

# Bazy Danych – Entity Framework Laboratorium

Bartłomiej Plewnia gr. wtorek 11.15 – 12.45 A

## I. Code First

- a) Stworzyłem klasę Product zawierającą atrybuty ProductID, Name oraz UnitsInStock

```
using System;
using System.Collections.Generic;
using System.Text;

namespace IPlewniaProductEF
{
    Odwołania: 4
    class Product
    {
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 2
        public string Name { get; set; }
        Odwołania: 0
        public int UnitsInStock { get; set; }
    }
}
```

- b) Stworzyłem jej kontekst – ProdContext zawierający listę Produktów oraz nadpisujący metodę korzystającą z bazy Sqlite

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.EntityFrameworkCore;

namespace IPlewniaProductEF
{
    Odwołania: 4
    class ProdContext : DbContext
    {
        1 odwołanie
        public DbSet<Product> Products { get; set; }

        Odwołania: 0
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) =>
            optionsBuilder.UseSqlite("DataSource=Product.db");
    }
}
```

c) W pliku program.cs napisałem kod, który zczytuje nazwę produktu, a następnie wypisuje zawartość ProdContext

```
namespace IPlewniaProductEF
{
    Odwołania: 0
    class Program
    {
        Odwołania: 0
        static void Main(string[] args)
        {
            Console.WriteLine("Podaj nazwę produktu:");
            string name = Console.ReadLine();
            Product prod = new Product { Name = name };
            ProdContext db = new ProdContext();
            db.Add(prod);
            db.SaveChanges();

            IQueryable<Product> query = from p in db.Products
                                         select p;

            foreach (var Prod in query)
            {
                Console.WriteLine("Nazwa produktu to: " + Prod.Name);
            }
        }
    }
}
```

Konsola debugowania programu Microsoft Visual Studio

```
Podaj nazwę produktu:
Kremówka
Nazwa produktu to: Koala
Nazwa produktu to: Koala
Nazwa produktu to: Kalarepa
Nazwa produktu to: Kremówka
```

## II. Zadanie

a) Dodałem DbSet Suppliers do kontekstu ProdContext

```
Odwołania: 4
class ProdContext : DbContext
{
    Odwołania: 4
    public DbSet<Product> Products { get; set; }
    Odwołania: 2
    public DbSet<Supplier> Suppliers { get; set; }

    Odwołania: 0
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) =>
        optionsBuilder.UseSqlite("DataSource=Product.db");
}
```

b) Dodałem klasę Supplier

```
using System;
using System.Collections.Generic;
using System.Text;

namespace IPlewniaProductEF
{
    Odwołania: 5
    class Supplier
    {
        Odwołania: 0
        public int SupplierID { get; set; }
        Odwołania: 4
        public string CompanyName { get; set; }
        Odwołania: 0
        public string Street { get; set; }
        Odwołania: 0
        public string City { get; set; }
    }
}
```

c) Napisałem kod, który dodaje Produkt oraz Dostawcę ,  
następnie przypisuje danego Dostawcę do tego produktu, a  
następnie listuje wszystkie produkty wraz z ich Dostawcami

```
Odwolań: 0
static void Main(string[] args)
{
    Console.WriteLine("Podaj nazwę produktu:");
    string nameProd = Console.ReadLine();
    Product prod = new Product { Name = nameProd };

    ProdContext dbProd = new ProdContext();
    dbProd.Products.Add(prod);

    Console.WriteLine("Podaj dostawcę:");
    string nameSupplier = Console.ReadLine();
    Supplier supplier = new Supplier { CompanyName = nameSupplier };

    dbProd.Suppliers.Add(supplier);
    dbProd.SaveChanges();

    // Connecting Supplier to Product
    Product prodChange = dbProd.Products.Single(p => p.Name == nameProd);
    prodChange.Suppliers = supplier;
    dbProd.Products.Update(prodChange);
    dbProd.SaveChanges();

    var query = from Product p in dbProd.Products
                join Supplier s in dbProd.Suppliers on p.Suppliers.CompanyName equals s.CompanyName
                select new { p.Name, s.CompanyName };

    foreach (var list in query)
    {
        Console.WriteLine(list.Name + " " + list.CompanyName);
    }
}
```

Konsola debugowania programu Microsoft Visual Studio

```
Podaj nazwę produktu:
K0as
Podaj dostawcę:
Boas
K0as Boas
```

### III Zadanie

- a) Dodałem do klasy Supplier atrybut zawierający listę produktów przez tą klasę dostarczanych.

```
namespace IPlewniaProductEF
{
    Odwołania: 4
    class Supplier
    {
        1 odwołanie
        public Supplier()
        {
            Products = new Collection<Product>();
        }

        Odwołania: 0
        public int SupplierID { get; set; }
        Odwołania: 2
        public string CompanyName { get; set; }
        Odwołania: 0
        public string Street { get; set; }
        Odwołania: 0
        public string City { get; set; }
        Odwołania: 6
        public ICollection<Product> Products { get; set; }
    }
}
```

b) Napisałem program, który wprowadza produkty oraz dostawcę, a następnie zwraca nazwę dostawcy oraz nazwy produktów przez niego dostarczanych.

```
ProdContext dbProd = new ProdContext();

Console.WriteLine("Podaj nazwę produktu:");
string nameProd1 = Console.ReadLine();
Product prod1 = new Product { Name = nameProd1 };
dbProd.Products.Add(prod1);

Console.WriteLine("Podaj nazwę produktu:");
string nameProd2 = Console.ReadLine();
Product prod2 = new Product { Name = nameProd2 };
dbProd.Products.Add(prod2);

Console.WriteLine("Podaj nazwę produktu:");
string nameProd3 = Console.ReadLine();
Product prod3 = new Product { Name = nameProd3 };
dbProd.Products.Add(prod3);

Console.WriteLine("Podaj dostawcę:");
string nameSupplier = Console.ReadLine();
Supplier supplier = new Supplier { CompanyName = nameSupplier };
dbProd.Suppliers.Add(supplier);

supplier.Products.Add(prod1);
supplier.Products.Add(prod2);
supplier.Products.Add(prod3);

dbProd.SaveChanges();

var query = dbProd.Suppliers.Include(p => p.Products).ToList();

foreach (var supp in query)
{
    Console.WriteLine(supp.CompanyName + " dostarcza: ");
    foreach (var prod in supp.Products)
    {
        Console.WriteLine(prod.Name);
    }
}
```

Konsola debugowania programu Microsoft Visual Studio

```
Podaj nazwę produktu:
Jabłko
Podaj nazwę produktu:
Ananas
Podaj nazwę produktu:
Gruszka
Podaj dostawcę:
Owoce & Warzywa
Owoce & Warzywa dostarcza:
Jabłko
Ananas
Gruszka
```



## IV Zadanie

- a) Dodałem do klasy Produkt atrybut Supplier, tak aby relacja działała w obie strony.

```
namespace IPlewniaProductEF
{
    Odwołania: 9
    class Product
    {
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 4
        public string Name { get; set; }
        Odwołania: 0
        public int UnitsInStock { get; set; }
        Odwołania: 3
        public Supplier Supplier { get; set; }
    }
}
```

b) Napisałem program, który na początku wczytuje dostawcę, następnie 3 produkty, a następnie wypisuje dostawcę i produkty przez niego dostarczane

```
static void Main(string[] args)
{
    ProdContext dbProd = new ProdContext();

    Console.WriteLine("Podaj dostawcę:");
    string nameSupplier = Console.ReadLine();
    Supplier supplier = new Supplier { CompanyName = nameSupplier };
    dbProd.Suppliers.Add(supplier);

    Console.WriteLine("Podaj nazwę produktu:");
    string nameProd1 = Console.ReadLine();
    Product prod1 = new Product { Name = nameProd1, Supplier = supplier };
    dbProd.Products.Add(prod1);


    Console.WriteLine("Podaj nazwę produktu:");
    string nameProd2 = Console.ReadLine();
    Product prod2 = new Product { Name = nameProd2, Supplier = supplier };
    dbProd.Products.Add(prod2);

    Console.WriteLine("Podaj nazwę produktu:");
    string nameProd3 = Console.ReadLine();
    Product prod3 = new Product { Name = nameProd3, Supplier = supplier };
    dbProd.Products.Add(prod3);

    dbProd.SaveChanges();

    var query = dbProd.Suppliers.Include(p => p).ToList();

    foreach (var supp in query)
    {
        Console.WriteLine(supp.CompanyName + " dostarcza: ");
        foreach (var prod in supp.Products)
        {
            Console.WriteLine(prod.Name);
        }
    }
}
```

 Konsola debugowania programu Microsoft Visual Studio

```
Podaj dostawcę:
Google AI
Podaj nazwę produktu:
UR GOOGLE
Podaj nazwę produktu:
KOMPUTER
Podaj nazwę produktu:
LAPTOP
Google AI dostarcza:
UR GOOGLE
KOMPUTER
LAPTOP
```



# V Zadanie

## a) Stworzenie klasy Category

```
namespace IPlewniaProductEF
{
    Odwołania: 7
    class Category
    {
        Odwołania: 2
        public Category()
        {
            Products = new Collection<Product>();
        }
        Odwołania: 0
        public int CategoryID { get; set; }
        Odwołania: 3
        public string Name { get; set; }
        Odwołania: 2
        public ICollection<Product> Products { get; set; }
    }
}
```

## b) Dodanie do klasy Produkt atrybutu Category

```
namespace IPlewniaProductEF
{
    Odwołania: 9
    class Product
    {
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 3
        public string Name { get; set; }
        Odwołania: 0
        public int UnitsInStock { get; set; }
        Odwołania: 0
        public Supplier Supplier { get; set; }
        Odwołania: 2
        public Category Category { get; set; }
    }
}
```

### c) Dodanie do kontekstu zbioru Categories

```
namespace IPlewniaProductEF
{
    Odwołania: 4
    class ProdContext : DbContext
    {
        Odwołania: 2
        public DbSet<Product> Products { get; set; }
        Odwołania: 0
        public DbSet<Supplier> Suppliers { get; set; }
        Odwołania: 3
        public DbSet<Category> Categories { get; set; }

        Odwołania: 0
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) =>
            optionsBuilder.UseSqlite("DataSource=Product.db");
    }
}
```

- d) Napisanie programu, który przyjmuje kategorie oraz produkty, następnie dołącza produkty do ich kategorii, a następnie listuje wszystkie kategorie oraz ich produkty

```
ProdContext dbProd = new ProdContext();

Console.WriteLine("Podaj kategorię:");
string nameCategory1 = Console.ReadLine();
Category category1 = new Category { Name = nameCategory1};
dbProd.Categories.Add(category1);

Console.WriteLine("Podaj kategorię:");
string nameCategory2 = Console.ReadLine();
Category category2 = new Category { Name = nameCategory2 };
dbProd.Categories.Add(category2);

Console.WriteLine("Podaj nazwę produktu:");
string nameProd1 = Console.ReadLine();
Product prod1 = new Product { Name = nameProd1, Category = category1};
dbProd.Products.Add(prod1);

Console.WriteLine("Podaj nazwę produktu:");
string nameProd2 = Console.ReadLine();
Product prod2 = new Product { Name = nameProd2, Category = category2};
dbProd.Products.Add(prod2);

dbProd.SaveChanges();

var query = dbProd.Categories.Include(p => p).ToList();

Console.WriteLine("DOSTĘPNE KATEGORIE: ");
foreach (var cat in query)
{
    Console.WriteLine(cat.Name + " zawiera: ");
    foreach (var prod in cat.Products)
    {
        Console.WriteLine(prod.Name);
    }
}
```

Konsola debugowania programu Microsoft Visual Studio

```
Podaj kategorię:
Owoce
Podaj kategorię:
Warzywa
Podaj nazwę produktu:
Pomarańcza
Podaj nazwę produktu:
Seler
DOSTĘPNE KATEGORIE:
Owoce zawiera:
Pomarańcza
Warzywa zawiera:
Seler
```

## VI Zadanie

- a) Aby uzyskać pożądaną relację wielu-do-wielu stworzyłem klasę pośredniczącą w wymianie między Invoice a Product

```
{
    Odwołania: 10
    class MediatorProduct
    {
        Odwołania: 0
        public int MediatorProductID { get; set; }
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 3
        public Product Product { get; set; }
        Odwołania: 4
        public int Quantity { get; set; }
        Odwołania: 0
        public int InvoiceID { get; set; }
        Odwołania: 4
        public Invoice Invoice { get; set; }
    }
}
```

- b) Stworzyłem klasę Invoice

```
{
    Odwołania: 7
    class Invoice
    {
        Odwołania: 2
        public Invoice()
        {
            MediatorProducts = new Collection<MediatorProduct>();
        }

        Odwołania: 0
        public int InvoiceID { get; set; }
        Odwołania: 3
        public int InvoiceNumber { get; set; }

        Odwołania: 4
        public ICollection<MediatorProduct> MediatorProducts { get; set; }
    }
}
```

c) Do klasy Product dodałem atrybut zawierający listę mediatorów

```
{
    Odwołania: 13
    class Product
    {
        Odwołania: 0
        public Product()
        {
            MediatorProducts = new Collection<MediatorProduct>();
        }
        Odwołania: 0
        public int ProductID { get; set; }
        Odwołania: 4
        public string Name { get; set; }
        Odwołania: 0
        public int UnitsInStock { get; set; }
        Odwołania: 0
        public Supplier Supplier { get; set; }
        Odwołania: 0
        public Category Category { get; set; }

        public ICollection<MediatorProduct> MediatorProducts { get; set; }
    }
}
```

d) Do ProdContext dołączyłem zbiór Invoices

```
Odwołania: 4
public DbSet<Product> Products { get; set; }
Odwołania: 0
public DbSet<Supplier> Suppliers { get; set; }
Odwołania: 0
public DbSet<Category> Categories { get; set; }
Odwołania: 2
public DbSet<Invoice> Invoices { get; set; }

Odwołania: 0
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) =>
    optionsBuilder.UseSqlite("DataSource=Product.db");
```



- e) Następnie stworzyłem kilka produktów oraz faktur, a następnie “sprzedałem” produkty na kilku transakcjach

```
Console.WriteLine("Podaj nazwę produktu:");
string nameProd1 = Console.ReadLine();
Product prod1 = new Product { Name = nameProd1 };
dbProd.Products.Add(prod1);

Console.WriteLine("Podaj nazwę produktu:");
string nameProd2 = Console.ReadLine();
Product prod2 = new Product { Name = nameProd2 };
dbProd.Products.Add(prod2);

Console.WriteLine("Podaj nazwę produktu:");
string nameProd3 = Console.ReadLine();
Product prod3 = new Product { Name = nameProd3 };
dbProd.Products.Add(prod3);

Console.WriteLine("Podaj numer faktury:");
int InvoiceNumber1 = Int32.Parse(Console.ReadLine());
Invoice invoice1 = new Invoice { InvoiceNumber = InvoiceNumber1 };
dbProd.Invoices.Add(invoice1);

Console.WriteLine("Podaj numer faktury:");
int InvoiceNumber2 = Int32.Parse(Console.ReadLine());
Invoice invoice2 = new Invoice { InvoiceNumber = InvoiceNumber2 };
dbProd.Invoices.Add(invoice2);

//Tworzenie sprzedaży
MediatorProduct mediator1 = new MediatorProduct { Invoice = invoice1, Product = prod1, Quantity = 30 };
MediatorProduct mediator2 = new MediatorProduct { Invoice = invoice1, Product = prod2, Quantity = 40 };
MediatorProduct mediator3 = new MediatorProduct { Invoice = invoice2, Product = prod3, Quantity = 100 };

prod1.Invoices.Add(invoice1);
prod2.Invoices.Add(invoice1);
prod3.Invoices.Add(invoice2);

invoice1.MediatorProducts.Add(mediator1);
invoice1.MediatorProducts.Add(mediator2);
invoice2.MediatorProducts.Add(mediator3);
```

- f) Następnie wykonałem polecenie b. (Pokaż produkty sprzedane w ramach wybranej faktury/transakcji)

```
var query = dbProd.Invoices.Include(p => p).ToList();

Console.WriteLine("FAKTURY: ");
foreach (var inv in query)
{
    Console.WriteLine("Numer faktury: " + inv.InvoiceNumber);
    foreach (var prod in inv.MediatorProducts)
    {
        Console.WriteLine(prod.Product.Name);
    }
}
```



```
Konsola debugowania programu Microsoft Visual Studio

Podaj nazwę produktu:
Kilof
Podaj nazwę produktu:
Łopata
Podaj nazwę produktu:
Sadzonki
Podaj numer faktury:
12345
Podaj numer faktury:
54321
FAKTURY:
Numer faktury: 12345
Kilof
Łopata
Numer faktury: 54321
Sadzonki
```

g) Następnie wykonałem polecenie c. (Pokaż faktury w ramach których był sprzedany wybrany produkt)

```
var query = dbProd.Products.Include(p => p).ToList();

Console.WriteLine("PRODUKTY W FAKTURACH: ");
foreach (var prod in query)
{
    Console.WriteLine("Nazwa produktu: " + prod.Name);
    foreach (var inv in prod.MediatorProducts)
    {
        Console.WriteLine("NUMER FAKTURY: " + inv.Invoice.InvoiceNumber);
        Console.WriteLine("ILOSC ZAKUPIONYCH: " + inv.Quantity);
    }
}
```

```
Konsola debugowania programu Microsoft Visual Studio

Podaj nazwę produktu:
Kilof
Podaj nazwę produktu:
Łopata
Podaj nazwę produktu:
Sadzonki
Podaj numer faktury:
12345
Podaj numer faktury:
54321
PRODUKTY W FAKTURACH:
Nazwa produktu: Kilof
NUMER FAKTURY: 12345
ILOSC ZAKUPIONYCH: 30
Nazwa produktu: Łopata
NUMER FAKTURY: 12345
ILOSC ZAKUPIONYCH: 40
Nazwa produktu: Sadzonki
NUMER FAKTURY: 54321
ILOSC ZAKUPIONYCH: 100
```

## VII Zadanie

a) Stworzyłem klasę Company

```
class Company
{
    Odwołania: 0
    public int CompanyID { get; set; }
    Odwołania: 0
    public string CompanyName { get; set; }
    Odwołania: 0
    public string Street { get; set; }
    Odwołania: 0
    public string City { get; set; }
    Odwołania: 0
    public string ZipCode { get; set; }
}
```

b) Stworzyłem klasę Customer dziedziczącą po klasie Company

```
class Customer : Company
{
    Odwołania: 0
    public int Discount { get; set; }
}
```

c) Klasa Supplier teraz dziedziczy po klasie Company

```
class Supplier : Company
{
    Odwołania: 0
    public Supplier()
    {
        Products = new Collection<Product>();
    }
    Odwołania: 0
    public int BankAccountNumber { get; set; }
    1 odwołanie
    public ICollection<Product> Products { get; set; }
}
```

d) W ProdContext dodałem zbiór Companies. Następnie aby wykorzystać strategię mapowania dziedziczenia - TablePerHierarchy należało nadpisać metodę OnModelCreating.

```
class ProdContext : DbContext
{
    Odwołania: 4
    public DbSet<Product> Products { get; set; }
    Odwołania: 0
    public DbSet<Supplier> Suppliers { get; set; }
    Odwołania: 0
    public DbSet<Category> Categories { get; set; }
    Odwołania: 2
    public DbSet<Invoice> Invoices { get; set; }
    Odwołania: 0
    public DbSet<Company> Companies { get; set; }

    Odwołania: 0
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) =>
        optionsBuilder.UseSqlite("DataSource=Product.db");

    Odwołania: 0
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Supplier>();
        modelBuilder.Entity<Customer>();
    }
}
```

e) Następnie napisałem program, który pobiera nazwy firm wprowadzone przez użytkownika a następnie wypisuje je

```
ProdContext dbProd = new ProdContext();

Console.WriteLine("Podaj nazwę firmy:");
string nameComp1 = Console.ReadLine();
Company comp1 = new Customer{ CompanyName = nameComp1, Discount = 50 };
dbProd.Companies.Add(comp1);

Console.WriteLine("Podaj nazwę firmy:");
string nameComp2 = Console.ReadLine();
Company comp2 = new Customer { CompanyName = nameComp2, Discount = 0 };
dbProd.Companies.Add(comp2);

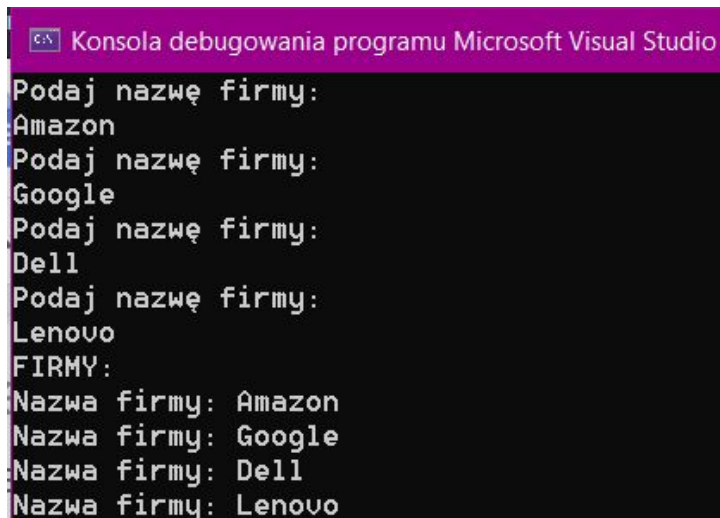
Console.WriteLine("Podaj nazwę firmy:");
string nameComp3 = Console.ReadLine();
Company comp3 = new Supplier { CompanyName = nameComp3, BankAccountNumber = 1234567};
dbProd.Companies.Add(comp3);

Console.WriteLine("Podaj nazwę firmy:");
string nameComp4 = Console.ReadLine();
Company comp4 = new Supplier { CompanyName = nameComp4, BankAccountNumber = 7654321 };
dbProd.Companies.Add(comp4);

dbProd.SaveChanges();

var query = dbProd.Companies.Include(p => p).ToList();

Console.WriteLine("FIRMY: ");
foreach (var comp in query)
{
    Console.WriteLine("Nazwa firmy: " + comp.CompanyName);
}
```



```
Podaj nazwę firmy:
Amazon
Podaj nazwę firmy:
Google
Podaj nazwę firmy:
Dell
Podaj nazwę firmy:
Lenovo
FIRMY:
Nazwa firmy: Amazon
Nazwa firmy: Google
Nazwa firmy: Dell
Nazwa firmy: Lenovo
```

Adnotacja do zadania VII:

Od wersji 3.0 Entity Frameworku strategie mapowania dziedziczenia TablePerClass oraz TablePerType nie działają.