Assert.equal w teście jednostkowym – sprawdza czy gdzies cos zwraca

Ctorf – konstruktor

Odwracanie zależności (IOC: )

1. Tworzymy konstruktor w klasie user service
2. Konstruktor przyjmuje jako parametr interfejs

Instancje interfejsu nazywamy od \_

Delegate = interfejs function w java

Action – Consumer

Func – Function (Ostatni parametr definiuje co funkcja zwraca)

LAMBDA = ()=>{}

API

Controller – dziedziczymy po ControllerBase

builder.Services.AddControllers() – dodaje kontrolery do Maina

app.mapcontrollers – mapuje kontrolery

[Route (‘’api/students”)] – przekierowuje żądania kierowane na adres api/stud do tej klasy

[HttpGet] – tutaj sa kierowane żadania

[HttpGet{id:int}] – http z templatka i przekazaniem id (najlepiej pola w kodzie nazywać tak samo)

IActionResult – zwraca jakas wartość

[HttpPut] – usuwa i dodaje na nowo (Podmiana)

Można okreslic czy int x([FromQuery]) czy from body

builder.Services.AddScoped<IStudentService, StudentService>();

transient – zawsze tworzymy nową instancje

scoped – ta sama instancja do wszystkich obiektow

singleton – tylko jeden obiekt

linq – interfejs symulujący sqla na kolekcjach

Przed sql injection:

1

Await tylko w metodach async

Dbtransaction tran = con.begintransactionAsync()

Comand.transaction = (sqltransaction) tran

Metody rozszerzeń:

Klasa static, metoda static z parametrami (this string s)

LINQ:

Linqsamples.com

.Include(e=>e.property) – join

var results = people.Join(countries,

arg => arg.CountryCode, // the key to join from the people list

arg => arg.Code, // the key to join from the country list

(person, country) => new { person.Name, country.CountryName }); // the select of the join

.add – Insert

.savechanges – zapisz zmiany

.single – zwraca błąd gdy więcej niż 1 rekord (alt. FirstorDefault)

Entity framework – ORM

* Microsoft.entityFrameworkCore
* Microsoft.entityFrameworkCore.SqlServer
* Microsoft.entityFrameworkCore.design – zarzadzanie konsolowe

Zmienić folder na folder projektu!

Zmiana wesji:



Generowanie – dotnet ef dbcontext scaffold “Data Source=…” Microsoft.EntityFrameworkCore.SqlServer –output-dir Models –context-dir Context

<optional>

project>edit csproj > Invariant Globalization>False

Entity framework – Usage

Var Job = new job{x=5} – tworzymy nowy obiekt anon

.attach – dodaje obiekt

.entry(obj) – otrzymuje obiekt

A close up of words

Description automatically generated

Entry.property(x=>x.id).ismodified = true

.remove()

Entry.state = entitystate.added

Entr.state = entitystate.removed

CODE FIRST:

Tworzymy public partial klase context w folder Context dziedziczącą po DbContext plus dwa konstruktory

A screenshot of a computer code

Description automatically generated

Public DbSet<Student> Students w apbd context

Dotnet ef migrations add WhatChanged

Dotnet ef database update 0 – wraca do stanu sprzed migracji

Constraintsy:

[ConcurrencyCheck] – public byte[] RowVersion

Najważniejsze komendy:

dotnet new tool-manifest

dotnet tool install dotnet-ef --version 8.0.0

dotnet ef migrations add WhatChanged

dotnet ef database update

Do dodania w Program.cs

builder.Services.AddDbContext<WebAppContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("Default")));