Component In Angular 2 and How to Create Nested Component in Angular 2

**Angular 2 Components**

We all know that Angular 2 is a component based framework because everything is a component in Angular 2. Basically, components are the basic building blocks of Angular 2 application. It also allows us to create reusable UI templates.

A component in Angular 2 is a class with a template and a decorator. There are basically the following parts of Angular 2 component –

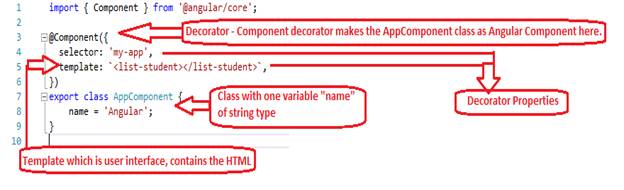
* ***Class***  
  It is very similar to the C# class or java class etc. It contains the constructor, variables and methods code which is required for the template/user interface.
* ***Decorator***  
  A decorator is used to store the metadata about the class. Basically, the decorator provided by Angular makes a class an Angular component when it is decorated with the component decorator.

Angular 2 provides us basically 4 types of decorators,

1. ***Class Decorators***  
   For Ex. @NgModule, @Component & @Directive
2. ***Property Decorators***  
   For Ex. @Input & @Output
3. ***Method Decorators***  
   For Ex. @HostListener
4. ***Parameter Decorators***  
   For Ex. @Inject

Each decorator has a basic configuration using several properties. We are taking a look at some possible configuration properties right here, that you can use when creating a component,

1. **Selector**– This is used for identifying this component in templates.
2. **Template**– This is used for defining HTML template inline for the view.
3. **TemplateUrl**– This is used for defining a URL to an external file containing a template for the view.
4. **Styles**– This is used for defining inline CSS to be applied to the template of this component.
5. **StyleUrls**– This is used for defining URLs to external style sheets to be applied to the templates of this component.

Let’s understand the component and its parts with the help of below screenshot and component code.  
  
  
  
This shows where the decorators are actually applied to a class and how they are actually responsible for making a class an Angular component.

**App.component.ts**

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

@Component({

  selector: 'my-app',

  template: `<list-student></list-student>`,

})

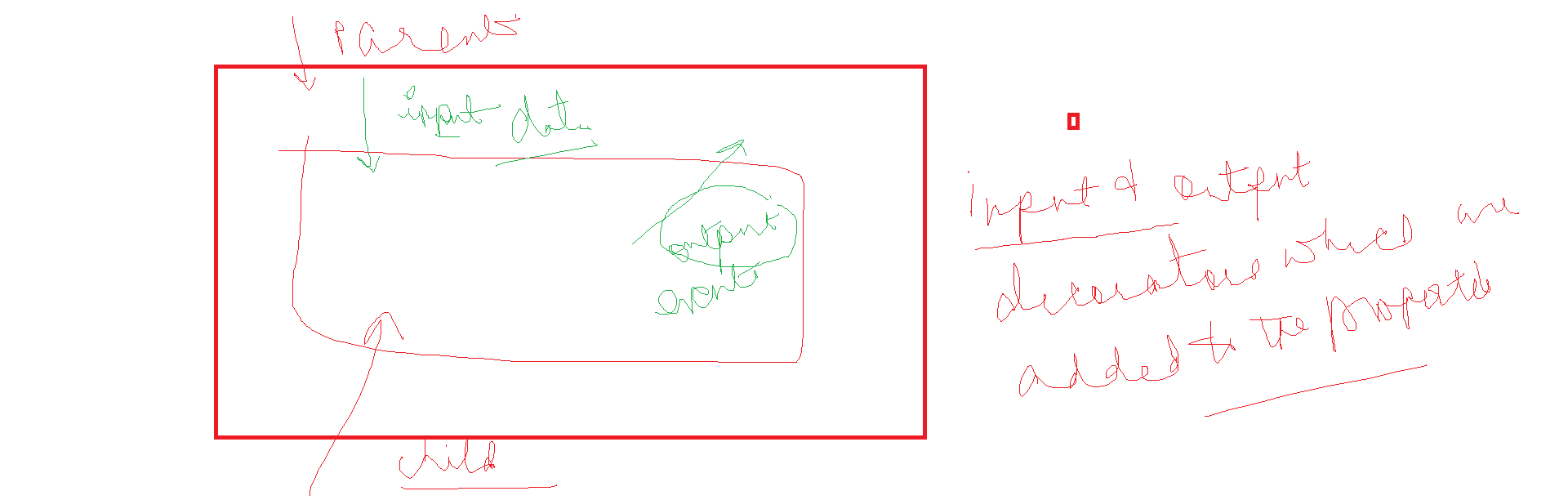
**export** **class** AppComponent {

    name = 'Angular';

}

**Nested Components**

How to create a nested component (parent component and child component) in Angular 2 applications and how parent and child components will communicate with each other.



## A tool to exchange data

First of all, the idea of Input and Output is to exchange data between components. They are a mechanism to send/receive data from one component to another.

* Input is used to receive data in
* whereas Output is used to send data out.
* Output sends data out by exposing event producers, usually EventEmitterobjects.

@Component({  
 selector: 'todo-item',  
 ...  
})export class TodoItemComponent {  
 @Input() item  
 @Output() onChange = new EventEmitter()  
}

that means

* hey, I am expecting data being sent to me. I will receive it and store it into my *item*property
* by the way, I will produce and send data out via the *onChange*property.

Main comp

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

@Component({

selector: 'main-comp',

templateUrl: './main-comp.component.html',

styleUrls: ['./main-comp.component.css']

})

export class MainCompComponent implements OnInit {

constructor() { }

name : string="Ajay";

ngOnInit() {

}

}

Main comp.html

<called-comp [data]= "name"> </called-comp>

In child comp

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

@Component({

selector: 'called-comp',

templateUrl: './called-comp.component.html',

styleUrls: ['./called-comp.component.css']

})

export class CalledCompComponent implements OnInit {

@Input() data;

constructor() { }

ngOnInit() {

}

}

Called comp.html

<p>called-comp works!</p>

<h1> Hi I am in called comp {{data}} </h1>

<input type="text" class="form-control" [value]="data">

OutPut Dec

In Main.comp

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

@Component({

selector: 'main-comp',

templateUrl: './main-comp.component.html',

styleUrls: ['./main-comp.component.css']

})

export class MainCompComponent implements OnInit {

constructor() { }

name : string="Ajay";

ngOnInit() {

}

IamCalledFromChild(value : string)

{

console.log(value);

}

}

Main.html

<called-comp [data]= "name" (onClick)="IamCalledFromChild($event)"> </called-comp>

Called.comp

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

@Component({

selector: 'called-comp',

templateUrl: './called-comp.component.html',

styleUrls: ['./called-comp.component.css']

})

export class CalledCompComponent implements OnInit {

@Input() data;

@Output() onClick= new EventEmitter();

constructor() { }

ngOnInit() {

}

}

Called.html

<button (click)="onClick.emit('Hi')"> CLICK HERE </button>

With a method in called comp

Main comp

IamCalledFromChild2(value : string)

{

alert("I am back");

alert(value);

console.log(value);

}

Main.html

<called-comp [data]= "name" (OnClick2)="IamCalledFromChild2($event)"> </called-comp>

Called comp

@Input() data;

@Output() onClick= new EventEmitter();

@Output() OnClick2= new EventEmitter<string>();

constructor() { }

onClick2(value : string) : void

{

alert(value);

this.OnClick2.emit(value.toUpperCase());

}

Called html

<p>called-comp works!</p>

<input type="text" class="form-control" [value]="data">

<h1> Hi I am in called comp {{data}} </h1>

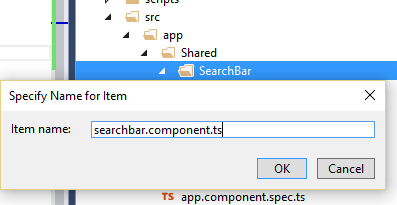
<button (click)="onClick.emit('Hi')"> CLICK HERE </button>

<button (click)="onClick2(data)"> CLICK HERE FOR CLICK 2</button>

Let’s understand it with the following steps –

Since we want to make search-bar component as a shared component, we will keep related files in a new folder i.e. shared/searchbar.

**Step 1.** **Add the search bar component TypeScript file**

Add a new folder named “Shared” inside app folder and create another folder “SearchBar” inside this “Shared” folder to add the component files. And now, create a TypeScript file “searchbar.component.ts” under this “SearchBar” folder.  
  
  
  
Add the below code into the “searchbar.component.ts” file.

**searchbar.component.ts**

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

@Component({

    selector: 'search-bar',

    templateUrl: 'searchbar.component.html',

    moduleId: module.id

})

**export** **class** SearchBarComponent {

    @Output()

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

    OnStudentSearch(searchTerm:string): **void** {

**this**.Search.emit(searchTerm);

    }

}

In the above code, we imported two components, Output and EventEmitter, from Angular core module. With the help of these two, we will inform the parent component when OnStudentSearch method will be called. We created a custom event “Search” which will emit when OnStudentSearch method will be called, so parent component will know about the event. The emit function has been used to fire this custom event.

We have mentioned the search bar template URL in templateUrl property, which we will create in a little bit.

**Step 2.** **Add the search bar template file**

Add an html searchbar.component.html file inside the same app -> Shared -> “SearchBar” folder and add the below code into this file,

<div **class**="form-group">

    <div **class**="md-col-4">

        <label>Enter Student Name:</label>

    </div>

    <div **class**="md-col-4">

        <input **class**="form-control" #searchInput type="text" />

    </div>

</div>

<button type="submit" **class**="btn btn-default" (click)="OnStudentSearch(searchInput.value)">

    Search

</button>

Now our searchbar component is complete to nest inside the existing StudentList component.

**Step 3.** **Nest the SearchBar component inside the StudentList component**

Open the existing student.component.html file from app -> student folder and added the below code as highlighted,

<search-bar (Search)="OnStudentSearch($event)"></search-bar>

<table **class**="table table-responsive table-bordered table-striped">

    <thead>

        <tr>

            <th>Student ID</th>

            <th>Name</th>

            <th>Gender</th>

            <th>Age</th>

            <th>Course</th>

            <th>DOB</th>

            <th>Grade</th>

            <th>Rating</th>

        </tr>

    </thead>

    <tbody>

        <tr \*ngFor="let s of students;">

            <td>{{s.studentID}}</td>

            <td>{{s.studentName | uppercase}}</td>

            <td>{{s.gender | lowercase}}</td>

            <td>{{s.age}}</td>

            <td>{{s.course | courseCategory}}</td>

            <td>{{s.DOB | date:'yMMMMd' | uppercase }}</td>

            <td>{{s.grade | percent:'.2'}}</td>

            <td>{{s.rating | number:'2.1-2'}}</td>

        </tr>

    </tbody>

</table>

In the above code, you can see how it has been nested with the StudentList component. This is how we can create nested component in Angular 2.

**Step 4.** **Update StudentList component for search functionality**

To filter the student list, we will update the LoadStudents method which will provide us the result after filtering the students with the help of search term passed from child component i.e. SearchBar component. Take a look at the code highlighted below.

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

@Component({

    selector: 'list-student',

    templateUrl: 'app/student/student.component.html'

})

**export** **class** StudentListComponent **implements** OnInit {

    students: any[];

**public** LoadStudents(filterText: string): **void** {

**this**.students = [

            { studentID: 1, studentName: 'Steve', gender: 'Male', age: 35, course: 'MCA', DOB: '10/12/1982', grade:0.7500,rating:7.5123 },

            { studentID: 2, studentName: 'Bobby', gender: 'Male', age: 32, course: 'MBA', DOB: '12/1/1985', grade: 0.7850, rating: 7.8223 },

            { studentID: 3, studentName: 'Rina', gender: 'Female', age: 45, course: 'B.Tech', DOB: '9/11/1972', grade: 0.8525, rating: 8.5263 },

            { studentID: 4, studentName: 'Alex', gender: 'Female', age: 24, course: 'M.Tech', DOB: '1/1/1993', grade: 0.5540, rating: 5.5123 },

            { studentID: 5, studentName: 'Rahul', gender: 'Male', age: 26, course: 'MCA', DOB: '1/21/1991', grade: 0.9550, rating: 9.5534 },

        ];

**if** (filterText != "") {

**var** filterStudentList: any[] = [];

**this**.students.forEach(stu => {

**if** (stu.studentName.toLowerCase().includes(filterText)) {

                    filterStudentList.push(stu);

                }

            })

**this**.students = filterStudentList;

        }

    }

    Ng

OnInit() {

**this**.LoadStudents("");

    }

    OnStudentSearch(searchTerm: string): **void** {

**this**.LoadStudents(searchTerm);

    }

}

**Step 5.** **Register the SearchBar component in Angular app module**

Now, we will include our SearchBar component in the Angular app module. Add the code as highlighted below.

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

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

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

**import** { StudentListComponent } from './student/student.component'

**import** { courseCategoryPipe } from './student/student.coursepipe'

**import** { SearchBarComponent } from './Shared/SearchBar/searchbar.component'

@NgModule({

    imports: [BrowserModule],

    declarations: [AppComponent, StudentListComponent, courseCategoryPipe, SearchBarComponent],

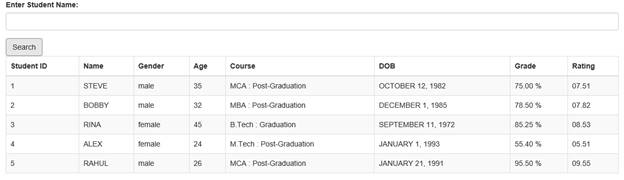
  bootstrap:    [ AppComponent ]

})

**export** **class** AppModule { }

**Step 6. Now, run the application to get the expected result**

Run the application using F5 or Ctrl + F5 and see the search bar as expected in the browser.

**Output  
  
**Let’s take a look at search functionality. Let’s enter the term “Rahul” in the search box and hit the "Search" button to get the student list.

**Output on filter  
  
**

<https://www.c-sharpcorner.com/article/what-are-components-in-angular-and-nested-components/>