Angular Singleton Service

1 Comment / 4 minutes of reading / March 9, 2023

← How Dependency Injection Works
Angular Tutorial
ProvidedIn →

A singleton service is a service for which only one instance exists in an app. In this tutorial, we will show how to create a singleton service in Angular when the service is in the root module, eagerly loaded module, or lazy loaded module.

Table of Contents

How to Create a Singleton Service in Angular

There are two ways in which you can create a Singleton Service

Using the root option of the `providedIn` property. This works irrespective of your service is in an **eager module** or **lazy loaded module**. Using the `providedIn` is the preferred way as it makes the service tree shakeable.

```

@Injectable({
  providedIn: 'root'
})
export class AppService {
...
}

```

The second option is to add it in the `Providers` array of `@NgModule`.

If the `NgModule` is **root module** or **eagerly loaded module**, then the `AppService` is available as a Singleton service to the entire application

But if the NgModule is **lazy-loaded module**, then `AppService` is available **only in that** **Lazy loaded module** and not outside of it.

```

@NgModule({
  imports: [],
  declarations: [],
  bootstrap: [],
  providers: [AppService]     <===
})
export class AppModule {}

```

Angular Singleton Service Example

Let us try to create the Singleton Service using an example from StackBlitz

The app has three modules. `AppModule`, `EagerModule` & `LazyModule`.

All the modules contain a random generation service. `AppService`, `LazyService`, & `EagerService`. The code for the `AppService` is as shown below. The `LazyService` & `EagerService` also has the same code. We just prefixed the random number with the service name to distinguish it from other services.

app.service.ts

```

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

@Injectable()
export class AppService {
  private _randomNo = '';

  constructor() {
    console.log('AppService Constructed');
    this._randomNo = 'App ' + Math.floor(Math.random() * 1000 + 1);
  }

  get RandomNo() {
    return this._randomNo;
  }
}

```

We have four components. `AppComponent` & `HelloComponent` from the `Root Module`, `EagerComponent` from `EagerModule` and `LazyComponent` from `LazyModule`

The following code is from the `HelloComponent`.

hello.component.ts

```

import { Component, Input, Optional } from '@angular/core';
import { AppService } from './app.service';
import { EagerService } from './eager/eager.service';
import { LazyService } from './lazy/lazy.service';

@Component({
  selector: 'hello',
  providers:[],
  template: `
    Hello Works {{ randomApp }} {{ randomEager }} {{ randomLazy }}
  `,
  styles: [
    `
      h1 {
        font-family: Lato;
      }
    `
  ]
})
export class HelloComponent {
  randomApp = 'App : Not defined';
  randomEager = 'Eager : Not defined';
  randomLazy = 'Lazy : Not defined';

  constructor(
    @Optional() private appService: AppService,
    @Optional() private eagerService: EagerService,
    @Optional() private laztyService: LazyService
  ) {
    if (appService) this.randomApp = this.appService.RandomNo;
    if (eagerService) this.randomEager = this.eagerService.RandomNo;
    if (laztyService) this.randomLazy = this.laztyService.RandomNo;
  }

  ngOnInit() {}
}


```

We have injected all three services into `HelloComponent`. The Optional Decorator ensures that if the service not available then the Angular returns **null** instead of throwing an error.

```

constructor(
  @Optional() private appService: AppService,
  @Optional() private eagerService: EagerService,
  @Optional() private laztyService: LazyService
)

```

All the other components also have similar codes. The AppComponent has a navigation menu.

Initially, we start off by removing the providedIn and also making the provider's array empty. This will make all the component's display Not defined . No errors thrown as we are using the optional decorator

Service in the root module

The AppService is in the root module.

Add it to the Providers array of the @NgModul of the AppModule .

```

@NgModule({
  imports: [BrowserModule,FormsModule,RouterModule.forRoot(routes),EagerModule],
  declarations: [AppComponent, HelloComponent],
  bootstrap: [AppComponent],
  providers: [AppService]
})
export class AppModule {}


```

You can see that all components display the same value for the Random No (including those from EagerModule & LazyModule). You can also see the AppService Constructed in the console only once.

Another way to achieve this is to add the providedIn: 'root' in the @Injectable decorator in the AppService

```

@Injectable({ providedIn: 'root' })
export class AppService {

```

Service in the eagerly loaded module

The services in the eagerly loaded module can be made singleton in the same way as in the root module.

You can add it in the providedIn :'root' with the @Injectable decorator

```


@Injectable({ providedIn: 'root' })
export class EagerService {

```

Or add it to the providers array of the NgModule decorator of the EagerModule.

```

@NgModule({
  imports: [CommonModule, RouterModule.forChild(routes)],
  declarations: [EagerComponent],
  providers: [EagerService]
})
export class EagerModule {}

```

Service in the lazy loaded module

Adding the `LazyService` in the `providers` array of the `LazyModule`. But this makes the `LazyService` available only in the `LazyModule` and not outside of it.

```

@NgModule({
  imports: [CommonModule, RouterModule.forChild(routes)],
  declarations: [LazyComponent],
  providers: [LazyService]
})
export class LazyModule {}

```

The only way to achieve it by using the `providedIn : 'root'` in the `Injectable` of the `LazyService`

```


@Injectable({ providedIn: 'root' })
export class LazyService {

```

That’s it.

Read More

1. Angular Tutorial
2. Services