**Lesson23 Angular 2 http service tutorial**

**Notes:-**

**1-Import the angular HTTP module : The first step is to import**

**HttpModule which is present in a separate javascript file - @angular/http.**

**import { HttpModule } from '@angular/http';**

**@NgModule({**

**imports: [**

**BrowserModule,**

**FormsModule,**

**HttpModule],**

**declarations: [**

**AppComponent,**

**EmployeeComponent,**

**EmployeeListComponent,**

**EmployeeTitlePipe,**

**EmployeeCountComponent,**

**SimpleComponent],**

**bootstrap: [AppComponent]})**

**export class AppModule { }**

**2-Modify angular EmployeeService to issue a GET request using the builtin http service: The angular EmployeeService is in employee.service.ts file.**

**A-Use the EmployeeService class constructor to inject Angular Http service. The inject http service can then be used anywhere in this class to call a web service over http.**

**B-Since this Angular EmployeeService class has an injected dependency, @Injectable() decorator is required on this class. If there are no injectable dependencies then we may omit the @Injectable() decorator, but angular strongly recommends to use the @Injectable() decorator irrespective of there are injectable dependencies or not for consistency and future proof.**

**C-Notice in the getEmployees() method, we are using the get() method of the angular http service to issue a get request over http. If you right click on get() method and go to it's definition you will notice that this method return Observable<Response>.**

**D-Observable<Response> is not that useful to us, so we have set the return type of getEmployees() method to Observable<IEmployee[]>**

**E-To convert Observable<Response> to Observable<IEmployee[]> we are using the map operator provided by rxjs.**

**F-At the moment, we are not handling exceptions. We will discuss how to handle exceptions in our upcoming videos.**

**What is an Observable?**

**1-Observable is an asynchronous pattern. In the Observable pattern we have an Observable and an Observer. Observer observes the Observable. In many implementations an Observer is also called as a Subscriber.**

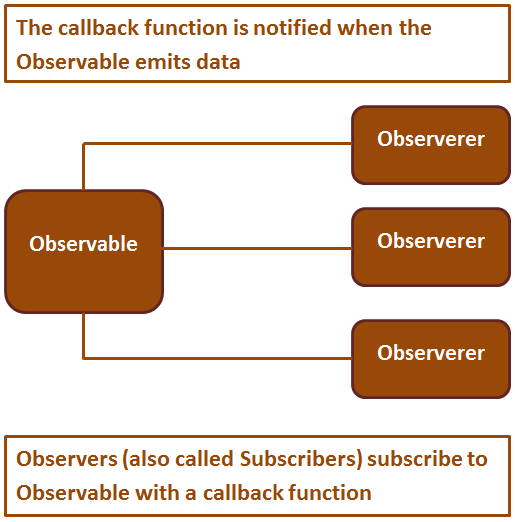
**2-An Observable can have many Observers (also called Subscribers).**

**3-Observable emits items or notifications over time to which an Observer (also called Subscriber) can subscribe.**

**4-When a subscriber subscribes to an Observable, the subscriber also specifies a callback function.**

**5-This subscriber callback function is notified as and when the Observable emits items or notifications.**

**6-Within this callback function we write code to handle data itmes or notifications received from the Observable.**



**3-on the Employee-Service.ts write the following code**

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

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

**// Import Http & Response from angular HTTP module**

**import { Http, Response } from '@angular/http';**

**// Import Observable from rxjs/Observable**

**import { Observable } from 'rxjs/Observable';**

**// Import the map operator**

**import 'rxjs/add/operator/map';**

**@Injectable()**

**export class EmployeeService {// Inject Angular http service**

**constructor(private \_http: Http) { }**

**// Notice the method return type is Observable<IEmployee[]>**

**getEmployees(): Observable<IEmployee[]> {**

**// To convert Observable<Response> to Observable<IEmployee[]>**

**// we are using the map operator**

**return this.\_http.get('http://localhost:24535/api/employees')**

**.map((response: Response) => <IEmployee[]>response.json());}}**

**4-on the component employee-list.component.ts we invoke it using the subscribe method**

ngOnInit() {

    this.\_employeeService.getEmployees()

        .subscribe(employeesData => this.employees = employeesData);

}

**Notice to the subscribe() function we are passing another arrow function as a parameter. This arrow function is called when the Observable emits an item. In our case the Observable emits an array of IEmployee objects. employees Data parameter receives the array of Employee objects, which we are then using to initialize employees property of the EmployeeListComponent class. We can specify up to 3 callback functions as parameters to the subscribe () method as shown below.**

|  |  |
| --- | --- |
| **Callback Method** | **Purpose** |
| onNext | The Observable calls this method whenever the Observable emits an item. The emitted item is passed as a parameter to this method |
| onError | The Observable calls this method if there is an error |
| onCompleted | The Observable calls this method after it has emitted all items, i.e after it has called onNext for the final time |

**5-we have to apply the check that the list employee have objects by writing**

//we have to make check that the array having objects or not

<app-employee-count \*ngIf="employeeService.Employees" [all]="GetAllEmployees()" [male]="GetMaleEmployees()" [female]="GetFemaleEmployees()" (countRadioButtonSelectionChanged)="onEmployeeCountRadioButtonChange($event)"></app-employee-count>

<table>

<thead>

<tr><th colspan="4">Employee Details</th></tr>

<tr><td></td><th style="text-align: left;">Full Name</th>

<th style="text-align: left;">Gender</th>

<th style="text-align: left;">Salary</th></tr>

</thead>

<tbody>

<ng-container \*ngFor="let employee of employeeService.Employees;">

<tr \*ngIf="selectedEmployeeCountRadioButton=='All' ||

selectedEmployeeCountRadioButton==employee.Gender">

<td>{{count}}</td>

<td>{{GetFullName(employee.Fname,employee.Lname) | employeeTitle:employee.Gender}}</td>

<td>{{employee.Gender}}</td>

<td>{{employee.Salary | currency:'USD':true:'1.4-4'}}</td>

</tr>

</ng-container>

//we make check that the array having objects or not

<tr \*ngIf="!employeeService.Employees">

<td colspan="4">

Loading Employees .... Please Wait

</td>

</tr>

<tr \*ngIf="employeeService.Employees && employeeService.Employees.length == 0"><td colspan="4">No Employees to display</td></tr>

</tbody>

</table>

**6-on the web.config on the WEBAPI , we have to allow cross origin by filling the code below**

**<system.webServer>**

**<httpProtocol>**

**<customHeaders>**

**<add name="Access-Control-Allow-Origin" value="\*" />**

**<add name="Access-Control-Allow-Headers" value="Content-Type" />**

**<add name="Access-Control-Allow-Methods" value="GET, POST, PUT, DELETE, OPTIONS" />**

**</customHeaders>**

**</httpProtocol>**

**</system.webServer>**