Contents

Angular 10 CRUD Application Tutorial By `Techiediaries.com`

Throughout this tutorial, We'll be learning how to build an Angular 10 CRUD application to consume a REST API, create, read, modify and search data.

You can also read this tutorial online from:

`https://www.techiediaries.com/angular-10-crud-example-web-api/`

Introducing our Angular 10 CRUD Application

We will learn how to build an Angular 10 front-end application that fetches data from a REST API of products:

- Each product has id, name, description, availability status.
- Users would be able to create, retrieve, update, and delete products.
- Users can search for products by name.

The REST API Endpoints

We'll be building a Angular 10 frontend app for a presumed REST API exporting the following REST API endpoints:

- POST /api/products create new product

- GET /api/products retrieve all products
- GET /api/products/:id retrieve a product by `:id`
- PUT /api/products/:id update a product by `:id`
- DELETE /api/products/:id delete a product by `:id`
- DELETE /api/products delete all products
- GET /api/products?name=[keyword] find all products which name contains the passed `keyword`.

All of them can work well with this Angular App.

Angular 10 CRUD App Structure

These are the components of our CRUD app:

- The `App` component is the parent of all other components and contains a `router-outlet` directive where the router will be inserting any matched component. It also contains a navigation bar that contains links to the app routes using `routerLink` directive.

– `ProductListComponent` which displays the list of products.
– `ProductUpdateComponent` which displays a form for editing product's details by `:id`.
– `ProductCreateComponent` which displays a form for creating a new product.

The components use the `ProductService` methods for actually making CRUD operations against the REST API. The service makes use of Angular 10 `HTTPClient` to send HTTP requests to the REST and process responses.

Step 1 — Creating a New Angular 10 Project

Let's get started by generating a new Angular 10 project using the CLI. You need to run the following command:

```
$ ng new Angular10CRUDExample
```

The CLI will ask you a couple of questions — If **Would you like to add Angular routing?** Type **y** for Yes and **Which stylesheet format would you like to use?** Choose **CSS**.

Step 2 — Generating Angular 10 CRUD Components and Service

Next, we need to generate a bunch of components and a service using the Angular CLI as follows:

```
$ ng generate service services/product
$ ng g c components/product-create
$ ng g c components/product-details
$ ng g c components/product-list
```

We have generated three components `product-list`, `product-details`, `product-create` and a product service that provides the necessary methods for sending HTTP requests to the server.

We also have the following artifacts:

– The `src/app/app-routing.module.ts` module will contain routes for each component. This file is automatically generated by Angular CLI when you answered Yes for routing.
– The `App` component contains the router view and navigation bar.
– The `src/app/app.module.ts` module declares our Angular components and import the necessary modules such Angular `HttpClient`.

Step 3 — Importing `FormsModule` and `HttpClientModule`

We'll be using the http client and forms in our CRUD application which are provided in its own modules in Angular so we'll need to import these modules — `FormsModule` and `HttpClientModule`.

Open `src/app/app.module.ts` file and import `FormsModule`, `HttpClientModule` as follows:

```
// [...]
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [ ... ],
  imports: [
    ...
    FormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 4 — Adding Routes for the CRUD Components

We have the components that compose our application UI but we need to link them with their routes to be able to navigate between them using the Angular 10 Router.

We'll create three routes:

– `/products` for the `product-list` component,
– `/products/:id` for the `product-details` component,
– `/create` for the`product-create` component.

Open the `src/app/app-routing.module.ts` file and update it as follows:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { ProductListComponent } from './components/product-list/product-list.component';
import { ProductDetailsComponent } from './components/product-details/product-details.compon
import { ProductCreateComponent } from './components/product-create/product-create.component

const routes: Routes = [
  { path: '', redirectTo: 'products', pathMatch: 'full' },
  { path: 'products', component: ProductListComponent },
  { path: 'products/:id', component: ProductDetailsComponent },
  { path: 'create', component: ProductCreateComponent }
];

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

Step 5 — Adding and Styling a Navigation Bar Using Bootstrap 4

Next, let's add Bootstrap 4 to our application and a navigation bar.

We'll be using Bootstrap 4 for styling the UI so you'll need to install it in your project. Check out three ways for how to add bootstrap to your Angular project.

Open the `src/app/app.component.html` file, and update as follows:

```
<div>
  <nav class="navbar navbar-expand navbar-dark bg-dark">
    <a href="#" class="navbar-brand">Techiediaries</a>
    <div class="navbar-nav mr-auto">
```

```
      <li class="nav-item">
        <a routerLink="products" class="nav-link">Products</a>
      </li>
      <li class="nav-item">
        <a routerLink="create" class="nav-link">Create</a>
      </li>
    </div>
  </nav>

  <div class="container mt-5">
    <router-outlet></router-outlet>
  </div>
</div>
```

We have created a bootstrap navigation bar and wrapped the router outlet with a container div.

Step 6 — Creating an Angular 10 CRUD Service

Next, we need to create a CRUD service that will use Angular 10 `HTTPClient` to send HTTP requests to the REST API server.

Open the `src/services/product.service.ts` file and update it as follows:

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

const baseURL = 'http://localhost:8080/api/products';

@Injectable({
  providedIn: 'root'
})
export class ProductService {

  constructor(private httpClient: HttpClient) { }

  readAll(): Observable<any> {
    return this.httpClient.get(baseURL);
  }

  read(id): Observable<any> {
    return this.httpClient.get(`${baseURL}/${id}`);
  }

  create(data): Observable<any> {
```

```
    return this.httpClient.post(baseURL, data);
  }

  update(id, data): Observable<any> {
    return this.httpClient.put(`${baseURL}/${id}`, data);
  }

  delete(id): Observable<any> {
    return this.httpClient.delete(`${baseURL}/${id}`);
  }

  deleteAll(): Observable<any> {
    return this.httpClient.delete(baseURL);
  }

  searchByName(name): Observable<any> {
    return this.httpClient.get(`${baseURL}?name=${name}`);
  }
}
```

Step 7 — Implementing the Angular 10 CRUD Components

We have previously generated three components and added them to the router, let's now implement the actual functionality of each component.

Creating a New Product Component

This component provides a form for submitting a new product with two fields, `name` and `description`. It injects and calls the `ProductService.create()` method.

Open the `src/components/product-create/product-create.component.ts` file and update it as follows:

```
import { Component, OnInit } from '@angular/core';
import { ProductService } from 'src/app/services/product.service';

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

```
    name: '',
    description: '',
    available: false
  };
  submitted = false;

  constructor(private productService: ProductService) { }

  ngOnInit(): void {
  }

  createProduct(): void {
    const data = {
      name: this.product.name,
      description: this.product.description
    };

    this.productService.create(data)
      .subscribe(
        response => {
          console.log(response);
          this.submitted = true;
        },
        error => {
          console.log(error);
        });
  }

  newProduct(): void {
    this.submitted = false;
    this.product = {
      name: '',
      description: '',
      available: false
    };
  }

}
```

Next, open the `src/components/product-create/product-create.component.html` file and update it as follows:

```
<div style="width: 500px; margin: auto;">
  <div class="submit-form">
    <div *ngIf="!submitted">
      <div class="form-group">
        <label for="name">Name</label>
```

```
        <input
          type="text"
          class="form-control"
          id="name"
          required
          [(ngModel)]="product.name"
          name="name"
        />
      </div>

      <div class="form-group">
        <label for="description">Description</label>
        <input
          class="form-control"
          id="description"
          required
          [(ngModel)]="product.description"
          name="description"
        />
      </div>

      <button (click)="createProduct()" class="btn btn-success">Create</button>
    </div>

    <div *ngIf="submitted">
      <h3>You successfully created a product!</h3>
      <button class="btn btn-success" (click)="newProduct()">New..</button>
    </div>
  </div>
</div>
```

Displaying the List of Products Component

Next, let's implement the product list component, which makes use of the following service methods:

- `readAll()`
- `deleteAll()`
- `searchByName()`

Open the `src/components/product-list/product-list.component.ts` file and update it as follows:

```
import { Component, OnInit } from '@angular/core';
import { ProductService } from 'src/app/services/product.service';
```

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

  products: any;
  currentProduct = null;
  currentIndex = -1;
  name = '';

  constructor(private productService: ProductService) { }

  ngOnInit(): void {
    this.readProducts();
  }

  readProducts(): void {
    this.productService.readAll()
      .subscribe(
        products => {
          this.products = products;
          console.log(products);
        },
        error => {
          console.log(error);
        });
  }

  refresh(): void {
    this.readProducts();
    this.currentProduct = null;
    this.currentIndex = -1;
  }

  setCurrentProduct(product, index): void {
    this.currentProduct = product;
    this.currentIndex = index;
  }

  deleteAllProducts(): void {
    this.productService.deleteAll()
      .subscribe(
        response => {
          console.log(response);
```

```
          this.readProducts();
        },
        error => {
          console.log(error);
        });
  }

  searchByName(): void {
    this.productService.searchByName(this.name)
      .subscribe(
        products => {
          this.products = products;
          console.log(products);
        },
        error => {
          console.log(error);
        });
  }
}
```

Open the `src/components/product-list/product-list.component.html` file and update it as follows:

```
<div class="list row">
  <div class="col-md-8">
    <div class="input-group mb-4">
      <input
        type="text"
        class="form-control"
        placeholder="Search by name"
        [(ngModel)]="name"
      />
      <div class="input-group-append">
        <button
          class="btn btn-outline-secondary"
          type="button"
          (click)="searchByName()"
        >
          Search
        </button>
      </div>
    </div>
  </div>
  <div class="col-md-6">
    <h4>Product List</h4>
    <ul class="list-group">
      <li
```

```
        class="list-group-item"
        *ngFor="let product of products; let i = index"
        [class.active]="i == currentIndex"
        (click)="setCurrentProduct(product, i)"
      >
        {{ product.name }}
      </li>
    </ul>

    <button class="m-4 btn btn-sm btn-danger" (click)="deleteAllProducts()">
      Delete All
    </button>
  </div>
  <div class="col-md-6">
    <div *ngIf="currentProduct">
      <h4>Product</h4>
      <div>
        <label><strong>Name:</strong></label> {{ currentProduct.name }}
      </div>
      <div>
        <label><strong>Description:</strong></label>
        {{ currentProduct.description }}
      </div>
      <div>
        <label><strong>Status:</strong></label>
        {{ currentProduct.available ? "Available" : "Not available" }}
      </div>

      <a class="badge badge-warning" routerLink="/products/{{ currentProduct.id }}">
        Edit
      </a>
    </div>

    <div *ngIf="!currentProduct">
      <br />
      <p>Please click on a product</p>
    </div>
  </div>
</div>
```

If you click on **Edit** button of any product, you will be directed to the product details page with the `/products/:id` URL.

The Product Details Component

Next, let's implement the product details component of our Angular 10 CRUD application.

Open the `src/components/product-details/product-details.component.ts` file and update it as follows:

```
import { Component, OnInit } from '@angular/core';
import { ProductService } from 'src/app/services/product.service';
import { ActivatedRoute, Router } from '@angular/router';

@Component({
  selector: 'app-product-details',
  templateUrl: './product-details.component.html',
  styleUrls: ['./product-details.component.css']
})
export class ProductDetailsComponent implements OnInit {
  currentproduct = null;
  message = '';

  constructor(
    private productService: ProductService,
    private route: ActivatedRoute,
    private router: Router) { }

  ngOnInit(): void {
    this.message = '';
    this.getProduct(this.route.snapshot.paramMap.get('id'));
  }

  getProduct(id): void {
    this.productService.read(id)
      .subscribe(
        product => {
          this.currentProduct = product;
          console.log(product);
        },
        error => {
          console.log(error);
        });
  }

  setAvailableStatus(status): void {
    const data = {
      name: this.currentProduct.name,
```

```
      description: this.currentProduct.description,
      available: status
    };

    this.productService.update(this.currentProduct.id, data)
      .subscribe(
        response => {
          this.currentProduct.available = status;
          console.log(response);
        },
        error => {
          console.log(error);
        });
  }

  updateProduct(): void {
    this.productService.update(this.currentProduct.id, this.currentProduct)
      .subscribe(
        response => {
          console.log(response);
          this.message = 'The product was updated!';
        },
        error => {
          console.log(error);
        });
  }

  deleteProduct(): void {
    this.productService.delete(this.currentProduct.id)
      .subscribe(
        response => {
          console.log(response);
          this.router.navigate(['/products']);
        },
        error => {
          console.log(error);
        });
  }
}
```

Open the `src/components/product-details/product-details.component.html` file and update it as follows:

```
<div style="width: 500px; margin: auto;">
  <div *ngIf="currentProduct" class="edit-form">
    <h4>Product</h4>
    <form>
```

```
  <div class="form-group">
    <label for="title">Name</label>
    <input
      type="text"
      class="form-control"
      id="name"
      [(ngModel)]="currentProduct.name"
      name="name"
    />
  </div>
  <div class="form-group">
    <label for="description">Description</label>
    <input
      type="text"
      class="form-control"
      id="description"
      [(ngModel)]="currentProduct.description"
      name="description"
    />
  </div>

  <div class="form-group">
    <label><strong>Status:</strong></label>
    {{ currentProduct.available ? "Available" : "Not available" }}
  </div>
</form>

<button
  class="badge badge-primary mr-2"
  *ngIf="currentProduct.available"
  (click)="setAvailableStatus(false)"
>
  Set Not Available
</button>
<button
  *ngIf="!currentProduct.available"
  class="badge badge-primary mr-2"
  (click)="setAvailableStatus(true)"
>
  Set Available
</button>

<button class="badge badge-danger mr-2" (click)="deleteProduct()">
  Delete
</button>
```

```
    <button
      type="submit"
      class="badge badge-success"
      (click)="updateProduct()"
    >
      Update
    </button>
    <p>{{ message }}</p>
  </div>

  <div *ngIf="!currentProduct">
    <br />
    <p>This product is not accessible</p>
  </div>
</div>
```

Step 8 — Serving the Angular 10 CRUD App

Head back to your command-line interface, make sure your are navigated at the root of your project's directory and run the following command to start a live development server in your local machine:

```
$ cd Angular10CRUDExample
$ ng serve
```

Go to your web browser and navigate to the `http://localhost:4200/` address to start testing your app.

Conclusion

In this tutorial, we've built an Angular 10 CRUD application for a Web REST API. Make sure to visit us at **Techiediaries** for more indepth tutorials about Angular and modern web development.

By Techiediaries.com