Angular Services

Services refers to a class in the angular which can be used to provide or share logic between components.

Features of services:

- In default scenario, the service is singleton. This feature can be configured by the programmer.
- A service doesn't have direct relationship with template.
- By default, the service is tree shakable.

Note: Tree shaking is a term commonly used within a JavaScript context to describe the removal of dead code. In Angular, the tree shaking is done by using webpack module.

Steps of using a service:

1. Creation of service: Angular-cli command used:

   `ng g s <service-name>`
2. Providing service to the components: In the default scenario (i.e. the service is not configured for a particular use by the programmer), it is provided to all the components within the angular project.
3. Injecting service in the components: This can be done by-

   `contructor(private <field-name> : <service-name>)`
4. Use of service in the component: The methods and fields in the service then can be used within the component class using:

   `this.<field_name>.<service_field_name>)`

Steps in which the services get resolved by the angular:

1. The service got found when used by the component.
2. Angular, will try to resolve it by checking whether it is injected into the component.
3. Then, it is checked that the component has access to the service or not (simply, whether or not the service is provided to the component).
4. Lastly, whether the service is created or not.

Different ways to provide a service to the component:

1. Using ProvideIn: It is present in @Injectable decorator within the service. If a service is provided to the component via the provideIn then the service is tree-shakable.
   - The default value of the ProvideIn is 'root', making the service class a singleton class i.e. only single onbject of the class can be created.
   - Its value can also be changed to 'any', representing different object will be created for every lazy loaded module and one object for all eagerly loaded objects.

2. Using Providers: It exists only in the module or in the component or both. The main difference between this and the providerIn is that services provided to the component via Providers array are not tree-shakable.
   The service can be directly specified in the providers array:
   Providers: [<Service_Name>]
   Note- The name mentioned directly in the providers array is the token of the service which is same as the name of the service-class.
   The way to provide services to the components can be configured using the fields:
   - Provide – specifies the token-name of the service. If the name of the service-class and the token-name is same then, there is no need to explicitly define it in the providers array.
   - UseClass - represents the underlying service-class represented by the token-name.
     Providers: [{provide:<token_Name>, useClass:<class-name>}]
     The class-name can be different according to the need. If a class is associated with multiple tokens, then each distinct object of the class is associated with the corresponding token.
   - UseExisting – represents another token-name of another service.
     Providers: [{provide:<token_Name>, useExisting:<another-token-name>}]
     The object of the class associated with the another-token-name is also being used by the token-name.
   - UseFactory – represents a logic which will return the class-name according to certain parameters which are subject to change according to the user's information.
     Providers: [{provide:<token_Name>, useFactory:<logic>}]
   - UseValue – representing a data which is shared among multiple components, unrelated or not. To use this:
     First, the custom token must be created using:
     const <token-name> = new InjectionToken<type> ("Information about the data")
     Then provided in the providers with the provide:
     Providers: [{provide:<token_Name>, useValue:<data>}]

How the Angular resolves the services?

As the angular services can be provided to the components using both Providers array and ProvideIn simultaneously. Angular follows a hierarchy of resolving token:

1. Search in the same component where the services is being used.
2. Searched in parent component from immediate parent of the component to the last parent.
3. Search in the module which consists of the sub-tree of the component.
4. Search in the main module.
5. Lastly, in the provideIn in the service itself.