**1) Create a basic Angular Application**

Lets first create a basic Angular application:

* 'ng new sample-app'
* 'cd sample-app'

Now let's add 2 components in our app that we want to share data between, and load these components within our AppComponent template:

* 'ng generate component task-list' - This component will need to be able to retrieve the list of tasks from inside of our TaskService.
* 'ng generate component task-new' - This component will need to be able to add a task in the list of tasks inside of our TaskService.

**...app/app.component.html**

<app-task-list></app-task-list>

<app-task-new></app-task-new>

### 2) Create a Task Service

Now, let's create our Task Service. We can do this with the following command in the terminal:

* ng generate service task

Once we have done so, we will have created a class 'TasksService' file.

##### .../app/task.service.ts

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

##### @Injectable()

##### export class TaskService {

##### constructor() { }

##### }

### 3) Inject the Task Service in our AppModule

We need to **import** the TaskService file in our AppModule file 'app/app.module.ts' and **include** it in the providers array. This will 'register' the newly created Service with our Angular application, allowing us to later inject it in a component and utilize it for data management (on the client side).

##### ...app/app.module.ts

##### import { BrowserModule } from '@angular/platform-browser';

##### import { NgModule } from '@angular/core';

##### import { AppComponent } from './app.component';

##### import { TaskListComponent } from './task-list/task-list.component';

##### import { TaskNewComponent } from './task-new/task-new.component';

##### import { TaskService } from './task.service'; // <-- Imported

##### @NgModule({

##### declarations: [

##### AppComponent,

##### TaskListComponent,

##### TaskNewComponent

##### ],

##### imports: [

##### BrowserModule

##### ],

##### providers: [TaskService], // <-- Included

##### bootstrap: [AppComponent]

##### })

##### export class AppModule { }

### 4) Add Data and Methods in the Service

Now, we will add an array of tasks in our service, this array will be where we store all the task information for our application. Components will later be able to retrieve and update this array of tasks, if we ever need to retrieve or update any task information, we will always come to this service variable.

Let's also add 2 methods in our service, a method named retrieveTasks() { } and another named createTask() { }'. 'retrieveTasks' should return the array of tasks from our service, while the 'createTask' method should expect a parameter of a task, using that parameter, it should append the task in our Service tasks array. These methods we are creating will be used by the components calling into our service.

##### .../app/task.service.ts

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

##### @Injectable()

##### export class TaskService {

##### 

##### tasks = ['drink coffee or tea', 'build a web app'];

##### 

##### constructor() { }

##### 

##### retrieveTasks(){

##### return this.tasks;

##### }

##### 

##### createTask(task){

##### this.tasks.push(task);

##### }

##### 

##### }

**5) Component retrieving the tasks array from the TaskService**

Now, to retrieve the task array from the TaskService, and into our Component (TaskListComponent in this case), we need to do a couple of steps (steps are referenced in comments within the sample code below):

* **1) Create a local variable called ‘tasks’** within our component, we will be retrieving the service data and storing it in this variable. We do this because each Component needs its own local variables to utilize for printing (interpolation or data binding) within it's template. Initially, let’s also define the variable to an empty array. tasks = [];
* **2)**Ensure we have **injected the service** in our component
* **3) Decide when we want the data** to be retrieved. It's most likely the case that we want to retrieve the data from the service, as soon as our component loads; for this, we will retrieve the data in our Component's constructor (constructor methods are invoked as soon as the component loads).
* **4)**Invoke the service method to **retrieve the data** and store it in our component's local variable.
* **5) Print the tasks array in our Component’s template**, in our example, we will use \*ngFor to view them all in a list.

**../app/task-list/task-list.component.ts**

import { TaskService } from './../task.service';

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

@Component({

selector: 'app-task-list',

templateUrl: './task-list.component.html',

styleUrls: ['./task-list.component.css']

})

export class TaskListComponent implements OnInit {

tasks = []; // <-- Step 1

constructor(private \_taskService: TaskService) { // <-- Step 2 & 3

this.tasks = this.\_taskService.retrieveTasks(); // <-- Step 4

}

ngOnInit() {

}

}

##### ../app/task-list/task-list.component.html

##### <ul>

##### <li \*ngFor="let task of tasks">{{ task }}</li>

##### </ul>

Now, our Component has the same data as our Service. The data was pulled as soon as our Component loaded. This is an important distinction. We retrieved the data from the service 1 time, after this point in time, if for whatever reason the service data changes, our component’s data will not automatically present the new data. We would need to re-invoke the this.taskService.retrieveTasks() method, to retrieve the new set of data from our service. This kind of situation would only occur if we had other components on the page that were altering the service data.

As we mentioned earlier, Services are simply classes instantiated into objects. Therefore, when we invoke the Service methods, we are simply calling on the Service object (though dependency injection) and invoke methods. Since the Service is simply an object, we can also call directly on the variables (and not just the methods of the service object). This way, we can simply redefine our Component's variable tasks, to be the value of our Service's variable tasks.

##### ../app/task-list/task-list.component.ts

import { TaskService } from './../task.service';

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

@Component({

selector: 'app-task-list',

templateUrl: './task-list.component.html',

styleUrls: ['./task-list.component.css']

})

export class TaskListComponent implements OnInit {

tasks = [];

constructor(private \_taskService: TaskService) {

this.tasks = this.\_taskService.tasks; // <-- adjusted code, retrieving the tasks data directly.

}

ngOnInit() {

}

}

This will work for now, until our Service makes HTTP requests for the data, and then, we will need to use the functions coupled with a callback to return the data to our Component, when and if the HTTP request made by the service was successful. More on this in the next module.