Monday: Creating a service

Angular implements Dependency Injection using **services**. A **service** shares data and information among classes that don't know each other. A service is basically a class with a well-defined purpose. This means that it should do something specific and do it well. For example, we can have a service that retrieves our goals list from either local storage, a database(in a server), or a hard-coded list.

Why Use Services?

So far, we have been storing our hard-coded goals inside the goal component(more specifically goal.component.ts file). This is not a good practice. Ideally, a component's job is to enable the user experience. It should mediate between the view and the application logic. In other words, components should not fetch or save data directly. They should focus on presenting data and delegate data access to a service. There are other tasks that a component can delegate to services such as validating user input or logging error messages to the console.

Now, let's remove our goals list from the goal component. Ideally, if we had a backend for our application, we would be getting this list from our database in the server but since we don't have a backend server we are going to store our goals list in a separate file called *goalList.ts.*

Create the goalList.ts file inside the app folder.

Here's how the goalList.ts file looks like

src/app/goalList.ts

```
import { Goal} from './goal';
export let Goals: Goal[] = [
    new Goal(1, 'Watch finding Nemo', 'Find an online version and watch merlin find his son',new Da
te(2021,9,14)),
    new Goal(2,'Buy Cookies','I have to buy cookies for the parrot',new Date(2021,6,9)),
    new Goal(3,'Get new Phone Case','Diana has her birthday coming up soon',new Date(2021,1,12)),
    new Goal(4,'Get Dog Food','Pupper likes expensive snacks',new Date(2020,11,18)),
    new Goal(5,'Solve math homework','Damn Math',new Date(2021,2,14)),
    new Goal(6,'Plot my world domination plan','Cause I am an evil overlord',new Date(2021,3,14)),
  ]
```

Remember to delete the list from the goal.component.ts file.

Here's how the goal.component.ts file looks like after deleting the list

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-goal',
  templateUrl: './goal.component.html',
  styleUrls: ['./goal.component.css']
})
export class GoalComponent implements OnInit {  addNewGoal(goal){
    let goalLength = this.goals.length;
    goal.id = goalLength+1;
    goal.completeDate = new Date(goal.completeDate)
    this.goals.push(goal)
  }
  toggleDetails(index){
    this.goals[index].showDescription = !this.goals[index].showDescription;
  }  completeGoal(isComplete, index){
    if (isComplete) {
```

```
      this.goals.splice(index,1);
    }
  }
  deleteGoal(isComplete, index){
    if (isComplete) {
      let toDelete = confirm(`Are you sure you want to delete ${this.goals[index].name}?`)      if
 (toDelete){
        this.goals.splice(index,1)
      }
    }
  }
  constructor() { }  ngOnInit(): void {
  }}
```

Right now the app will not run because we have not connected the goal list to the goal component.

Next, let's create a service to access our goal list.

In our terminal, let's use this command to create the service:

```
$ ng generate service goal-service/goal
```

This command creates a folder named goal-service and inside it creates the goal service class file. It also generates a skeleton *GoalService* class as follows;

src/app/goal-service/goal.service.ts

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class GoalService {

  constructor() { }
}
```

The service imports the Angular **Injectable** symbol imported and annotates the class with the @Injectable() decorator. This annotation is what tell Angular that this class is a service (just like @component annotation tells Angular that a particular class is a component). This marks this class as one that participates in dependency injection. The decorator accepts metadata for the class which means our service can also have its own dependencies. The **providedIn** property has a value **'root'** which means that this service is injectable throughout the application. This means that this service can be used by any component in the application.

Let's write code to make this service access our goals.

src/app/goal-service/goal.service.ts

```
import { Injectable } from '@angular/core';
import { Goals } from '../goalList';

@Injectable({
  providedIn: 'root'
})
export class GoalService {

  getGoals(){
    return Goals
  }

  constructor() { }
}
```

At the top, we have imported our **Goals** array. In the service class, we have created a method **getGoals()** which returns our Goals array. Now, our GoalService is able to access our goals list. All we need to do now is to call the getGoals() function in the goals component.

How do we achieve that though?

Well, the answer is very simple; we'll make use of Dependency Injection. We'll inject the GoalService class in the Goal class (which is inside our goal component), then through the GoalService class, we can access the getGoals() function.

But before we perform the Dependency Injection, let's learn how to register the services we created with Angular. We'll do this in the next lesson