Angular 7/8/9/10/11/12 HttpClient & Http Services

O serviço HttpClient é uma API muito útil em Angular para se comunicar com o servidor remoto. Para se trabalhar com estrutura de CRUD (Create, Read, Update e Delete) utilizamos esse serviço de modo significativo.

Recursos da API HttpClient

- Suporte Observable
- Teste de API sem complicações
- Solicitações e mecanismo de resposta
- Melhor tratamento de erros

HttpClient é um serviço injetável, que possue vários métodos poderosos para se comunicar com o servidor remoto. A API HttpClient pode enviar solicitações Http POST, GET, PUT e DELETE facilmente.

Métodos de serviço HttpClient desde Angular 7

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

Para construirmos nossa aplicação de cadastro – utilizando os métodos GET, PUT, POST e DELETE - criaremos um novo projeto do zero; esses métodos permitem que nossa aplicação front-end se comunique com um servidor REST API.

O que vamos fazer:

- configurar o HttpClientModule no aplicativo Angular
- fazer uma solicitação usando um servidor local com pacote NPM servidor JSON.
- fazer uma solicitação GET, POST, PUT & DELETE com Angular usando HttpClient API

Como configurar o Angular Project nas versões 7/8/9/10/11/12 Project usando Angular CLI

Vamos criar um sistema de gerenciamento de registros de funcionários com Angular – pode ser executado as partir das versões 7/8/9/10. Nossa aplicação irá trabalhar com uma API RESTful via serviço HttpClient.

Execute o seguinte comando no Angular CLI.

```
ng new angular-httpclient-json-server
```

Como visto anteriormente, durante a istalçao, serão feitas as seguintes perguntas:

Você gostaria de adicionar o roteamento angular? Selecione y e pressione Enter.

Qual formato de folha de estilo você gostaria de usar? (Use as setas) Escolha CSS e pressione Enter

Depois disso, seu projeto começará a ser criado. Assim que o projeto for criado, não se esqueça de entrar na pasta do projeto.

```
cd angular-httpclient-json-server
```

Vamos usar a estrutura CSS Bootstrap 4 com Angular para consumir a API RESTful com o serviço HttpClient. Insira o seguinte comando para obter o Bootstrap em seu aplicativo Angular.

```
npm install bootstrap
```

Depois disso, vá para o arquivo ***angular.json*** e substitua o código abaixo com “styles”: [] array.

```
"styles": [

"node_modules/bootstrap/dist/css/bootstrap.min.css",
            "src/styles.css"
          ]
```

Execute seu projeto inserindo o comando ng serve:

```
ng serve
```

Criando os componentes necessários para a aplicação

Com a inserção do comnado abaixo – via prompt de comando ou shell do VS Code – é possível criar os componentes necessários para nosso projeto. Vamos criar três: employee-create, employee-edit, employee-list. Observe os comando abaixo:

```
ng g c employee-create
ng g c employee-edit
ng g c employee-list
```

Configurando o servidor JSON no projeto

Vamos criar um ***mock server*** em nosso projeto para testar nosso aplicativo Angular, portanto, vamos ter a ajuda do pacote NPM json-server para resolver nosso problema.
Instale o servidor JSON em nosso projeto, execute o seguinte comando no Angular CLI.

```
npm i json-server --save
```

Em seguida, crie uma pasta com o nome do servidor e mantenha seu arquivo de banco de dados para gerenciar as APIs localmente.

```
mkdir server && cd server

server/db.json
```

Depois que o arquivo db.json for criado, adicione alguns dados a ele. Observe o código abaixo:

```
{
  "employees": [{
    "id": 1,
    "name": "Tonynho Stark",
    "email": "tonynho@aveng.com",
    "phone": 15-91660-5478"
  }, {
    "id": 2,
    "name": "Viuvinha Hulk",
    "email": "v_hulk@aveng.com",
    "phone": "11-98070-9090"
  }]
}
```

Depois disso, execute o seguinte comando para executar o servidor JSON.

```
json-server --watch server/db.json
```

Agora, se você fizer qualquer solicitação com a **postagem a partir da aplicação Angular – via Http - , put , get ou delete** seu arquivo **db.json** será atualizado localmente.

Você pode verificar seu arquivo **db.json** local neste endereço URL http://localhost:3000/employees. Será possível ver as alterações

Habilitar o serviço de roteamento

Para navegar entre componentes no Angular, precisamos ativar o serviço de roteamento (routing) em nosso aplicativo, para habilitar as rotas, vá para o arquivo ***app-routing.module.ts*** e adicione o código a seguir.

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { EmployeeCreateComponent } from './employee-create/employee-
create.component';
import { EmployeeEditComponent } from './employee-edit/employee-
edit.component';
import { EmployeesListComponent } from './employee-list/employees-
list.component';

const routes: Routes = [
  { path: '', pathMatch: 'full', redirectTo: 'create-employee' },
  { path: 'create-employee', component: EmployeeCreateComponent },
  { path: 'employees-list', component: EmployeesListComponent },
  { path: 'employee-edit/:id', component: EmployeeEditComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})

export class AppRoutingModule { }
```

Configurar a API HttpClient no aplicativo Angular para consumir a API RESTful

Para usar a API HttpClient para fazer a comunicação com o servidor remoto Http, você deve configurar este serviço em seu aplicativo Angular. Vamos observar as intruções abaixo:

Vá para ***app.module.ts*** e implemente o código a seguir.

```
import { HttpClientModule } from '@angular/common/http';
```

Inclua o ***HttpClientModule*** no array @NgModule de importações (imports).

```
@NgModule({
  imports: [
    HttpClientModule
   ]
})
```

Você injetou o **HttpClientModule** em seu aplicativo, agora você pode usá-lo facilmente em seu aplicativo Angular.

Criar um serviço Angular para consumir a API RESTful usando HttpClient

Para criar uma API RESTful de consumo usando o serviço Angular HttpClient, precisamos criar um arquivo de serviço em nosso aplicativo. Este arquivo conterá a lógica central de nosso aplicativo:

Funcionalidades do projeto:

- Create Employee
- Delete Employee
- Update Employee
- Manage Employee List

Para criar operações CRUD usando a API RESTful no Angular, precisamos gerar classes ***employee.ts*** e arquivos ***rest-api.service.ts***.

Gerar classe de interface employee (model)

```
ng g cl shared/Employee
```

Acesse ***shared/employee.ts*** e defina os tipos de dados na classe ***Employee***.

```
export class Employee {
   id: string;
   name: string;
   email: string;
   phone: number;
}
```

Gerar classe RestApiService

```
ng g s shared/rest-api
```

Este é o arquivo que usamos para consumir a API RESTful usando a API HttpClient. Também usaremos RxJS para lidar com operações assíncronas e erros neste aplicativo de demonstração.

Vamos ao arquivo ***shared/rest-api.service.ts*** e adicionar o seguinte código.

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Employee } from '../shared/employee';
import { Observable, throwError } from 'rxjs';
import { retry, catchError } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})

export class RestApiService {

  // Define API
  apiURL = 'http://localhost:3000';

  constructor(private http: HttpClient) { }

  /*========================================
    Métodos CRUD para consumir a RESTful API
  ========================================*/

  // Http Options
  httpOptions = {
    headers: new HttpHeaders({
      'Content-Type': 'application/json'
    })
  }

  // HttpClient API método get() => Busca a Lista employee
  getEmployees(): Observable<Employee> {
    return this.http.get<Employee>(this.apiURL + '/employees')
    .pipe(
      retry(1),
      catchError(this.handleError)
    )
  }

  // HttpClient API método get()=> busca employee por id
  getEmployee(id): Observable<Employee> {
    return this.http.get<Employee>(this.apiURL + '/employees/' + id)
    .pipe(
      retry(1),
      catchError(this.handleError)
    )
  }

  // HttpClient API método post()=> Cria employee
  createEmployee(employee): Observable<Employee> {
    return this.http.post<Employee>(this.apiURL + '/employees',
JSON.stringify(employee), this.httpOptions)
    .pipe(
      retry(1),
      catchError(this.handleError)
    )
```

```
  }

  // HttpClient API método put()=> Update (atualiza) employee
  updateEmployee(id, employee): Observable<Employee> {
    return this.http.put<Employee>(this.apiURL + '/employees/' + id,
JSON.stringify(employee), this.httpOptions)
    .pipe(
      retry(1),
      catchError(this.handleError)
    )
  }

  // HttpClient API método delete() => "Deleta" (exclui) employee
  deleteEmployee(id){
    return this.http.delete<Employee>(this.apiURL + '/employees/' +
id, this.httpOptions)
    .pipe(
      retry(1),
      catchError(this.handleError)
    )
  }

  // Error handling (Manipulador de erro)
  handleError(error) {
     let errorMessage = '';
     if(error.error instanceof ErrorEvent) {
       // Get client-side error
       errorMessage = error.error.message;
     } else {
       // Get server-side error (manipulador de erro do servidor)
       errorMessage = `Error Code: ${error.status}\nMessage:
${error.message}`;
     }
     window.alert(errorMessage);
     return throwError(errorMessage);
  }

}
```

Acesse a API HttpClient a partir do componente angular

Nós criamos com sucesso serviços RESTful usando Angular HttpClient API, é hora de acessar ***rest-api.service.ts*** através de componentes Angular.

Navegue para o seu arquivo ***app.module.ts*** e adicione o seguinte código, este arquivo contém os serviços principais para executar nosso aplicativo.

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

// HttpClient module para RESTful API
import { HttpClientModule } from '@angular/common/http';

// Routing module para router service
import { AppRoutingModule } from './app-routing.module';

// Forms module
import { FormsModule } from '@angular/forms';
```

```
// Components
import { EmployeeCreateComponent } from './employee-create/employee-
create.component';
import { EmployeeEditComponent } from './employee-edit/employee-
edit.component';
import { EmployeesListComponent } from './employees-list/employees-
list.component';

@NgModule({
  declarations: [
    AppComponent,
    EmployeeCreateComponent,
    EmployeeEditComponent,
    EmployeesListComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})

export class AppModule { }
```

Criar employee fazendo uma solicitação HTTP POST

Navegue até o arquivo ***employee-create.component.html*** e adicione o seguinte código.

```
<div class="container custom-container">
  <div class="col-md-12">
    <h3 class="mb-3 text-center">Create Employee</h3>

    <div class="form-group">
      <input type="text" [(ngModel)]="employeeDetails.name"
class="form-control" placeholder="Name">
    </div>

    <div class="form-group">
      <input type="text" [(ngModel)]="employeeDetails.email"
class="form-control" placeholder="Email">
    </div>

    <div class="form-group">
      <input type="text" [(ngModel)]="employeeDetails.phone"
class="form-control" placeholder="Phone">
    </div>

    <div class="form-group">
      <button class="btn btn-success btn-lg btn-block"
(click)="addEmployee()">Create Employee</button>
    </div>

  </div>
</div>
```

Navegue até o arquivo employee-create.component.ts e adicione o seguinte código.

```
import { Component, OnInit, Input } from '@angular/core';
import { Router } from '@angular/router';
import { RestApiService } from "../shared/rest-api.service";

@Component({
  selector: 'app-employee-create',
  templateUrl: './employee-create.component.html',
  styleUrls: ['./employee-create.component.css']
})
export class EmployeeCreateComponent implements OnInit {

  @Input() employeeDetails = { name: '', email: '', phone: 0 }

  constructor(
    public restApi: RestApiService,
    public router: Router
  ) { }

  ngOnInit() { }

  addEmployee() {
    this.restApi.createEmployee(this.employeeDetails).subscribe((data:
{}) => {
      this.router.navigate(['/employees-list'])
    })
  }

}
```

Ao adicionar o código fornecido acima no componente ***create-employee,*** podemos criar facilmente um funcionário fazendo uma solicitação HTTP POST por meio do componente Angular.

Enviar solicitações HTTP GET e DELETE para

Neste passo, iremos gerenciar a lista de funcionários que criamos – a partir dos código acima. Vamos usar o serviço API RESTful enviando as solitações ***get()*** e ***delete()*** via nossas APIs personalizadas.

employees-list.component.html

```
<div class="container custom-container-2">

  <!-- Show it when there is no employee -->
  <div class="no-data text-center" *ngIf="Employee.length == 0">
    <p>There is no employee added yet!</p>
    <button class="btn btn-outline-primary" routerLink="/create-
employee">Add Empoyee</button>
  </div>

  <!-- Employees list table, fica oculta enquanto no existir employee
-->
  <div *ngIf="Employee.length !== 0">
    <h3 class="mb-3 text-center">Employees List</h3>

    <div class="col-md-12">
```

```
      <table class="table table-bordered">
        <thead>
          <tr>
            <th scope="col">User Id</th>
            <th scope="col">Name</th>
            <th scope="col">Email</th>
            <th scope="col">Phone</th>
            <th scope="col">Action</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let employee of Employee">
            <td>{{employee.id}}</td>
            <td>{{employee.name}}</td>
            <td>{{employee.email}}</td>
            <td>{{employee.phone}}</td>
            <td>
              <span class="edit" routerLink="/employee-
edit/{{employee.id}}">Edit</span>
              <span class="delete"
(click)="deleteEmployee(employee.id)">Delete</span>
            </td>
          </tr>
        </tbody>
      </table>
    </div>

  </div>

</div>
```

employees-list.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RestApiService } from "../shared/rest-api.service";

@Component({
  selector: 'app-employees-list',
  templateUrl: './employees-list.component.html',
  styleUrls: ['./employees-list.component.css']
})
export class EmployeesListComponent implements OnInit {

  Employee: any = [];

  constructor(
    public restApi: RestApiService
  ) { }

  ngOnInit() {
    this.loadEmployees()
  }

  // Get employees list
  loadEmployees() {
    return this.restApi.getEmployees().subscribe((data: {}) => {
      this.Employee = data;
    })
  }
```

```
  // Delete employee
  deleteEmployee(id) {
    if (window.confirm('Are you sure, you want to delete?')){
      this.restApi.deleteEmployee(id).subscribe(data => {
        this.loadEmployees()
      })
    }
  }

}
```

Fazer solicitação HTTP PUT para atualizar os dados

Agora, vamos enviar uma solicitação HTTP PUT para atualizar os dados dos employees em nosso aplicativo. É muito simples, basta seguir as etapas a seguir.

employee-edit.component.html

```
<div class="container custom-container">
  <div class="col-md-12">

    <h3 class="mb-3 text-center">Update Employee</h3>

    <div class="form-group">
      <input type="text" [(ngModel)]="employeeData.name" class="form-
control" placeholder="Name">
    </div>

    <div class="form-group">
      <input type="text" [(ngModel)]="employeeData.email" class="form-
control" placeholder="Email">
    </div>

    <div class="form-group">
      <input type="text" [(ngModel)]="employeeData.phone" class="form-
control" placeholder="Phone">
    </div>

    <div class="form-group">
      <button class="btn btn-success btn-lg btn-block"
(click)="updateEmployee()">Update Employee</button>
    </div>

  </div>
</div>
```

employee-edit.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RestApiService } from "../shared/rest-api.service";
import { ActivatedRoute, Router } from '@angular/router';

@Component({
  selector: 'app-employee-details',
  templateUrl: './employee-edit.component.html',
  styleUrls: ['./employee-edit.component.css']
})
```

```
export class EmployeeEditComponent implements OnInit {
  id = this.actRoute.snapshot.params['id'];
  employeeData: any = {};

  constructor(
    public restApi: RestApiService,
    public actRoute: ActivatedRoute,
    public router: Router
  ) {
  }

  ngOnInit() {
    this.restApi.getEmployee(this.id).subscribe((data: {}) => {
      this.employeeData = data;
    })
  }

  // Update employee data
  updateEmployee() {
    if(window.confirm('Are you sure, you want to update?')){
      this.restApi.updateEmployee(this.id,
this.employeeData).subscribe(data => {
        this.router.navigate(['/employees-list'])
      })
    }
  }

}
```

Para finalizar nosso projeto, o arquivo ***app.component.html*** deve ficar da seguinte forma – observe o código abaixo:

```
<div class="d-flex flex-column flex-md-row align-items-center p-3 px-
md-4 mb-3 bg-white border-bottom shadow-sm">
  <h5 class="my-0 mr-md-auto font-weight-
normal">Employee Management</h5>
  <nav class="my-2 my-md-0 mr-md-3">
    <a class="btn btn-outline-
primary" routerLinkActive="active" routerLink="/create-
employee">Create Employee</a>
  </nav>
  <a class="btn btn-outline-
primary" routerLinkActive="active" routerLink="/employees-
list">Employees List</a>
</div>

<router-outlet></router-outlet>
```

Na sequencia, finalize a implementação com a implementação do arquivo styles.css (css global do seu projeto); insira o código abaixo:

```
.custom-container {
    max-width: 500px;
  }

  .custom-container-2 {
    max-width: 800px;
  }

  .table-bordered td {
    font-size: 14px;
  }

  .edit {
    margin-right: 10px;
  }

  .edit,
  .delete {
    cursor: pointer;
    display: inline-block;
    color: blue
  }

  .no-data {
    padding-top: 50px;
  }

  .no-data p {
    font-size: 27px;
    color: rgba(0, 0, 0, .5);
    letter-spacing: .2px;
    padding-bottom: 10px;
  }
```

Agora você pode testar seu aplicativo Angular HttpClient no navegador, basta digitar ***ng serve*** no terminal.

Referencia:
Traduzido e adaptado do original que pode ser obtido através do link abaixo:
https://www.positronx.io/angular-7-httpclient-http-service/