Child/Nested Components in Angular

4 Comments / 6 minutes of reading / March 9, 2023

← ngmodelchange Angular Tutorial Angular Directives →

In this tutorial, we will learn how to Add a child Component to our Angular Application. We use the Angular CLI command `ng new` to create the new Angular Project. It creates `AppComponent`, which is the root component of our application. The `AppComponent` is bootstrapped in the `AppModule` and loaded in the `index.html` file using the selector `<app-root>Loading…</app-root>`. In this guide let us add another component to the app.

Table of Contents

What is a Child/Nested Component

Create a new application

How to add Child Component

Adding a Child Component in Angular

- Create the Child Component
- Import the Child Component in the Module
- Tell angular where to display the component
- Summary

What is a Child/Nested Component

The Angular follows component-based Architecture, where each Angular Component manages a specific task or workflow. Each Component is an independent block of the reusable unit.

In real life, angular applications will contain many Angular Components. The task of the root component is to just host these child components. These child components, in turn, can host the more child components creating a Tree-like structure called Component Tree.

In this tutorial, we will learn how to create a Child or nested components and host it in the App Component.

Create a new application

Create a new Angular application using the following command

```

ng new childComponent

```

Run the app and verify everything is ok.

```

ng serve

```

How to add Child Component

1. Create the Child Component. In the child Component, metadata specify the selector to be used
2. Import the Child Component in the module class and declare it in declaration Array

3. Use the CSS Selector to specify in the Parent Component Template, where you want to display the Child Component

Adding a Child Component in Angular

Now, let us add a Child Component to our project. In our child component, let us display a list of customers.

Create the Child Component

Creating the Child Component is no different from creating any other Parent Component. But, first, we need a customer class

Customer Class

Go to the app folder and create a file and name it as customer.ts . Copy the following code

```

export class Customer {

  customerNo: number;
  name:string ;
  address:string;
  city:string;
  state:string;
  country:string;

}

```

Note that we have used the `export` keyword. This enables us to use the above class in our components by importing it.

Create Child Component

In the `app` folder and create a new file and name it as `customer-list.component.ts` .

The code for `component-list.component.ts` is shown below

```

import { Component } from '@angular/core';
import { Customer } from './customer';

@Component({
  selector: 'customer-list',
  templateUrl: './customer-list.component.html'
})
export class CustomerListComponent
{
  customers: Customer[] = [

    {customerNo: 1, name: 'Rahuld Dravid', address: '', city: 'Banglaore', state: 'Karnataka',
    {customerNo: 2, name: 'Sachin Tendulkar', address: '', city: 'Mumbai', state: 'Maharastr
    {customerNo: 3, name: 'Saurrav Ganguly', address: '', city: 'Kolkata', state: 'West Beng
    {customerNo: 4, name: 'Mahendra Singh Dhoni', address: '', city: 'Ranchi', state: 'Bihar'
    {customerNo: 5, name: 'Virat Kohli', address: '', city: 'Delhi', state: 'Delhi', country: 'In

  ]
}

```

First, we import the required modules & classes. Our component requires `Customer` class, hence we import it along with the Component.

```
import { Component } from '@angular/core';
import { Customer } from './customer';
```

The next step is to add the @Component directive. The selector clause has the value `customer-list`. We need to use this in our parent view to display our view. The `templateURL` is `customer-list.component.html`, which we yet to build.

```
@Component({
  selector: 'customer-list',
  templateUrl: './customer-list.component.html'
})
```

The last step is to create the Component class. We name it as `CustomerListComponent`. The class consists of a single property, which is a collection of `customers`. We initialize the customers collection with some default values. In real-life situations, you will make use of the HTTP Client to get the data from the back end server.

```
export class CustomerListComponent
{
  customers: Customer[] = [

    {customerNo: 1, name: 'Rahuld Dravid', address: '', city: 'Banglaore', state: 'Karnataka',
    {customerNo: 2, name: 'Sachin Tendulkar', address: '', city: 'Mumbai', state: 'Maharastr
    {customerNo: 3, name: 'Saurrav Ganguly', address: '', city: 'Kolkata', state: 'West Beng
    {customerNo: 4, name: 'Mahendra Singh Dhoni', address: '', city: 'Ranchi', state: 'Bihar'
    {customerNo: 5, name: 'Virat Kohli', address: '', city: 'Delhi', state: 'Delhi', country: 'Inc

  ]
}

```

Creating the View

The next step is to create the View to display the list of customer. Go to the `app` folder and create the file with the name `customer-list.component.html`

```

<h2>List of Customers</h2>

<table class='table'>
  <thead>
    <tr>
      <th>No</th>
      <th>Name</th>
      <th>Address</th>
      <th>City</th>
      <th>State</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let customer of customers;">
      <td>{{customer.customerNo}}</td>
      <td>{{customer.name}}</td>
      <td>{{customer.address}}</td>
      <td>{{customer.city}}</td>
      <td>{{customer.state}}</td>
    </tr>
  </tbody>
</table>

```

To iterate through the Customer collection, we have used the ngFor Directive provided by the Angular. We have a separate tutorial, which discusses the ngFor Directive

The syntax for `ngFor` directive starts with `*ngFor`. The `*` indicates that it is a structural directive. i.e a directive that adds or removes the HTML elements to or from the DOM.

The expression `let customer of customers` is assigned to `*ngFor`. The `let` clause assigns the instance of `customer` object from the `Customers` collection to the template reference variable or local variable `customer`.

We use the template reference variable `customer` is to build the template to display the details of the customer to the user. The `ngFor` directive is applied to the `tr` element of the table. The Angular repeats everything inside the `tr` element in the DOM tree.

`{{customer.customerNo}}` is stands for `interpolation in Angular`. Angular evaluates everything within the `{{ }}` and replaces the string with the result.

Import the Child Component in the Module

Every [Component](), [directive](), [pipes]() we build must belong to an [Angular Module](). To do that we need to register our component in the Module. A Component, directive, pipes cannot be part of more than one module.

The Angular Modules (or NgModules) are Angular ways of organizing related components, directives, pipes and services, etc into a group. To add a component to a module, you need to declare it in the declarations metadata of the Angular Module

Angular creates a top-level root module (AppModule in file app.module.ts) when we create a new Angular app. That is where we need to register our CustomerListComponent

Open the app.module.ts under the app folder and update the code as shown below

```

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

import {NgbModule} from '@ng-bootstrap/ng-bootstrap';

import { AppComponent } from './app.component';
import {CustomerListComponent} from './customer-list.component';

@NgModule({
  declarations: [
    AppComponent, CustomerListComponent
  ],
  imports: [
    BrowserModule,NgbModule.forRoot()
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Registering the component or directive in the module requires two steps

First, import it

```
import { AppComponent } from './app.component';
import {CustomerListComponent} from './customer-list.component';
```

And, then declare it in declaration array

```
@NgModule({
declarations: [
   AppComponent, CustomerListComponent
]
```

Applies to: Angular 5 to the latest edition i.e. Angular 8, Angular 9. Angular 10, Angular 11

Tell angular where to display the component

Finally, we need to inform the Angular, where to display the child Component

We want our child Component as the child of the `AppComponent`. Open the `app.component.html` and add the following template

```
<h1>{{title}}. </h1>

<customer-list></customer-list>
```