**INF 354 Semester test notes:**

**VS 2022: (Homework 1)**

**Controller:**

using Architecture.Models;

using Architecture.ViewModel;

using Microsoft.AspNetCore.Components.Forms;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System.Reflection.Metadata.Ecma335;

namespace Architecture.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class CourseController : ControllerBase

{

private readonly ICourseRepository \_courseRepository;

public CourseController(ICourseRepository courseRepository)

{

\_courseRepository = courseRepository;

}

[HttpGet]

[Route("GetAllCourses")]

public async Task<IActionResult> GetAllCourses()

{

try

{

var results = await \_courseRepository.GetAllCourseAsync();

return Ok(results);

}

catch (Exception)

{

return StatusCode(500,"Internal Server Error. Please contact support.");

}

}

[HttpGet]

[Route("GetCourse/{courseId}")]

public async Task<IActionResult> GetCourseAsync(int courseId)

{

try

{

var result = await \_courseRepository.GetCourseAsync(courseId);

if (result == null) return NotFound("Course does not exist. You need to create it first");

return Ok(result);

}

catch (Exception)

{

return StatusCode(500, "Internal Server Error. Please contact support");

}

}

[HttpPost]

[Route("AddCourse")]

public async Task<IActionResult> AddCourse(CourseViewModel cvm)

{

var course = new Course { Name = cvm.Name, Duration = cvm.Duration, Description = cvm.Description };

try

{

\_courseRepository.Add(course);

await \_courseRepository.SaveChangesAsync();

}

catch (Exception)

{

return BadRequest("Invalid transaction");

}

return Ok(course);

}

[HttpPut]

[Route("EditCourse/{courseId}")]

public async Task<ActionResult<CourseViewModel>> EditCourse(int courseId, CourseViewModel courseModel)

{

try

{

var existingCourse = await \_courseRepository.GetCourseAsync(courseId);

if (existingCourse == null) return NotFound($"The course does not exist");

existingCourse.Name = courseModel.Name;

existingCourse.Duration = courseModel.Duration;

existingCourse.Description = courseModel.Description;

if (await \_courseRepository.SaveChangesAsync())

{

return Ok(existingCourse);

}

}

catch (Exception)

{

return StatusCode(500, "Internal Server Error. Please contact support.");

}

return BadRequest("Your request is invalid.");

}

[HttpDelete]

[Route("DeleteCourse/{courseId}")]

public async Task<IActionResult> DeleteCourse(int courseId)

{

try

{

var existingCourse = await \_courseRepository.GetCourseAsync(courseId);

if (existingCourse == null) return NotFound($"The course does not exist");

\_courseRepository.Delete(existingCourse);

if (await \_courseRepository.SaveChangesAsync()) return Ok(existingCourse);

}

catch (Exception)

{

return StatusCode(500, "Internal Server Error. Please contact support.");

}

return BadRequest("Your request is invalid.");

}

}

}

**Course Repository:**

using Microsoft.EntityFrameworkCore;

namespace Architecture.Models

{

public class CourseRepository : ICourseRepository

{

private readonly AppDbContext \_appDbContext;

public CourseRepository(AppDbContext appDbContext)

{

\_appDbContext = appDbContext;

}

public void Add<T>(T entity) where T : class

{

\_appDbContext.Add(entity);

}

public void Delete<T>(T entity) where T : class

{

\_appDbContext.Remove(entity);

}

public async Task<Course[]> GetAllCourseAsync()

{

IQueryable<Course> query = \_appDbContext.Courses;

return await query.ToArrayAsync();

}

public async Task<Course> GetCourseAsync(int courseId)

{

IQueryable<Course> query = \_appDbContext.Courses.Where(c => c.CourseId == courseId);

return await query.FirstOrDefaultAsync();

}

public async Task<bool> SaveChangesAsync()

{

return await \_appDbContext.SaveChangesAsync() > 0;

}

}

}

**DocumentType Repository:**

using AutoDocs.ViewModels;

using AutoDocs.Models;

using Microsoft.EntityFrameworkCore;

using System;

using System.Linq;

using System.Threading.Tasks;

// Define the namespace for the CourseRepository class

namespace AutoDocs.Models

{

// This class implements the ICourseRepository interface

public class Document\_TypeRepository : IDocument\_TypeRepository

{

// The AppDbContext instance used to interact with the database

private readonly AppDbContext \_appDbContext;

// Constructor receives an AppDbContext instance and assigns it to the private field

public Document\_TypeRepository(AppDbContext appDbContext)

{

\_appDbContext = appDbContext;

}

// This method retrieves all courses from the database

public async Task<Document\_Type[]> GetAllDocTypeAsync()

{

IQueryable<Document\_Type> query = \_appDbContext.Document\_Types;

return await query.ToArrayAsync();

}

// This method retrieves a single course by courseId from the database

public async Task<Document\_Type> GetDocTypeAsync(int Document\_TypeID)

{

IQueryable<Document\_Type> query = \_appDbContext.Document\_Types.Where(c => c.Document\_TypeID == Document\_TypeID);

return await query.FirstOrDefaultAsync();

}

// This method saves the changes made to the DbContext to the database

public async Task<bool> SaveChangesAsync()

{

return await \_appDbContext.SaveChangesAsync() > 0;

}

// This method adds a new course to the database

public async Task<int> AddDocTypeAsync(Document\_TypeViewModel DocTypeVm)

{

const int successCode = 200;

const int errorCode = 500;

try

{

// Create a new Course instance and populate it with data from the ViewModel

Document\_Type newDocType = new Document\_Type

{

Document\_TypeName = DocTypeVm.Document\_TypeName,

Document\_TypeDescription = DocTypeVm.Document\_TypeDescription,

Document\_TypeExtension = DocTypeVm.Document\_TypeExtension,

};

// Add the new course to the database and save changes

await \_appDbContext.Document\_Types.AddAsync(newDocType);

await \_appDbContext.SaveChangesAsync();

// Return success code if everything went well

return successCode;

}

catch (Exception)

{

// Return error code if an exception occurred

return errorCode;

}

}

// This method deletes a course by id from the database

public async Task<Document\_Type> DeleteDocTypeAsync(int id)

{

var DocType = await \_appDbContext.Document\_Types.FirstOrDefaultAsync(c => c.Document\_TypeID == id);

if (DocType != null)

{

\_appDbContext.Document\_Types.Remove(DocType);

await \_appDbContext.SaveChangesAsync();

}

return DocType;

}

// This method updates a course in the database

public async Task<Document\_Type> UpdateDocTypeAsync(int id, Document\_TypeViewModel updatedDocType)

{

var DocType = await \_appDbContext.Document\_Types.FirstOrDefaultAsync(c => c.Document\_TypeID == id);

// If the course exists, update its properties

if (DocType != null)

{

DocType.Document\_TypeName = updatedDocType.Document\_TypeName;

DocType.Document\_TypeDescription = updatedDocType.Document\_TypeDescription;

DocType.Document\_TypeExtension = updatedDocType.Document\_TypeExtension;

// Update the course in the database and save changes

\_appDbContext.Document\_Types.Update(DocType);

await \_appDbContext.SaveChangesAsync();

}

// Return the updated course

return DocType;

}

}

}

**ICourseRepository:**

namespace Architecture.Models

{

public interface ICourseRepository

{

void Add<T>(T entity) where T : class;

void Delete<T>(T entity) where T : class;

Task<bool> SaveChangesAsync();

// Course

Task<Course[]> GetAllCourseAsync();

Task<Course> GetCourseAsync(int courseId);

}

}

**ViewModel:**

namespace Architecture.ViewModel

{

public class CourseViewModel

{

public string Name { get; set; }

public string Duration { get; set; }

public string Description { get; set; }

}

}

**Appsettings.json:**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"DefaultConnection": "Server=servername.;Database=Assignment1;Trusted\_Connection=True;MultipleActiveResultSets=True"

}

}

**For Database:**

Add-migration (name)

Update-database

**VSCODE:**

**SERVICE.ts**

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

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

import { map, Observable, Subject } from 'rxjs';

import { Course } from '../shared/course';

@Injectable({

providedIn: 'root'

})

export class DataService {

apiUrl = 'http://localhost:5116/api/'

httpOptions ={

headers: new HttpHeaders({

ContentType: 'application/json'

})

}

constructor(private httpClient: HttpClient) {

}

GetCourses(): Observable<any>{

return this.httpClient.get(`${this.apiUrl}Course/GetAllCourses`)

.pipe(map(result => result))

}

AddCourse(course: Course): Observable<Course[]> {

return this.httpClient.post<Course[]>(`${this.apiUrl}Course/AddCourses`, course, this.httpOptions)

}

DeleteCourse(courseId: number): Observable<Course[]> {

const url = `${this.apiUrl}Course/DeleteCourse/${courseId}`;

return this.httpClient.delete<Course[]>(url, this.httpOptions);

}

EditCourse(courseId: number, course: Course): Observable<Course[]> {

const url = `${this.apiUrl}Course/EditCourse/${courseId}`;

return this.httpClient.put<Course[]>(url, course, this.httpOptions);

}

}

Course.ts

export interface Course {

courseId: number;

name:String;

duration:String;

description:String;

}

**Course.componenet.ts - Get and delete method:**

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

import { DataService } from '../services/data.service';

import { Course } from '../shared/course';

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

@Component({

selector: 'app-courses',

templateUrl: './courses.component.html',

styleUrls: ['./courses.component.scss']

})

export class CoursesComponent implements OnInit {

courses:Course[] = []

constructor(private dataService: DataService, private router : Router) { }

ngOnInit(): void {

this.GetCourses()

console.log(this.courses)

}

GetCourses()

{

this.dataService.GetCourses().subscribe(result => {

let courseList:any[] = result

courseList.forEach((element) => {

this.courses.push(element)

});

})

}

onDelete(course: Course) {

if(confirm(`Are you sure you want to delete ${course.name}?`)) {

this.dataService.DeleteCourse(course.courseId).subscribe(

response => {

// handle success

console.log(response);

// Remove course from array with lambda statement

this.courses = this.courses.filter(c => c.courseId !== course.courseId);

},

error => {

// handle error

console.error(error);

}

);

}

}

}

Coursehtml:  
<div class="table-responsive" style="height: 500px; overflow-y:auto;">

<table class="table table-striped">

<thead class="thead-dark">

<tr>

<th>Name</th>

<th>Duration</th>

<th>Description</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let course of courses">

<td>{{ course.name }}</td>

<td>{{ course.duration }}</td>

<td>{{ course.description }}</td>

<td>

<button class="btn btn-primary" [routerLink]="['/editcourses', course.courseId]">Edit</button>

<button type="button" class="btn btn-danger" (click)="onDelete(course)">Delete</button>

</td>

</tr>

</tbody>

</table>

</div>

<style>

body {

font-family: Arial, sans-serif;

}

.table-responsive {

height: 500px;

overflow-y: auto;

}

.table {

width: 100%;

border-collapse: collapse;

}

.table thead th {

background-color: #343a40;

color: white;

padding: 10px;

border: 1px solid #ffffff;

text-align: left;

}

.table tbody tr:nth-child(even) {

background-color: #686363;

}

.table tbody tr:hover {

background-color: #dddddd5b;

cursor: pointer;

}

.table tbody td {

padding: 8px;

border: 1px solid #ddd;

}

.btn {

margin-right: 5px;

}

</style>

Addcourse.component.ts

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

import { DataService } from '../services/data.service';

import { Course } from '../shared/course'

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

@Component({

selector: 'app-addcourse',

templateUrl: './addcourse.component.html',

styleUrls: ['./addcourse.component.scss']

})

export class AddcourseComponent implements OnInit {

courses: Course[] = []

newCourse: Course = { courseId: this.courses.length + 1, name: '', duration: '', description: '' };

constructor(private dataService: DataService, private router: Router) { }

ngOnInit(): void {

console.log(this.newCourse)

}

AddCourse() {

this.dataService.AddCourse(this.newCourse).subscribe(() => {

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

});

}

}

Addcourse.html:

<!DOCTYPE html>

<html lang="en" style="height: 100%;">

<head>

<title>Add Course</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

</head>

<body>

<div class="container" style="background: rgb(255, 255, 255);">

<h1 class="mt-5 mb-4">Add Course</h1>

<form (ngSubmit)="AddCourse()">

<div class="form-group">

<label for="courseName">Name:</label>

<input type="text" class="form-control" id="courseName" placeholder="Enter course name" required name="name" [(ngModel)]="newCourse.name">

</div>

<div class="form-group">

<label for="courseDuration">Duration:</label>

<input type="text" class="form-control" id="courseDuration" placeholder="Enter course duration" required name="duration" [(ngModel)]="newCourse.duration">

</div>

<div class="form-group">

<label for="courseDescription">Description:</label>

<textarea class="form-control" id="courseDescription" rows="3" placeholder="Enter course description" required name="description" [(ngModel)]="newCourse.description"></textarea>

</div>

<button type="submit" class="btn btn-success">Add</button>

<a href="/courses" class="btn btn-danger">Cancel</a>

</form>

</div>

</body>

</html>

Editcourse.component.ts

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

import { DataService } from '../services/data.service';

import { Course } from '../shared/course';

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

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

@Component({

selector: 'app-editcourse',

templateUrl: './editcourse.component.html',

styleUrls: ['./editcourse.component.scss']

})

export class EditcourseComponent implements OnInit {

currentcourse : Course = { courseId: -1, name: '', duration: '', description: ''};

editcourse : Course = { courseId: -1, name: '', duration: '', description: '' };

constructor(private dataservice: DataService, private router: Router, private route: ActivatedRoute) { }

ngOnInit(): void {

// Using paramMap to get course ID inside url.

const mycourseId = Number(this.route.snapshot.paramMap.get('courseId'));

console.log(this.editcourse);

}

EditCourse() {

const mycourseId = Number(this.route.snapshot.paramMap.get('courseId'));

this.dataservice.EditCourse(mycourseId, this.editcourse).subscribe(

response => {

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

},

error => {

// handle error

console.error(error);

})

}

}

Edithtml:

<!DOCTYPE html>

<html lang="en" style="height: 100%;">

<head>

<title>Edit Course</title>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

</head>

<body>

<div class="container">

<h1 class="mt-5 mb-4">Edit Course</h1>

<form (ngSubmit)="EditCourse()">

<div class="form-group">

<label for="courseName">Name:</label>

<input type="text" class="form-control" id="courseName" [placeholder]="currentcourse.name" required name="name" [(ngModel)]="editcourse.name">

</div>

<div class="form-group">

<label for="courseDuration">Duration:</label>

<input type="text" class="form-control" id="courseDuration" [placeholder]="currentcourse.duration" required name="duration" [(ngModel)]="editcourse.duration">

</div>

<div class="form-group">

<label for="courseDescription">Description:</label>

<textarea class="form-control" id="courseDescription" rows="3" [placeholder]="currentcourse.description" required name="description" [(ngModel)]="editcourse.description"></textarea>

</div>

<button type="submit" class="btn btn-warning">Make changes</button>

<a href="/courses" class="btn btn-danger">Cancel</a>

</form>

</div>

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

</body>

</html>

App-routing:

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

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

import { CoursesComponent } from './course/courses.component';

import { AddcourseComponent } from './addcourse/addcourse.component';

import { EditcourseComponent } from './editcourse/editcourse.component';

const routes: Routes = [

// Added add course path

{path: 'addcourses', component: AddcourseComponent },

{path: 'editcourses/:courseId', component: EditcourseComponent },

{path: 'courses', component: CoursesComponent},

{path: '', redirectTo: '/courses', pathMatch: 'full'},

]

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }