Tasks

1. делегаты/события +
2. дженерики +
3. сериализация +
4. стэкфрейм
5. интерфейсы +
6. yield+
7. Напишите программу, в которой определено событие и выполнена подписка на него. Продемонстрируйте смысл событий.

namespace Ex1

{

class Publisher

{

public event EventHandler<string> Notify;

public void DoSomething()

=> Notify.Invoke(this, "Some happened in Publisher");

}

}

namespace Ex1

{

class Subscriber

{

public void Subscribe(ref Publisher publisher) =>

publisher.Notify += Send;

public void Unsubscribe(ref Publisher publisher) =>

publisher.Notify += Send;

public void Send(object sender, string message) =>

Console.WriteLine($"Отправить сообщение от подписчика: {message}");

}

}

namespace Ex1

{

class Program

{

//1) Напишите программу,

//в которой определено событие и выполнена подписка на него.Продемонстрируйте смысл событий.

static void Main(string[] argv)

{

Publisher publisher = new Publisher();

Subscriber subscriber = new Subscriber();

Subscriber subscriber1 = new Subscriber();

subscriber.Subscribe(ref publisher);

subscriber1.Subscribe(ref publisher);

publisher.DoSomething();

subscriber1.Unsubscribe(ref publisher);

publisher.DoSomething();

}

}

}

1. Напишите метод-расширения для десериализации generic – типа +-
2. Напишите метод-расширения для сериализации generic-типа
3. Напишите программу, получающую доступ ко всем кадрам стека и выводящие их содержимое на консоль +-
4. Напишите программу, содержащий собственный generic-тип и пример его использования +

class GenericClass<T>

{

public T Value { get; set; }

public GenericClass(T Value)

{

this.Value = Value;

}

}

class Program

{

static void Main(string[] args)

{

GenericClass<int> first = new GenericClass<int>(1);

int b = 1;

int c = first.Value + b;

GenericClass<string> second = new GenericClass<string>("Няя");

string str = "Не";

//if (second is string)

//{

Console.WriteLine($"{second.Value + str}");

//}

}

}

1. Напишите программу, содержащую пример применения итератора yield +

namespace Ex6

{

class Program

{

static void Main(string[] argv)

{

Console.WriteLine("Hello, World!");

foreach(int num in GetNums())

{

Console.WriteLine(num);

}

}

public static IEnumerable<int> GetNums()

{

for(int i = 0; i < 10; i++)

{

yield return i;

}

}

}

}

1. Напишите программу, считывающую с клавиатуры цифры и сохраняющие их в файл. Продемонстрируйте чтение из этого файла.

using System;

using System.IO;

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введи набор символов");

string input = "was ist\ndas"; // Ваш ввод

string path = "data.txt";

// Запись

using (StreamWriter sw = new StreamWriter(path))

{

sw.Write(input);

}

// Чтение из файла

using (StreamReader sr = new StreamReader(path))

{

string line = sr.ReadToEnd();

Console.WriteLine("Прочитано из файла:");

Console.WriteLine(line);

}

Console.ReadLine();

}

}

1. Напишите программу, реализующую шаблон проектирования «cтратегия” (конкретный класс может не знать реализацию методов других классов, которые использует через интерфейс) +

namespace Ex8

{

interface ICar

{

public void Move();

}

}

namespace Ex8

{

class DieselCar : ICar

{

public void Move()

{

Console.WriteLine("Движется дизельная машина");

}

}

}

namespace Ex8

{

class ElectroCar : ICar

{

public void Move()

{

Console.WriteLine("Движется Электромобиль");

}

}

}

namespace Ex8

{

class Car

{

public string Name { get; set; }

public Car(string Name) => this.Name = Name;

public ICar TypeCar { get; set; }

public void Move()

{

TypeCar.Move();

}

}

}

namespace Ex8

{

class Program

{

static void Main(string[] args)

{

Car car = new Car("Volkswagen");

car.TypeCar = new DieselCar();

car.TypeCar.Move();

}

}

}

1. Напишите программу, реализующую шаблон проектирования «наблюдатель” +

namespace Ex9

{

interface IObservable

{

public void Subscribe(IObserver observer);

public void UnSubscribe(IObserver observer);

public void Notify(double Price);

}

}

namespace Ex9

{

interface IObserver

{

public void Update(double Price);

}

}

namespace Ex9

{

//Subscriber

class Observable : IObservable

{

//список издательств

//подписчик подписывается на разные издательства

private List<IObserver> observers = new List<IObserver>();

public void Subscribe(IObserver obs)

{

Console.WriteLine($"подпика на издательство ");

observers.Add(obs);

}

public void UnSubscribe(IObserver obs)

{

observers.Remove(obs);

}

public void Notify(double Price)

{

foreach (var obs in observers)

{

obs.Update(Price);

}

}

}

}

namespace Ex9

{

//Publisher

class Observer : IObserver

{

private double Price {get; set; }

public Observer(double Price)

{

this.Price = Price;

Console.WriteLine($"Цена на книгу изначально = {this.Price}");

}

public void Update(double Price)

{

this.Price = Price;

Console.WriteLine($"Цена на книгу обновилась = {this.Price}");

}

}

}

namespace Ex9

{

class Program

{

static void Main(string[] argv)

{

Observer Publisher = new Observer(1488);

Observable Subscriber0 = new Observable();

Observable Subscriber1 = new Observable();

// Subscribe the observer to the observable

Subscriber0.Subscribe(Publisher);

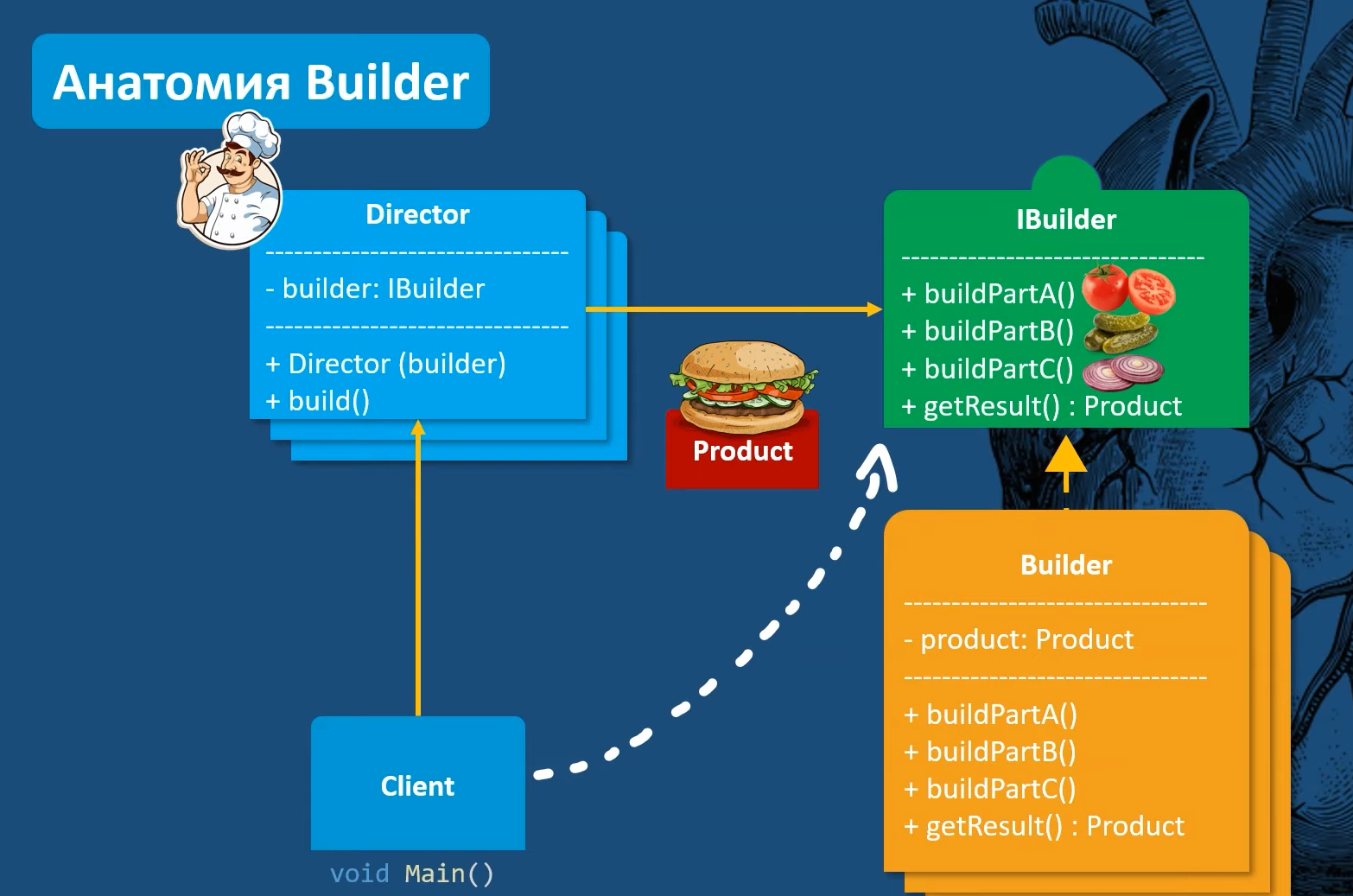
Subscriber1.Subscribe(Publisher);

}

}

}

1. Напишите программу, реализующую шаблон проектирования «builder”



1. Напишите программу, реализующую декларативную атрибутивную валидацию с использованием рефлексии.

using System;

using System.ComponentModel.DataAnnotations;

namespace Validation

{

public class Person

{

[Range(1, 5, ErrorMessage ="Ошибка: Интервал должен быть 1..5")]

public int Id { get; set; }

[Required(ErrorMessage = "Имя обязательно к заполнению")]

public string Name { get; set; }

public Person(int Id, string Name)

{

this.Id = Id;

this.Name = Name;

}

}

}

using System;

using System.ComponentModel.DataAnnotations;

using System.Reflection;

namespace Validation

{

class Program

{

static void Main(string[] args)

{

Person person = new Person(0, "Consta");

var validationContext = new ValidationContext(person);

var validationResults = new List<ValidationResult>();

var isValid = Validator.TryValidateObject(person, validationContext,

validationResults,

validateAllProperties: true);

if (!isValid)

{

foreach (var validation in validationResults)

{

Console.WriteLine(validation.ErrorMessage);

}

}

Type type = typeof(Person);

foreach(PropertyInfo property in type.GetProperties())

{

Console.WriteLine(property.Name, property.PropertyType);

}

}

}

}

1. Напишите программу с плагинной архитектурой.

namespace PlaguinApi

{

public interface IPlugin

{

void Execute();

}

}

namespace PlaguinApi

{

public class Plugin : IPlugin

{

public void Execute()

{

Console.WriteLine("Plugin 1 is executed.");

}

}

}

class Program

{

static void Main()

{

List<IPlugin> plugins = LoadPlugins();

foreach (var plugin in plugins)

{

plugin.Execute();

}

}

static List<IPlugin> LoadPlugins()

{

List<IPlugin> plugins = new List<IPlugin>();

// Создайте экземпляры плагинов и добавьте их в список.

plugins.Add(new Plugin());

return plugins;

}

}

1. Напишите программу, использующие флаговые перечисления. +

namespace Ex7

{

[Flags]

enum Days

{

Monday = 1,

Tuesday = 2,

Wednesday = 3,

Thursday = 4,

Friday = 5,

Saturday = 6,

Sunday = 7

}

}

class Program

{

static void Main(string[] args)

{

Days days = Days.Monday | Days.Tuesday;

Console.WriteLine("Рабочие дни: " + days);

if (days == Days.Wednesday)

{

Console.WriteLine("Среда - рабочий день");

}

//добавление

days = days | Days.Saturday;

Console.WriteLine("День недели " + days);

//удаление

days = days & ~Days.Saturday;

Console.WriteLine("День недели " + days);

}

}

Напишите программу, использующую yield return. +

namespace Ex6

{

class Program

{

static void Main(string[] argv)

{

Console.WriteLine("Hello, World!");

foreach(int num in GetNums())

{

Console.WriteLine(num);

}

}

public static IEnumerable<int> GetNums()

{

for(int i = 0; i < 10; i++)

{

yield return i;

}

}

}

}

1. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения списка курсов валют, хранящегося в xml-файле в файловой системе. Результат должен отдаваться на клиент как json. Путь к файлу должен быть указан в конфигурации appSettings.json
2. REST-API = ASP.NET (Шаблон WebAPI)
3. Swagger =  Swashbuckle .NET Core  (не менять)
4. Xml, json
5. Нужно поменять только класс с данными, файл контроллера и добавить Path в appsettings.json

using Microsoft.AspNetCore.Mvc;

using System.IO;

using System.Text.Json;

namespace Ex14.Controllers

{

[Route("[controller]")]

[ApiController]

public class DataController : ControllerBase

{

private List<Data> \_datas; //список рабочих данных

private string \_pathToJSON; //файл с данными Data

public DataController()

{

//можно заменить на путь к json файлу

// string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "appsettings.json");

//var config = new ConfigurationBuilder().AddJsonFile(filePath).Build();

//\_pathToJSON = config["Path"];

\_pathToJSON = "data.json";

//файл существует

if(System.IO.File.Exists(\_pathToJSON))

{

string jsonString = System.IO.File.ReadAllText(\_pathToJSON);

List<Data> datas = JsonSerializer.Deserialize<List<Data>>(jsonString);

if (datas != null)

\_datas = datas;

else

Console.WriteLine("File with data is null");

}

else

Console.WriteLine("File not exist");

}

/// <summary>

/// Get All Elements

/// </summary>

/// <returns></returns>

[HttpGet, Route("datas")]

public List<Data> GetDatas()

{

return \_datas;

}

/// <summary>

/// Get element from Id

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

/// <exception cref="Exception"></exception>

[HttpGet, Route("datas/{id}")]

public Data GetDataById(int id)

{

var data = \_datas.FirstOrDefault(x => x.Id == id);

if(data == null)

throw new Exception("data with this Id is null");

return data;

}

/// <summary>

/// Add element in Data

/// </summary>

/// <param name="id"></param>

/// <param name="name"></param>

/// <param name="exchangeRates"></param>ze

[HttpPost, Route("AddDatas/{Name}")]

public void AddData(int id, string name, double exchangeRates)

{

Data data = new Data { Id = id, Name = name, ExchangeRates = exchangeRates };

\_datas.Add(data);

string JSONString = JsonSerializer.Serialize(\_datas);

System.IO.File.WriteAllText(\_pathToJSON, JSONString);

}

}

}

appsettings.json

{

"Path": "data.json",

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

data.json

[{"Id":1,"Name":"Rouble","ExchangeRates":1.488},{"Id":2,"Name":"Euro","ExchangeRates":6.66},{"Id":3,"Name":"{Name}","ExchangeRates":2.12},{"Id":3,"Name":"{Name}","ExchangeRates":2.12},{"Id":4,"Name":"{Name}","ExchangeRates":2.12},{"Id":6,"Name":"{Name}","ExchangeRates":45.6},{"Id":6,"Name":"{Name}","ExchangeRates":45.6}]

Program

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

1. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения списка студентов, хранящегося в json-файле в файловой системе. Путь к файлу должен быть указан в конфигурации appSettings.json.
2. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения списка стран, хранящегося в json-файле в файловой системе. Путь к файлу должен быть указан в конфигурации appSettings.json. Названия стран должны быть представлены на разных языках.
3. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения списка курсов образовательной организации, хранящегося в json-файле в файловой системе. Путь к файлу должен быть указан в конфигурации appSettings.json.
4. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения списка строк, хранящегося в xml-файле в файловой системе. Путь к файлу должен быть указан в конфигурации appSettings.json.

<?xml version="1.0" encoding="utf-8"?>

<Root>

<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<Id>1</Id>

<Name>Rouble</Name>

<ExchangeRates>1.488</ExchangeRates>

</Data>

<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<Id>2</Id>

<Name>Euro</Name>

<ExchangeRates>6.66</ExchangeRates>

</Data>

</Root>

using System.Xml.Serialization;

namespace Ex18

{

[XmlRoot("Root")]

public class Root

{

[XmlElement("Data")]

public List<Data> Datas { get; set; }

}

public class Data

{

public int Id { get; set; }

public string? Name { get; set; }

public double? ExchangeRates { get; set; }

public Data() { }

}

}

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System.IO;

using System.Xml.Serialization;

namespace Ex18.Controllers

{

[Route("[controller]")]

[ApiController]

public class DataController : ControllerBase

{

private List<Data> \_datas;

private string PathXML;

public DataController()

{

PathXML = "data.xml";

XmlSerializer xmlSerializer = new XmlSerializer(typeof(Root));

using (FileStream fs = new FileStream(PathXML, FileMode.OpenOrCreate))

{

Root root = (Root)xmlSerializer.Deserialize(fs);

\_datas = root.Datas;

}

}

[HttpGet, Route("Datas")]

public List<Data> GetDatas()

{

return \_datas;

}

[HttpGet, Route("Datas/{id}")]

public Data GetDataById(int id)

{

var data = \_datas.FirstOrDefault(x => x.Id == id);

if (data == null)

throw new Exception("Data = null");

return data;

}

[HttpPost, Route("Datas/{Name}")]

public void AddData(int id, string name, double exchangeraites)

{

Data data = new Data { Id = id, Name = name, ExchangeRates= exchangeraites };

\_datas.Add(data);

XmlSerializer serializer = new XmlSerializer(typeof(Data));

using (FileStream fs = new FileStream(PathXML, FileMode.OpenOrCreate))

{

serializer.Serialize(fs, \_datas);

}

}

}

}

1. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения списка доступных для покупки пицц, хранящегося в json-файле в файловой системе. Путь к файлу должен быть указан в конфигурации appSettings.json.
2. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения списка доступных для покупки пицц, хранящегося в базе данных
3. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения сведений о нагрузке на компьютер в виде json- ответа. Метод должен принимать на вход логин и пароль, после чего проводить сверку учетных данных с указанными ранее в конфигурационном файле. Допустимо хранить только хэш от пароля.
4. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения сведений о параметрах подключившегося клиента (User Agent)
5. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения списка доступного съемного жилья по коду страны, переданного в качестве get - параметра, Список хранится в json-файле в файловой системе. Путь к файлу должен быть указан в конфигурации appSettings.json.
6. Напишите программу, предоставляющее REST-API и документацию посредством swagger, для получения списка курсов образовательной, хранящегося в json-файле в файловой системе. Путь к файлу должен быть указан в конфигурации appSettings.json.