STEP 14: REST API In Angular

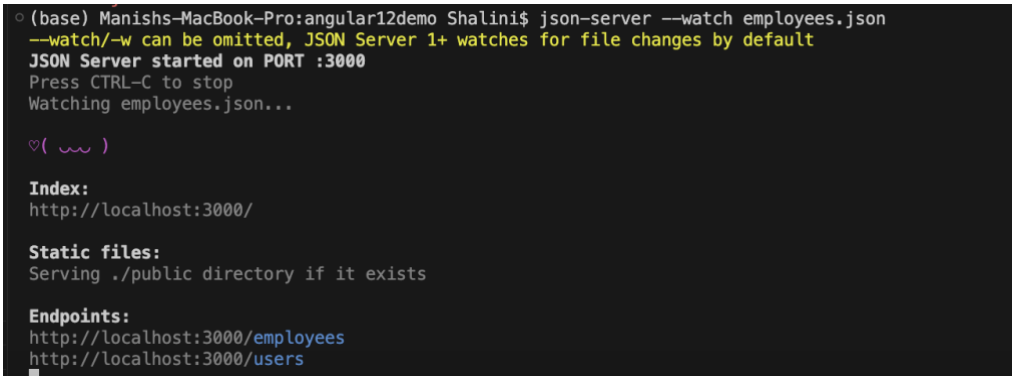
1. Open command prompt or terminal and install json-server that is a light weight server using below command:
   npm install -g json-server

2. Create a file employees.json within the angulardemo folder and add the below code:

```
    {
 "employees": [
  {
   "eid": 1,
   "ename": "shalini",
   "password": "shalini",
   "email": "shalini@gmail.com",
   "phone": "1321312312",
   "address": {
    "country": "India",
    "city": "Delhi",
    "zipcode": 787878
   },
   "id": 1
  },
  {
   "eid": 2,
   "ename": "shalini123",
   "password": "shalini123",
   "email": "shalini@gmail.com",
   "phone": "1321312312",
   "address": {
    "country": "India",
    "city": "Delhi",
    "zipcode": 909090
   },
   "id": 2
  },
  {
   "eid": 3,
   "ename": "shalini123",
   "password": "shalini123",
   "email": "shalini@gmail.com",
   "phone": "1321312312",
   "address": {
    "country": "India",
    "city": "Delhi",
    "zipcode": 909090
   },
   "id": 3
```

```
    },
    {
      "eid": "4",
      "ename": "dummy",
      "email": "dumy@f.c",
      "password": "dummy",
      "phone": "4567890",
      "address": {
        "city": "Mum",
        "country": "India",
        "zipcode": "4567"
      },
      "id": 4
    },
    {
      "eid": "5",
      "ename": "pooja",
      "email": "p@s.d",
      "password": "pooja",
      "phone": "567896789",
      "address": {
        "city": "Pune",
        "country": "India",
        "zipcode": "79799"
      },
      "id": 5
    },
    {
      "eid": "6",
      "ename": "Rushal",
      "email": "rush@d.com",
      "password": "rushil",
      "phone": "67890678",
      "address": {
        "city": "Delhi",
        "country": "India",
        "zipcode": "56789"
      },
      "id": 6
    }
  ],
  "users":[
    {"username":"abc", "password":"abc123"},
    {"username":"pqr", "password":"pqr123"},
    {"username":"user", "password":"user123"}
  ]
}
```

3. Open VSCode terminal from the root of angulardemo folder and execute below command:
   json-server --watch employees.json



4. Open endpoints on the browser and you will see the employees data exposed as a REST API

5. Lets understand observables since angular HTTP calls return data of type Observables:

6. Create observables component:
   ng g c observables

7. Update observables html as follows:

```
<p >Observable Basics</p>
<hr/>
<b>Observable Data </b>
<div *ngFor="let f of fruits"> {{ f | uppercase }}</div>
<hr>
<div>
 <b>Error Status :</b>
{{anyErrors ? 'error occured ' : 'It All Good'}}
 <hr>
</div>
<div> <b> completion status : </b> {{ finished ? 'Observer completed ': "
}}</div>
<hr>
<button (click)="Start()">Start Emitting</button>
```

8. Lets update observables ts file :

```
data:Observable<string> | null;
  fruits: Array<string> = [];
  anyErrors: boolean =false;
  finished: boolean = false;
  sub:any;

  Start(){
    this.data = new Observable (observer =>
    {
      setTimeout(() => {  observer.next('Apple'); }, 1000);
      setTimeout(() => {  observer.next('mango'); }, 2000);
      setTimeout(() => {  observer.next('Orannge'); }, 3000);
      setTimeout(() => {  observer.next('banana'); }, 4000);
      setTimeout(() => {  observer.next('grapes'); }, 5000);
      setTimeout(() => {  observer.next('watermelon'); }, 6000);
     // setTimeout(() => {  observer.error('something went wrong'); }, 4000);
      setTimeout(() => {  observer.complete(); }, 7000);
    })

     this.sub = this.data.subscribe(fruit => {
       console.log(fruit);
       this.fruits.push(fruit)
      },
      error => this.anyErrors = true,
      () => this.finished = true)

}
  constructor() {
    this.data = null;
  }
  ngOnDestroy(): void {
```

```
  this.sub.unsubscribe()
 }

 ngOnInit(): void {
 }
```

https://www.telerik.com/blogs/angular-basics-comparing-data-producers-javascript-functions-promises-iterables-observables

https://www.telerik.com/blogs/angular-basics-introduction-observables-rxjs-part-1

https://www.telerik.com/blogs/angular-basics-introduction-observables-rxjs-part-2

9. To configure for listening to HTTP endpoint in angular, add HttpClientModule in app.module.ts file

10. Execute below command from within the service folder:
    ng g s emphttp

11. Add below property in emphttp service
    url:string = "http://localhost:3000/employees";

12. Update emphttp service to make a REST API call:

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

getAllEmployees():Observable<any>
{
 return this.http.get<any>(this.url);
}
getEmployeeById(eid:number):Observable<Employee>
{
 return this.http.get<Employee>(this.url+'/'+eid);
}
addEmployee(employee:Employee):Observable<Employee>
{
 return this.http.post<Employee>(this.url, employee);
}
updateEmployee(employee:Employee):Observable<Employee>
{
 return this.http.put<Employee>(this.url+'/'+employee.eid, employee);
}
deleteEmployee(eid:number){
 this.http.delete(this.url+'/'+eid)
}
```

13. Update employees list to use the service to fetch data from REST API

```
constructor(private empservice:EmphttpService){}
```

```
  ngOnInit(): void {
 this.empservice.getAllEmployees()
 .subscribe(resp => {
  console.log('fetched employees')
  console.log(resp);
   this.employees = resp.employees;
 })
}
```

14. Likewise update emp form to make a POST request and respectively for update and delete as well.