# Angular -HTTP Client module

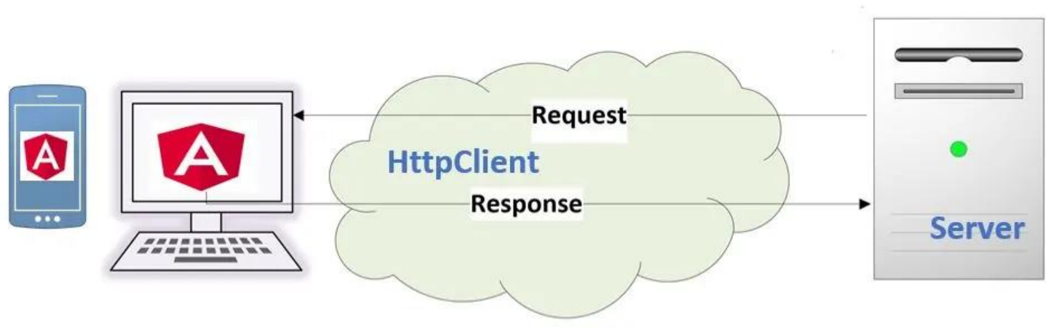
- HTTP protocol- (Hyper Text Transfer protocol)-Using HTTP protocol we can share information between client and server. Http is a stateless protocol.

- HTTP Specification Methods

  - GET( getting the information from the resource)
  - POST-(Creating a new resource)
  - PUT-(Updating existing resource)
  - DELETE-(Removing the resources)
  - TRACE
  - OPTIONS
  - HEAD

- To interact with backend services Rest API can be used. Rest API is designed using HTTP protocol.

- Most front-end applications need to communicate with a server over the HTTP protocol, in order to download or upload data and access other back-end services. Before Angular 4.0 HTTP module was used to communicate was used to communicate with backend applications . From Angular 4.0 version, it provides a simplified client HTTP API for Angular applications, the HttpClient service class in @angular/common/http.



- ➢ Most of all the web applications communicate with backend servers through REST APIs (which are based on the HTTP protocol) using either the XMLHttpRequest interface or the fetch() API. Angular HttpClient makes use of the XMLHttpRequest interface that supports both modern and legacy browsers.

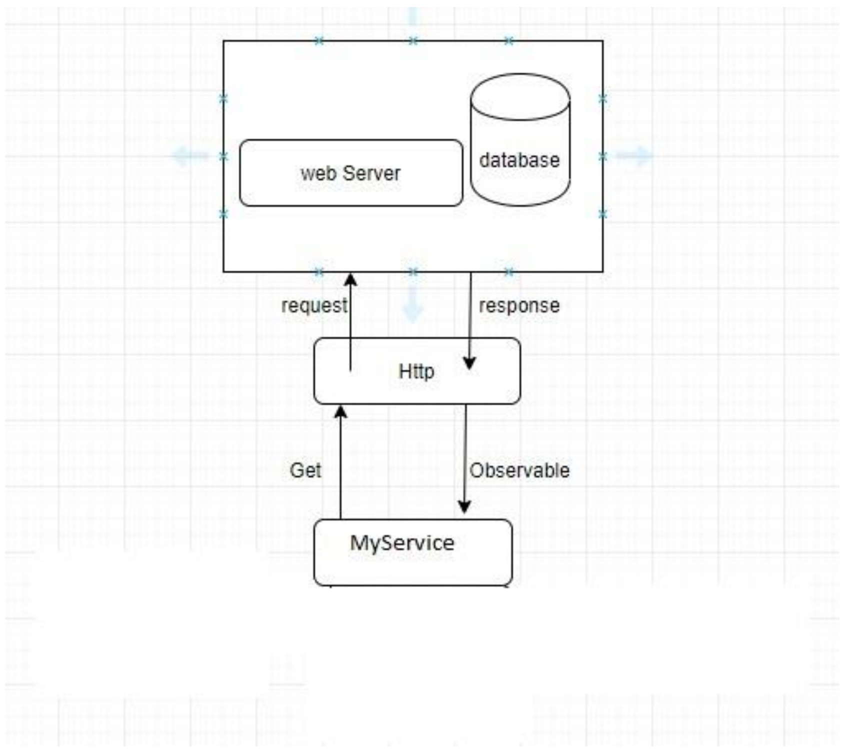
- ➢ Methods available in HTTP client module

  - ♦ get()
  - ♦ post()
  - ♦ put()
  - ♦ delete()
  - ♦ head()
  - ♦ jsonp()
  - ♦ options()
  - ♦ patch()

HTTP is just a two-way process, first is to send the request and second is to receive the response. The HTTP request will hit the web API or service that will in return fetch the data from the database and send it back as an HTTP response. HTTP response is nothing but the Observable returned by HttpClient.get(). Every method in HTTP client method ,the return type is Observable.

<u>Syntax of get() method</u>

- ✓ get(url:String|HttpRequest<any>,url?:String,options:{……}):Observable<any>
- ✓ get(url: String, options:{…}):Observable<any>



**Example 1:**

To get the data from a predefined API

**<u>Steps</u>**

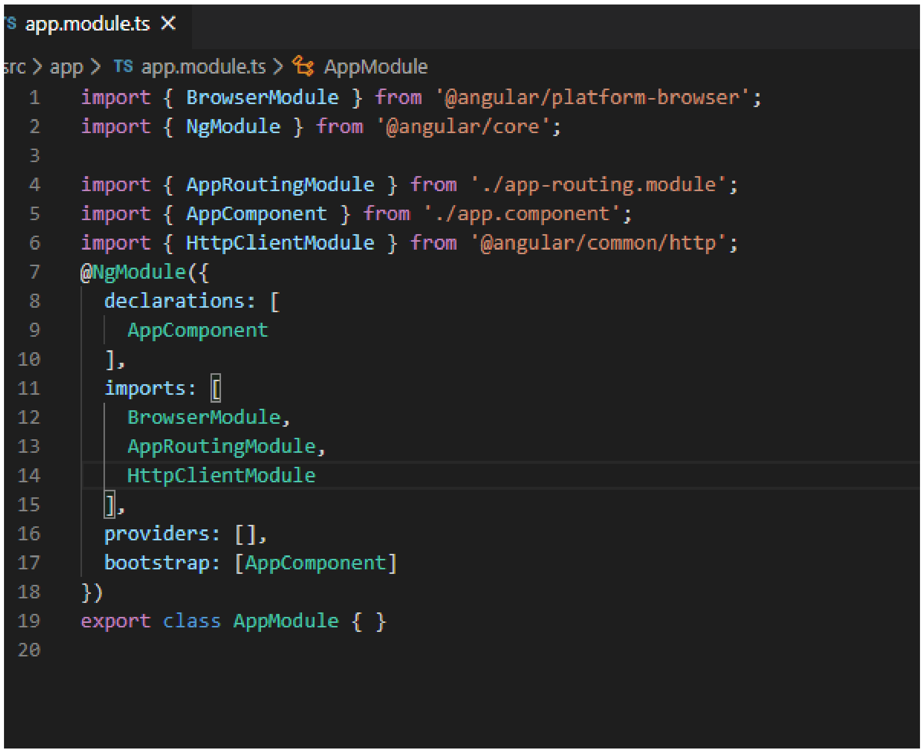
Let us start using the HTTP functionality.

**Step 1: Open your application.**

Open app.module.ts and add below contents,

```
import { HttpClientModule } from ‘@angular/common/http’;
imports: [
   BrowserModule,
   HttpClientModule
  ]
```

Note: Always try to include HttpClientModule only after the BrowserModule in imports.



```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

**Step 2: Create a service named MyService**

**ng g service MyService.**

In the MyServiceService class, define a predefined api ,

**private apiurl="https://jsonplaceholder.typicode.com/users";**

And add HttpClient to MyServiceService using Constructor injections as below:

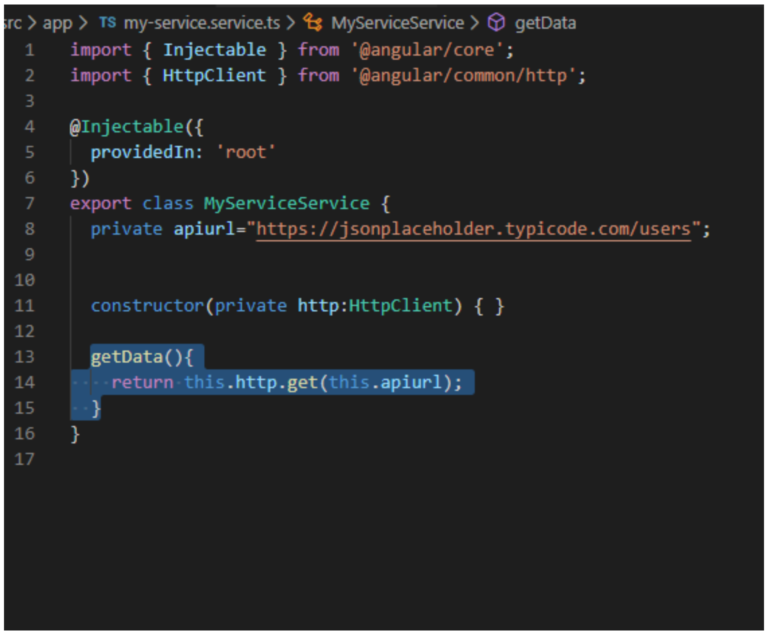
**constructor(private http:HttpClient);**

Define a method getData(), the data method to call the API. We are calling the HTTP GET method to get a list of all the users. We requested the API URL, https://jsonplaceholder.typicode.com/users.

**getData(){**

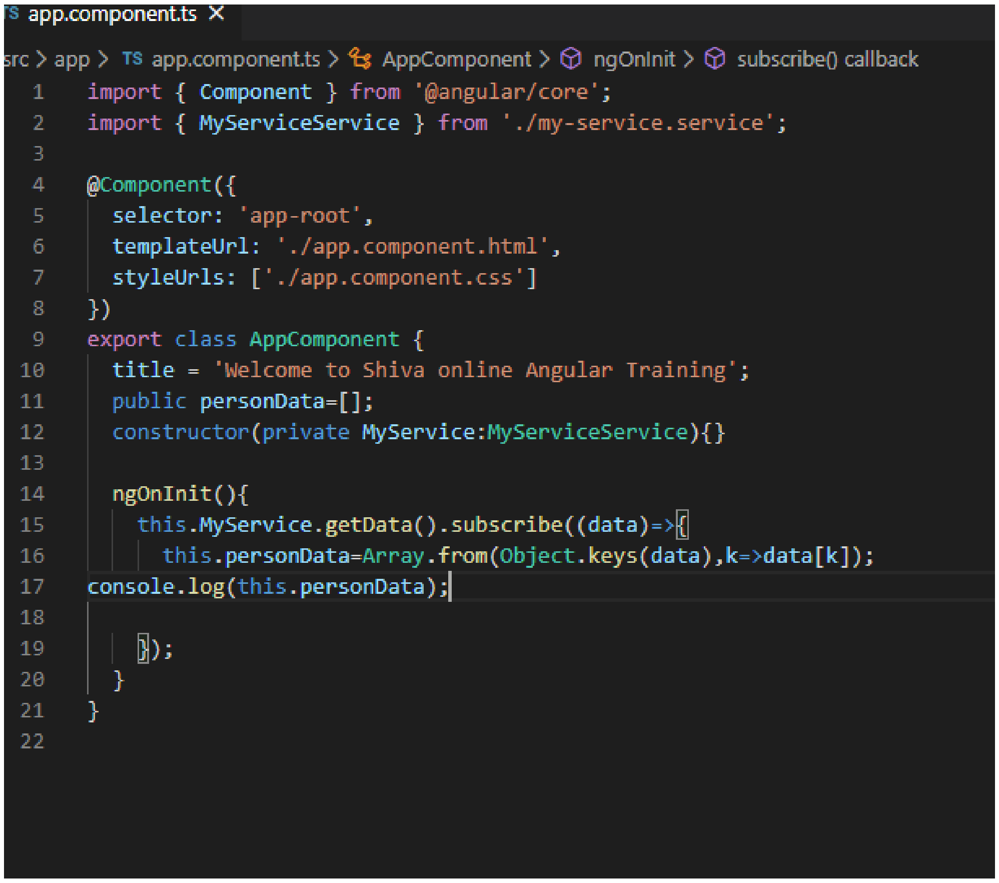
**return this.http.get(this.apiurl);**

**}**


rc > app > TS my-service.service.ts > MyServiceService > getData

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

@Injectable({
  providedIn: 'root'
})
export class MyServiceService {
  private apiurl="https://jsonplaceholder.typicode.com/users";


  constructor(private http:HttpClient) { }

  getData(){
    return this.http.get(this.apiurl);
  }
}

```

**Step 3: In app.component.ts, the AppComponent injects the MyService and calls the getData() method which is called at the *ngOnInit()* method. Subscribe method is used to connect an Observer to an Observable. The observable data that is obtained as response should be converted to Array format using Array.from() method. Use the Object.keys() method to obtain keys of the object. To print the data, use the console.log() method.**


app.component.ts ✕
src > app > TS app.component.ts > AppComponent > ngOnInit > subscribe() callback

```
import { Component } from '@angular/core';
import { MyServiceService } from './my-service.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Welcome to Shiva online Angular Training';
  public personData=[];
  constructor(private MyService:MyServiceService){}

  ngOnInit(){
    this.MyService.getData().subscribe((data)=>{
      this.personData=Array.from(Object.keys(data),k=>data[k]);
console.log(this.personData);

    });
  }
}
```

**Step 4: To display user data in the browser go to app.component.html and add the below code:**

**<h1>Customer Information</h1>**

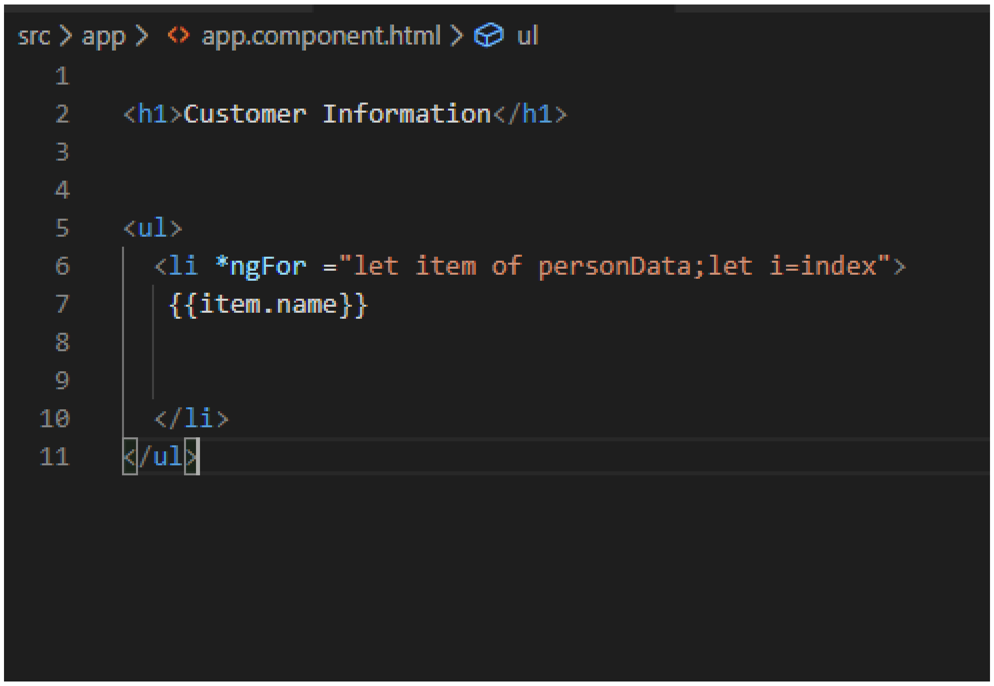
**<ul>**

**<li *ngFor ="let item of personData;let i=index">**

**{{item.name}}**

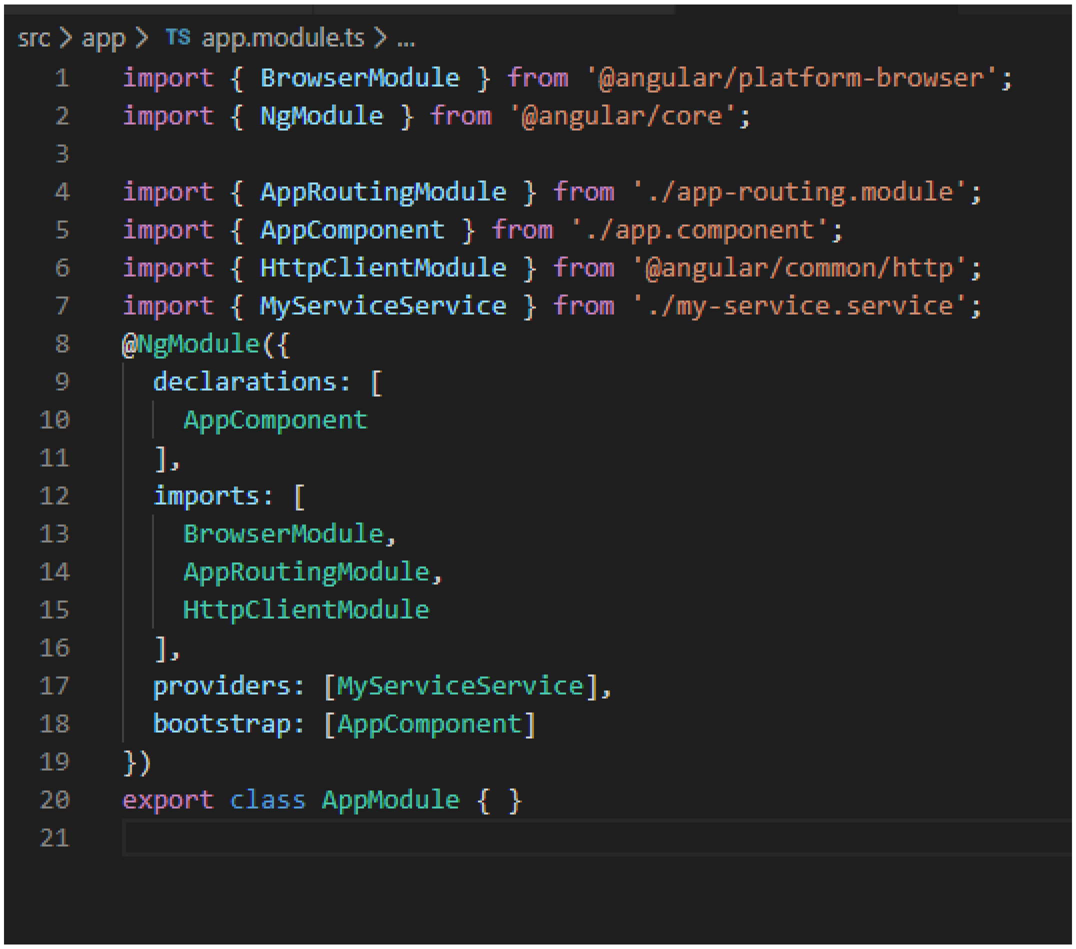
**</li>**

**</ul>**



```

<h1>Customer Information</h1>


<ul>
   <li *ngFor ="let item of personData;let i=index">
    {{item.name}}


   </li>
</ul>
```

**Step 5: Register the service with module.ts**


src > app > TS app.module.ts > ...

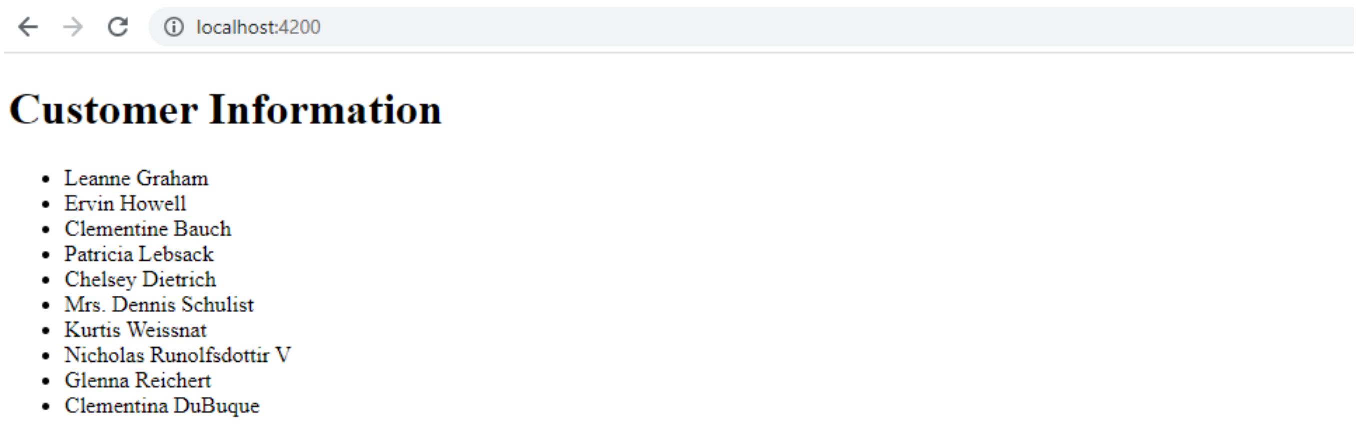
```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';
import { MyServiceService } from './my-service.service';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [MyServiceService],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

**Step 6: Next, navigate to your project's folder and run the local development server using the following command:**

**ng serve**

A local development server will start listening on the http://localhost:4200/ address.

Open your web browser and navigate to the http://localhost:4200/ address to see your app up and running. This is a screenshot at this point:


localhost:4200

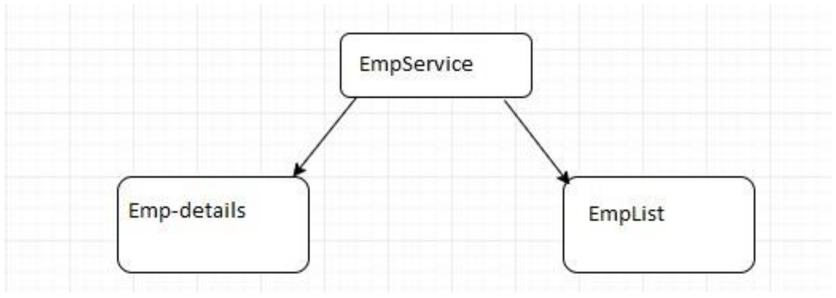
Customer Information

- Leanne Graham
- Ervin Howell
- Clementine Bauch
- Patricia Lebsack
- Chelsey Dietrich
- Mrs. Dennis Schulist
- Kurtis Weissnat
- Nicholas Runolfsdottir V
- Glenna Reichert
- Clementina DuBuque

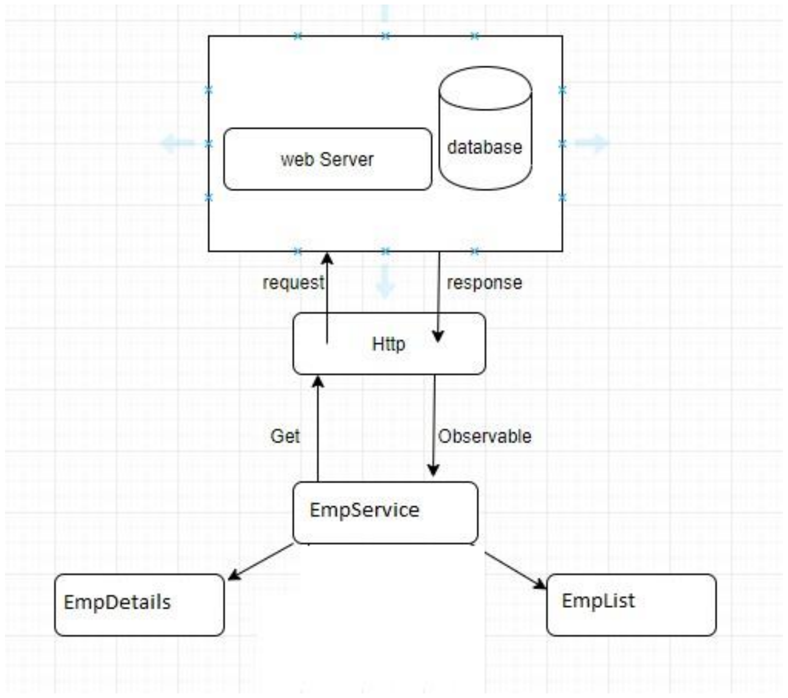
## <u>Example 2</u>

Create a .json file which contains the employee information  in the application and from that file fetching the data and displaying in the browser using HTTP client get method.

➢ Consider a service EmpService that will be injected into two components EmpList and EmpDetail and should solely be responsible to deliver the data to these components. (Here we are considering only Emp-details Component)

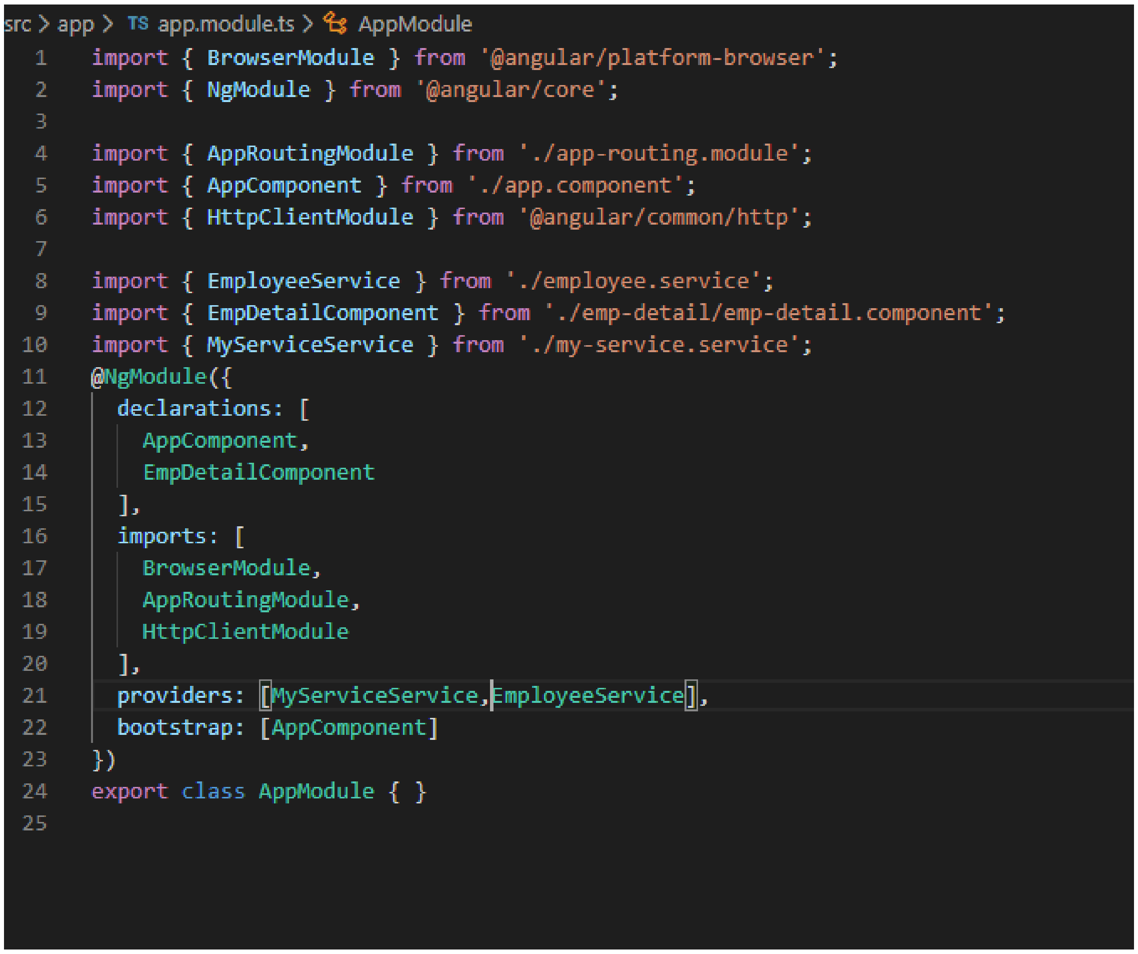


- Now, what we need to achieve is to return the same data but with the help of HTTP.



In the above diagram, the EmpService will call or make the HTTP request from the HTTP module. This HTTP request will hit the web API or service that will in return fetch the data from the database and send it back as an HTTP response. Now the observable needs to be cast to the particular type and the EmpService will cast this observable data to the array of employees and return it to the subscribed component. EmpDetails will display id and name of employees.

**Step 1: Open your application. Open app.module.ts and add below contents, Import the HttpClientModule.**

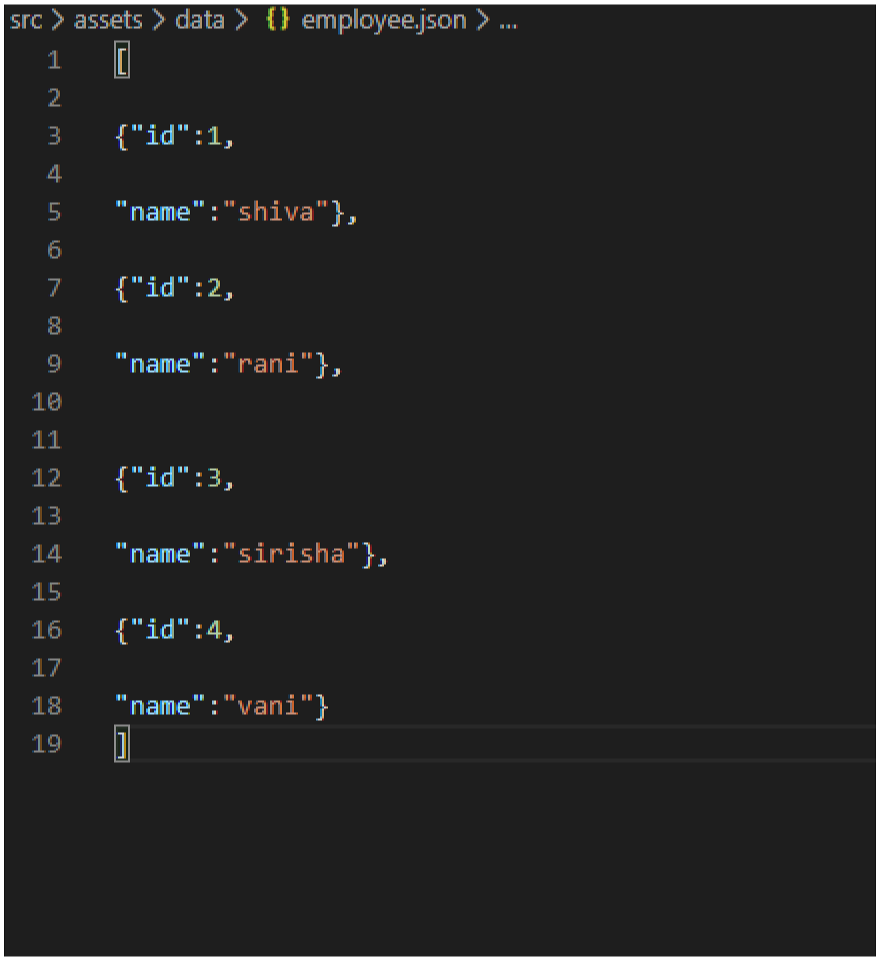


src > app > TS app.module.ts > AppModule

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';

import { EmployeeService } from './employee.service';
import { EmpDetailComponent } from './emp-detail/emp-detail.component';
import { MyServiceService } from './my-service.service';
@NgModule({
  declarations: [
    AppComponent,
    EmpDetailComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [MyServiceService,EmployeeService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

**Step 2: Create a json file on location '/assets/data/employee.json'.**

Add the below contents under employee.json



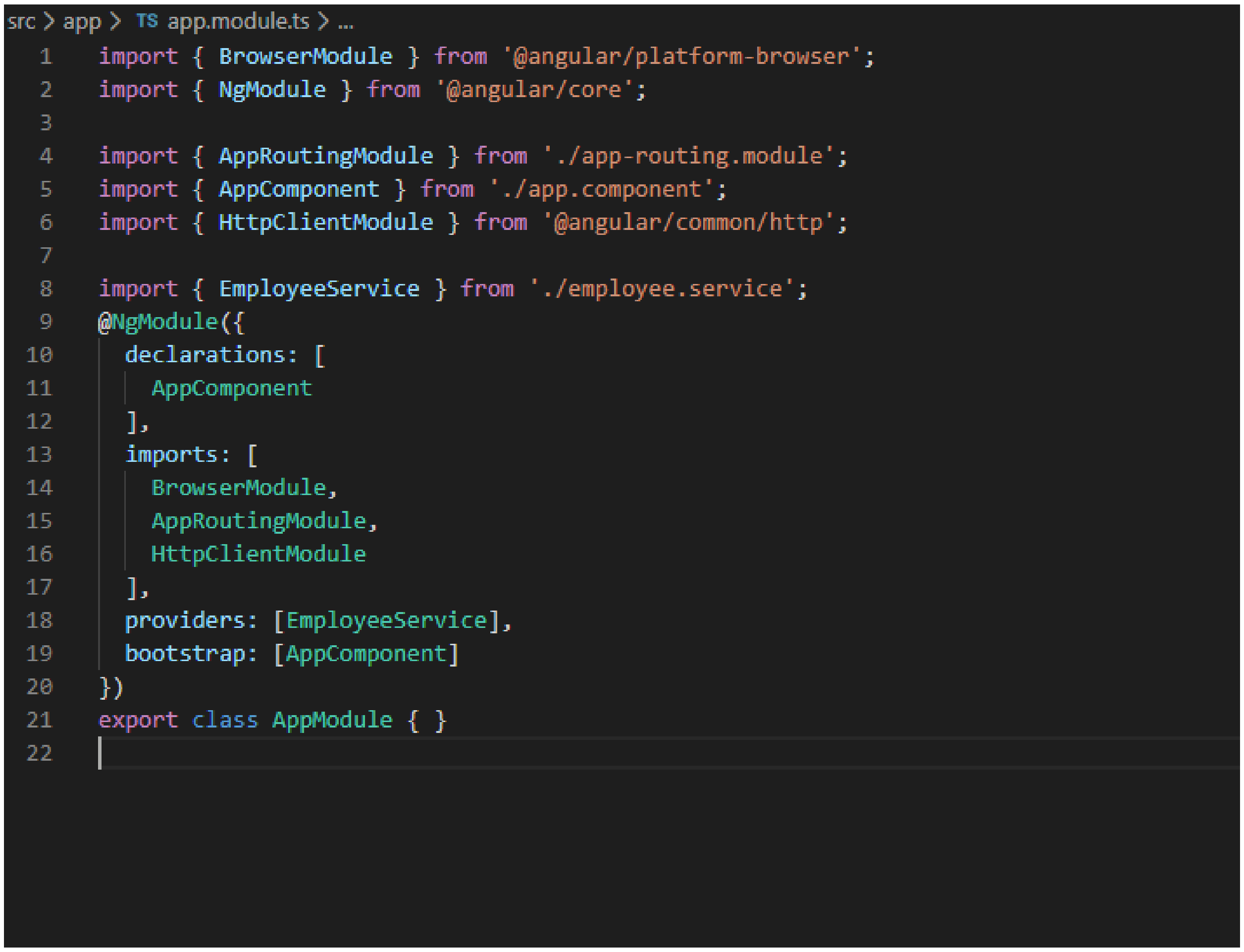
src > assets > data > {} employee.json > ...

```
[

{"id":1,

"name":"shiva"},

{"id":2,

"name":"rani"},


{"id":3,

"name":"sirisha"},

{"id":4,

"name":"vani"}
]
```

**Step 3: Create a service employee**

**ng g service employee**

Register the service with app.module.ts



src > app > TS app.module.ts > ...

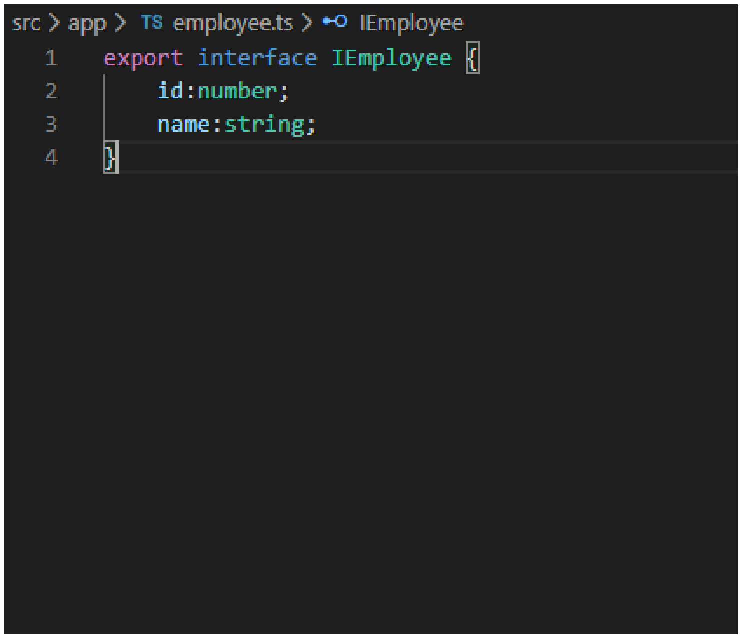
```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';

import { EmployeeService } from './employee.service';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [EmployeeService],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

**Step 4:**

**The return type of method that we are going to define in employeeservice.ts  is observable and we need to convert that data into the required type. To convert the data that we obtain in string format to the data in array format we can add a typescript file.**

**Add a file employee.ts under app directory and add the below contents,**

```
export interface IEmployee {
   id:number;
   name:string;
}
```



```
src > app > TS employee.ts > IEmployee
export interface IEmployee {
    id:number;
    name:string;
}
```

**Step 5: Open employeeService.ts and add the below code**

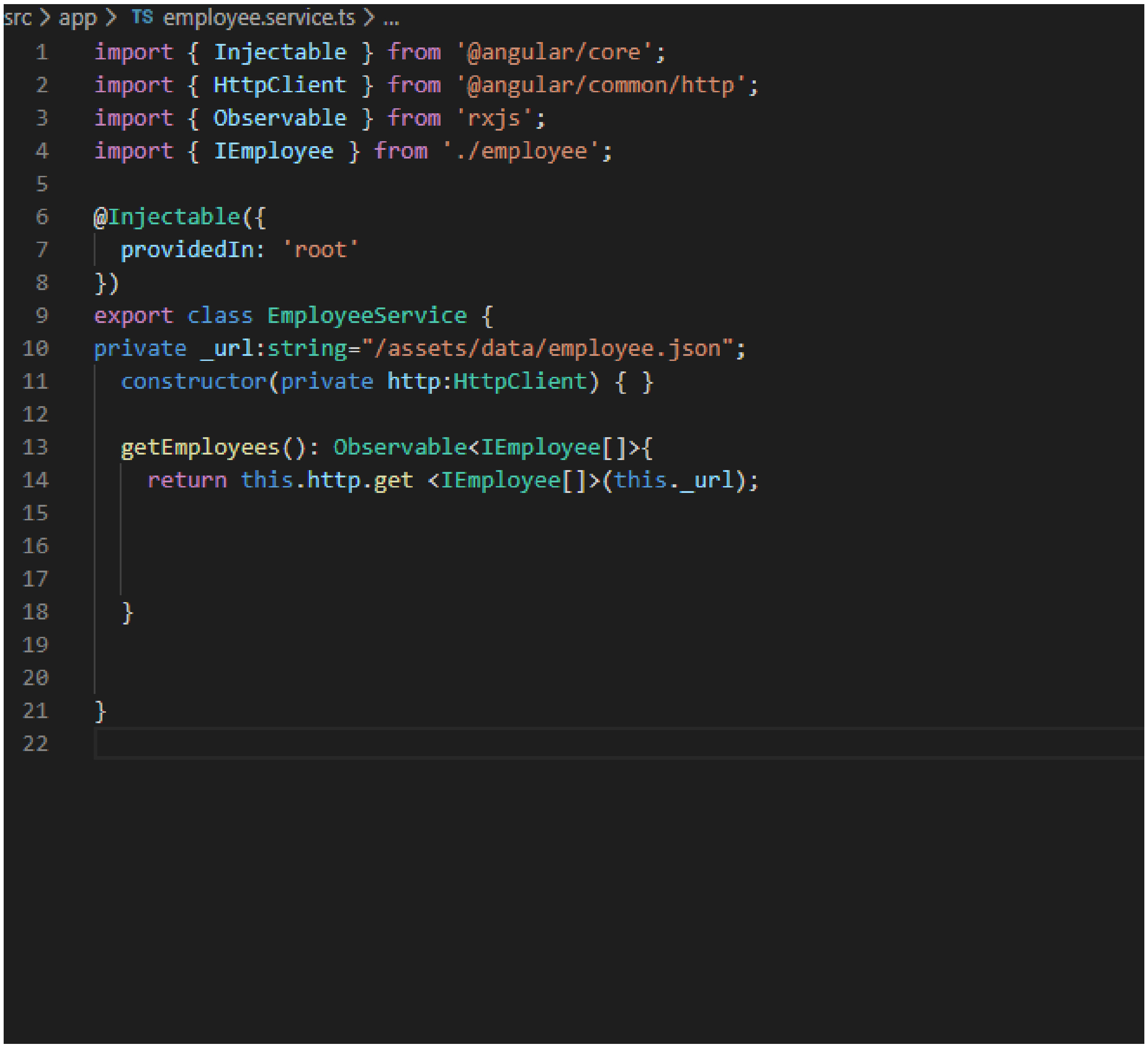
**private url:string=”/assests/data/employee.json”;** //location of the json file

And add HttpClient to EmployeeService using Constructor injections as below:

**constructor(private http:HttpClient) { }**

Define a method getEmployees(), the data method to call the API.

```
getEmployees(): Observable<IEmployee[]>{
   return this.http.get <IEmployee[]>(this._url);
 }
```



src > app > TS employee.service.ts > ...

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { IEmployee } from './employee';

@Injectable({
  providedIn: 'root'
})
export class EmployeeService {
private _url:string="/assets/data/employee.json";
  constructor(private http:HttpClient) { }

  getEmployees(): Observable<IEmployee[]>{
    return this.http.get <IEmployee[]>(this._url);



  }


}
```

**Step 6: Create a component named emp-details using the code given below:**

**ng g component emp-detail**

The Emp-details Component injects the EmployeeService and calls the getEmployees() method which is called at the *ngOnInit()* method.

**constructor(private employeeService : EmployeeService) {**

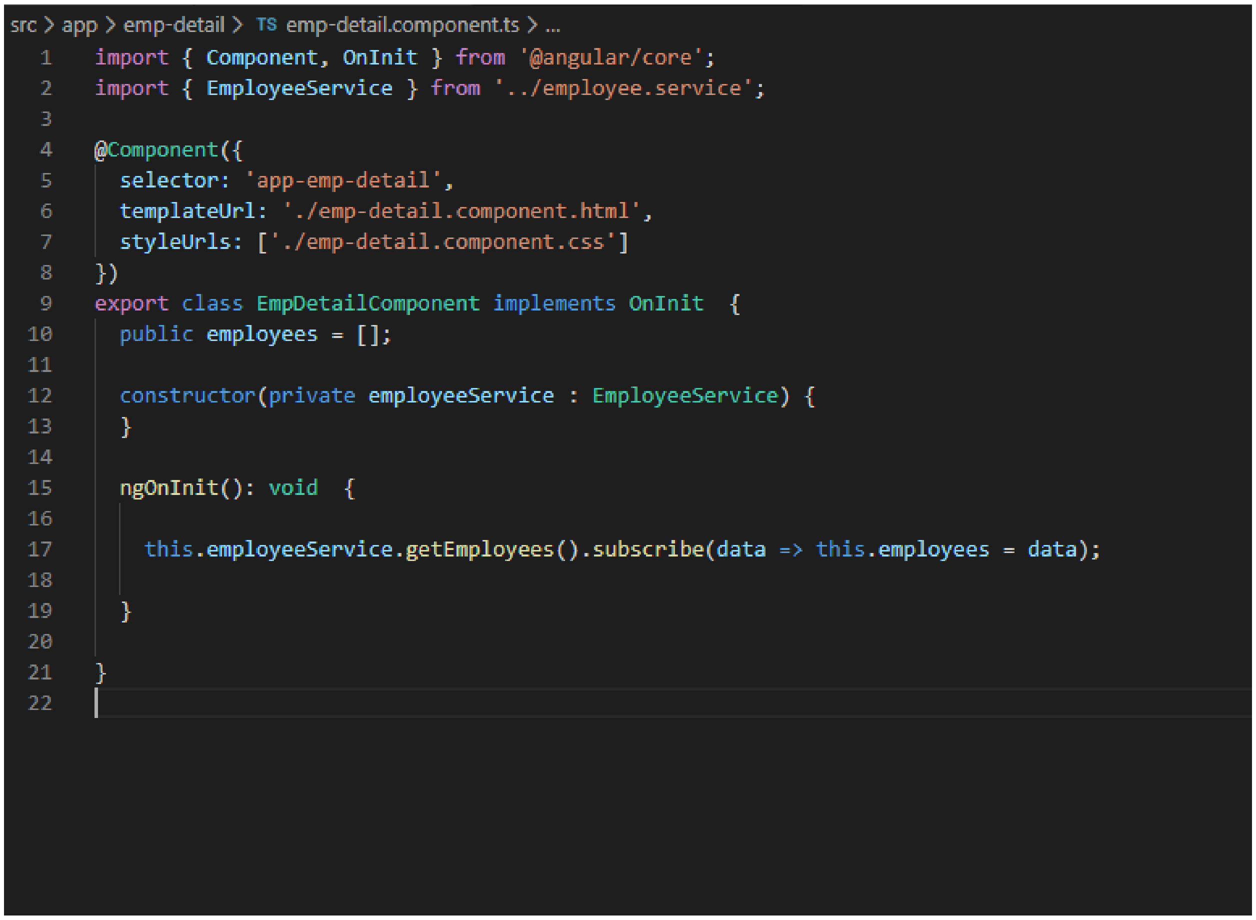
**}**

When we subscribe we get the data and assign it to the property in that component class.

**ngOnInit(): void {**

**this.employeeService.getEmployees().subscribe(data => this.employees = data);**

**}**



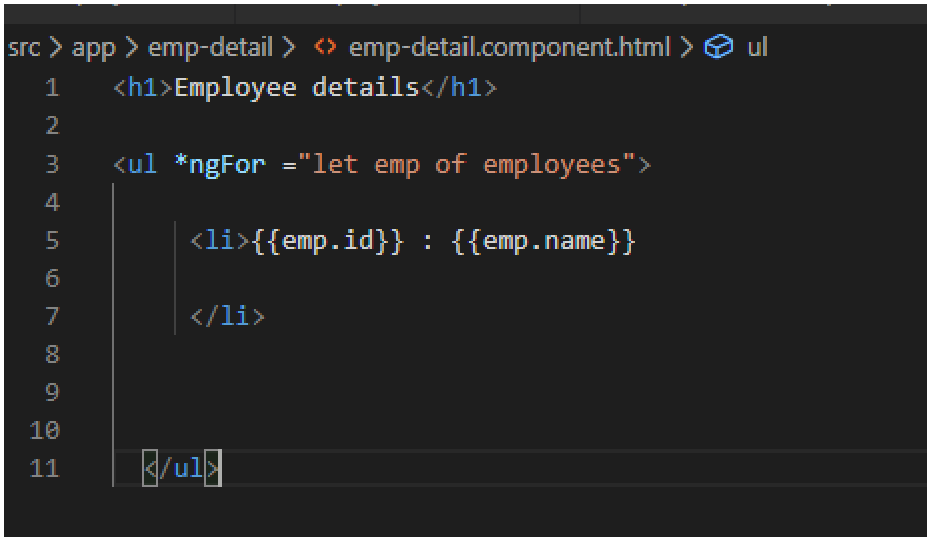
src > app > emp-detail > TS emp-detail.component.ts > ...

```
import { Component, OnInit } from '@angular/core';
import { EmployeeService } from '../employee.service';

@Component({
  selector: 'app-emp-detail',
  templateUrl: './emp-detail.component.html',
  styleUrls: ['./emp-detail.component.css']
})
export class EmpDetailComponent implements OnInit  {
  public employees = [];

  constructor(private employeeService : EmployeeService) {
  }

  ngOnInit(): void  {

    this.employeeService.getEmployees().subscribe(data => this.employees = data);

  }

}

```

**Step 7: Display the data in the browser  using data binding.**

**Add the below code to emp-detail.component.html**

```
<h1>Employee details</h1>
<ul *ngFor ="let emp of employees">
    <li>{{emp.id}} : {{emp.name}}
    </li>
  </ul>
```



src > app > emp-detail > emp-detail.component.html > ul

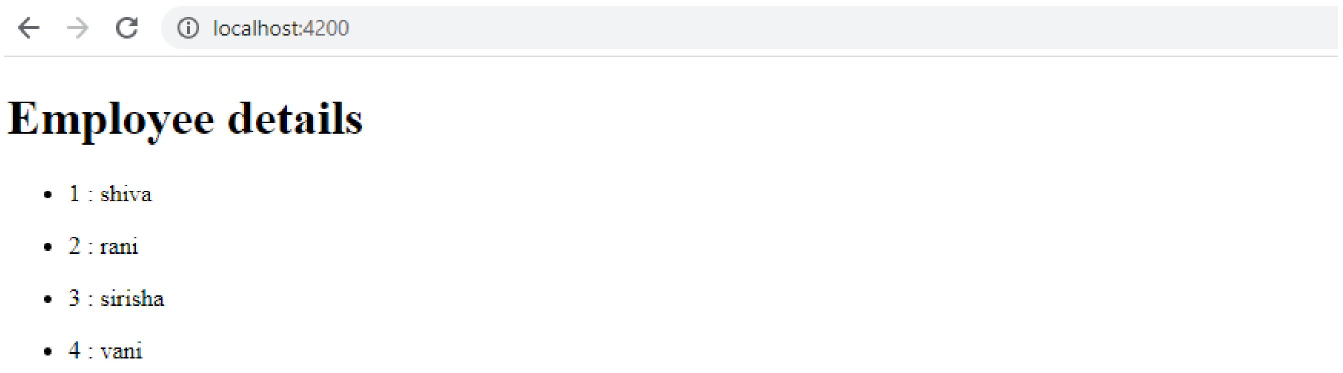
```
<h1>Employee details</h1>

<ul *ngFor ="let emp of employees">

    <li>{{emp.id}} : {{emp.name}}

    </li>



  </ul>
```

**Step 8: Next, navigate to your project's folder and run the local development server using the following command:**

**ng serve**

A local development server will start listening on
the http://localhost:4200/ address.

Open your web browser and navigate to the http://localhost:4200/ address to see your app up and running. This is a screenshot at this point:



localhost:4200

Employee details

- 1 : shiva
- 2 : rani
- 3 : sirisha
- 4 : vani