**Angular 12 CRUD App Component Diagram**

– The App component is a container with router-outlet. It has navbar that links to routes paths via routerLink.

– TutorialsList component gets and displays Tutorials.  
– TutorialDetails component has form for editing Tutorial’s details based on :id.  
– AddTutorial component has form for submission new Tutorial.

– These Components call TutorialService methods which use Angular HTTPClient to make HTTP requests and receive responses.

**Setup Angular 12 Project**

Let’s open cmd and use Angular CLI to create a new Angular Project as following command:

ng new Angular12Crud

? Would you like to add Angular routing? Yes

? Which stylesheet format would you like to use? CSS

We also need to generate some Components and Services:

ng g s services/tutorial

ng g c components/add-tutorial

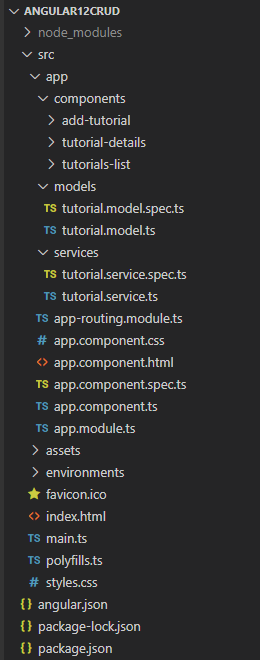
ng g c components/tutorial-details

ng g c components/tutorials-list

ng g class models/tutorial --type=model

Now you can see that our project directory structure looks like this.

**Project Structure**



Let me explain it briefly.

– tutorial.model.ts exports the main class model: Tutorial.  
– There are 3 components: tutorials-list, tutorial-details, add-tutorial.  
– tutorial.service has methods for sending HTTP requests to the [Apis](https://www.bezkoder.com/angular-12-crud-app/?__cf_chl_tk=PeLnOBXiAPyOBVzt_zswP4JetGRoZ4Ran2_70ZBTOZA-1723337049-0.0.1.1-3540).  
– app-routing.module.ts defines routes for each component.  
– app component contains router view and navigation bar.  
– app.module.ts declares Angular components and import necessary modules.

**Set up App Module**

Open [*app*](https://www.bezkoder.com/angular-12-crud-app/?__cf_chl_tk=PeLnOBXiAPyOBVzt_zswP4JetGRoZ4Ran2_70ZBTOZA-1723337049-0.0.1.1-3540)*.module.ts* and import FormsModule, HttpClientModule:

...

import { FormsModule } from '@angular/forms';

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

@NgModule({

declarations: [ ... ],

imports: [

...

FormsModule,

HttpClientModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**Define Routes for Angular 12 AppRoutingModule**

There are 3 main routes:  
– /tutorials for tutorials-list component  
– /tutorials/:id for tutorial-details component  
– /add for add-tutorial component

*app-routing.module.ts*

import { NgModule } from '@angular/core';

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

import { TutorialsListComponent } from './components/tutorials-list/tutorials-list.component';

import { TutorialDetailsComponent } from './components/tutorial-details/tutorial-details.component';

import { AddTutorialComponent } from './components/add-tutorial/add-tutorial.component';

const routes: Routes = [

{ path: '', redirectTo: 'tutorials', pathMatch: 'full' },

{ path: 'tutorials', component: TutorialsListComponent },

{ path: 'tutorials/:id', component: TutorialDetailsComponent },

{ path: 'add', component: AddTutorialComponent }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

**Import Bootstrap into Angular 12 Project**

Open *index.html*, we only need to add one line of code for Bootstrap 4.

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Angular12Crud</title>

...

<link type="text/css" rel="stylesheet" href="//unpkg.com/bootstrap@4.6.0/dist/css/bootstrap.min.css" />

</head>

<body>

<app-root></app-root>

</body>

</html>

**Add Navbar and Router View to Angular 12 CRUD App**

Let’s open **src**/*app.component.html*, this App component is the root container for our

[application](https://www.bezkoder.com/angular-12-crud-app/?__cf_chl_tk=PeLnOBXiAPyOBVzt_zswP4JetGRoZ4Ran2_70ZBTOZA-1723337049-0.0.1.1-3540), it will contain a nav element.

<div>

<nav class="navbar navbar-expand navbar-dark bg-dark">

<a href="#" class="navbar-brand">bezKoder</a>

<div class="navbar-nav mr-auto">

<li class="nav-item">

<a routerLink="tutorials" class="nav-link">Tutorials</a>

</li>

<li class="nav-item">

<a routerLink="add" class="nav-link">Add</a>

</li>

</div>

</nav>

<div class="container mt-3">

<router-outlet></router-outlet>

</div>

</div>

## Define Model Class

Our main model class Tutorial will be exported in tutorial.model.ts with 4 fields:

* id
* title
* description
* published

**models**/tutorial.model.ts

export class Tutorial {

id?: any;

title?: string;

description?: string;

published?: boolean;

}

## Create Data Service

This service will use [Angular](https://www.bezkoder.com/angular-12-crud-app/?__cf_chl_tk=PeLnOBXiAPyOBVzt_zswP4JetGRoZ4Ran2_70ZBTOZA-1723337049-0.0.1.1-3540) HTTPClient to send HTTP requests.  
You can see that its functions includes CRUD operations and finder method.

**services**/tutorial.service.ts

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

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

import { Tutorial } from '../models/tutorial.model';

const baseUrl = 'http://localhost:8080/api/tutorials';

@Injectable({

providedIn: 'root'

})

export class TutorialService {

constructor(private http: HttpClient) { }

getAll(): Observable<Tutorial[]> {

return this.http.get<Tutorial[]>(baseUrl);

}

get(id: any): Observable<Tutorial> {

return this.http.get(`${baseUrl}/${id}`);

}

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

return this.http.post(baseUrl, data);

}

update(id: any, data: any): Observable<any> {

return this.http.put(`${baseUrl}/${id}`, data);

}

delete(id: any): Observable<any> {

return this.http.delete(`${baseUrl}/${id}`);

}

deleteAll(): Observable<any> {

return this.http.delete(baseUrl);

}

findByTitle(title: any): Observable<Tutorial[]> {

return this.http.get<Tutorial[]>(`${baseUrl}?title=${title}`);

}

}

## Create Angular 12 Components

As you’ve known before, there are 3 components corresponding to 3 routes defined in AppRoutingModule.

### Add new Item Component

This component has a Form to submit new Tutorial with 2 fields: title & description. It calls TutorialService.create() method.

**components**/**add-tutorial**/add-tutorial.component.ts

import { Component, OnInit } from '@angular/core';

import { Tutorial } from 'src/app/models/tutorial.model';

import { TutorialService } from 'src/app/services/tutorial.service';

@Component({

selector: 'app-add-tutorial',

templateUrl: './add-tutorial.component.html',

styleUrls: ['./add-tutorial.component.css']

})

export class AddTutorialComponent implements OnInit {

tutorial: Tutorial = {

title: '',

description: '',

published: false

};

submitted = false;

constructor(private tutorialService: TutorialService) { }

ngOnInit(): void {

}

saveTutorial(): void {

const data = {

title: this.tutorial.title,

description: this.tutorial.description

};

this.tutorialService.create(data)

.subscribe(

response => {

console.log(response);

this.submitted = true;

},

error => {

console.log(error);

});

}

newTutorial(): void {

this.submitted = false;

this.tutorial = {

title: '',

description: '',

published: false

};

}

}

**components**/**add-tutorial**/add-tutorial.component.html

<div>

<div class="submit-form">

<div \*ngIf="!submitted">

<div class="form-group">

<label for="title">Title</label>

<input

type="text"

class="form-control"

id="title"

required

[(ngModel)]="tutorial.title"

name="title"

/>

</div>

<div class="form-group">

<label for="description">Description</label>

<input

class="form-control"

id="description"

required

[(ngModel)]="tutorial.description"

name="description"

/>

</div>

<button (click)="saveTutorial()" class="btn btn-success">Submit</button>

</div>

<div \*ngIf="submitted">

<h4>Tutorial was submitted successfully!</h4>

<button class="btn btn-success" (click)="newTutorial()">Add</button>

</div>

</div>

</div>

**components**/**add-tutorial**/add-tutorial.component.css

.submit-form {

max-width: 400px;

margin: auto;

}

### List of items Component

This component calls 3 TutorialService methods:

* getAll()
* deleteAll()
* findByTitle()

**components**/**tutorials-list**/tutorials-list.component.ts

import { Component, OnInit } from '@angular/core';

import { Tutorial } from 'src/app/models/tutorial.model';

import { TutorialService } from 'src/app/services/tutorial.service';

@Component({

selector: 'app-tutorials-list',

templateUrl: './tutorials-list.component.html',

styleUrls: ['./tutorials-list.component.css']

})

export class TutorialsListComponent implements OnInit {

tutorials?: Tutorial[];

currentTutorial: Tutorial = {};

currentIndex = -1;

title = '';

constructor(private tutorialService: TutorialService) { }

ngOnInit(): void {

this.retrieveTutorials();

}

retrieveTutorials(): void {

this.tutorialService.getAll()

.subscribe(

data => {

this.tutorials = data;

console.log(data);

},

error => {

console.log(error);

});

}

refreshList(): void {

this.retrieveTutorials();

this.currentTutorial = {};

this.currentIndex = -1;

}

setActiveTutorial(tutorial: Tutorial, index: number): void {

this.currentTutorial = tutorial;

this.currentIndex = index;

}

removeAllTutorials(): void {

this.tutorialService.deleteAll()

.subscribe(

response => {

console.log(response);

this.refreshList();

},

error => {

console.log(error);

});

}

searchTitle(): void {

this.currentTutorial = {};

this.currentIndex = -1;

this.tutorialService.findByTitle(this.title)

.subscribe(

data => {

this.tutorials = data;

console.log(data);

},

error => {

console.log(error);

});

}

}

**components**/**tutorials-list**/tutorials-list.component.html

<div class="list row">

<div class="col-md-8">

<div class="input-group mb-3">

<input

type="text"

class="form-control"

placeholder="Search by title"

[(ngModel)]="title"

/>

<div class="input-group-append">

<button

class="btn btn-outline-secondary"

type="button"

(click)="searchTitle()"

>

Search

</button>

</div>

</div>

</div>

<div class="col-md-6">

<h4>Tutorials List</h4>

<ul class="list-group">

<li

class="list-group-item"

\*ngFor="let tutorial of tutorials; let i = index"

[class.active]="i == currentIndex"

(click)="setActiveTutorial(tutorial, i)"

>

{{ tutorial.title }}

</li>

</ul>

<button class="m-3 btn btn-sm btn-danger" (click)="removeAllTutorials()">

Remove All

</button>

</div>

<div class="col-md-6">

<div \*ngIf="currentTutorial.id">

<h4>Tutorial</h4>

<div>

<label><strong>Title:</strong></label> {{ currentTutorial.title }}

</div>

<div>

<label><strong>Description:</strong></label>

{{ currentTutorial.description }}

</div>

<div>

<label><strong>Status:</strong></label>

{{ currentTutorial.published ? "Published" : "Pending" }}

</div>

<a class="badge badge-warning" routerLink="/tutorials/{{ currentTutorial.id }}">

Edit

</a>

</div>

<div \*ngIf="!currentTutorial">

<br />

<p>Please click on a Tutorial...</p>

</div>

</div>

</div>

If you click on **Edit** button of any Tutorial, You will be directed to Tutorial page with url: /tutorials/:id.

**components**/**tutorials-list**/tutorials-list.component.css

.list {

text-align: left;

max-width: 750px;

margin: auto;

}

You can add Pagination to this Component, just follow instruction in the post:  
[Angular Pagination example with ngx-pagination](https://bezkoder.com/angular-12-pagination-ngx/)

### Item details Component

For getting data & update, delete the Tutorial, this component will use 3 TutorialService methods:

* get()
* update()
* delete()

**components**/**tutorial-details**/tutorial-details.component.ts

import { Component, OnInit } from '@angular/core';

import { TutorialService } from 'src/app/services/tutorial.service';

import { ActivatedRoute, Router } from '@angular/router';

import { Tutorial } from 'src/app/models/tutorial.model';

@Component({

selector: 'app-tutorial-details',

templateUrl: './tutorial-details.component.html',

styleUrls: ['./tutorial-details.component.css']

})

export class TutorialDetailsComponent implements OnInit {

currentTutorial: Tutorial = {

title: '',

description: '',

published: false

};

message = '';

constructor(

private tutorialService: TutorialService,

private route: ActivatedRoute,

private router: Router) { }

ngOnInit(): void {

this.message = '';

this.getTutorial(this.route.snapshot.params.id);

}

getTutorial(id: string): void {

this.tutorialService.get(id)

.subscribe(

data => {

this.currentTutorial = data;

console.log(data);

},

error => {

console.log(error);

});

}

updatePublished(status: boolean): void {

const data = {

title: this.currentTutorial.title,

description: this.currentTutorial.description,

published: status

};

this.message = '';

this.tutorialService.update(this.currentTutorial.id, data)

.subscribe(

response => {

this.currentTutorial.published = status;

console.log(response);

this.message = response.message ? response.message : 'The status was updated successfully!';

},

error => {

console.log(error);

});

}

updateTutorial(): void {

this.message = '';

this.tutorialService.update(this.currentTutorial.id, this.currentTutorial)

.subscribe(

response => {

console.log(response);

this.message = response.message ? response.message : 'This tutorial was updated successfully!';

},

error => {

console.log(error);

});

}

deleteTutorial(): void {

this.tutorialService.delete(this.currentTutorial.id)

.subscribe(

response => {

console.log(response);

this.router.navigate(['/tutorials']);

},

error => {

console.log(error);

});

}

}

**components**/**tutorial-details**/tutorial-details.component.html

<div>

<div \*ngIf="currentTutorial.id" class="edit-form">

<h4>Tutorial</h4>

<form>

<div class="form-group">

<label for="title">Title</label>

<input

type="text"

class="form-control"

id="title"

[(ngModel)]="currentTutorial.title"

name="title"

/>

</div>

<div class="form-group">

<label for="description">Description</label>

<input

type="text"

class="form-control"

id="description"

[(ngModel)]="currentTutorial.description"

name="description"

/>

</div>

<div class="form-group">

<label><strong>Status:</strong></label>

{{ currentTutorial.published ? "Published" : "Pending" }}

</div>

</form>

<button

class="badge badge-primary mr-2"

\*ngIf="currentTutorial.published"

(click)="updatePublished(false)"

>

UnPublish

</button>

<button

\*ngIf="!currentTutorial.published"

class="badge badge-primary mr-2"

(click)="updatePublished(true)"

>

Publish

</button>

<button class="badge badge-danger mr-2" (click)="deleteTutorial()">

Delete

</button>

<button

type="submit"

class="badge badge-success mb-2"

(click)="updateTutorial()"

>

Update

</button>

<p>{{ message }}</p>

</div>

<div \*ngIf="!currentTutorial.id">

<br />

<p>Cannot access this Tutorial...</p>

</div>

</div>

**components**/**tutorial-details**/tutorial-details.component.css

.edit-form {

max-width: 400px;

margin: auto;

}

## Run the Angular 12 CRUD App

You can run this APP with command: ng serve.