponent].**

**import** **{** BrowserModule **}** from '@angular/platform-browser';

**import** **{** NgModule **}** from '@angular/core';

**import** **{** AppRoutingModule **}** from './app-routing.module';

**import** **{** AppComponent **}** from './app.component';

**import** **{** StudentListComponent **}** from './student-list/student-list.component';

**import** **{** StudentCountComponent **}** from './student-count/student-count.component';

@NgModule**({**

declarations: **[**

AppComponent,

StudentListComponent,

StudentCountComponent

**]**,

imports: **[**

BrowserModule,

AppRoutingModule

**]**,

providers: **[]**,

bootstrap: **[**StudentListComponent**]**

**})**

**export** **class** AppModule **{** **}**

##### **Modify student-count.component.html file:**

Open student-count.component.html file and then copy and paste the following code in it.

**<span** class="radioClass"**>**Show : **</span>**

**<input** type="radio" name="options" **/>**

**<span** class="radioClass"**>**{{"All(" + all + ")"}}**</span>**

**<input** name="options" type="radio"**>**

**<span** class="radioClass"**>**{{"Male(" + male + ")"}}**</span>**

**<input** name="options" type="radio"**>**

**<span** class="radioClass"**>**{{"Female(" + female + ")"}}**</span>**

As you can see in the above HTML, we have 3 radio buttons and bound them to the 3 properties (all, male, female) that we have in the component (**StudentCountComponent**) class. Here, we are using the interpolation technique for data-binding.

##### **Modify the student-list.component.html file:**

Now, we need to nest the **StudentCountComponent**in **StudentListComponent**and to do so, we need to use the StudentCountComponent selector (i.e. **app-student-coun**t) as a directive i.e. <**app-student-coun**></**app-student-coun**> on **StudentListComponent**.. So, open student-list.component.html file and then copy and paste the following code in it.

**<app-student-count></app-student-count>**

**<br/><br/>**

**<table>**

**<thead>**

**<tr>**

**<th>**ID**</th>**

**<th>**Name**</th>**

**<th>**Gender**</th>**

**<th>**DOB**</th>**

**<th>**Course Fee**</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr** \*ngFor='let student of students'**>**

**<td>**{{student.ID | uppercase}}**</td>**

**<td>**{{student.FirstName}}**</td>**

**<td>**{{student.Gender}}**</td>**

**<td>**{{student.DOB | date:'dd/MM/y'}}**</td>**

**<td>**{{student.CourseFee | currency:'USD':true:'1.2-2'}}**</td>**

**</tr>**

**<tr** \*ngIf="!students || students.length==0"**>**

**<td** colspan="10"**>**

No Students to display

**</td>**

**</tr>**

**</tbody>**

**</table>**

That’s it. We have done with the first phase of our implementation. At this point, if you run the application, then you should see the student count radio buttons and the student list. Except that nothing is going to work.

Goto index.html

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>DemoApp</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="icon" type="image/x-icon" href="favicon.ico">

</head>

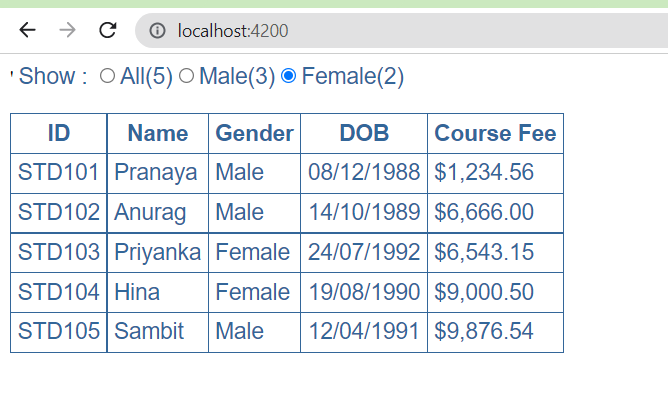
<body>'

  <app-student-list></app-student-list>

  <!-- <app-root></app-root> -->

</body>

</html>



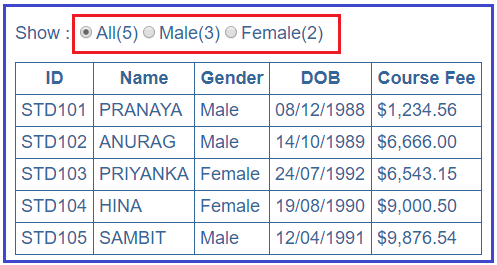
**Angular Component Input Properties**

##### **What are Angular Component Input Properties?**

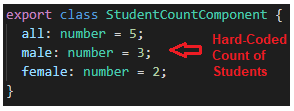
00

The Angular Component Input Properties are used to pass the data from container component to the nested component

As of now, we have hard-coded the count of students in the **StudentCountComponent**that are displayed next to each radio buttons as shown in the below image.



The following code in the **student-count.component.ts** file does this.



As shown in the above image the values for the 3 properties (all, male, female) are hard-coded. But, here we want the values for these 3 properties (all, male and female) to be passed from the container component i.e. **StudentListComponent**.

##### **How to pass the data from the container component to the nested component?**

We can pass the data from the container component to the nested component using the input property of the angular component. We can easily convert an angular property to an angular input property by using the **@Input** decorator.

In our example, to pass the values for these 3 properties (all, male and female) from the container component i.e. **StudentListComponent**to the nested component i.e. **StudentCountComponent**, we need to decorate the **StudentCountComponent**properties (all, male and female) with **@Input()** decorator.

When we decorate a property with **@Input()**decorator then that property becomes an input property. To use the **@Input()** decorator in your component, first, you need to import it from the **@angular/core**.

So, let’s modify the **StudentCountComponent**as shown below to use the Angular Component **Input**Properties. Open student-count.component.ts file and then copy and paste the following code in it.

**import** **{** Component, Input **}** from '@angular/core';

@Component**({**

selector: 'app-student-count',

templateUrl: './student-count.component.html',

styleUrls: **[**'./student-count.component.css'**]**

**})**

**export** **class** StudentCountComponent **{**

@Input**()**

all: **number**;

@Input**()**

male: **number**;

@Input**()**

female: **number**;

**}**

Here, you can to notice that we have removed the default hard-coded values, as we are going to pass the values for these properties from the Container Component i.e. **StudentListComponent**..

##### **How to Pass data from the parent component to the child component?**

We need to do two modifications in the **StudentListComponent**in order to pass the values from the parent component i.e. **StudentListComponent**to the child component i.e. **StudentCountComponent**.

##### **Modifying student-list.component.ts file:**

So open the student-list.component.ts file and then copy and paste the below code. As you can see in the below code, we have created three methods that are going to return the male students count, female students count and total students count.

**import** **{** Component**}** from '@angular/core';

@Component**({**

selector: 'app-student-list',

templateUrl: './student-list.component.html',

styleUrls: **[**'./student-list.component.css'**]**

**})**

**export** **class** StudentListComponent **{**

students: **any[]** = **[**

**{**

ID: 'std101', FirstName: 'Pranaya', LastName: 'Rout',

DOB: '12/8/1988', Gender: 'Male', CourseFee: 1234.56

**}**,

**{**

ID: 'std102', FirstName: 'Anurag', LastName: 'Mohanty',

DOB: '10/14/1989', Gender: 'Male', CourseFee: 6666.00

**}**,

**{**

ID: 'std103', FirstName: 'Priyanka', LastName: 'Dewangan',

DOB: '7/24/1992', Gender: 'Female', CourseFee: 6543.15

**}**,

**{**

ID: 'std104', FirstName: 'Hina', LastName: 'Sharma',

DOB: '8/19/1990', Gender: 'Female', CourseFee: 9000.50

**}**,

**{**

ID: 'std105', FirstName: 'Sambit', LastName: 'Satapathy',

DOB: '4/12/1991', Gender: 'Male', CourseFee: 9876.54

**}**

**]**;

getTotalStudentsCount**()**: **number** **{**

**return** this.students.length;

**}**

getMaleStudentsCount**()**: **number** **{**

**return** this.students.filter**(**std => std.Gender === 'Male'**)**.length;

**}**

getFemaleStudentsCount**()**: **number** **{**

**return** this.students.filter**(**std => std.Gender === 'Female'**)**.length;

**}**

**}**

The points to remember is that in the filter method we are using the triple equals (===) instead of the double equals (==). The meaning single, double and triple equals in TypeScript are as follows

1. = Assign a value
2. == Compare two values
3. === Compare two values and their types

##### **Modify student-list.component.html file:**

Open student-list.component.html file and then copy and paste the following code in it. As you can see in the above HTML, within the **<app-student-count>** directive we are using the property binding to bind the properties (all, male, female) of the nested component i.e. **StudentCountComponent** with the 3 methods in the container component i.e. **StudentListComponent**.

**<app-student-count** [all]="getTotalStudentsCount()"

[male]="getMaleStudentsCount()"

[female]="getFemaleStudentsCount()"**></app-student-count>**

**<br/><br/>**

**<table>**

**<thead>**

**<tr>**

**<th>**ID**</th>**

**<th>**Name**</th>**

**<th>**Gender**</th>**

**<th>**DOB**</th>**

**<th>**Course Fee**</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr** \*ngFor='let student of students'**>**

**<td>**{{student.ID | uppercase}}**</td>**

**<td>**{{student.FirstName | uppercase}}**</td>**

**<td>**{{student.Gender}}**</td>**

**<td>**{{student.DOB | date:'dd/MM/y'}}**</td>**

**<td>**{{student.CourseFee | currency:'USD':true:'1.2-2'}}**</td>**

**</tr>**

**<tr** \*ngIf="!students || students.length==0"**>**

**<td** colspan="10"**>**

No Students to display

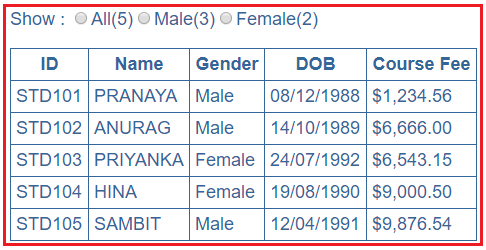
**</td>**

**</tr>**

**</tbody>**

**</table>**

That’s it. We have done with our part 2 implementation. So save all the changes and run the application and you will see the correct count of students next to each radio button as shown in the image below.



To confirm whether it’s working or not, let’s add the following student object to the student’s array in the **StudentListComponent** as shown below.

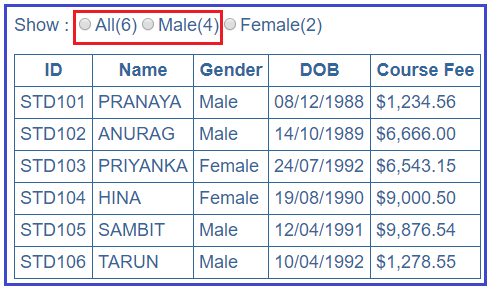
{

ID: 'std106', FirstName: 'Tarun', LastName: 'Mallick',

DOB: '4/10/1992', Gender: 'Male', CourseFee: 1278.55

}

Now, save all the changes and then reload the web page and you will notice that All count and the Male count is increased by 1 as expected as shown in the image below.



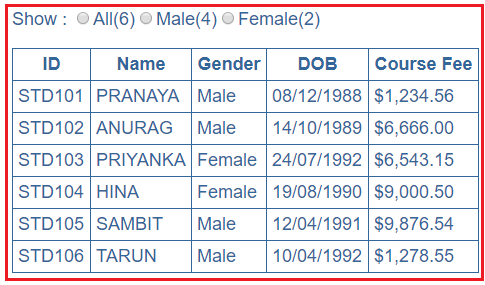
At the moment when we click on the radio buttons, then nothing is happening. So in the next article we will discuss [**how to pass the data from the child component to the parent component**](https://dotnettutorials.net/lesson/angular-component-output-properties/)i.e. when a radio button’s checked event is raised in the child component i.e. **StudentCountComponent**, we want to know about it in the parent component i.e. **StudentListComponent** so that we can react to them and then decide which students to show in the table depending on the selection of the radio button.

## ****Angular Component Output Properties****

1. **How to pass the user actions or user entered values or selections from the child component to the parent component using output properties.**
2. **How to create custom events using angular EventEmitter class**
3. **What are ng-container directive and its use**

##### **What are Angular Component Output Properties?**

The **Angular Component Output Properties**are used to to pass the data from the nested component to the container component.



At the moment when you click any of the radio buttons, then nothing is happening. Let first discuss what we want to do here.

When All(6) radio button is clicked then we need to display all the students in the table. When Male(4) radio button is clicked then we need to display only the 4 Male students in the table similarly when the Female(2) radio button is clicked then we only need to display the 2 Female students in the table.

##### **How to achieve this?**

To achieve this here we are going to make use of the **Angular component Output Properties**. First, let’s discuss the changes that are required in the nested component i.e. **StudentCountComponent**.

##### **Modify student-count.component.ts file:**

Open student-count.component.ts file and then copy and paste the following code in it. The changes we have done here are self-explained so please go through the comments.

// Import Output and EventEmitter from angular

**import** **{** Component, Input, Output, EventEmitter **}** from '@angular/core';

@Component**({**

selector: 'app-student-count',

templateUrl: './student-count.component.html',

styleUrls: **[**'./student-count.component.css'**]**

**})**

**export** **class** StudentCountComponent **{**

@Input**()**

all: **number**;

@Input**()**

male: **number**;

@Input**()**

female: **number**;

// This variable holds the selected value of the radio button

selectedRadioButtonValue: **string** = 'All';

// The Output decorator makes the property of an Component as an Output property

// The EventEmitter class in Angular is used to create the custom event

// When the radio button selection changes, the selected radio button

// value which is a string gets passed to the event handler method.

// Hence, the event payload is string.

@Output**()**

countRadioButtonSelectionChanged: EventEmitter**<string>** =

new EventEmitter**<string>()**;

// This method raises the custom event. We will bind this

// method to the change event of all the 3 radio buttons

onRadioButtonSelectionChange**()** **{**

this.countRadioButtonSelectionChanged

.emit**(**this.selectedRadioButtonValue**)**;

**}**

**}**

##### 

##### **Modify student-count.component.html**

The following are the changes that are required in the view template of StudentCountComponent i.e. **student-count.component.html**. Notice here we have made 3 changes on each radio button.

1. The value attribute is set to (All, Male or Female)
2. We implemented the 2-way data-binding using the ngModel directive. Notice the ngModel is bound to selectedRadioButtonValue property of the component class. This 2-way data-binding ensures whenever the radio button selection changes, the selectedRadioButtonValue property is updated with the value of the selected radio button.
3. The onRadioButtonSelectionChange() method is binded to the “change” event of the radio button. That means whenever, the selection of the radio button changes, onRadioButtonSelectionChange() method raises the custom event “countRadioButtonSelectionChanged”. We defined this custom event using Angular EventEmitter class.

##### **Modify student-list.component.ts file**

Now let’s have a look at the changes that are required in the parent component i.e. StudentListComponent. The following are the changes that are required in the StudentListComponent class. The changes are commented and self-explained.

**import** **{** Component**}** from '@angular/core';

@Component**({**

selector: 'app-student-list',

templateUrl: './student-list.component.html',

styleUrls: **[**'./student-list.component.css'**]**

**})**

**export** **class** StudentListComponent **{**

// This property will keep track of the radio button which is selected

// We have set the default value to All, so all the students

// are displayed in the table by default

selectedStudentCountRadioButton: **string** = 'All';

students: **any[]** = **[**

**{**

ID: 'std101', FirstName: 'Pranaya', LastName: 'Rout',

DOB: '12/8/1988', Gender: 'Male', CourseFee: 1234.56

**}**,

**{**

ID: 'std102', FirstName: 'Anurag', LastName: 'Mohanty',

DOB: '10/14/1989', Gender: 'Male', CourseFee: 6666.00

**}**,

**{**

ID: 'std103', FirstName: 'Priyanka', LastName: 'Dewangan',

DOB: '7/24/1992', Gender: 'Female', CourseFee: 6543.15

**}**,

**{**

ID: 'std104', FirstName: 'Hina', LastName: 'Sharma',

DOB: '8/19/1990', Gender: 'Female', CourseFee: 9000.50

**}**,

**{**

ID: 'std105', FirstName: 'Sambit', LastName: 'Satapathy',

DOB: '4/12/1991', Gender: 'Male', CourseFee: 9876.54

**}**,

**{**

ID: 'std106', FirstName: 'Tarun', LastName: 'Mallick',

DOB: '4/10/1992', Gender: 'Male', CourseFee: 1278.55

**}**

**]**;

getTotalStudentsCount**()**: **number** **{**

**return** this.students.length;

**}**

getMaleStudentsCount**()**: **number** **{**

**return** this.students.filter**(**std => std.Gender === 'Male'**)**.length;

**}**

getFemaleStudentsCount**()**: **number** **{**

**return** this.students.filter**(**std => std.Gender === 'Female'**)**.length;

**}**

// Depending on the radio button which is selected, this method updates

// the selectedStudentCountRadioButton property

// This method is going to called when the child component (StudentCountComponent)

// raises the custom event - countRadioButtonSelectionChanged

// The event binding is specified in StudentList.component.html

onStudentCountRadioButtonChange**(**selectedRadioButtonValue: **string)**: **void** **{**

this.selectedStudentCountRadioButton = selectedRadioButtonValue;

**}**

##### **Modify student-list.component.html**

The following are the changes that are required in the view template of StudentListComponent i.e. student-list.component.html.

The **onStudentCountRadioButtonChange($event)** method is bound to the custom event – countRadioButtonSelectionChanged. The $event object will have the selected radio button value as that is what is passed as the event payload from the nested component. The event handler method (onStudentCountRadioButtonChange()) in the component class updates the property “selectedStudentCountRadioButton”. This property is then used along with the \*ngIf structural directive to decide which student objects to display in the table.

On the <tr> element, we are using “ngIf” directive along with selectedStudentCountRadioButton property which controls the student objects to display.

Notice, just above the <tr> element, we have introduced <ng-container> element and the “ngFor” directive is placed on this element. If you are wondering why we have done this, Angular does not allow multiple structural directives to be placed on one element as shown below.

<app-student-count [all]="getTotalStudentsCount()" [male]="getMaleStudentsCount()" [female]="getFemaleStudentsCount()" (countRadioButtonSelectionChanged)="onStudentCountRadioButtonChange($event)"> </app-student-count> <br/><br/> <table> <thead> <tr> <th>ID</th> <th>Name</th> <th>Gender</th> <th>DOB</th> <th>Course Fee</th> </tr> </thead> <tbody> <ng-container \*ngFor="let student of students;"> <tr \*ngIf="selectedStudentCountRadioButton=='All' || selectedStudentCountRadioButton==student.Gender"> <td>{{student.ID | uppercase}}</td> <td>{{student.FirstName | uppercase}}</td> <td>{{student.Gender}}</td> <td>{{student.DOB | date:'dd/MM/y'}}</td> <td>{{student.CourseFee | currency:'USD':true:'1.2-2'}}</td> </tr> </ng-container> <tr \*ngIf="!students || students.length==0"> <td colspan="10"> No Students to display </td> </tr> </tbody> </table>

##### **Modify app.module.ts:**

Open app.module.ts file and then copy and paste the following code in it.

**import** **{** BrowserModule **}** from '@angular/platform-browser';

**import** **{** NgModule **}** from '@angular/core';

**import** **{** FormsModule **}** from '@angular/forms';

**import** **{** AppRoutingModule **}** from './app-routing.module';

**import** **{** AppComponent **}** from './app.component';

**import** **{** StudentListComponent **}** from './student-list/student-list.component';

**import** **{** StudentCountComponent **}** from './student-count/student-count.component';

@NgModule**({**

declarations: **[**

AppComponent,

StudentListComponent,

StudentCountComponent

**]**,

imports: **[**

BrowserModule,

AppRoutingModule,

FormsModule

**]**,

providers: **[]**,

bootstrap: **[**StudentListComponent**]**

**})**

**export** **class** AppModule **{** **}**

That’s it. We have done with our implementation. Now save all the changes and then run the application and you will see based on the selected radio button, the students are displayed in the table.

