# Introduction to services

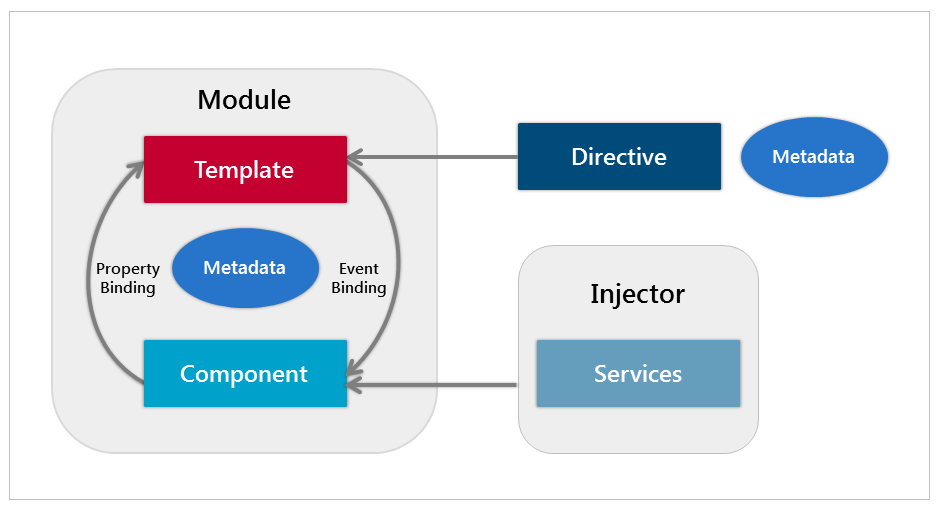
Services is a class in Angular with a specific purpose. You may write service for the following scenarios:

• Logging

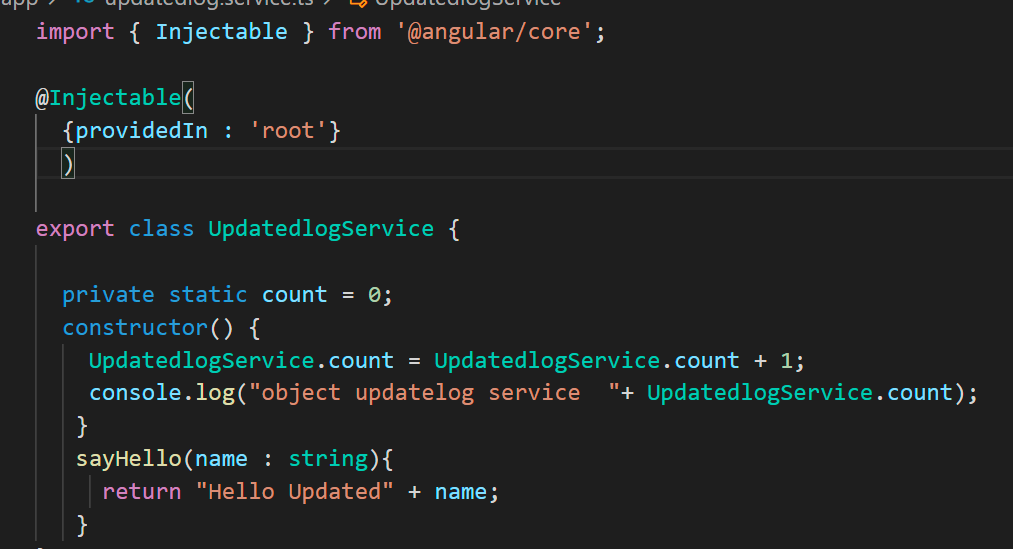
• Authentication

• Authorization

• Sharing data between unrelated components



To define a class as a service in Angular, use the @Injectable() decorator.



CREATE A SERVICES

**ng g s UpdatedlogService**

generate service service name

**Are the Angular services singleton?**

NO, angular services are not singleton always. The main objective of angular services is to share data across Angular application. Practically an angular service can be shared between all the components or can be limited to some component. Hence Angular service can be a singleton as well as non-singleton in nature.

A service for which only one instance exists in the application is called to be a singleton in nature. By default service is singleton (**providedIn)**

A service for which more than one instance exists in the application is not singleton in nature. (**providers)**

**Providers**

**Not Tree shakeable**

**Components**

**Array type**

@Component({

 Providers:[{provide:LogService,useClass : LogService}]

 })

**ProvideIn**

Tree shakeable

Services

Value type

@Injectable(

  {providedIn : 'root'}

  )

**Steps in Services**

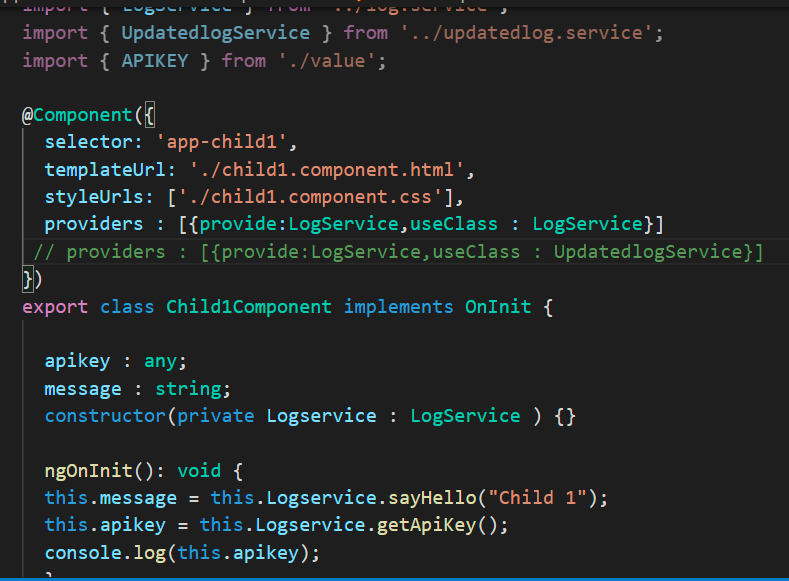
**Steps in service**

**CREATE A SERVICE**

**PROVIDE A SERVICE**

**INJECT A SERVICE**

**USE A SERVICE**

****

Use service

InjectService

ProvideService

**useClass ,useValue ,useFactory ,useExisting ?**

**useClass** : is used when you want to provide instance of the class .

**useExisting :** use existing object of already existing token.

**useFactory :** There could be scenario where, until runtime, you do not have idea about what instance is needed. You need to create dependency on the basis of information you do not have until the last moment.

**useValue :** Both useClass and useExisting create instance of a service class to inject for a particular token, but sometimes you want to pass value directly instead of creating instance. So, if you want to pass readymade object instead of instance of a class, you can use useValue

  providers:[{provide:UpdatedlogService,useClass:UpdatedlogService},

            {provide:LogService,useFactory: getApiKey()},

            {provide:UpdatedlogService,useExisting:LogService},

            {provide:APIKEY,useValue:{apikey : '123456'}}

            ],

useClass

getApiKey – some business logic (which class object is created)

useFactory

useValue

useExisting