Get started Log in

Parul Saxena

Understanding the Purpose of Services in Angular

Parul Saxena

Jul 12, 2020 • 5 Min read • 7,144 Views

Jul 12, 2020 • 5 Min read • 7,144 Views

Languages Framework... Front End Web Develo...
Client-side Frameworks Angular

Introduction

33

Introduction

A reusable Angular service is designed to encapsulate business logic and data with different components of Angular. It is basically a class that has a well-defined purpose to do something. You can create a service class for data or logic that is not associated with any specific view to share across components.

Why Services Are Needed

If you write all the business logic in components, you will have the following problems:

- You will not be able to reuse that logic anywhere else and you will have to re-code the entire logic in the target component.
- Your components will become hard to maintain as you will have to maintain two copies of the same code.

So, if you wrap your business logic in a reusable Angular service, you will not only keep your components clean but also get the benefits of reusability and maintainability.

Best Practices

If any component method has logic that goes beyond ten lines of code, it should be considered a candidate to be wrapped in a reusable Angular service. Always make sure to decorate your service class with the `@Injectable()` decorator so that when you inject any other service in this service, you don't get an error.

Dependency Injection

Dependency injection is used to provide components with the services they can use. To define a class as a service in Angular, the `@Injectable()` decorator is used. It provides the metadata that allows Angular to inject it into a component as a dependency. Similarly, the `@Injectable()` decorator is used to indicate that a component or other class (such as another service, a pipe, or an NgModule) has a dependency.

Registering a Service in Dependency Container

There are two ways to register a service in the providers array of any module:

- Longhand method
- Shorthand method

Longhand Method

In the longhand method, you register an object with two properties, `provide` and `useValue`, in the `providers` array of any module, as shown below:

```
import { BrowserModule } from '@angular/platfor
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing
import { AppComponent } from './app.component';
import { EventService } from './events/shared/e
import { VoterService } from './events/event-de

```

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule
  ],
  providers: [
              {
                provide: EventService,
                useValue: EventService
              }
              ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Shorthand Method

In the shorthand method, you specify the name of your service class in the `providers` array in your module as shown below:

```
import { BrowserModule } from '@angular/platfor
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing
import { AppComponent } from './app.component';
import { EventService } from './events/shared/e
import { VoterService } from './events/event-de

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule
  ],
  providers: [
              EventService
              ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Component Communication Using Angular Services

Reusable Angular services can also be used to establish communication between two components. The components can be in a parent-child relationship or can be siblings. Irrespective of the relationship type, the services can be used to share data between two components. All you need is public properties in the service class which one component will set and the other component will consume and vice-versa. The service that is being used as a bridge between the two components has to be injected into both the components.

Service Instances in Angular

No matter how many components you try to consume a service in, Angular creates only one instance of the service. So, Angular services by default are singletons. This helps services to establish communication between two or more components.

Conclusion

Congratulations! You are now familiar with Angular services, the methods used to register them, and how to establish component communication in Angular. For more information, please refer to the Fundamentals of Angular.

33

LEARN MORE

SOLUTIONS

Pluralsight Skills (/product/skills)

Pluralsight Flow (/product/flow)

Government (/industries/government)

Gift of Pluralsight (/gift-of-pluralsight)

PLATFORM

Browse library (/browse)

Role IQ (/product/role-iq)

Skill IQ (/product/skill-iq)

Iris (/product/iris)

View Pricing (/pricing)

Contact Sales (/product/contact-sales)

Skill up for free (/product/skills/free)

Authors (/authors)

Professional Services (/product/professional-services)

Technology Index (/tech-index)