### Bazy Danych 2

### Laboratorium 4

Entity Framework

Mateusz Łopaciński

### Kod po wprowadzeniu

#### **Zaimplementowane klasy**

#### **Klasa Product**

namespace MateuszLopacinskiEFProducts

{

internal class Product

{

public int ProductID { get; set; }

public string ProductName { get; set; }

public int UnitsOnStock { get; set; }

}

}

#### **Klasa ShopContext**

using Microsoft.EntityFrameworkCore;

namespace MateuszLopacinskiEFProducts

{

internal class ShopContext : DbContext

{

public DbSet<Product> Products { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

base.OnConfiguring(optionsBuilder);

optionsBuilder.UseSqlite("Datasource=ProductsDatabase.db");

}

}

}

#### **Klasa Program**

namespace MateuszLopacinskiEFProducts

{

class Program

{

static void Main()

{

Console.WriteLine("Podaj nazwę produktu");

String prodName = Console.ReadLine();

Console.WriteLine("Poniżej lista produktów zarejestrowanych w naszej bazie danych");

ShopContext shopContext = new ShopContext();

Product product = new Product { ProductName = prodName };

shopContext.Products.Add(product);

shopContext.SaveChanges();

var query = from prod in shopContext.Products

select prod.ProductName;

foreach (var pName in query)

{

Console.WriteLine(pName);

}

}

}

}

#### **Przykład działania**

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

### Wprowadzenie pojęcia Dostawcy

#### **Zaimplementowane klasy**

#### **Klasa Product**

namespace MateuszLopacinskiEFProducts

{

internal class Product

{

public int ProductID { get; set; }

public string ProductName { get; set; }

public int UnitsOnStock { get; set; }

public Supplier? Supplier { get; set; } = null;

public override string ToString()

{

return $"{ProductName} ({UnitsOnStock} szt.)";

}

}

}

#### **Klasa Supplier**

namespace MateuszLopacinskiEFProducts

{

internal class Supplier

{

public int SupplierID { get; set; }

public string CompanyName { get; set; }

public string Street { get; set; }

public string City { get; set; }

public override string ToString()

{

return CompanyName;

}

}

}

#### **Klasa ShopContext**

using Microsoft.EntityFrameworkCore;

namespace MateuszLopacinskiEFProducts

{

internal class ShopContext : DbContext

{

public DbSet<Product> Products { get; set; }

public DbSet<Supplier> Suppliers { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

base.OnConfiguring(optionsBuilder);

optionsBuilder.UseSqlite("Datasource=ProductsDatabase.db");

}

}

}

#### **Klasa Program**

namespace MateuszLopacinskiEFProducts

{

class Program

{

static void Main()

{

ShopContext shopContext = new ShopContext();

Product product = createNewProduct();

Supplier? supplier = null;

bool isCorrectChoice = false;

bool createdNewSupplier = false;

do {

Console.WriteLine("Dodać nowego dostawcę? (tak/nie)");

string choice = Console.ReadLine();

switch (choice)

{

case "tak":

isCorrectChoice = true;

supplier = createNewSupplier();

createdNewSupplier = true;

break;

case "nie":

isCorrectChoice = true;

displayAllSuppliers(shopContext);

supplier = findSupplier(shopContext);

break;

}

} while (!isCorrectChoice);

Console.WriteLine("Dodaję dostawcę do produktu...");

product.Supplier = supplier;

Console.WriteLine("Zapisuję dane do bazy...");

if (createdNewSupplier) shopContext.Suppliers.Add(supplier);

shopContext.Products.Add(product);

shopContext.SaveChanges();

}

private static Product createNewProduct()

{

Console.Write("Podaj nazwę produktu\n>>> ");

string prodName = Console.ReadLine();

Console.Write("Podaj liczbę dostępnych sztuk produktu\n>>> ");

int quantity = Int32.Parse(Console.ReadLine());

Console.WriteLine("Tworzę nowy produkt...");

Product product = new Product

{

ProductName = prodName,

UnitsOnStock = quantity

};

Console.WriteLine($"Stworzono produkt: {product}");

return product;

}

private static Supplier createNewSupplier()

{

Console.Write("\nPodaj nazwę dostawcy\n>>> ");

string companyName = Console.ReadLine();

Console.Write("Podaj miasto\n>>> ");

string city = Console.ReadLine();

Console.Write("Podaj ulicę\n>>> ");

string street = Console.ReadLine();

Console.WriteLine("Tworzę nowego dostawcę...");

Supplier supplier = new Supplier

{

CompanyName = companyName,

City = city,

Street = street

};

Console.WriteLine($"Stworzono dostawcę: {supplier}");

return supplier;

}

private static Supplier findSupplier(ShopContext shopContext)

{

Console.Write("Wprowadź id dostawcy, który ma zostać przypisany do nowego produktu\n>>>");

int choice = Int32.Parse(Console.ReadLine());

var query = from sup in shopContext.Suppliers

where sup.SupplierID == choice

select sup;

return query.FirstOrDefault();

}

private static void displayAllSuppliers(ShopContext shopContext)

{

Console.WriteLine("Lista wszystkich dostawców");

foreach (Supplier supplier in shopContext.Suppliers)

{

Console.WriteLine($"[{supplier.SupplierID}] {supplier}");

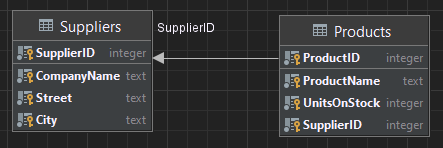
}

}

}

}

#### **Diagram bazy danych**



#### **Przykład działania**

#### **Przykładowe wykonania programu**

* **Z dodawaniem nowego dostawcy**

Obraz zawierający tekst

Opis wygenerowany automatycznie

* **Z dodawaniem istniejącego dostawcy do nowego produktu**

Obraz zawierający tekst

Opis wygenerowany automatycznie

#### **Tabele po kilkukrotnym dodaniu produktów**

* **Produkty**

Obraz zawierający tekst, sprzęt elektroniczny, ekran, zrzut ekranu

Opis wygenerowany automatycznie

* **Dostawcy**

**Obraz zawierający tekst

Opis wygenerowany automatycznie**

### Odwrócenie relacji

#### **Zmienione klasy**

#### **Klasa Product**

Usunięty został atrybut public Supplier? Supplier. Poza tym, klasa pozostaje bez zmian względem poprzedniej wersji.

namespace MateuszLopacinskiEFProducts

{

internal class Product

{

public int ProductID { get; set; }

public string ProductName { get; set; }

public int UnitsOnStock { get; set; }

public override string ToString()

{

return $"{ProductName} ({UnitsOnStock} szt.)";

}

}

}

#### **Klasa Supplier**

W tej klasie dodana została kolekcja public ICollection<Product> Products. Poza tym, nie wprowadzałem innych zmian.

namespace MateuszLopacinskiEFProducts

{

internal class Supplier

{

public int SupplierID { get; set; }

public string CompanyName { get; set; }

public string Street { get; set; }

public string City { get; set; }

public ICollection<Product> Products { get; set; } = new List<Product>();

public override string ToString()

{

return CompanyName;

}

}

}

#### **Klasa Program**

Zmienione zostały jednie 2 linijki w metodzie static void Main(), dlatego poniżej zamieszczam tylko fragment kodu, który został zamieniony oraz kod, którym go zastąpiłem.

* **Poprzedni kod (fragment)**

static void Main()

{

**…**

Console.WriteLine("Dodaję dostawcę do produktu...");

product.Supplier = supplier;

**…**

}

* **Nowy kod (fragment)**

static void Main()

{

**…**

Console.WriteLine("Dodaję produkt to dostawcy...");

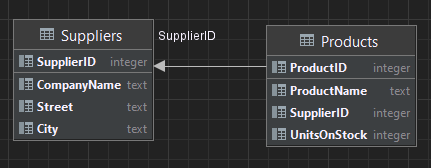
supplier.Products.Add(product);

**…**

}

#### **Diagram bazy danych**

Jak możemy zauważyć, pomimo zapisania relacji w Entity Frameworku w odwrotny sposób, w bazie danych relacja wciąż wygląda tak samo. Widzimy więc, że Entity Framework „pod spodem” dokonał optymalizacji, dzięki czemu nie musimy trzymać w tabeli **Suppliers** powielonych danych dostawców, różniących się jedynie **SupplierID** oraz kluczem obcym, wskazującym na produkt z tabeli **Products**. Zauważmy, że w takiej sytuacji również mielibyśmy problem z rozróżnianiem dostawców, ponieważ jeden dostawca musał by sięznaleźć w tabeli wielokrotnie, mając za każdym razem przypisane inne id (traktowany byłby jako inny dostawca).

****

#### **Przykład działania**

#### **Przykładowe wykonania programu**

* **Z dodawaniem produktu do istniejącego dostawcy**

Obraz zawierający tekst

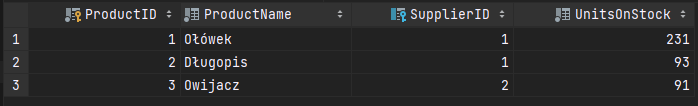
Opis wygenerowany automatycznie

* **Z dodawaniem nowego dostawcy**

#### Obraz zawierający tekst Opis wygenerowany automatycznie

#### **Tabele po dodaniu kilku dostawców i produktów**

* **Produkty**



* **Dostawcy**

Obraz zawierający tekst

Opis wygenerowany automatycznie

### Relacja dwustronna

#### **Zmienione klasy**

#### **Klasa Product**

Dodany został atrybut public Supplier ?Supplier { get; set; } = null; (taki jak w pierwszej implementacji klasy Product). Klasa wygląda więc identycznie jak ta z podpunktu **2.1.1.**, dlatego nie umieściłem poniżej jej kodu.

#### **Klasa Supplier**

Ta klasa nie została zmieniona względem implementacji z punktu **3.1.2.**, dlatego nie wkleiłem jej implementacji poniżej.

#### **Klasa Program**

Ponownie została dodana linijka , (taka jak w implementacji z punktu **2.1.4.**). Poniżej umieściłem jedynie fragment kodu, przedstawiający modyfikację względem implementacji z punktu **3.1.3.**).

* **Poprzedni kod (fragment)**

static void Main()

{

**…**

Console.WriteLine("Dodaję produkt to dostawcy...");

supplier.Products.Add(product);

**…**

}

* **Nowy kod (fragment)**

static void Main()

{

**…**

Console.WriteLine("Dodaję produkt to dostawcy...");

product.Supplier = supplier;

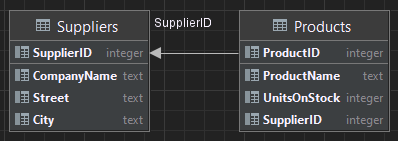
supplier.Products.Add(product);

**…**

}

#### **Diagram bazy danych**

Ponownie obserwujemy taki sam diagram. Możemy więc dojść do wniosku, że Entity Framework pozwala nam na stworzenie relacji dwukierunkowej (lub w odwrotnym kierunku do tego, w którym relacja zostanie zapisana, jak widzieliśmy w poprzednim przykładzie), po to, aby łatwiej móc manipulować powiązanymi ze sobą obiektami. Mimo to, „pod spodem” zapisane przez nas relacje są przekształcane na relacje dające się zapisać w bazie danych.



#### **Przykład działania**

#### **Przykładowe wykonania programu**

* **Z dodawaniem nowego dostawcy**

Obraz zawierający tekst

Opis wygenerowany automatycznie

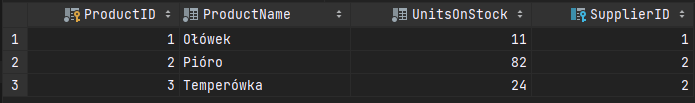
* **Z dodawaniem produktu do istniejącego dostawcy**

Obraz zawierający tekst

Opis wygenerowany automatycznie

#### **Tabele po dodaniu kilku dostawców i produktów**

* **Produkty**



* **Dostawcy**

Obraz zawierający tekst

Opis wygenerowany automatycznie

### Relacja wiele do wielu

#### **Zmienione klasy**

#### **Klasa Product**