Tutorial Angular 6

Mateo Agudelo Toro

Septiembre 2018

1 Introducción

Angular es un framework para front end de código abierto muy popular. Utiliza principalmente TypeScript (JavaScript con tipos de datos y más) para crear aplicaciones web tanto para móvil como para desktop. Este es un tutorial simple basado en este tutorial en inglés.
En nuestro caso crearemos una aplicación que consume un API REST que nos permite tanto hacer reviews sobre servicios prestados por automóviles, como ver todas las reviews.

2 Prerequisitos

Angular propiamente se instala como un paquete de Node, por lo que necesitaremos tener instalado tanto Node como su manejador de paquetes (NPM, Node Package Manager).
Sin entrar en detalles sobre este aspecto (pues son muy variados y dependen de muchos factores), se recomienda instalar la versión LTS (actualmente la más reciente de este tipo es la 8.12.0, pero versiones como la 8.11.4 también están bien, pues la versión 8 es la LTS). Esta deberá incluir NPM en versión 5.6.0 o similar.
Para verificar la instalación de Node.js podemos correr en la terminal es siguiente comando:

```
$ node -v
```

El cual, de ser correcta la instalación, deberá responder con algo como

```
8.11.4
```

Del mismo, modo, para verificar la instalación de NPM:

```
$ npm -v
```

Y deberá responder con algo como

```
5.6.0
```

3 Instalación

Ya que tenemos NPM, para instalar Angular podemos simplemente ejecutar

```
$ npm install -g @angular/cli
```

Lo que realmente estamos haciendo es instalar la interfaz de linea de comandos de Angular (Angular Command Line Interface o CLI) de manera global (contrario a para algún proyecto específico de Node).

4 Creación de Proyecto

Para esto utilizaremos el CLI de Angular que hará gran parte del trabajo repetitivo por nosotros. Para crear el proyecto de este ejemplo utilizaremos el siguiente comando:

```
$ ng new ejempl-ng6 --routing
```

Lo que hacemos es, mediante el CLI (llamado con *ng*) crear un nuevo proyecto (especificado mediante el argumento *new*) llamado *ejemplo-ng6* con un parámetro adicional (*–routing*) que nos ayudará con el enrutamiento de nuestra aplicación.
Una vez este comando finalice, se habrá creado el proyecto base en la carpeta *ejmplo-ng6*, sobre la cual construiremos nuestra aplicación.

5 API Service

Aquí definiremos la forma de conectarnos al API REST que tenemos. Primero inicializamos el servicio utilizando el CLI y luego lo modificamos a nuestra necesidad:

```
$ ng generate service api
```

api.service.ts:

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

@Injectable({
  providedIn: 'root'
})
export class ApiService {
  readonly headers: HttpHeaders = new HttpHeaders().set('ZUMO-API-VERSION', '2.0.0');
  readonly endpoint = 'https://reportrans.azurewebsites.net/tables/report';

  constructor(private httpClient: HttpClient) { }

  getPosts() {
    return this.httpClient.get(this.endpoint, { headers: this.headers });
  }

  addPost(post: Object) {
    this.httpClient.post(this.endpoint, post, { headers: this.headers })
      .subscribe(
        res => console.log(res),
        err => console.log(`Couldn't add post: ${JSON.stringify(err)}`)
      );
  }
}
```

6 Componentes

Angular crea lo que llaman *aplicaciones de página única* o *SPAs, Single Page Applications*, por lo que la forma en la que se crea la interactividad en la aplicación es mostrando en pantalla únicamente los componentes adecuados con base en el input del usuario.
Existen por lo menos dos formas de generar componentes utilizando el CLI. La forma larga:

```
$ ng generate component <nombre-del-componente>
```

Y la forma abreviada:

```
$ ng g c <nombre-del-componente>
```

Los componentes en Angular se conforman a su vez de 4 partes:

- **Controlador**: es el archivo que maneja la lógica "del negocio". Por defecto se llama *nombre-del-componente*.component.ts.
- **Vista**: es el archivo que estructura lo que el usuario ve. Por defecto se llama *nombre-del-componente*.component.html.
- **Estilos**: es el archivo que hace que lo que ve el usuario sea bonito. Por defecto se llama *nombre-del-componente*.component.css. No nos enfocaremos en este aspecto.
- **Pruebas**: es el archivo que comprueba la funcionalidad del componente. Por defecto se llama *nombre-del-componente*.component.spec.ts.

Para nuestro ejemplo crearemos 4 componentes:

- **Add**: componente para agregar una review.
- **Home**: componente para darle la bienvenida al usuario. No lo extenderemos mucho.
- **Posts**: componente para ver rodas las reviews.
- **Sidebar**: barra de navegación lateral que le permitirá al usuario utilizar las diferentes funcionalidades de la aplicación. En nuestro caso probablemente no sea lateral (por defecto se sitúan en la parte superior de la página) ya que para esto necesitaríamos utilizar CSS y no queremos enforcarnos en este aspecto.

Para esto los inicializamos así:

```
$ ng g c add
$ ng g c home
$ ng g c posts
$ ng g c sidebar
```

6.1 Integración

Para integrar los componentes y crear la noción de interactividad en el usuario tenemos que trabajar directamente con el componente raíz: `app.component.html`.
Le agregaremos la barra de navegación y dejaremos que el contenido sea definido dinámicamente por el enrutador (controlado por el usuario mediante la barra de navegación):

```
<div id="container">
  <app-sidebar></app-sidebar>

  <div id="content">
    <router-outlet></router-outlet>
  </div>
</div>
```

6.2 Estilos básicos

Si bien Angular es un framework para el front end, el objetivo de este tutorial es más sobre el consumo de un API que sobre la estética del producto. Es por esto que simplemente utilizaremos Bootstrap (y algo de CSS luego, pero muy poco).
Para esto debemos interactuar directamente con la única página que realmente tenemos (`index.html`) y agregar Bootstrap al `head`:

```
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
  integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
    crossorigin="anonymous"></script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
```

```
  integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
    crossorigin="anonymous"></script>

  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
  integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
    crossorigin="anonymous">

  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
  integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
    crossorigin="anonymous"></script>
```

6.3 Add

Tal y como habíamos mencionado, este componente permite agregar reviews, por lo que es simplemente un formulario que envía los valores recolectados al servidor mediante el API REST que se nos provee.

add.component.ts:

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

@Component({
  selector: 'app-add',
  templateUrl: './add.component.html',
  styleUrls: ['./add.component.css']
})
export class AddComponent {
  post: any = {};

  constructor(private apiService: ApiService) { }

  onSubmit() {
    this.apiService.addPost(this.post);
  }

}
```

add.component.html:

```
<h1>Add</h1>

<ul>
  <li>
    <form name="form" (ngSubmit)="f.form.valid && onSubmit()" #f="ngForm" novalidate>
      <div class="form-group">
        <label for="carPlate">Car Plate</label>
        <input type="text" class="form-control" name="carPlate"
        [(ngModel)]="post.carPlate" #firstName="ngModel" [ngClass]="false"
          required />
      </div>
      <div class="form-group">
        <label for="type">Car Type</label>
        <input type="text" class="form-control" name="type"
        [(ngModel)]="post.type" #firstName="ngModel" [ngClass]="false"
          required />
      </div>
      <div class="form-group">
```

```
        <label for="score">Score</label>
        <input type="text" class="form-control" name="score"
        [(ngModel)]="post.score" #firstName="ngModel" [ngClass]="false"
          required />
      </div>
      <div class="form-group">
        <label for="comment">Comment</label>
        <input type="text" class="form-control" name="comment"
        [(ngModel)]="post.comment" #firstName="ngModel" [ngClass]="false"
          required />
      </div>
      <div class="form-group">
        <button class="btn btn-primary">Register</button>
      </div>
    </form>
  </li>
</ul>
```

6.4 Home

En este componente le daríamos la bienvenida al usuario y podríamos poner links y demás, pero en nuestro caso no lo implementaremos.

6.5 Posts

Este es similar al de agregar un review pero lo que hace es pedirlas todas.

posts.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { ApiService } from '../api.service';
import { Observable } from 'rxjs';

@Component({
  selector: 'app-posts',
  templateUrl: './posts.component.html',
  styleUrls: ['./posts.component.css']
})
export class PostsComponent implements OnInit {
  posts$: Object;

  constructor(private apiService: ApiService) { }

  ngOnInit() {
    this.apiService.getPosts().subscribe(data => this.posts$ = data);
  }

}
```

posts.component.html:

```
<h1>Posts</h1>

<ul>
  <li *ngFor="let post of posts$">
    {{ post.id }}
```

```
    <ul>
      <li>Comment: {{ post.comment }}</li>
      <li>Plate: {{ post.carPlate }}</li>
      <li>Type: {{ post.type }}</li>
      <li>Score: {{ post.score }}</li>
      <li>Created: {{ post.createdAt }}</li>
      <li>Update: {{ post.updatedAt }}</li>
    </ul>
  </li>
</ul>
```

6.6 Sidebar

Esta barra permite al usuario entre la dos funcionalidades que hemos implmentado hasta ahora. Los estilos (aunque incompletos) fueron tomados de la página original del tutorial, la vista son dos enlaces a Add y Posts, y la lógica indica qué funcionalidad está seleccionada para mostrarlo al usuario más facilmente.

sidebar.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { Router, NavigationEnd } from '@angular/router';

@Component({
  selector: 'app-sidebar',
  templateUrl: './sidebar.component.html',
  styleUrls: ['./sidebar.component.css']
})
export class SidebarComponent implements OnInit {
  currentUrl: string;

  constructor(private router: Router) {
    router.events.subscribe((_: NavigationEnd) => {
      if (_.url !== undefined)
        this.currentUrl = _.url
    });
  }

  ngOnInit() {
  }

}
```

sidebar.component.html:

```
<nav>
  <ul>
    <li>
      <a routerLink="add" [class.activated]="currentUrl == '/add'">
        <i class="material-icons">comment</i>
      </a>
    </li>
    <li>
      <a routerLink="posts" [class.activated]="currentUrl == '/posts'">
        <i class="material-icons">book</i>
      </a>
    </li>
```

```
  </ul>
</nav>
```

sidebar.component.css (adaptado utilizando SassMeister):

```
nav {
  background: #2D2E2E;
  height: 100%;
}
nav ul {
  list-style-type: none;
  padding: 0;
  margin: 0;
}
nav ul li a {
  color: #FFF;
  padding: 20px;
  display: block;
}
nav ul li .activated {
  background-color: #00A8FF;
}
```

7 Enrutamiento

Aquí simplemente asocionamos los componentes a las rutas que les queremos asignar.

app-routing.module.ts:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AddComponent } from './add/add.component';
import { HomeComponent } from './home/home.component';
import { PostsComponent } from './posts/posts.component';

const routes: Routes = [
  {
    path: 'add',
    component: AddComponent
  },
  {
    path: '',
    component: HomeComponent
  },
  {
    path: 'posts',
    component: PostsComponent
  }
];

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

8 Integración final

Esta parte realmente se va haciendo a medida que se hacen cambios; en nuestro caso la dejamos para el final:

app.module.ts:

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

import { AppRoutingModule } from './app-routing.module';
import { HttpClientModule } from '@angular/common/http';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
import { SidebarComponent } from './sidebar/sidebar.component';
import { PostsComponent } from './posts/posts.component';
import { AddComponent } from './add/add.component';
import { HomeComponent } from './home/home.component';

@NgModule({
  declarations: [
    AppComponent,
    SidebarComponent,
    PostsComponent,
    AddComponent,
    HomeComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

9 Correr la aplicación

Para correr la aplicación en modo desarrollador (la página se recarga automáticamente cuando hay cambios en el código) podemos usar el siguiente comando:

```
$ ng serve
```

Para usarla, visiten http://localhost:4200/ (por defecto).