Єсьман Віктор Святославович КІТ-119б

***Лабораторна робота №8***

Робота з ASP.NET Core MVC

# Розробник

Інформація про розробника:

• Єсьман Віктор Святославович;

* КІТ-119Б;

ХІД РОБОТИ

У папці для зберігання моделей створимо новий клас Student, що представляє дані про студента. Обираємо MS SQL Server як сховище для цієї моделі. Для взаємодії з базою даних створюємо клас Context, який зберігає у папці моделей. Щоб підключатися до бази даних, нам треба задати параметри підключення. Для цього змінимо файл appsettings.json. Далі змінюємо файл Startup.cs. У ньому нам треба змінити метод ConfigureServices ().

Щоб у нас в базі даних вже були початкові дані під час запуску програми, нам потрібен якийсь клас. який би виконував роль ініціалізатора бази даних. Для цього додамо в проект новий клас, який назвемо SampleData. анний клас визначає один статичний метод Initialize (), в якому відбувається додавання трьох початкових елементів - об'єктів Students. Для додавання об'єктів в бд в метод Initialize передається контекст даних. І якщо дані в таблиці Students в бд відсутні (if (! Context.Students.Any ())), то додаються три об'єкти.

Щоб ініціалізатор бази даних викликався при старті програми, змінимо клас Program. Для виводу на веб-сторінку даних ми перейдемо до папки Controllers, де знаходиться єдиний контролер HomeController та змінимо його. По-перше, тут видалені всі непотрібні нам методи - все крім методу Index, який буде використовуватися для передачі користувачу даних про студентів. По-друге, доданий конструктор, в якому отримуємо контекст даних. Тут застосовується вбудований механізм впровадження залежностей.

Метод Index буде використовувати уявлення Index.cshtml. Відкриємо це представлення і змінимо його код.

Тепер запустимо додаток на виконання. За замовчуванням, як і визначено вище маршрутом default, спрацює звернення до методу Index контролера

Home, який поверне користувачеві сторінку з даними.

ТЕКСТ ПРОГРАМИ

# Student.cs

using System;

namespace WebApplication2

{

public class Student

{ public int Id { get; set; } public string Name { get; set; } public string Lastname { get; set; } public string Patronym { get; set; } public DateTime Birthday { get; set; } public DateTime Date { get; set; } public char Index { get; set; } public string Faculty { get; set; } public int Specialty { get; set; } public int Progress { get; set; }

public Student() { }

public Student(string name, string lastname, string patronym, DateTime birthday,

DateTime date, char index, string faculty, int specialty, int progress)

{

this.Name = name; this.Lastname = lastname; this.Patronym = patronym; this.Birthday = birthday; this.Date = date; this.Index = index; this.Faculty = faculty; this.Specialty = specialty; this.Progress = progress;

} public override string ToString()

{ return $”{Name}\t {Lastname}\t {Patronym}\t {Birthday}\t {Date}\t {Index}\t

{Faculty}\t {Specialty}\t {Progress} \n”;

}

}

}

# Context.cs

using Microsoft.EntityFrameworkCore;

namespace WebApplication2

{

public class Context : DbContext

{ public DbSet<Student> Students { get; set; } public Context(DbContextOptions<Context> options)

: base(options)

{

Database.EnsureCreated();

}

} }

# appsettings.json

{

“ConnectionStrings”: { “DefaultConnection”:

“Server=(localdb)[\\mssqllocaldb;](file://mssqllocaldb/)Database=studentsdb;Trusted\_Connection=True;MultipleActiv eResultSets=true” },

“Logging”: {

“LogLevel”: {

“Default”: “Information”,

“Microsoft”: “Warning”,

“Microsoft.Hosting.Lifetime”: “Information”

}

},

“AllowedHosts”: “\*”

}

# Startup.cs

using Microsoft.AspNetCore.Builder; using Microsoft.AspNetCore.Hosting; using Microsoft.Extensions.Configuration; using Microsoft.Extensions.DependencyInjection; using Microsoft.Extensions.Hosting; using WebApplication2; using Microsoft.EntityFrameworkCore;

namespace vyesman8

{ public class Startup

{ public Startup(Iconfiguration configuration)

{

Configuration = configuration;

}

public Iconfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

Public void ConfigureServices(IserviceCollection services) {

string connection = Configuration.GetConnectionString(“DefaultConnection”); services.AddDbContext<Context>(options => options.UseSqlServer(connection)); services.AddControllersWithViews();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

Public void Configure(IapplicationBuilder app, IwebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

} else {

app.UseExceptionHandler(“/Home/Error”);

// The default HSTS value is 30 days. You may want to change this for production scenarios, see [https://aka.](https://aka/)ms/aspnetcore-hsts.

App.UseHsts();

}

app.UseHttpsRedirection(); app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute( name: “default”,

pattern: “{controller=Home}/{action=Index}/{id?}”);

});

}

} }

# SampleData.cs

using System;

using System.Collections.Generic; using System.Linq; using System.Threading.Tasks; using WebApplication2;

namespace vyesman8

{ public class SampleData

{

public static void Initialize(Context context)

{ if (!context.Students.Any())

{

context.Students.AddRange( new Student

{

Name = “Марина”,

Lastname = “Сергієнко”,

Patronym = “Володимирівна”,

Birthday = new DateTime(2000,07, 12),

Date = new DateTime(2017, 08, 22),

Index = ‘а’,

Faculty = “КІТ”,

Specialty = 123,

Progress = 95

}, new Student

{

Name = “Олег”,

Lastname = “Ткаченко”,

Patronym = “Сергійович”,

Birthday = new DateTime(2002, 11, 14),

Date = new DateTime(2019, 09, 3),

Index = ‘в’,

Faculty = “КІТ”,

Specialty = 121,

Progress = 93

}, new Student

{

Name = “Ольга”,

Lastname = “Кравченко”,

Patronym = “Олександрівна”,

Birthday = new DateTime(2001, 2, 20),

Date = new DateTime(2018, 09, 11),

Index = ‘г’,

Faculty = “СГТ”,

Specialty = 93,

Progress = 97

} );

context.SaveChanges();

}

}

} }

# Program.cs

using System;

using Microsoft.AspNetCore.Hosting; using Microsoft.Extensions.DependencyInjection; using Microsoft.Extensions.Hosting; using Microsoft.Extensions.Logging; using WebApplication2;

namespace WebApplication2

{ public class Program

{ public static void Main(string[] args)

{

var host = CreateHostBuilder(args).Build(); using (var scope = host.Services.CreateScope())

{ var services = scope.ServiceProvider;

try {

var context = services.GetRequiredService<Context>();

SampleData.Initialize(context);

}

catch (Exception ex)

{

var logger = services.GetRequiredService<Ilogger<Program>>(); logger.LogError(ex, “An error occurred seeding the DB.”); }

}

host.Run();

}

public static IhostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

} }

# HomeController.cs

using System.Linq; using Microsoft.AspNetCore.Mvc; using WebApplication2;

namespace WebApplication2

{

public class HomeController : Controller

{

Context db;

public HomeController(Context context)

{

db = context;

}

public IactionResult Index()

{

return View(db.Students.ToList());

}

} }

# Index.cshtml

|  |  |
| --- | --- |
| @model | |
| @{ |  |

Ienumerable< WebApplication2>

//ViewData[“Title”] = “Список студентів”;

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<title>Список студентів</title>

</head>

<body>

<h3>Студенти</h3>

<table>

<tr>

<td>Ім’я</td>

<td>Прізвище</td>

<td>По-батькові</td>

<td>Дата народження</td>

<td>Дата вступу</td>

<td>Індекс групи</td>

<td>Факультет</td>

<td>Спеціальність</td>

<td>Успішність</td>

<td></td>

</tr>

@foreach (var student in Model)

{

<tr>

<td>@student.Name</td>

<td>@student.Lastname</td>

<td>@student.Patronym</td>

<td>@student.Birthday</td>

<td>@student.Date</td>

<td>@student.Index</td>

<td>@student.Faculty</td>

<td>@student.Specialty</td>

<td>@student.Progress</td>

</tr>

}

</table>

</body>

</html>

РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

Результат роботи програми показано на рис. 1.

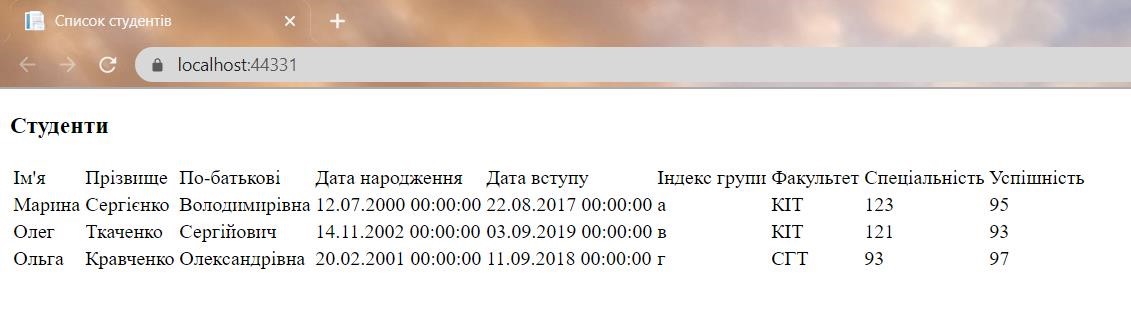


Рисунок 1 – Результат роботи програми

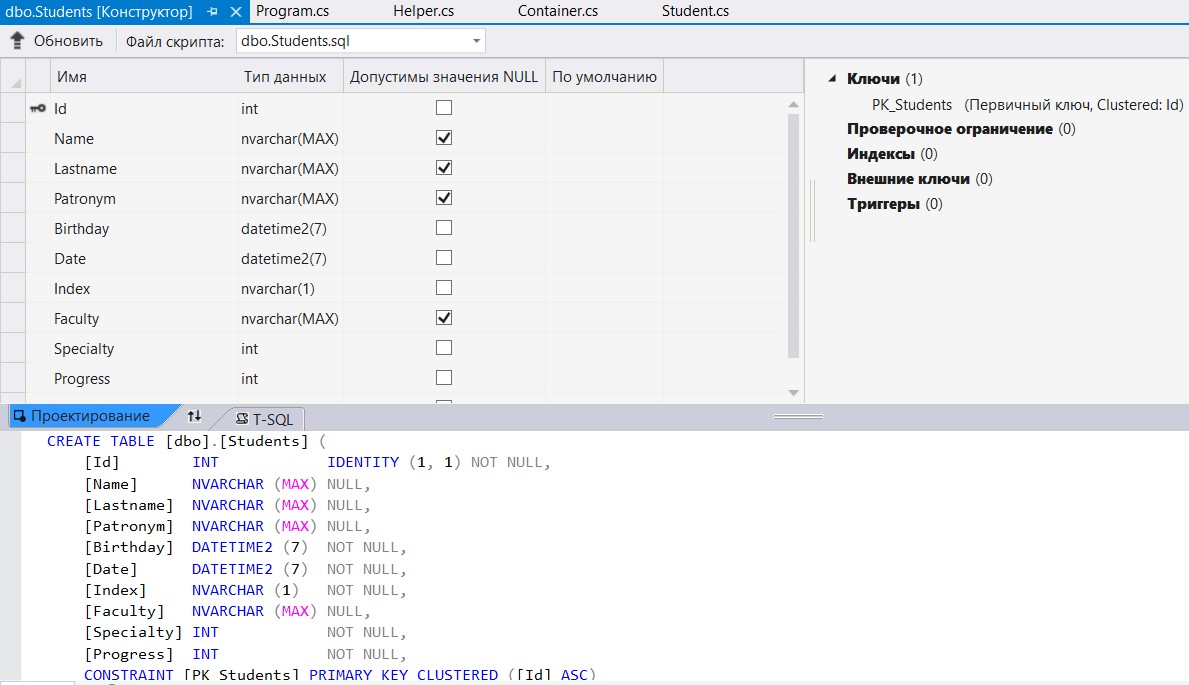


Рисунок 2 – Таблиця у базі даних

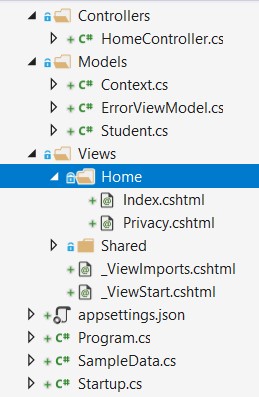


Рисунок 3 – Структура проекту

ВИСНОВОК

Під час виконання даної лабораторної роботи я навчився працювати з ASP.NET Core MVC та дізнався про паттерн MVC(model-view-controller).