**Services**

We might come across a situation where we need some code to be used everywhere on the page. It can be for data connection that needs to be shared across components, etc. Services help us achieve that. With services, we can access methods and properties across other components in the entire project.

To create a service, we need to make use of the command line. The command for the same is −

touch myservice.ts

ng g c compname

ng g service myservice

C:\projectA4\Angular 4-app>ng g service myservice

installing service

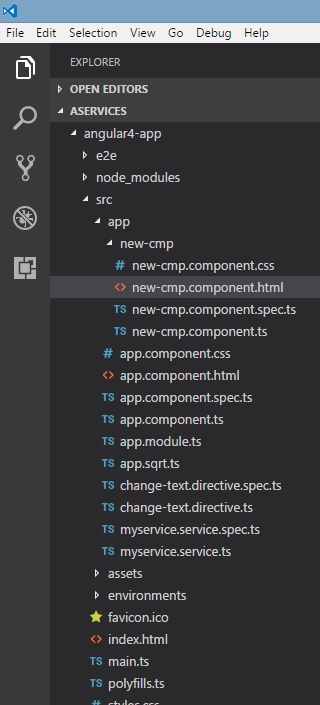
create src\app\myservice.service.spec.ts

create src\app\myservice.service.ts

WARNING Service is generated but not provided, it must be provided to be used

C:\projectA4\Angular 4-app>

The files are created in the app folder as follows −



Following are the files created at the bottom - **myservice.service.specs.ts** and **myservice.service.ts**.

myservice.service.ts

import { Injectable } from '@angular/core';

@Injectable()

export class MyserviceService {

constructor() { }

fun1(){}

fun2(){}

fun3(){}

}

Here, the Injectable module is imported from the **@angular/core**.

It contains the **@Injectable** method and a class called **MyserviceService**. We will create our service function in this class.

Before creating a new service, we need to include the service created in the main parent **app.module.ts**.

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

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

import { RouterModule} from '@angular/router';

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

import { MyserviceService } from './myservice.service';

import { NewCmpComponent } from './new-cmp/new-cmp.component';

import { ChangeTextDirective } from './change-text.directive';

import { SqrtPipe } from './app.sqrt';

@NgModule({

declarations: [

SqrtPipe,

AppComponent,

NewCmpComponent,

ChangeTextDirective

],

imports: [

BrowserModule,

RouterModule.forRoot([

{

path: 'new-cmp',

component: NewCmpComponent

}

])

],

providers: [MyserviceService],

bootstrap: [AppComponent]

})

export class AppModule { }

We have imported the Service with the class name and the same class is used in the providers. Let us now switch back to the service class and create a service function.

In the service class, we will create a function which will display today’s date. We can use the same function in the main parent component **app.component.ts** and also in the new component **new-cmp.component.ts** that we created in the previous chapter.

Let us now see how the function looks in the service and how to use it in components.

import { Injectable } from '@angular/core';

@Injectable()

export class MyserviceService {

constructor() { }

showTodayDate() {

let ndate = new Date();

return ndate;

}

}

In the above service file, we have created a function **showTodayDate**. Now we will return the new Date () created. Let us see how we can access this function in the component class.

app.component.ts

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

import { MyserviceService } from './myservice.service';

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Angular 4 Project!';

todaydate;

constructor(private myservice: MyserviceService) {}

ngOnInit() {

this.todaydate = this.myservice.showTodayDate();

}

}

The **ngOnInit** function gets called by default in any component created. The date is fetched from the service as shown above. To fetch more details of the service, we need to first include the service in the component **ts** file.

We will display the date in the **.html** file as shown below −

{{todaydate}}

<app-new-cmp></app-new-cmp>

// data to be displayed to user from the new component class.

Let us now see how to use the service in the new component created.

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

import { MyserviceService } from './../myservice.service';

@Component({

selector: 'app-new-cmp',

templateUrl: './new-cmp.component.html',

styleUrls: ['./new-cmp.component.css']

})

export class NewCmpComponent implements OnInit {

todaydate;

newcomponent = "Entered in new component created";

constructor(private myservice: MyserviceService) {}

ngOnInit() {

this.todaydate = this.myservice.showTodayDate();

}

}

In the new component that we have created, we need to first import the service that we want and access the methods and properties of the same. Please see the code highlighted. The todaydate is displayed in the component html as follows −

<p>

{{newcomponent}}

</p>

<p>

Today's Date : {{todaydate}}

</p>

The selector of the new component is used in the **app.component.html** file. The contents from the above html file will be displayed in the browser as shown below −



If you change the property of the service in any component, the same is changed in other components too. Let us now see how this works.

We will define one variable in the service and use it in the parent and the new component. We will again change the property in the parent component and will see if the same is changed in the new component or not.

In **myservice.service.ts**, we have created a property and used the same in other parent and new component.

import { Injectable } from '@angular/core';

@Injectable()

export class MyserviceService {

serviceproperty = "Service Created";

constructor() { }

showTodayDate() {

let ndate = new Date();

return ndate;

}

}

Let us now use the **serviceproperty** variable in other components. In **app.component.ts**, we are accessing the variable as follows −

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

import { MyserviceService } from './myservice.service';

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Angular 4 Project!';

todaydate;

componentproperty;

constructor(private myservice: MyserviceService) {}

ngOnInit() {

this.todaydate = this.myservice.showTodayDate();

console.log(this.myservice.serviceproperty);

this.myservice.serviceproperty = "component created"; // value is changed.

this.componentproperty = this.myservice.serviceproperty;

}

}

We will now fetch the variable and work on the console.log. In the next line, we will change the value of the variable to “**component created**”. We will do the same in **new-cmp.component.ts**.

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

import { MyserviceService } from './../myservice.service';

@Component({

selector: 'app-new-cmp',

templateUrl: './new-cmp.component.html',

styleUrls: ['./new-cmp.component.css']

})

export class NewCmpComponent implements OnInit {

todaydate;

newcomponentproperty;

newcomponent = "Entered in newcomponent";

constructor(private myservice: MyserviceService) {}

ngOnInit() {

this.todaydate = this.myservice.showTodayDate();

this.newcomponentproperty = this.myservice.serviceproperty;

}

}

In the above component, we are not changing anything but directly assigning the property to the component property.

Now when you execute it in the browser, the service property will be changed since the value of it is changed in **app.component.ts** and the same will be displayed for the **new-cmp.component.ts**.

Also check the value in the console before it is changed.



Http Service

Http Service will help us fetch external data, post to it, etc. We need to import the http module to make use of the http service. Let us consider an example to understand how to make use of the http service.

To start using the http service, we need to import the module in **app.module.ts** as shown below −

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

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

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { HttpModule } from '@angular/http';

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

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

BrowserAnimationsModule,

HttpModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

If you see the highlighted code, we have imported the HttpModule from @angular/http and the same is also added in the imports array.

Let us now use the http service in the **app.component.ts**.

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

import { Http } from '@angular/http';

import 'rxjs/add/operator/map';

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

constructor(private http: Http) { }

ngOnInit() {

this.http.get("http://jsonplaceholder.typicode.com/users").

map((response) ⇒ response.json()).

subscribe((data) ⇒ console.log(data))

}

}

Let us understand the code highlighted above. We need to import http to make use of the service, which is done as follows −

import { Http } from '@angular/http';

In the class **AppComponent**, a constructor is created and the private variable http of type Http. To fetch the data, we need to use the **get API** available with http as follows

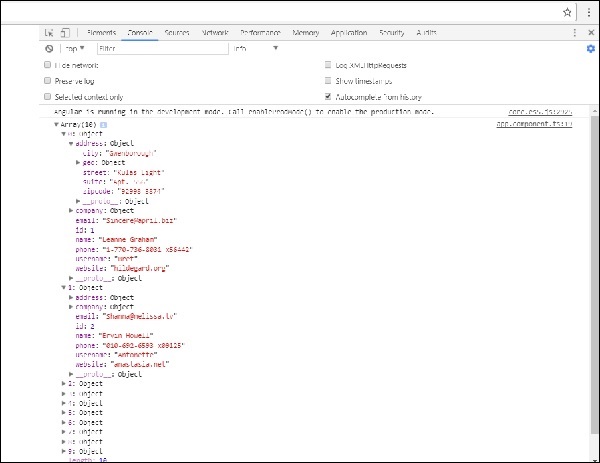
this.http.get();

It takes the url to be fetched as the parameter as shown in the code.

We will use the test url - https://jsonplaceholder.typicode.com/users to fetch the json data. Two operations are performed on the fetched url data map and subscribe. The Map method helps to convert the data to json format. To use the map, we need to import the same as shown below −

import 'rxjs/add/operator/map';

Once the map is done, the subscribe will log the output in the console as shown in the browser −



If you see, the json objects are displayed in the console. The objects can be displayed in the browser too.

For the objects to be displayed in the browser, update the codes in **app.component.html** and **app.component.ts** as follows −

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

import { Http } from '@angular/http';

import 'rxjs/add/operator/map';

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

constructor(private http: Http) { }

httpdata;

ngOnInit() {

this.http.get("http://jsonplaceholder.typicode.com/users").

map(

(response) ⇒ response.json()

).

subscribe(

(data) ⇒ {this.displaydata(data);}

)

}

displaydata(data) {this.httpdata = data;}

}

In **app.component.ts**, using the subscribe method we will call the display data method and pass the data fetched as the parameter to it.

In the display data method, we will store the data in a variable httpdata. The data is displayed in the browser using **for** over this httpdata variable, which is done in the **app.component.html** file.

<ul \*ngFor = "let data of httpdata">

<li>Name : {{data.name}} Address: {{data.address.city}}</li>

</ul>

The json object is as follows −

{

"id": 1,

"name": "Leanne Graham",

"username": "Bret",

"email": "Sincere@april.biz",

"address": {

"street": "Kulas Light",

"suite": "Apt. 556",

"city": "Gwenborough",

"zipcode": "92998-3874",

"geo": {

"lat": "-37.3159",

"lng": "81.1496"

}

},

"phone": "1-770-736-8031 x56442",

"website": "hildegard.org",

"company": {

"name": "Romaguera-Crona",

"catchPhrase": "Multi-layered client-server neural-net",

"bs": "harness real-time e-markets"

}

}

The object has properties such as id, name, username, email, and address that internally has street, city, etc. and other details related to phone, website, and company. Using the **for** loop, we will display the name and the city details in the browser as shown in the **app.component.html** file.

This is how the display is shown in the browser −



Let us now add the search parameter, which will filter based on specific data. We need to fetch the data based on the search param passed.

Following are the changes done in **app.component.html** and **app.component.ts** files −

app.component.ts

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

import { Http } from '@angular/http';

import 'rxjs/add/operator/map';

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'app';

searchparam = 2;

jsondata;

name;

constructor(private http: Http) { }

ngOnInit() {

this.http.get("http://jsonplaceholder.typicode.com/users?id="+this.searchparam).

map(

(response) ⇒ response.json()

).

subscribe((data) ⇒ this.converttoarray(data))

}

converttoarray(data) {

console.log(data);

this.name = data[0].name;

}

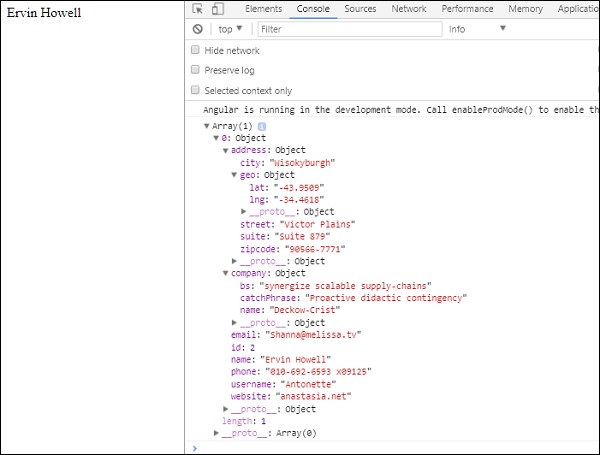
}

For the **get api**, we will add the search param id = this.searchparam. The searchparam is equal to 2. We need the details of **id=2** from the json file.

app.component.html

{{name}}

This is how the browser is displayed −



We have consoled the data in the browser, which is received from the http. The same is displayed in the browser console. The name from the json with **id=2** is displayed in the browser.