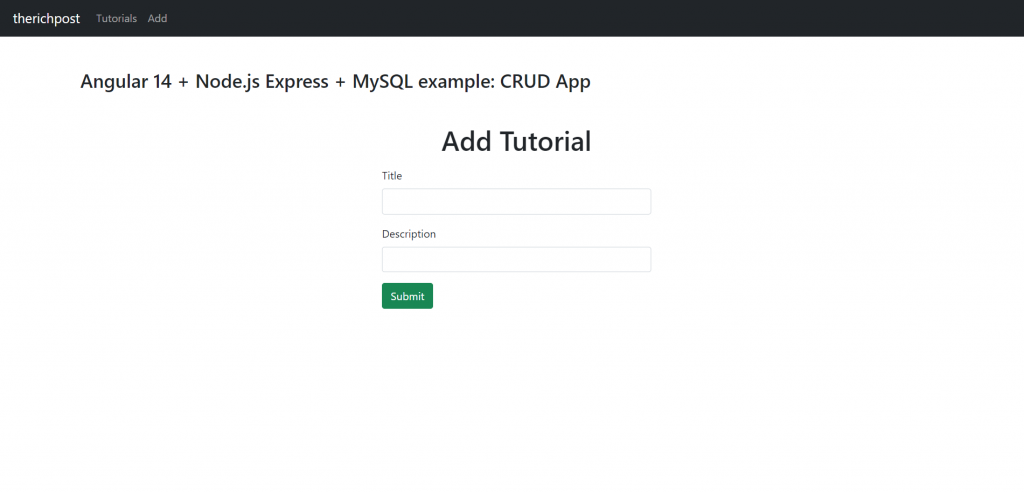
* Angular 14
* Bootstrap 5
* RxJs
* Nodejs
* Express
* Cors
* Mysql
* Add, update, delete, deleteAll, searchFilter

[**Angular 14**](https://therichpost.com/category/angular-14/) came and if you are new then please check below links:

**Angular 14 + Node.js Express + MySQL example: CRUD App**

Guys here is the working complete code snippet and please use carefully and if you have any kind of doubt then please comment below:

**1. Very first, we need run below commands into our terminal to get angular 14 fresh setup:**

npm install -g @angular/cli //Setup Angular14 atmosphere

ng new angularcrud //Install New Angular App

/\*\*You need to update your Nodejs also for this verison\*\*/

cd angularcrud //Go inside the Angular 14 Project

**2. Now guys we need to run below commands into your terminal to get components, services, models and bootstrap 5 modules into your angular 14 application:**

ng g class models/tutorial --type=model

ng g c components/add-tutorial

ng g c components/tutorial-details

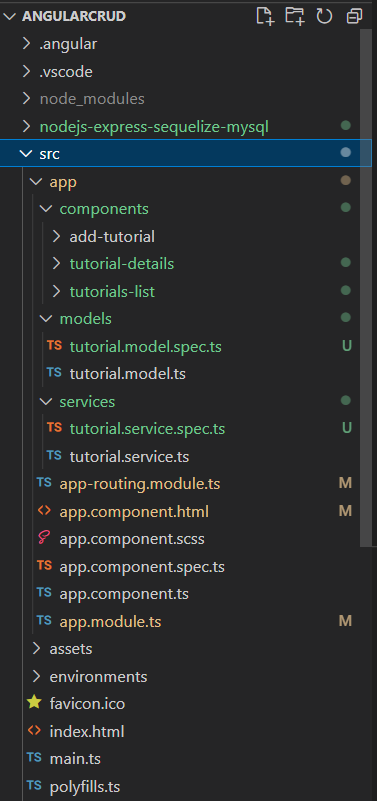
ng g c components/tutorials-list

ng g s services/tutorial

npm install bootstrap

npm i @popperjs/core

**After all above commands we will below image like folder structure:**

**Angular 14 nodejs express MySQL crud application folder structure**

**3. Now add below code into your angular.json file for bootstrap styles:**

"styles": [

...

"node\_modules/bootstrap/dist/css/bootstrap.min.css",

],

"scripts": [

...

"node\_modules/bootstrap/dist/js/bootstrap.min.js",

]

**4. Now add below code into your src/app/app.module.ts file:**

...

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

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

@NgModule({

declarations: [ ... ],

imports: [

...

FormsModule,

HttpClientModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**5. Now add below code into your src/app/app-routing.module.ts file:**

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 { }

**6. Now add below code into your src/app/app-component.html file:**

<div>

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

<a href="#" class="navbar-brand">therichpost</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-5">

<router-outlet></router-outlet>

</div>

</div>

**7. Now add below code into your src/app/models/tutorial.model.ts file for our four fields:**

export class Tutorial {

id?: any;

title?: string;

description?: string;

published?: boolean;

}

**8. Now add below code into your src/app/services/tutorial.service.ts file for api  crud request:**

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}`);

}

}

**9. Now add below code into your src/app/components/add-tutorial/add-tutorial.component.ts file:**

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.scss']

})

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

};

}

}

**10. Now add below code into your src/app/components/add-tutorial/add-tutorial.component.html file:**

<div class=" mt-5">

<h1 class="mb-3 text-center">Add Tutorial</h1>

<div class="submit-form">

<div \*ngIf="!submitted">

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

<label for="title" class="mb-2">Title</label>

<input

type="text"

class="form-control"

id="title"

required

[(ngModel)]="tutorial.title"

name="title"

/>

</div>

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

<label for="description" class="mb-2">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>

**11. Now add below code into your src/app/components/add-tutorial/add-tutorial.component.scss file:**

.submit-form {

max-width: 400px;

margin: auto;

}

**12. Now add below code into your src/app/components/tutorial-details/tutorial-details.component.ts file:**

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.scss']

})

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)

});

}

}

**13. Now add below code into your src/app/components/tutorial-details/tutorial-details.component.html file:**

<div \*ngIf="viewMode; else editable">

<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="btn btn-warning"

routerLink="/tutorials/{{ currentTutorial.id }}"

>

Edit

</a>

</div>

<div \*ngIf="!currentTutorial">

<br />

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

</div>

</div>

<ng-template #editable>

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

<h4>Tutorial</h4>

<form>

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

<label for="title" class="mb-2">Title</label>

<input

type="text"

class="form-control"

id="title"

[(ngModel)]="currentTutorial.title"

name="title"

/>

</div>

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

<label for="description" class="mb-2">Description</label>

<input

type="text"

class="form-control"

id="description"

[(ngModel)]="currentTutorial.description"

name="description"

/>

</div>

<div class="form-group mb-5">

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

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

</div>

</form>

<button

class="btn btn-primary me-2 mb-2"

\*ngIf="currentTutorial.published"

(click)="updatePublished(false)"

>

UnPublish

</button>

<button

\*ngIf="!currentTutorial.published"

class="btn btn-primary me-2 mb-2"

(click)="updatePublished(true)"

>

Publish

</button>

<button class="btn btn-danger me-2 mb-2" (click)="deleteTutorial()">

Delete

</button>

<button

type="submit"

class="btn btn-success mb-2"

(click)="updateTutorial()"

>

Update

</button>

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

</div>

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

<br />

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

</div>

</ng-template>

**14. Now add below code into your src/app/components/tutorial-details/tutorial-details.component.scss file:**

.list {

text-align: left;

max-width: 750px;

margin: auto;

}

.edit-form {

max-width: 400px;

margin: auto;

}

**15. Now add below code into your src/app/components/tutorials-list/tutorials-list.component.ts file:**

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.scss']

})

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)

});

}

}

**16. Now add below code into your src/app/components/tutorials-list/tutorials-list.component.html file:**

<div class="list row mt-5">

<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 mt-3">

<h4 class="mb-3">Tutorials List</h4>

<ul class="list-group mb-3">

<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="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>

Guys now we have all the angular code and below I am going to write nodejs+express code and please follow it carefully also:

**1. Now guys create folder `nodejs-express-sequelize-mysql` inside angular project root and run below command inside the folder:**

npm init

name: (nodejs-express-sequelize-mysql)

version: (1.0.0)

description: Node.js Rest Apis with Express, Sequelize & MySQL.

entry point: (index.js) server.js

test command:

git repository:

keywords: nodejs, express, sequelize, mysql, rest, api

author: therichpost

license: (ISC)

Is this ok? (yes) yes

**2. Guys now inside `nodejs-express-sequelize-mysql` folder create server.js file and add below code inside it:**

const express = require("express");

const cors = require("cors");

const app = express();

var corsOptions = {

origin: "http://localhost:4200"

};

app.use(cors(corsOptions));

// parse requests of content-type - application/json

app.use(express.json());

// parse requests of content-type - application/x-www-form-urlencoded

app.use(express.urlencoded({ extended: true }));

const db = require("./app/models");

db.sequelize.sync()

.then(() => {

console.log("Synced db.");

})

.catch((err) => {

console.log("Failed to sync db: " + err.message);

});

// simple route

app.get("/", (req, res) => {

res.json({ message: "Welcome to therichpost application." });

});

// set port, listen for requests

require("./app/routes/turorial.routes")(app);

const PORT = process.env.PORT || 8080;

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}.`);

});

**3. Guys also run below command inside `nodejs-express-sequelize-mysql` folder:**

npm install express sequelize mysql2 cors --save

**4. Now guys create `app` folder inside `nodejs-express-sequelize-mysql` folder.**

**5. Now guys create `config` folder inside `nodejs-express-sequelize-mysql/app` folder and then create `db.config.js` file  inside nodejs-express-sequelize-mysql/app/config and add below code inside it for mysql connection:**

module.exports = {

HOST: "localhost",

USER: "root",

PASSWORD: "",

DB: "testdb",

dialect: "mysql",

pool: {

max: 5,

min: 0,

acquire: 30000,

idle: 10000

}

};

**6. Now guys create `controllers` folder inside `nodejs-express-sequelize-mysql/app` folder and then create `tutorial.controller.js` file  inside nodejs-express-sequelize-mysql/app/controllers and add below code inside it for crud functions:**

const db = require("../models");

const Tutorial = db.tutorials;

const Op = db.Sequelize.Op;

// Create and Save a new Tutorial

exports.create = (req, res) => {

// Validate request

if (!req.body.title) {

res.status(400).send({

message: "Content can not be empty!"

});

return;

}

// Create a Tutorial

const tutorial = {

title: req.body.title,

description: req.body.description,

published: req.body.published ? req.body.published : false

};

// Save Tutorial in the database

Tutorial.create(tutorial)

.then(data => {

res.send(data);

})

.catch(err => {

res.status(500).send({

message:

err.message || "Some error occurred while creating the Tutorial."

});

});

};

// Retrieve all Tutorials from the database.

exports.findAll = (req, res) => {

const title = req.query.title;

var condition = title ? { title: { [Op.like]: `%${title}%` } } : null;

Tutorial.findAll({ where: condition })

.then(data => {

res.send(data);

})

.catch(err => {

res.status(500).send({

message:

err.message || "Some error occurred while retrieving tutorials."

});

});

};

// Find a single Tutorial with an id

exports.findOne = (req, res) => {

const id = req.params.id;

Tutorial.findByPk(id)

.then(data => {

if (data) {

res.send(data);

} else {

res.status(404).send({

message: `Cannot find Tutorial with id=${id}.`

});

}

})

.catch(err => {

res.status(500).send({

message: "Error retrieving Tutorial with id=" + id

});

});

};

// Update a Tutorial by the id in the request

exports.update = (req, res) => {

const id = req.params.id;

Tutorial.update(req.body, {

where: { id: id }

})

.then(num => {

if (num == 1) {

res.send({

message: "Tutorial was updated successfully."

});

} else {

res.send({

message: `Cannot update Tutorial with id=${id}. Maybe Tutorial was not found or req.body is empty!`

});

}

})

.catch(err => {

res.status(500).send({

message: "Error updating Tutorial with id=" + id

});

});

};

// Delete a Tutorial with the specified id in the request

exports.delete = (req, res) => {

const id = req.params.id;

Tutorial.destroy({

where: { id: id }

})

.then(num => {

if (num == 1) {

res.send({

message: "Tutorial was deleted successfully!"

});

} else {

res.send({

message: `Cannot delete Tutorial with id=${id}. Maybe Tutorial was not found!`

});

}

})

.catch(err => {

res.status(500).send({

message: "Could not delete Tutorial with id=" + id

});

});

};

// Delete all Tutorials from the database.

exports.deleteAll = (req, res) => {

Tutorial.destroy({

where: {},

truncate: false

})

.then(nums => {

res.send({ message: `${nums} Tutorials were deleted successfully!` });

})

.catch(err => {

res.status(500).send({

message:

err.message || "Some error occurred while removing all tutorials."

});

});

};

// Find all published Tutorials

exports.findAllPublished = (req, res) => {

Tutorial.findAll({ where: { published: true } })

.then(data => {

res.send(data);

})

.catch(err => {

res.status(500).send({

message:

err.message || "Some error occurred while retrieving tutorials."

});

});

};

**7. Now guys create `routes` folder inside `nodejs-express-sequelize-mysql/app` folder and then create `turorial.routes.js` file  inside nodejs-express-sequelize-mysql/app/routes and add below code inside it for api routing:**

module.exports = app => {

const tutorials = require("../controllers/tutorial.controller.js");

var router = require("express").Router();

// Create a new Tutorial

router.post("/", tutorials.create);

// Retrieve all Tutorials

router.get("/", tutorials.findAll);

// Retrieve all published Tutorials

router.get("/published", tutorials.findAllPublished);

// Retrieve a single Tutorial with id

router.get("/:id", tutorials.findOne);

// Update a Tutorial with id

router.put("/:id", tutorials.update);

// Delete a Tutorial with id

router.delete("/:id", tutorials.delete);

// Delete all Tutorials

router.delete("/", tutorials.deleteAll);

app.use('/api/tutorials', router);

};

**8. Now guys create `models` folder inside `nodejs-express-sequelize-mysql/app` folder and then create `index.js` file  inside nodejs-express-sequelize-mysql/app/models and add below code inside it for models:**

const dbConfig = require("../config/db.config.js");

const Sequelize = require("sequelize");

const sequelize = new Sequelize(dbConfig.DB, dbConfig.USER, dbConfig.PASSWORD, {

host: dbConfig.HOST,

dialect: dbConfig.dialect,

operatorsAliases: false,

pool: {

max: dbConfig.pool.max,

min: dbConfig.pool.min,

acquire: dbConfig.pool.acquire,

idle: dbConfig.pool.idle

}

});

const db = {};

db.Sequelize = Sequelize;

db.sequelize = sequelize;

db.tutorials = require("./tutorial.model.js")(sequelize, Sequelize);

module.exports = db;

**9. Now guys create `models` folder inside `nodejs-express-sequelize-mysql/app` folder and then create `tutorial.model.js` file  inside nodejs-express-sequelize-mysql/app/models and add below code inside it for models**

**(This Sequelize Model represents tutorials table in MySQL database. These columns will be generated automatically: id, title, description, published, createdAt, updatedAt.):**

module.exports = (sequelize, Sequelize) => {

const Tutorial = sequelize.define("tutorial", {

title: {

type: Sequelize.STRING

},

description: {

type: Sequelize.STRING

},

published: {

type: Sequelize.BOOLEAN

}

});

return Tutorial;

};