Components should not fetch or save data directly from the backend(rest api service or middle ware). They are focus on representing data

Services in Angular are a great way to share information among components that **do not know each other.**

When we are developing an Angular app, we will most likely run into a scenario in which we need to use the same code across multiple components. In that case, **Services**will help us to get rid of this problem. We can share the services code among various components.

Angular service plays an important role to communicate with the backend layer(service layer or middleware) of any application from the component level to send or retrieve data.

[**Components**](https://appdividend.com/2018/01/28/angular-components-tutorial-example-scratch/)use to display and present the data. **Services**use to fetch the data from the API.

## Benefits of using Service

Angular services are single objects that normally get instantiated only once during the lifetime of the Angular application. This Angular service maintains data throughout the life of an application. It means data does not get replaced or refreshed and is available all the time. The main objective of the Angular service is to use shared business logic, models, or data and functions with multiple different components of an Angular application.

The main objective of using an Angular service is the Separation of Concern. An Angular service is basically a stateless object, and we can define some useful functions within an Angular service. These functions can be invoked from any component of the application elements like Components, Directives, etc. This will help us to divide the entire application into multiple small, different, logical units so that those units can be reusable.

**Creating Angular Service**

**Using Angular CLI**

Creating service is easy if you use Angular CLI. It is just one command which does the all work. It will generate the service code just like above example.

|  |
| --- |
|  |
| **ng g s service/calc** |

**Global service vs Local Service Injection**

To inject the service, you have two options.

**1) Inject as ‘global service’**

To inject as global service, inject the **service into root module**.

|  |
| --- |
| **Global service injection - app.module.ts** |
| **import { BrowserModule } from '@angular/platform-browser';**  **import { NgModule } from '@angular/core';**    **import { AppComponent } from './app.component';**  **import { CalcService } from './service/calc.service';**    **@NgModule({**  **declarations: [**  **AppComponent**  **],**  **imports: [**  **BrowserModule**  **],**  **providers: [CalcService],**  **bootstrap: [AppComponent]**  **})**  **export class AppModule { }** |

**2) Inject as ‘local service’**

To inject as local service, inject the **service into component** directly.

|  |
| --- |
| **Local service injection - app.component.ts** |
| import { Component } from '@angular/core';  import { CalcService } from './service/calc.service';    @Component({    selector: 'app-root',    templateUrl: './app.component.html',    styleUrls: ['./app.component.css'],    providers: [CalcService]  })  export class AppComponent {    title = 'app';      constructor(calc:CalcService){      //Use calc    }  } |

## What is Dependency Injection?

**Dependency injection** is an important application design pattern. Angular has its own dependency injection framework

Dependency Injection is one of the main benefits of Angular Framework.

With the help of this feature, we can inject any types of dependency like service or class, external utility module in our application module.

If we are using **Dependency Injection** then, we do not need to create the instances in the **constructor**.

constructor(calc:CalcService){}