Angular Services

**What are Angular Services?**

The Angular Services are the piece of code or logic that are used to perform some specific task. A service can contain a value or function or combination of both. The Services in angular are injected into the application using the dependency injection mechanism.

**Why do we need a service in Angular?**

Whenever you need to reuse the same data and logic across multiple components of your application, then you need to go for angular service. That means whenever you see the same logic or data-access code duplicated across multiple components, then you need to think about refactoring the same logic or data access code into a service.

There is a principle in software development i.e. DRY (Don’t Repeat Yourself) and using services in angular safely follow this principle. The logic or data is implemented in a services and the service can be used across multiple components of your angular application. So, services is a mechanism used to share the functionality between the components.

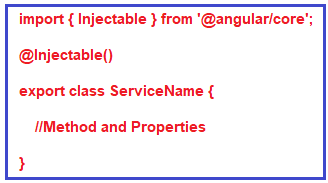
Without Angular Services, you would have to repeat the same logic or code in multiple components wherever you want this code or logic. Now think about the overhead in terms of time and effort required to develop, debug, test and maintain the duplicated code across multiple places instead of having that duplicated code at one central place like a service and reusing that service where required.

**Understanding Angular Service with an example:**

Let us Angular Service step by step with an example. First we will understand how to create angular service and then we will discuss how to use angular service within a component.

**Step1: Creating Angular Service**

The angular service is composed of three things. You need to create an **export**class and you need to decorate that class with **@Injectable** decorator and you need to import the Injectable decorator from the **angular core** library. The syntax to create angular service is given below.



Let say you want to create an angular service for fetching the student details and this student details is going to be used by many components. So, open terminal and type **ng generate service Student** and press enter as shown below.

How to create a service in Angular?

Once you press the enter button it will create two files within the app folder as shown below.

Why do we need a service in Angular?

**Modifying student.service.ts file:**

Open **student.service.ts** file and then copy and paste the following code in it. At the moment we have hard-coded the student data, later in this article series we will discuss how to get this data from restful APIs. Here, you need to focus on two things. The **@Injectable** decorator and the **getStudents** method which returns the student data. As with all other angular decorators, we preceded the name with an @ symbol, and we do not add a semicolon (;) at the end.

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

@Injectable**()**

**export** **class** StudentService **{**

getStudents**()**: **any[]** **{**

**return** **[**

**{**

ID: 'std101', FirstName: 'Preety', LastName: 'Tiwary',

Branch: 'CSE', DOB: '29/02/1988', Gender: 'Female'

**}**,

**{**

ID: 'std102', FirstName: 'Anurag', LastName: 'Mohanty',

Branch: 'ETC', DOB: '23/05/1989', Gender: 'Male'

**}**,

**{**

ID: 'std103', FirstName: 'Priyanka', LastName: 'Dewangan',

Branch: 'CSE', DOB: '24/07/1992', Gender: 'Female'

**}**,

**{**

ID: 'std104', FirstName: 'Hina', LastName: 'Sharma',

Branch: 'ETC', DOB: '19/08/1990', Gender: 'Female'

**}**,

**{**

ID: 'std105', FirstName: 'Sambit', LastName: 'Satapathy',

Branch: 'CSE', DOB: '12/94/1991', Gender: 'Male'

**}**

**]**;

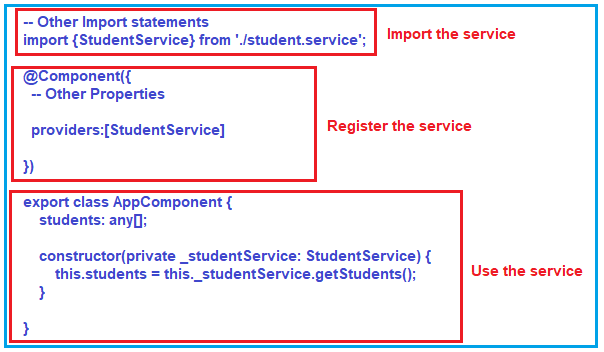
**}**

**}**

**Note:** The **@Injectable()** decorator in angular is used to inject other dependencies into the service. At the moment our service does not have any other dependencies, so, yo can remove the **@Injectable()** decorator and the service should works. However, the Angular Team recommends to always use **@Injectable()** decorator to ensures consistency.

**Step2: Using the Service in Angular**

Once you created the service, the next and the important step is to inject and use the service within the components. It is a three step process. Import the service, register the service and use the service. The syntax is given below.



Here, first we import the service. Then we need to register the Service in the component by declaring it in the providers array. Then we need to Inject StudentService using the constructor. The private variable \_studentService which points to StudentService singleton instance is then available throughout the class.

Here, we want to use the Student Service within our AppComponent class.

**Modifying app.component.ts file:**

Open **app.component.ts** file and then copy and paste the following code in it.

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

**import** **{**StudentService**}** from './student.service';

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

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

providers:**[**StudentService**]**

**})**

**export** **class** AppComponent **{**

students: **any[]**;

**constructor(private** \_studentService: StudentService**)** **{**

this.students = this.\_studentService.getStudents**()**;

**}**

**}**

As you can see in the above code, we declare one variable called students and then initialize this variable using the student service.

**Modifying app.component.html file:**

Now, open app.component.html file and then copy and paste the following code in it.

**<h2>**Angular Service Example**</h2>**

**<table>**

**<thead>**

**<tr>**

**<th>**ID**</th>**

**<th>**First Name**</th>**

**<th>**Last Name**</th>**

**<th>**Branch**</th>**

**<th>**DOB**</th>**

**<th>**Gender**</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr** \*ngFor='let student of students'**>**

**<td>**{{student.ID}}**</td>**

**<td>**{{student.FirstName}}**</td>**

**<td>**{{student.LastName}}**</td>**

**<td>**{{student.Branch}}**</td>**

**<td>**{{student.DOB}}**</td>**

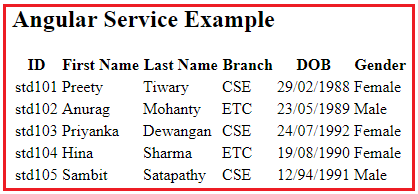
**<td>**{{student.Gender}}**</td>**

**</tr>**

**</tbody>**

**</table>**

With the above changes in place, now browse the application and you should see the student data as expected in the browser as shown in the below image.



**Using ngOnInit() life cycle hook to call the getStudents() service method:**

Let us see how to use ngOnInit to call the service method and then we will see the difference between calling the service method from a constructor and from ngOnInit life cycle hook. So, modify the app.component.ts file as shown below to use ngOnInit life cycle hook to call the getStudents method of Student Service.

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

**import** **{**StudentService**}** from './student.service';

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

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

providers:**[**StudentService**]**

**})**

**export** **class** AppComponent **{**

students: **any[]**;

**constructor(private** \_studentService: StudentService**)** **{** **}**

ngOnInit**()** **{**

this.students = this.\_studentService.getStudents**()**;

**}**

**}**

With the above changes, you will also get the same output.

**Difference between constructor and ngOnInit**

Whenever you create an instance of a class, the class constructor is automatically called. Like other programming languages, the class constructor in angular is also used to initialize the members of the class and it’s sub classes.

The ngOnInit is a life cycle hook method provided by Angular which 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 server. We can also do the same using a class constructor, but the general rule of thumb is, tasks that are time consuming should use ngOnInit instead of the constructor. As fetching data from a remote server is time consuming, the better place for calling the service method is ngOnInit.

In our example, the dependency injection is done using the class constructor and the actual service method call is issued from ngOnInit life cycle hook as shown below.

