


Hey, dev peeps: DevReach is back, face-to-face, and in Boston! Join us and our incredible line-up of speakers covering JS, .NET, UI, A11y, and everything in-between.



Angular Basics: How To Use Services in Angular

 by Nwose Lotanna Victor

|

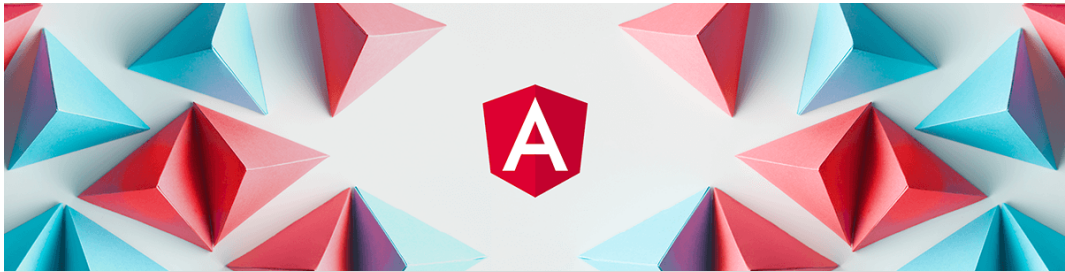
Ä June 04, 2021

|

Web, Angular

|

1 Comment



Ê	Č
â	÷
Û	

Now that we learned how data was shared among classes before services and that this wasn't very DRY or scalable, let's learn how to use services for dependency injection.

This is the second piece of a two-part article about dependency injection. In the first, we learned **why** we would want to use services in Angular. In this post, we will look at **how** to use them.

Prerequisites

To be able to follow through in this article's demonstration, you should have:

- An integrated development environment like VS Code
- Node version 11.0 installed on your machine
- Node Package Manager version 6.7 (it usually ships with Node installation)
- Angular CLI version 8.0 or above
- The latest version of Angular (version 12)

```
// run the command in a terminal
ng version
```

Confirm that you are using version 12, and update to 12 if you are not.

- Create a new Angular app with the command below:

```
ng new serviceapp
cd serviceapp
```

Other nice-to-haves include:

- Working knowledge of the Angular framework at a beginner level

Before Services Were Born

In the first part of this dependency injection series, we made a strong case for the use of services in our Angular applications. Benefits like writing modular code, efficiency and not having to

repeat ourselves among other things make this a great approach. Make sure to read the first part of the post [here](#).

Introduction to Services

Simply put, services in Angular let you define code or functionalities that are then accessible and reusable in many other components in your Angular project. Services help you with the abstraction of logic and data that is hosted independently but can be shared across other components.

The service class has a single, well-defined function, helping you make your application structure very modular. It is different from other classes in Angular because of the injection process. Dependency injection is the concept that makes it possible for you to receive dependencies from one class to another.

How We Will Be Using a Service

One of the biggest use cases for Angular services is managing or manipulating or even storing data. We will see how to create a service, register it and share data between two components today.

What We Are Building

We are going to re-create the application that displays artist information and locations like it is in the first part of the dependency injection series, but this time around, we will use a service.

Hi, this is the serviceapp

This is the list of Top African Music Artists

- Davido who is currently number 1
- Burna Boy who is currently number 2
- Diamondz Platinum who is currently number 3
- Sarkodie who is currently number 4
- Mr. Eazi who is currently number 5

This is the location list of Top African Music Artists

- Our number 1 artist in Africa is from Nigeria
- Our number 2 artist in Africa is from Nigeria
- Our number 3 artist in Africa is from Tanzania
- Our number 4 artist in Africa is from Ghana
- Our number 5 artist in Africa is from Nigeria

Getting Started



Open your new app which you created at the start of this post, and in the terminal inside your VS Code (or any other IDE) generate the two components:

```
ng generate component Artists
ng generate component Artistnames
```

Navigate into the artist component and change the content to the code blocks below:

```
// copy inside component.ts file
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-artists',
  templateUrl: './artists.component.html',
  styleUrls: ['./artists.component.css']
})
export class ArtistsComponent implements OnInit {
  public artists = [
    {grade:1, 'name':'Davido', 'country':'Nigeria'},
    {grade:2, 'name':'Burna Boy', 'country':'Nigeria'},
    {grade:3, 'name':'Diamondz Platinum', 'country':'Tanzania'},
    {grade:4, 'name':'Sarkodie', 'country':'Ghana'},
    {grade:5, 'name':'Mr. Eazi', 'country':'Nigeria'}
  ]
  constructor() { }
  ngOnInit(): void {
  }
}
```

```
<!-- copy into component.html file -->
<h2>
  This is the list of Top African Music Artists
</h2>
<ul *ngFor="let artist of artists">
  <li>
    {{artist.name}} who is currently number {{artist.grade}}
  </li>
</ul>
```

Now, in the second component, replace the content with the code blocks below:

```
// copy inside artistsname component.ts file
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-artistnames',
  templateUrl: './artistnames.component.html',
  styleUrls: ['./artistnames.component.css']
})
export class ArtistnamesComponent implements OnInit {
  public artists = [
    {grade:1, 'name':'Davido', 'country':'Nigeria'},
    {grade:2, 'name':'Burna Boy', 'country':'Nigeria'},
    {grade:3, 'name':'Diamondz Platinum', 'country':'Tanzania'},
    {grade:4, 'name':'Sarkodie', 'country':'Ghana'},
    {grade:5, 'name':'Mr. Eazi', 'country':'Nigeria'}
  ]
  constructor() { }
  ngOnInit(): void {
  }
}
```

```
<!-- copy into artistsname component.html file -->
<h2>
  This is the location list of Top African Music Artists
</h2>
<ul *ngFor="let artist of artists">
  <li>
    Our number {{artist.grade}} artist in Africa is from {{artist.co
  </li>
</ul>
```

Finally, in the app component HTML file, replace the content with the code block:

```
<div>
  <h2>
    Hi, this is the {{title}}
  </h2>
</div>
<app-artists></app-artists>
<app-artistnames></app-artistnames>
```

If you save all files and run the application in development like this:

```
ng serve
```

You will see it looks exactly like what we have in the picture at the start of this section.

The task now is to have the data we have repeated in both components inside a service from where it can be referenced anytime it is needed.

Creating the Service

To create a service in Angular, you need to run the generate service command:

```
ng generate service data
```

Two new files will be created. Navigate to the data service.ts file, and make sure the content is the same as this:

```
import { Injectable } from '@angular/core';
@Injectable({
  providedIn: 'root'
})
export class DataService {
  constructor() {}
  getList(){
    return [
      {grade:1, 'name':'Davido', 'country':'Nigeria'},
      {grade:2, 'name':'Burna Boy', 'country':'Nigeria'},
      {grade:3, 'name':'Diamondz Platinum', 'country':'Tanzania'},
      {grade:4, 'name':'Sarkodie', 'country':'Ghana'},
      {grade:5, 'name':'Mr. Eazi', 'country':'Nigeria'}
```

```

    ];
  }
}

```

This data service has now been created and hard-coded data stored in the `getList` function.

Registration of Angular Service

At this stage, Angular takes this service as any other Angular class, so to make sure Angular counts it as a service, we have to register it. It is also important to note the hierarchical way Angular handles things like dependency injection. Anywhere you register your service, dependencies can only be injected into the said component or the child nodes. So for our case, we are going to register it at the root module.

Navigate to your app module file and make sure it looks like this:

```

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { ArtistsComponent } from './artists/artists.component';
import { ArtistnamesComponent } from './artistnames/artistname';
import { DataService } from './data.service'
@NgModule({
  declarations: [
    AppComponent,
    ArtistsComponent,
    ArtistnamesComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [DataService],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Using the Service

To use the service we have set up in our project, all we need to do is to import it where needed and bring functions from it through the component's constructor.

In your artist component, copy this code block below into it:

```

import { Component, OnInit } from '@angular/core';
import { DataService } from '../data.service';
@Component({
  selector: 'app-artists',
  templateUrl: './artists.component.html',
  styleUrls: ['./artists.component.css']
})
export class ArtistsComponent implements OnInit {
  public artists = []
  constructor(private list: DataService) { }
  ngOnInit(): void {
    this.artists = this.list.getList();
  }
}

```



You can see how this is done and now we have access to the functions declared inside of the service we created.

Do the same for the Artistnames component:

```
import { Component, OnInit } from '@angular/core';
import { DataService } from '../data.service';
@Component({
  selector: 'app-artists',
  templateUrl: './artistname.component.html',
  styleUrls: ['./artistname.component.css']
})
export class ArtistnameComponent implements OnInit {
  public artists = []
  constructor(private list: DataService) { }
  ngOnInit(): void {
    this.artists = this.list.getList();
  }
}
```

If you save all the files and refresh your browser, you will see that the display is exactly the same as before:

Hi, this is the serviceapp

This is the list of Top African Music Artists

- Davido who is currently number 1
- Burna Boy who is currently number 2
- Diamondz Platinum who is currently number 3
- Sarkodie who is currently number 4
- Mr. Eazi who is currently number 5

This is the location list of Top African Music Artists

- Our number 1 artist in Africa is from Nigeria
- Our number 2 artist in Africa is from Nigeria
- Our number 3 artist in Africa is from Tanzania
- Our number 4 artist in Africa is from Ghana
- Our number 5 artist in Africa is from Nigeria

This time, though, it is more efficient and very modular as we have abstracted the function of data management to a data service and now components can serve their purposes without a need to repeat code or do more than they were built to do.

Conclusion



This has been a great introduction to using services in Angular. We looked at how we overburdened components in the past and how Angular has extensive features that help us encapsulate some things, thereby making the structures we build more efficient and modular. Happy hacking!

© Angular, Angular Basics



ABOUT THE AUTHOR

[Nwose Lotanna Victor](#)

Nwose Lotanna Victor is a web technology enthusiast who documents his learning process with technical articles and tutorials. He is a freelance frontend web developer based in Lagos, Nigeria. Passionate about inclusion, community-building and movies in Africa, he enjoys learning new things and traveling.

RELATED POSTS

WEB ANGULAR

Angular Basics: Why You Should Use Services in Angular

WEB ANGULAR

Angular Basics: 10 Helpful Native Web APIs Every New JavaScript Developer Should Know

WEB ANGULAR

Angular Basics: Using Pipes in Angular

WEB ANGULAR

Angular Basics: Style Binding in Angular with ngStyle

COMMENTS

ALSO ON TELERIK BLOGS

5 months ago • 1 comment

Blazor for .NET MAUI: What, How and When

2 months ago • 2 comments

World of Windows 11 (New Theme)

4 months ago • 1 comment

Windows 11: What, How and When

Comments Community Privacy Policy Login ▾

Favorite Tweet Share Sort by Best ▾

Join the discussion...

LOG IN WITH OR SIGN UP WITH DISQUS

Name

AmmoniaFlows • 4 months ago

Very nice boss... Very very good explanation of services. You did in two articles what I have been searching for in a while. Thanks

| • Reply • Share ▾

All articles

TOPICS

Web	0
Mobile	0
Desktop	0
Design	0
Productivity	0
People	0
Release	0

search blogs...





Latest Stories in Your Inbox

Subscribe to be the first to get our expert-written articles and tutorials for developers!

Email

Country/Territory

Subscribe



Telerik and Kendo UI are part of Progress product portfolio. Progress is the leading provider of application development and digital experience technologies.

[Company](#) [Technology](#) [Awards](#) [Press Releases](#) [Media Coverage](#) [Careers](#) [Offices](#)

Copyright © 2022 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. Progress, Telerik, Ipswitch, Chef, Kemp, Flowmon and certain product names used herein are trademarks or registered trademarks of Progress Software Corporation and/or one of its subsidiaries or affiliates in the U.S. and/or other countries. See Trademarks for appropriate markings.