НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«DotNet»

*Звiт з лабораторної роботи №9*

*Тема: «ASP.NET. MVC»*

Виконав:

ст. гр. КІТ-119а

Капелька Я. І.

Перевірив:

Бартош М. В.

Харків – 2021

**Тема:** ASP.NET. MVC

**Мета**: Розробка застосунку на фреймворку ASP.NET з використанням MVC.

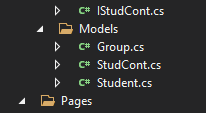
**Розробник**: Капелька Ярослав, КІТ-119а, варіант №11.

**Опис програми:**

**Засоби ООП**: клас, метод класу, поле класу.

**Ієрархія та структура класів:**

До проекту додано попередньо розроблені класи.



**Важливі фрагменти програми:**

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Students

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

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

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.AspNetCore.Http;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Students.Data.Models;

using Students.Data.Interfaces;

using Microsoft.AspNetCore.Mvc;

namespace Students

{

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)

{

services.AddTransient<IStudCont, StudCont>();

services.AddMvc(options => options.EnableEndpointRouting = false) ;

services.AddRazorPages();

}

// 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("/Error");

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

app.UseHsts();

}

app.UseStatusCodePages();

app.UseStaticFiles();

app.UseMvcWithDefaultRoute();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Students.Data.Models;

using Microsoft.AspNetCore.Mvc;

using Students.Data.Interfaces;

namespace Students.Controllers

{

public class StudContController : Controller

{

private IStudCont cont;

public StudContController(IStudCont cont)

{

this.cont = cont;

}

public ViewResult List()

{

cont = new StudCont();

cont.Add(new Student("Luhoviy Oleksander Evgeniyovich", new DateTime(2002, 4, 2), new DateTime(2019, 9, 1), "B", 119, "CIT", "Videogames", 95));

cont.Add(new Student("Orlova Vlada Stanislavovna", new DateTime(2001, 12, 8), new DateTime(2019, 9, 1), "B", 119, "CIT", "Videogames", 94));

cont.Add(new Student("Myhlin Andrew Aleksandrovich", new DateTime(2002, 3, 31), new DateTime(2019, 9, 1), "B", 119, "CIT", "Videogames", 67));

return View(cont.toList());

}

}

}

using Students.Data.Models;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Students.Data.Interfaces

{

public interface IStudCont

{

public void Add(Student stud)

{

throw new NotImplementedException();

}

public IEnumerable<Student> toList()

{

throw new NotImplementedException();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Students.Data.Models

{

public class Group

{

public readonly int \_groupNum;

public readonly string \_faculty;

public readonly string \_groupIndex;

public readonly string[] faculties = { "Э", "МИТ", "И", "ХТ", "БЭМ", "МО", "СГТ", "КН", "КИТ" , "CIT" };

public string FullName

{

get

{

return $"{\_faculty}-{\_groupNum}{\_groupIndex}";

}

}

public Group(int groupNum, string faculty, string groupIndex)

{

if (faculties.Contains(faculty.ToUpper()))

{

\_groupIndex = groupIndex;

\_faculty = faculty;

\_groupNum = groupNum;

}

else throw new Exception("faculty value is not in facultie names list");

}

public override bool Equals(object obj)

{

if (obj.GetType() == this.GetType())

{

Group other = (Group)obj;

return other.FullName == FullName;

}

return false;

}

public override string ToString()

{

return $"{\_faculty};{\_groupNum};{\_groupIndex}";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

using Students.Data.Interfaces;

namespace Students.Data.Models

{

public class StudCont : IStudCont, IEnumerable, IEnumerator

{

private List<Student> list;

private int position = -1;

public enum SortType

{

Name,

Age,

Group,

Year,

Performance

}

public StudCont()

{

list = new List<Student>();

}

public StudCont(List<Student> studList)

{

list = studList;

}

public void Add(Student stud)

{

list.Add(stud);

}

public bool Delete(int index)

{

if (index < list.Count && index >= 0)

{

list.RemoveAt(index);

return true;

}

return false;

}

public void PrintAll(bool brief=true)

{

int i = 1;

IEnumerable<string> query =

from Student stud in list

select brief? stud.Name : stud.ToText();

foreach(string line in query)

{

//if (!brief) Console.WriteLine(i + "." + stud.ToText() + "\n");

//else Console.WriteLine(i + "." + stud.Name + "\n");

Console.WriteLine(i + "." + line + "\n");

i++;

}

}

public Student GetStudent(int index)

{

//if(index<list.Count && index >= 0)

//{

// return list.ElementAt<Student>(index);

//}

//else throw new IndexOutOfRangeException();

IEnumerable<Student> query =

from Student stud in list

where list.IndexOf(stud) == index

select stud;

return query.First();

}

public Student GetStudent(string name)

{

IEnumerable<Student> query =

from Student stud in list

where stud.Name == name

select stud;

return query.First();

}

public bool ChangeStudent(int index, Student stud)

{

if (index < list.Count && index >= 0)

{

list[index] = stud;

return true;

}

else return false;

}

public bool ChangeStudent(string name, Student stud)

{

int i = 0;

foreach(Student stu in list)

{

if (stu.Name == name)

{

list[i] = stud;

return true;

}

i++;

}

return false;

}

public int Count()

{

return list.Count;

}

public IEnumerator GetEnumerator()

{

return (IEnumerator)this;

}

//IEnumerator

public bool MoveNext()

{

position++;

return (position < list.Count());

}

//IEnumerable

public void Reset()

{

position = -1;

}

//IEnumerable

public object Current

{

get { return list.ElementAt<Student>(position); }

}

public IEnumerable<Student> toList()

{

return list;

}

public void RemoveByComp(Student.Compare comp, string line)

{

//foreach(Student stud in list)

//{

// if(comp(stud, line))

// {

// list.Remove(stud);

// }

//}

IEnumerable<Student> query =

from Student stud in list

where comp(stud, line)

select stud;

foreach(Student stud in query)

{

list.Remove(stud);

}

}

public void Sort(SortType type)

{

IOrderedEnumerable<Student> result;

switch (type)

{

case SortType.Name:

result = from s in list

orderby s.Name

select s;

list = result.ToList();

break;

case SortType.Age:

result = from s in list

orderby s.Age

select s;

list = result.ToList();

break;

case SortType.Performance:

result = from s in list

orderby s.Performance

select s;

list = result.ToList();

break;

case SortType.Group:

result = from s in list

orderby s.\_group

select s;

list = result.ToList();

break;

case SortType.Year:

result = from s in list

orderby s.Year

select s;

list = result.ToList();

break;

}

}

public void Unite(StudCont cont)

{

var res = from Student s in cont

select s;

foreach(Student s in res)

{

Add(s);

}

}

}

}

using System;

using System.Linq;

using System.Collections;

using System.Text;

namespace Students.Data.Models

{

public class Student

{

private string \_name;

private DateTime \_dateOfBirth;

private DateTime \_dateOfAdmission;

private string \_specialty;

private int \_performance;

public Group \_group;

private string[] \_faculties = { "Э", "МИТ", "И", "ХТ", "БЭМ", "МО", "СГТ", "КН", "КИТ" , "CIT" };

public delegate bool Compare(Student stud, string line);

public static bool CompareGroup(Student stud, string line)

{

if (stud.\_group.FullName == line) return true;

else return false;

}

public static bool CompareSpecialty(Student stud, string line)

{

if (stud.Specialty == line) return true;

else return false;

}

public static bool CompareFaculty(Student stud, string line)

{

if (stud.\_group.\_faculty == line) return true;

else return false;

}

public Student(string name, DateTime dateOfBirth, DateTime dateOfAdmission, string groupIndex, int groupNum, string faculty, string specialty, int performance)

{

\_name = name ?? throw new ArgumentNullException(nameof(name));

\_dateOfBirth = dateOfBirth;

\_dateOfAdmission = dateOfAdmission;

\_group = new Group(groupNum, faculty, groupIndex);

\_specialty = specialty ?? throw new ArgumentNullException(nameof(specialty));

\_performance = performance;

}

public string Name { get { return \_name; } }

public DateTime DateOfBirth { get { return \_dateOfBirth; } }

public DateTime DateOfAdmission { get { return \_dateOfAdmission; } }

public int Year

{

get

{

int year = (int)((DateTime.Now - \_dateOfAdmission).TotalDays/365.2425) + 1;

if (year > 6) year = 6;

else if (year < 0)

{

year = 0;

}

return year;

}

}

public int Semester

{

get

{

int day = (int)((DateTime.Now - \_dateOfAdmission).TotalDays % 365.2425);

if (day < 150)

{

return 1;

}

else return 2;

}

}

public int Age

{

get

{

return (int)((DateTime.Now - \_dateOfBirth).TotalDays / 365.2425);

}

}

public string Specialty { get { return \_specialty; } private set { \_specialty = value; } }

public int Performance {

get { return \_performance; }

set {

if (value <= 100 && value >= 0)

{

\_performance = value;

}

}

}

public override string ToString()

{

string output = $"{\_name};{\_dateOfBirth.ToString()};{\_dateOfAdmission};{\_group.ToString()};{\_specialty};{\_performance}";

return output;

}

public static Student ParseString(string line)

{

string[] arr = line.Split(";");

Student stud = new Student(arr[0], DateTime.Parse(arr[1]), DateTime.Parse(arr[2]), arr[5], Int32.Parse(arr[4]), arr[3], arr[6], Int32.Parse(arr[7]));

return stud;

}

public string ToText()

{

string output = $"Ф.И.О.: {\_name}\nДата рождения: {\_dateOfBirth.ToString()}\nДата поступления: {\_dateOfAdmission}\nГруппа: {\_group.FullName}\nСпециальность: {\_specialty}\nУспеваемость: {\_performance}%";

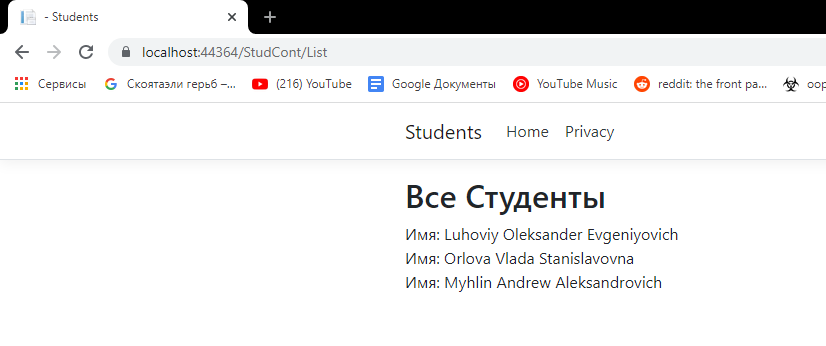
return output;

}

}

}

**Результат роботи програми:**



**Висновок:**Було розроблено програму на платформі ASP.NET з використанням фреймворку MVC.