**Lesson22 Services in Angular**

**Notes:-**

**1-Angular is generally used when you need to reuse data or logic across multiple components. Anytime you see logic or data-access duplicated across multiple components, think about refactoring that piece of logic or data-access code into a service.**

**(To apply the principle of the DRY (Don’t repeat yourself))**

**Steps:-**

**1-we will create service**

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

**import { IEmployee } from './employee';**

**// The @Injectable() decorator is used to inject other dependencies**

**// into this service. As our service does not have any dependencies**

**// at the moment, we may remove the @Injectable() decorator and the**

**// service works exactly the same way. However, Angular recomends**

**// to always use @Injectable() decorator to ensures consistency**

**@Injectable()**

**export class EmployeeService {**

**getEmployees(): IEmployee[] {**

**return [**

**{code: 'emp101', name: 'Tom', gender: 'Male',**

**annualSalary: 5500, dateOfBirth: '6/25/1988'},**

**{code: 'emp102', name: 'Alex', gender: 'Male',**

**annualSalary: 5700.95, dateOfBirth: '9/6/1982'},**

**{code: 'emp103', name: 'Mike', gender: 'Male',**

**annualSalary: 5900, dateOfBirth: '12/8/1979'},**

**{code: 'emp104', name: 'Mary', gender: 'Female',**

**annualSalary: 6500.826, dateOfBirth: '10/14/1980'},**

**{code: 'emp105', name: 'Nancy', gender: 'Female',**

**annualSalary: 6700.826, dateOfBirth: '12/15/1982'},**

**{code: 'emp106', name: 'Steve', gender: 'Male',**

**annualSalary: 7700.481, dateOfBirth: '11/18/1979'},];}}**

**2-on the employee-list.component.ts we will write the following code**

**// Import OnInit Life Cycle Hook interface**

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

**import { IEmployee } from './employee';**

**// Import EmployeeService**

**import { EmployeeService } from './employee.service';**

**@Component({**

**selector: 'list-employee',**

**templateUrl: 'app/employee/employeeList.component.html',**

**styleUrls: ['app/employee/employeeList.component.css'],**

**// Register EmployeeService in this component by**

**// declaring it in the providers array**

**providers: [EmployeeService]})**

**// Make the class implement OnInit interface**

**export class EmployeeListComponent implements OnInit {**

**employees: IEmployee[];**

**selectedEmployeeCountRadioButton: string = 'All';**

**// Inject EmployeeService using the constructor**

**// The private variable \_employeeService which points to**

**// EmployeeService singelton instance is then available**

**// throughout this class**

**constructor(private \_employeeService: EmployeeService) {}**

**// In ngOnInit() life cycle hook call the getEmployees()**

**// service method of EmployeeService using the private**

**// variable \_employeeService**

**--This is the best place to call the service**

**ngOnInit()**

**{this.employees = this.\_employeeService.getEmployees();}**

**getTotalEmployeesCount(): number {**

**return this.employees.length;}**

**getTotalMaleEmployeesCount(): number {**

**return this.employees**

**.filter(e => e.gender === 'Male').length;}**

**getTotalFemaleEmployeesCount(): number {**

**return this.employees.filter(e => e.gender === 'Female').length;}**

**onEmployeeCountRadioButtonChange(selectedRadioButtonValue: string): void {**

**this.selectedEmployeeCountRadioButton = selectedRadioButtonValue;}}**

**Difference between constructor and ngOnInit** 

**1-class constructor is automatically called when an instance of the class is created. It is generally used to initialize the fields of the class and it's sub classes.   
  
2-ngOnInit is a life cycle hook method provided by Angular. ngOnInit is called after the constructor and is generally used to perform tasks related to Angular bindings. For example, ngOnInit is the right place to call a service method to fetch data from a remote**