STEP 14: Service In Angular

1. Create a service folder and execute the below command from within the service folder
   ng g s demo

2. Add a property message and a method to update this message
   ```
   message:string ;
   constructor() {
           this.message = 'from service';
           console.log('demo service')
       }
   setMessage(msg:string){
           this.message = msg
   }
   ```

3. Create a component within the service folder
   ng g c service --flat

4. Inject this service via constructor in service component as well as app component

   ```
   constructor(public service:DemoService){}
   ```

5. Add <app-service></app-service> in app html component

6. Update the html of service component as follows:

   ```
   <h1>Service Component</h1>
   <p>Service Message : {{service.message}}</p>
   <p> Title : {{title}}</p>
   <p><input type="text" [(ngModel)]="title"/></p>
   <p><button (click)="changeTitle()">
      Change</button></p>
   ```

7. Update service component to handle the changeTitle method toupdate the message of demo service

   ```
   changeTitle(){
     this.service.setMessage(this.title);
   }
   ```

8. Update app component html as below:
   ```
   <h1>App Component</h1>
   <p>{{service.message}}</p>
    <app-service></app-service>
   ```

9. Modifying the message in service component will modify the message in app component as well since angular injects a single instance of service throughout angular application

10. Now in service component, inject the demo service in providers array within the @Component decorator as follows:

```
@Component({
  selector: 'app-serv',
  templateUrl: './serv.component.html',
  styleUrls: ['./serv.component.css'],
  providers:[DemoService]
})
```

11. Now when the message value is changed by service component will not reflect in app component as now there are 2 different instances of service created.