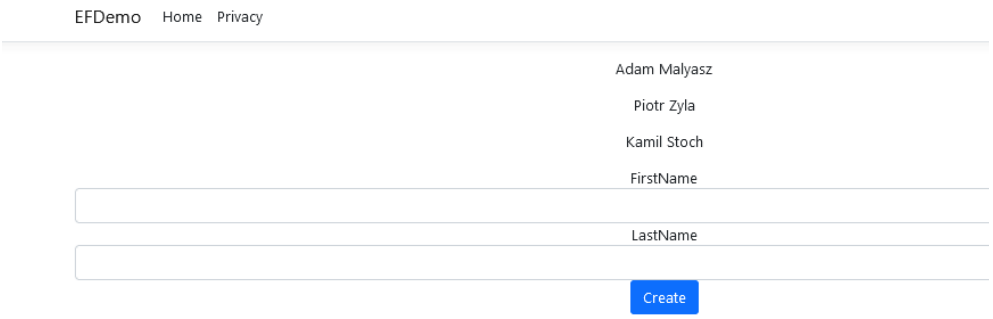
Dostęp do bazy z wykorzystaniem Entity Framework Core cz.1/2

Programowanie aplikacji WWW w technologii .NET, 2021/2022

Przygotowała: I. Kartowicz-Stolarska

CEL

1. Tworzenie lokalnej bazy danych i podłączenia jej do projektu aplikacji WWW.
2. Tworzenie tabel w bazie danych według zasady code-first.
3. Zapytania w składni LINQ
4. Zapis danych do bazy przez formularz



PAKIETY NUGET

- Micorosoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.EntityFrameworkCore.Design

ROZSZERZENIA DO VS CODE

- https://marketplace.visualstudio.com/items?itemName=ms-mssql.mssql

TUTORIAL

Utwórz nowy projekt o nazwie EFDemo.

Instalacja pakietów

1. **Microsoft Visual Studio**
   Z poziomu Projekt -> Zarządzanie projektami NuGet zainstaluj biblioteki Microsoft.EntityFrameworkCore:
   - Microsoft.EntityFrameworkCore
   - Microsoft.EntityFrameworkCore.SqlServer
   - Microsoft.EntityFrameworkCore.Tools
   - Microsoft.EntityFrameworkCore.Design
2. **Visual Studio Code**
   - dotnet add package Microsoft.EntityFrameworkCore
   - dotnet add package Microsoft.EntityFrameworkCore.SqlServer
   - dotnet add package Microsoft.EntityFrameworkCore.Tools
   - dotnet add package Microsoft.EntityFrameworkCore.Design

Metoda Code-first

1. W strukturze projektu utwórz nowy folder Models.
2. Do katalogu Models dodaj klasę
   a. *Person.cs* o następującej strukturze:

```
public class Person {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
    }
```

3. W katalogu projektu utwórz katalog Data z klasą kontekstu dla danych *PeopleContext.cs:*

```
public class PeopleContext : DbContext {
        public DbSet<Person> Person { get; set; }
    }
```

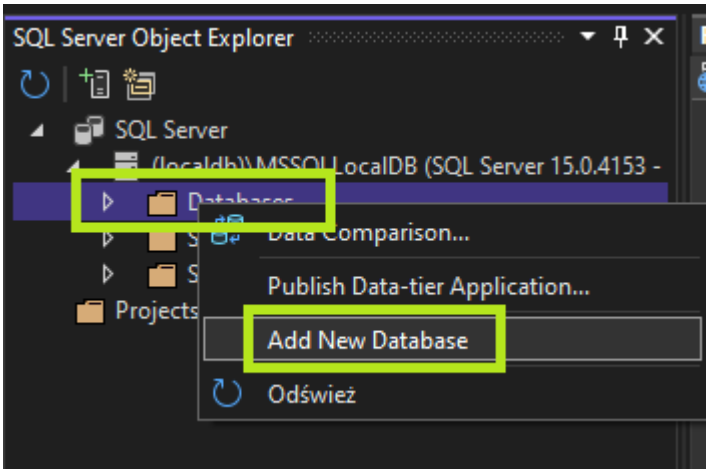
   Dodaj wymagane przestrzenie nazw:

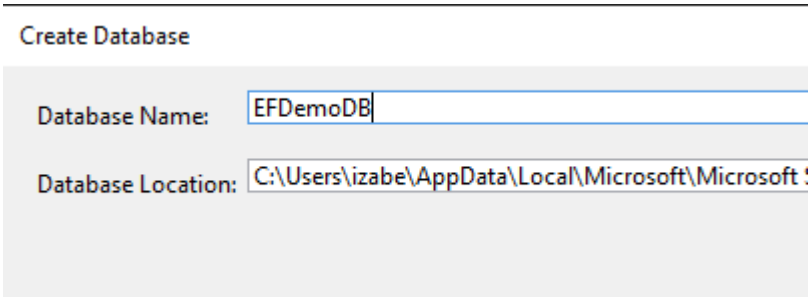
```
using EFDemo.Models;
using Microsoft.EntityFrameworkCore;
```

Tworzenie bazy danych

Domyślna nazwa serwera dla środowiska lokalnego: (localdb)\MSSQLLocalDB

1. Microsoft Visual Studio
   a. Utwórz nową bazę danych o nazwie EFDemoDB (Widok->SQL Server Object Explorer -> Databases -> Add new database)

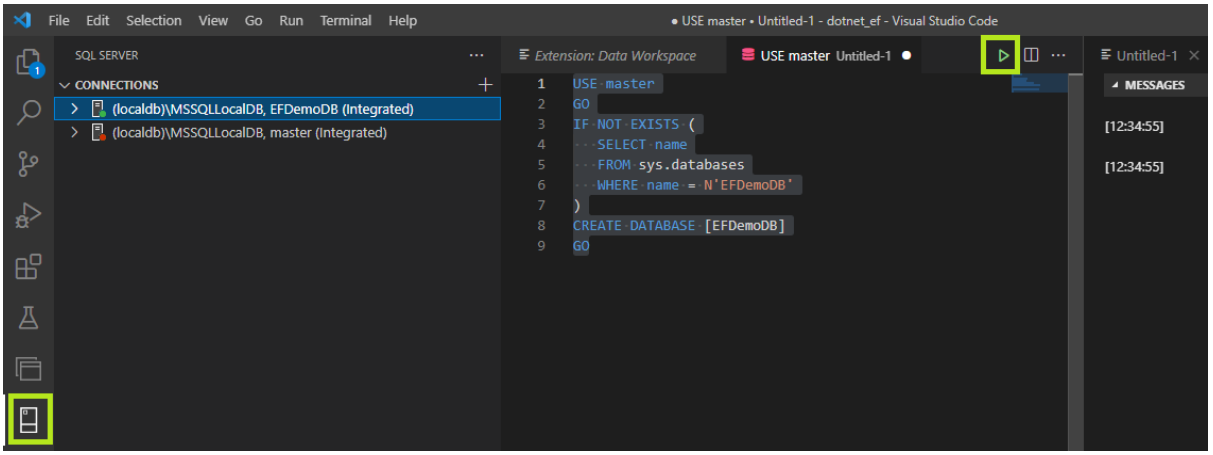




2. Visual Studio Code (2 sposoby)
    a. Sposób 1:
        i. Stworzenie bazy EFDemoDB przez SQL Server Management Studio
        ii. Połączenie do bazy z poziomu VSCode (Ctrl+Shift+P)
            1. Nazwa serwera: (localdb)\MSSQLLocalDB
            2. Nazwa bazy: EFDemoDB
            3. Authentication type: Integrated
    b. Sposób 2:
        i. Wykonanie zapytania:

```
USE master
GO
IF NOT EXISTS (
   SELECT name
   FROM sys.databases
   WHERE name = N'EFDemoDB'
)
CREATE DATABASE [EFDemoDB]
GO
```

z wykorzystaniem T-SQL IntelliSense. Instrukacja https://thinkaboutit.be/2017/10/how-do-i-use-visual-studio-code-instead-of-ssms/



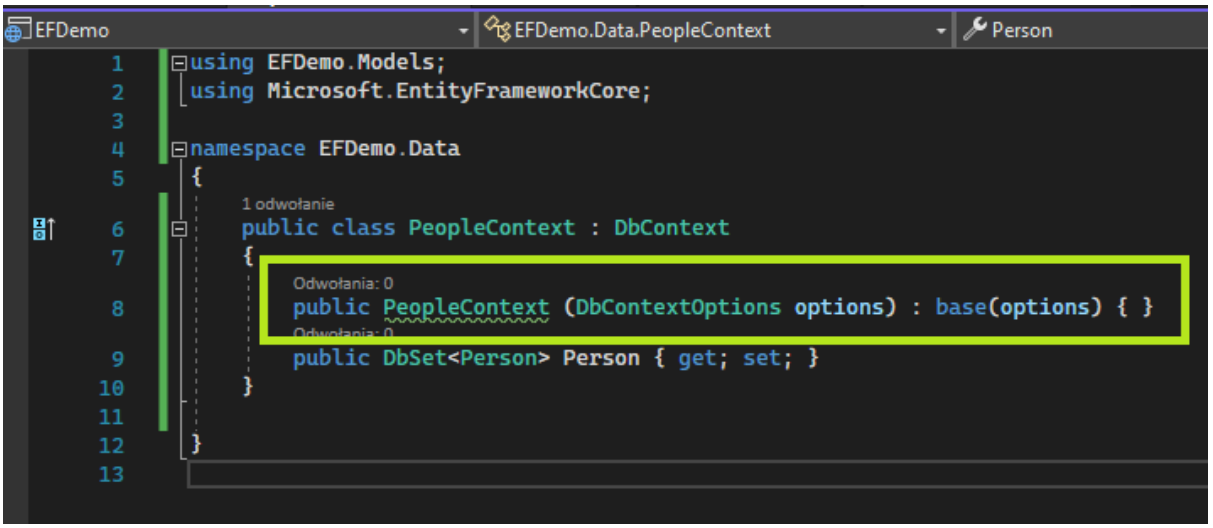
Połączenie do bazy danych

3. Z właściwości bazy danych pobierz wartość ConnectionString i zapisz ją w `appsettings.json` w kluczu “ConnectionStrings” pod nazwą “EFDemoDB”:

```
{
….,
  "ConnectionStrings": {
    "EFDemoDB": "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=EFDemoDB;Integrated Security=True;"
  },
….
}
```

4. W pliku kontekstu danych PeopleContext.cs rozszerz konstruktor klasy o parameter options:

```
public PeopleContext(DbContextOptions options) : base(options) { }
```



5. W pliku Startup.cs w funkcji ConfigureServices skonfiguruj połączenie z bazą EFDemoDB:
    a. wersja .NET 6 - Program.cs

```
….
builder.Services.AddRazorPages();
builder.Services.AddDbContext<PeopleContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("EFDemoDB")));
….
```

    b. wersja niższa niż .NET 6 Startup.cs

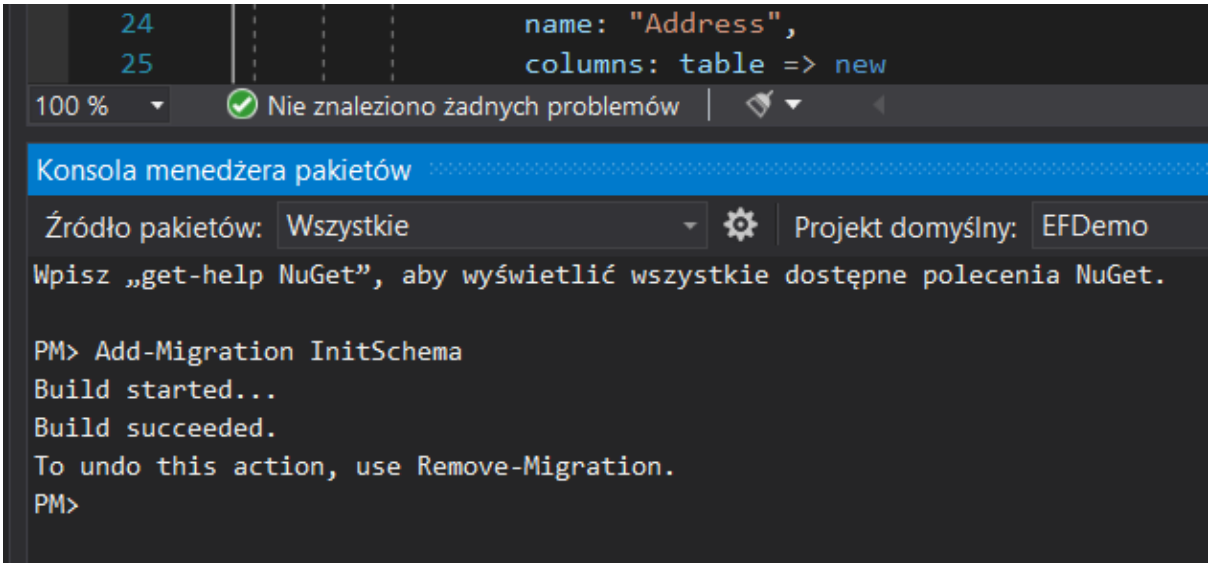
```
public void ConfigureServices(IServiceCollection services)
        {
            services.AddDbContext<PeopleContext>(options => {
options.UseSqlServer(Configuration.GetConnectionString("EFDemoDB"));
            });
            services.AddRazorPages();
        }
```

Dodaj do przestrzeni nazw:

- `using EFDemo.Data;`
- `using Microsoft.EntityFrameworkCore;`

Zarządzanie schematem danych za pomocą migracji

1. Uruchom konsolę menedżera pakietów (Widok->Inne okna->Konsola Menedżera pakietów)
2. W konsoli uruchom komendę tworzącą pierwszą migrację o nazwie InitSchema:
   `Add-Migration InitSchema`

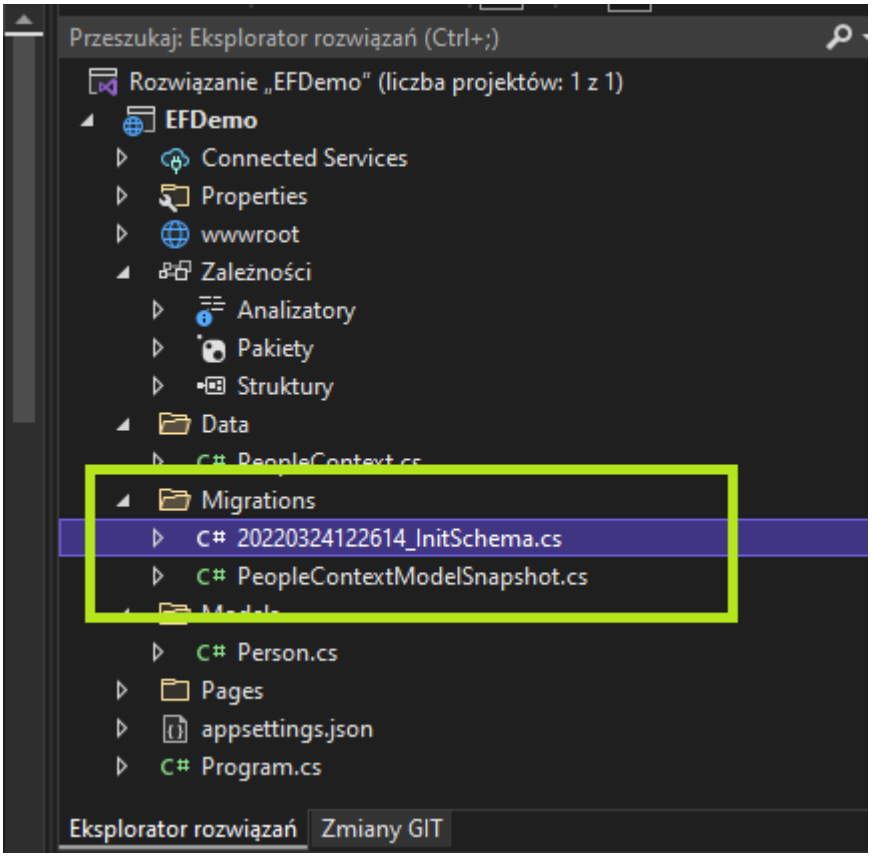


Polecenie Add-Migration tworzy plik ze strukturą zmian bazy danych i zapisuje go pod podaną nazwą w katalogu Migrations. Katalog jest tworzony wraz z uruchomieniem tworzenia pierwszej migracji.

Aby użyć narzędzi EntityFrameworkCore w Visual Studio Code lub w konsoli należy w konsoli uruchomić polecenie instalujące komendy linii poleceń dla Entity Framework:

```
dotnet tool install --global dotnet-ef

dotnet ef migrations add InitSchema
```



3. Przeanalizuj utworzony plik migracji (funkcje Up i Down).
4. Za pomocą polecenia `Update-Database (lub dontnet ef database update)` zaktualizuj bazę danych.
5. Czy w bazie danych pojawiły się tabele? Jakie?
6. Zaktualizuj model Person.cs:

```
public class Person {
        public int Id { get; set; }
        [Required]
        [MaxLength(100)]
        [Column(TypeName="varchar(100)")]
        public string FirstName { get; set; }
        [Required]
        [MaxLength(100)]
        public string LastName { get; set; }
    }
```

7. Utwórz nową migrację o dowolnej nazwie np.
   `Add-Migration AddDataAnnotation`
   lub
   `dotnet ef migrations add nazwa`
8. Zaktualizuj bazę danych.
   `Update-Database`

lub

```
dotnet ef database update
```

9. Zaobserwuj zmiany w bazie danych.

Budowanie zapytań

1. Dodaj losowe dane do tabeli Person np. przez interface graficzny.
2. W Pages/Index.cshtml.cs:
    a. dodaj `private readonly PeopleContext _context;`
    b. rozszerz konstruktor o pobieranie konfiguracji:

```
public IndexModel(ILogger<IndexModel> logger, PeopleContext
context) {
      _logger = logger;
       _context = context;
 }
```

    c. zaimplementuj pobieranie danych z tabeli Person w metodzie onGet np.

```
public IList<Person> People { get; set; }
public void OnGet() {
   People = _context.Person.ToList();
}
```

    d. zaimplementuj wyświetlanie danych w szablonie:

```
@foreach (var item in Model.People)
{
<p>@Html.DisplayFor(modelItem => item.FirstName)
@Html.DisplayFor(modelItem => item.LastName)</p>
}
```

3. Zmodyfikuj metodę onGet, aby pobierała tylko osoby o podanym imieniu np. Adam:

```
public void OnGet() {
      People = _context.Person
                 .Where(p => p.FirstName == "Adam").ToList();
}
```

4. Możesz zrealizować te same zapytania w dwóch różnych składaniach LINQ.

LINQ Fluent	LINQ Query

`People = _context.Person.ToList();`	`var PeopleQuery = from person in _context.Person select person;`
	
	`People = PeopleQuery.ToList();`

`People = _context.Person .Where(p => p.FirstName == "Adam").ToList();`	`var PeopleQuery = from person in _context.Person where person.FirstName == "Adam" select person;`
	
	`People = PeopleQuery.ToList();`

5. Spróbuj posortować wprowadzone dane. Jakiej konstrukcji użyjesz?

Przykładowy zapis danych z formularza do bazy danych

1. Do Pages/Index.html.cs dodaj metodę OnPost oraz właściwość Person

```
    [BindProperty]
    public Person Person { get; set; }
    public IActionResult OnPost()
    {
        People = _context.Person.ToList();
        if (!ModelState.IsValid)
        {
            return Page();
        }
        _context.Person.Add(Person);
        _context.SaveChanges();
        return RedirectToPage("./Index");
    }
```

2. W szablonie Pages/Index.html utwórz formularz:

```
<form method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
            <label asp-for="Person.FirstName" class="control-label"></label>
            <input asp-for="Person.FirstName" class="form-control" />
            <span asp-validation-for="Person.FirstName"
class="text-danger"></span>
            </div>
            <div class="form-group">
            <label asp-for="Person.LastName" class="control-label"></label>
            <input asp-for="Person.LastName" class="form-control" />
            <span asp-validation-for="Person.LastName" class="text-danger"></span>
            </div>
            <div class="form-group">
            <input type="submit" value="Create" class="btn btn-primary" />
            </div>
    </form>
```

PRZYDATNE LINKI

- EntityFramework
- EF - Budowanie zapytań
- Struktura zapytań LINQ
- Przykład problemu z EntityFramework
- Różnice między nvarchar a varchar
- Metoda Code-first
- Metoda Database-first

NARZĘDZIA

- SQL Server Express (instalacja Basic)
  https://www.microsoft.com/pl-pl/sql-server/sql-server-downloads
- SSMS - SQL Server Management Studio
  https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15