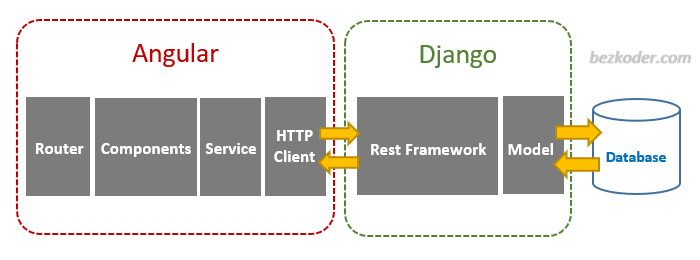
Project MySQL + Django + Angular 14:

Django

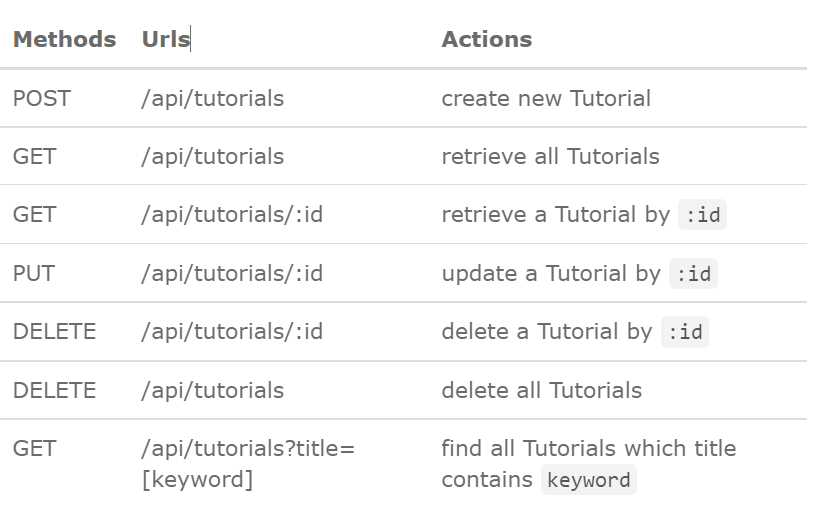


Django là một nền tảng web framework sử dụng ngôn ngữ Python năm 2003. Django được phát triển dựa trên kiến trúc MVT (Model-View-Template)

Django REST framework – một thư viện giúp người dùng có thể xây dựng các Rest API một cách đơn giản, nhanh chóng. (REST API (còn được biết với tên gọi RESTful API) là một giao diện lập trình ứng dụng (API) tuân thủ các ràng buộc và quy ước kiến trúc REST được sử dụng trong việc giao tiếp giữa client và server.)

húng ta thường tương tác dữ liệu qua lại bằng các gọi API tới 1 server khác. Angular đã cung cấp cho chúng ta một module tên là HTTPClientModule

Xậy dựng API cho Django App:



*From rest\_framework import serializers*

serializers.py để viết các serializer cho các models của project (Serializer giúp chuyển các dữ liệu có cấu trúc phức tạp như querysets hoặc model instance thành các kiểu dữ liệu Python truyền thống, từ đó có thể dễ dàng chuyển sang các định dạng khác như json, xml,… Serializer cũng cung cấp tính năng deserialization là quá trình ngược lại với serialization để chuyển dữ liệu lại thành các dữ liệu ban đầu, có cấu trúc phức tạp).

*from rest\_framework.decorators import api\_view*

@api\_view (sử dụng cho function-based view): cung cấp các tính năng như bảo đảm view nhận được đúng loại Request, xử lý lỗi với request.data,…

*from django.http.response import JsonResponse*

JsonResponse là một lớp con HTTPRESPONSE giúp tạo phản hồi được mã hóa JSON

*from rest\_framework.parsers import JSONParser*

JSONParser: Nó phân tích nội dung JSON yêu cầu đến vào loại nội dung Python.

is\_valid(): Phương pháp chạy xác thực và trả về Boolean chỉ định xem dữ liệu có hợp lệ hay không

filter(title\_\_icontains): Sẽ trả về tất cả các đối tượng có tiêu đề chứa giá trị, nhưng bất kỳ chữ cái hoa/chữ thường nào cũng sẽ khớp.

CORS\_ORIGIN\_ALLOW\_ALL: False // Chỉ chấp nhận truy cập từ các miền được nêu trong CORS\_ORIGIN\_WHITELIST

CORS\_ORIGIN\_WHITELIST = (

'http://localhost:8081',

)

Angular 14

1. Tạo project

ng new angular-14-crud-example

? Would you like to add Angular routing? Yes

? Which stylesheet format would you like to use? CSS

1. Thiết lập các Components và Services:

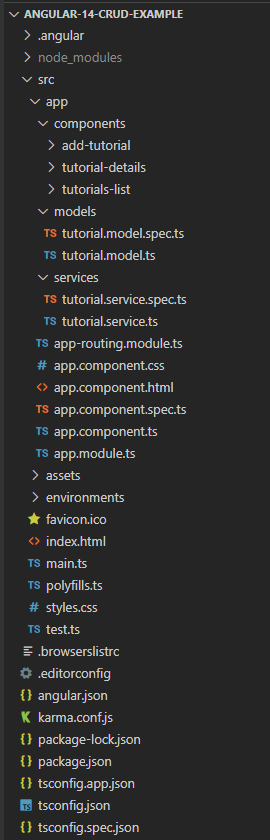
ng g class models/tutorial --type=model

ng g c components/add-tutorial

ng g c components/tutorial-details

ng g c components/tutorials-list

ng g s services/tutorial

tutorial.model.ts exports the main class model: Tutorial.  
–There are 3 components: tutorials-list, tutorial-details, add-tutorial.

– tutorial.service chứa methods cho việc gửi HTTP requests to the APIs.

– app-routing.module.ts chứa các đường dẫn đến các chức năng riêng( Componets)

– app component contains router view and navigation bar.

– app.module.ts declares Angular components and import necessary modules.

– style.css contains import for Bootstrap

Cài đặt App Module

Open app.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 { }

FormsModule: Chứa các thành phần và chỉ thị cần thiết cho các biểu mẫu dựa trên mẫu, làm cho chúng có sẵn để nhập bởi Ngmodules nhập mô-đun này.

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 { }

NgModel chỉ thị liên kết giá trị của HTML điều khiển (đầu vào, chọn, vùng văn bản) đến dữ liệu ứng dụng.

Import Bootstrap into Angular 14 Project:

npm install bootstrap@4.6.1.

**src**/style.css:

@import "~bootstrap/dist/css/bootstrap.css";

Add Navbar and Router View

**src**/app.component.html

<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>

Router là một module được đặt tại @angular/router, cung cấp cho ứng dụng Angluar của chúng ta khả năng điều hướng và hiển thị nội dung phù hợp với địa chỉ URL.

<app-items-list></app-items-list> : Cài đặt set tĩnh khi thay đổi trong html của component.

<router-outlet></router-outlet>: nội dung của component đã được chỉ định được load cùng với url path, giúp ta chỉ định rõ phần thay đổi và kiểm soát nội dung trong trang tốt hơn.

Thiết lập thành phần của Class Tutorial

**models**/tutorial.model.ts

export class Tutorial {

id?: any;

title?: string;

description?: string;

published?: boolean;

}

“ ? ” Dùng để define 1 thuộc tính, những thuộc tính đó có thể có hoặc không đối với đối tượng Tutorial.

Create Data Service

**Dịch vụ này sẽ sử dụng angular httpclient để gửi các yêu cầu HTTP. Bạn có thể thấy rằng các chức năng của nó bao gồm các hoạt động CRUD và phương thức tìm kiếm.**

**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<Tutorial>(`${baseUrl}/${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}`);

}

}

*import { Observable } from 'rxjs';*

Observable được dùng để chỉ các hàm đưa ra giá trị. Các Observable mở ra một kênh liên lạc liên tục trong đó nhiều giá trị dữ liệu có thể được phát ra theo thời gian. Observable trong Angular để quản lý các dữ liệu bất đồng bộ, Observable quản lý các đối tượng subscribes

HTTP Client để gọi các service bên ngoài (Django)

Inject HttpClient vào Service nơi sẽ gọi các webservice bên ngoài thông qua constructor

Create Angular 14 Components

**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({

next: (res) => {

console.log(res);

this.submitted = true;

},

error: (e) => console.error(e)

});

}

newTutorial(): void {

this.submitted = false;

this.tutorial = {

title: '',

description: '',

published: false

};

}

}

Phương thức khởi tạo (constructor hay là hàm khởi tạo) là một phương thức đặc biệt để tạo và khởi tạo một đối tượng từ một class nào đó.

Next(): xử lý dữ liệu được yêu cầu

. subscribe: Đăng kí nhận dữ liệu

**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({

next: (data) => {

this.tutorials = data;

console.log(data);

},

error: (e) => console.error(e)

});

}

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({

next: (res) => {

console.log(res);

this.refreshList();

},

error: (e) => console.error(e)

});

}

searchTitle(): void {

this.currentTutorial = {};

this.currentIndex = -1;

this.tutorialService.findByTitle(this.title)

.subscribe({

next: (data) => {

this.tutorials = data;

console.log(data);

},

error: (e) => console.error(e)

});

}

}

Constructor là hàm tạo của một class, khi khởi tạo một instance của class thì nó sẽ được tự động chạy, và chỉ chạy duy nhất một lần. Cũng là nơi đưa ra các dịch vụ.

ngOninit là một life-cycle method,  tự động gọi khi component được khởi tạo

**Observable** thì sẽ thường xuyên sử dụng phương thức **subscribe** để theo dõi các thay đổi của dữ liệu

**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">

<app-tutorial-details

[viewMode]="true"

[currentTutorial]="currentTutorial"

></app-tutorial-details>

</div>

</div>

<app-tutorial-details [viewMode]="true" [currentTutorial]="currentTutorial"></apptutorial-details>

Chuyển đến tutorial-details Component để hiển thị toàn bộ dữ liệu hiện có trang database

### Item details Component ( Chi tiết về 1 Tutorial )

if viewMode is true: Hiển thị Tutorial được chọn

if viewMode is false hiển thị list Tutorial hiện có để cho người dùng chọn lựa

This component will use 3 TutorialService methods:

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

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

import { Component, Input, 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 {

@Input() viewMode = false;

@Input() currentTutorial: Tutorial = {

title: '',

description: '',

published: false

};

message = '';

constructor(

private tutorialService: TutorialService,

private route: ActivatedRoute,

private router: Router) { }

ngOnInit(): void {

if (!this.viewMode) {

this.message = '';

this.getTutorial(this.route.snapshot.params["id"]);

}

}

getTutorial(id: string): void {

this.tutorialService.get(id)

.subscribe({

next: (data) => {

this.currentTutorial = data;

console.log(data);

},

error: (e) => console.error(e)

});

}

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({

next: (res) => {

console.log(res);

this.currentTutorial.published = status;

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

},

error: (e) => console.error(e)

});

}

updateTutorial(): void {

this.message = '';

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

.subscribe({

next: (res) => {

console.log(res);

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

},

error: (e) => console.error(e)

});

}

deleteTutorial(): void {

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

.subscribe({

next: (res) => {

console.log(res);

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

},

error: (e) => console.error(e)

});

}

}

params: Is A Observable

Navigate : điều hướng

## @INPUT DECORATOR

@Input() progress = 0; Khai báo ràng component nhận vào một biến tên là progress và có giá trị mặc định là 0

## Run the Angular 14 CRUD example

Fontend: ng serve --port 8081 Vì đã setup CORS port: 8081

Backend: python manage.py runserver 8080. Setup in the tutorial.service.ts

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