Q

What are services?

Services

A service is a piece of reusable code that you will use it in many components across your application.
Services are easier to test, debug, reuse.
An Angular service is simply a Javascript function.

A component can delegate certain tasks to services, such as fetching data from the server, validating user input, or logging directly to the console.
By defining such processing tasks in an *injectable service class*, you make those tasks available to any component.

Often the http access is put in in a service.

The following example is adapted from www.tektutorialshub.com

https://www.tektutorialshub.com/angular/angular-services/

Services

```
export class Product {
  constructor(productID:number,name: string ,price:number) {
    this.productID=productID;
    this.name=name;
    this.price=price;
  }
  productID:number ;
  name: string ;
  price:number;
}
```

product.ts

```
import {Product} from './Product'
export class ProductService{
  public  getProducts() {
    let products:Product[];
    products=[
      new Product(1,'Memory Card',500),
      new Product(1,'Pen Drive',750),
      new Product(1,'Power Bank',100)
    ]
    return products;
  }
}
```

product.service.ts

Services

app.component.ts

```
import { Component } from '@angular/core';
import { ProductService } from './product.service';
import { Product } from './product';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})
export class AppComponent
{
   products:Product[];
   productService;
   constructor(){
     this.productService=new ProductService();
   }
   getProducts() {
     this.products=this.productService.getProducts();
   }
}
```

instantiate the service

forward the call to the service

Services

```
<div class="container">
  <h1 class="heading">Services Demo</h1>
  <button type="button" (click)="getProducts()">
  Get Products</button>
  <div class='table-responsive'>
     <table class='table'>
        <thead>
           <tr>
              <th>ID</th>
              <th>Name</th>
              <th>Price</th>
           </tr>
        </thead>
        <tbody>
           <tr *ngFor="let product of products;">
               <td>{{product.productID}}</td>
               <td>{{product.name}}</td>
               <td>{{product.price}}</td>
           </tr>
        </tbody>
     </table>
  </div>
</div>
```

ask the component

app.component.html

Services Dependeny Injection

Dependency injection, or DI, is a design pattern in which a class requests **dependencies** from external sources rather than creating them.

Angular's DI framework provides **dependencies** to a class upon instantiation. You can use **Angular** DI to increase flexibility and modularity in your applications.

In our example we will now decouple AppComponent from ProductService.

Services Dependency Injection

From the previous project, we only change the service and the component.

```
import { Injectable } from '@angular/core';
import {Product} from './Product'
@Injectable
export class ProductService{
  public  getProducts() {
    let products:Product[];
    products=[
      new Product(1,'Memory Card',500),
      new Product(1,'Pen Drive',750),
      new Product(1,'Power Bank',100)
    ]
    return products;
  }
}
```

product.service.ts

Services Dependeny Injection

```
import { Component } from '@angular/core';
import { ProductService } from './product.service';
import { Product } from './product';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  providers: [ProductService]
})
export class AppComponent
{
   products:Product[];
   productService;
   constructor(){
     this.productService=new ProductService();
   }
   constructor(private productService:ProductService){
   getProducts() {
     this.products=this.productService.getProducts();
   }
}
```

app.component.ts

see https://stackblitz.com/edit/servicedemo-ivy-gkwrbm?file=src%2Fapp%2Fapp.component.ts

Making services transversal

To make the dependencies available to the entire application, we need to register it in the root module.

Hence we must remove the provider(s) from
the AppComponent and move it to the AppModule.

Providing the service in the root module will create a single, shared instance of service and injects into any class that asks for it, thanks to the Tree of Injectors with parent child relationship similar to the Component Tree.

Sharing data using services

Just build a simple service to gather/deliver data from/to any component that injects the service.

Examples:

- you may retrieve some user preferences and retain them for use by any component in the application.
- you may store some user-entered values, such as the selected currency, throughout the application.

Q

What is routing?

Routing

Association of an URL with a component or state.

Useful for solving the bookmarking and SEO (Search Engine Optimization) problems.

The **Router** is a separate module in Angular (library package *@angular/router)* It provides the service providers and directives for navigating through application views, e.g.:

- Navigate to a specific view by typing a URL in the address bar
- Pass optional parameters to the View
- Bind the clickable elements to the View and load the view when the user performs application tasks
- Handle back and forward buttons of the browser
- Dynamically load the view
- Protect the routes from unauthorized users using Guards

Basic elements

- **Router**: object that enables navigation from one component (navigate(), navigateByUrl())
- **Route**: association path-component
- **Routes**: array of Route
- **RouterLink**: directive that binds the HTML element to a Route
- **RouterOutlet**: directive that indicate where the product of routing should be shown

- **NavigationExtras**: special object to keep extra info, e.g. the state

Basic operations

- Set the <base href> (in index.html)
- Define routes for the view (create the Route objects, and its Routes container)
- Register the Router Service with Routes: RouterModule.forRoot(*Routes instance*)
- Map HTML Element actions to Route (using router.navigate or RouterLink)
- Choose where you want to display the view (using <router-outlet>)

Basic operations

- Set the <base href> (in index.html)
- Define routes for the view (create the Route objects, and its Routes container)
- Register the Router Service with Routes: RouterModule.forRoot(*Routes instance*)
- Map HTML Element actions to Route (using router.navigate or RouterLink)
- Choose where you want to display the view (using <router-outlet>)

Example – 1 - simple components

https://stackblitz.com/edit/angular-router-y6v9gc?file=src%2Fapp%2Fapp.module.ts

home.component.ts

```
import {Component} from '@angular/core';
@Component({
    template: `<h1>Welcome!</h1>
               <p>This is Home Component </p>`})
export class HomeComponent {}
```

contact.component.ts

```
import { Component } from "@angular/core";
@Component({
  template: `<h1>Contact Us</h1>
                 <p><a href=
"mailto://pippo@topolinia.com">send a mail to us</a> </p>`})
export class ContactComponent {}
```

Example - 2 -simple components

product.component.ts

```
import { Component } from "@angular/core";@Component({
template: `     <h1>Product list</h1>
    <table><thead>
        <tr><th>ID</th><th>Name</th><th>Price</th></tr>
     </thead><tbody>
       <tr><td>1</td><td>Memory card</td><td>500</td></tr>
       <tr><td>2</td><td>Pen Drive</td><td>750</td></tr>
     </tbody></table>  `})
export class ProductsComponent {}
```

error.component.ts

```
import {Component} from '@angular/core';
@Component({
    template: `<h1>Page not found</h1>
                <p>This is a Error Page</p>`})
export class ErrorComponent {}
```

Example- 3

```
import { Routes } from "@angular/router";
import { HomeComponent } from "./home.component";
import { ContactComponent } from "./contact.component";
import { ProductsComponent } from "./products.component";
import { ErrorComponent } from "./error.component";
export const appRoutes: Routes = [
  { path: "home", component: HomeComponent },
  { path: "contact", component: ContactComponent },
  { path: "product", component: ProductsComponent },
  { path: "", redirectTo: "home", pathMatch: "full" },
  { path: "**", component: ErrorComponent }
];
```

app.routes.ts

```
<!DOCTYPE html>
<html><head>
    <base href="/" />
    <meta charset="utf-8" />
    <title>Angular Routing</title>
    <meta name="viewport" content="width=device-width,
       initial-scale=1" />
  </head><body>
    <my-app>loading...</my-app>
  </body></html>
```

index.html

Example- 4

app.component.ts

```
import { Component, VERSION } from '@angular/core';
import { Router } from '@angular/router';
@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: [ './app.component.css' ]
})
export class AppComponent  {
  name = 'Angular ' + VERSION.major;
   constructor(private router: Router){
     //this is only needed to define the variable
     //used in the following function
  }
  navigateToProducts(){
    this.router.navigate(['product']);
  }
}
```

Example- 5

app.component.html

```
<h1>
  Welcome to the routing demo
</h1>
<div class="container">
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      Click below to navigate
    </div>
    <ul class="nav navbar-nav">
        <li><a [routerLink]="['home']">Home</a></li>
        <li><a [routerLink]="['product']">Product</a></li>
        <li><a [routerLink]="['contact']">Contact us</a></li>
    </ul>
  </div>
</nav>
 <button (click)="navigateToProducts()">Products</button>
 <router-outlet></router-outlet>
</div>
```

Example- 6

app.module.ts

```
import { RouterModule } from '@angular/router';
import { HomeComponent} from './home.component'
import { ContactComponent} from './contact.component'
import { ProductsComponent} from './products.component'
import { ErrorComponent} from './error.component'
import { appRoutes } from './app.routes';
@NgModule({
  imports:      [ BrowserModule, FormsModule,
                  HttpClientModule,
                  RouterModule.forRoot(appRoutes) ],
  declarations: [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Multiple router-outlets

You can have multiple router-outlets, by giving them names:

```
<router-outlet></router-outlet>
<router-outlet name="sidebar"></router-outlet>
```

For details, see:

https://www.techiediaries.com/angular-router-multiple-outlets/

Q

How can we protect routing, like we do with Java Filters?

Guards

Sometimes you want users to navigate based on some specific condition, such as e.g.:

- users should be authenticated in order to have access to some resouce
- prevent the user from accidentally navigating away without saving data.

Guards

The problem is similar to the one we dealt with filters.

The Angular way to solve this problem is by using route-guards: they provide built-in interfaces that can be implemented to routes to control their navigation.

Five types of route guards are provided by angular :

- CanActivate (preventing access to the specific route)
- CanActivateChild (prevent access to child routes of a given route)
- CanDeactivate (prevent a user from accidentally navigating away without saving or some other undone tasks.)

- CanLoad (prevent downloading of a module)

Guards

Resolve

Complex angular applications involve data communication between components.

Sometimes data is so heavy that it is not possible to pass data through query params.

To handle this situation angular has provided Resolve Guard.

Q

What else ?

Custom directives

https://www.tektutorialshub.com/angular/custom-directive-in-angular/

Migration

The Angular Team release a new version of the Angular at regular intervals.

To keep up with the latest version, we need to update or upgrade our Angular Application.

See here:

https://www.tektutorialshub.com/angular/angular-how-to-update-to-latest-version/