

Dawid Majchrowski

.Net, Entity Framework

Sprawozdanie

Sprawozdanie kontynuujemy od miejsca zakończenia ćwiczeń na następującym stanie (IV j.):

- CategoryForm

```
ConsoleApplication1.CategoryForm
CategoryForm_Load

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Entity;

namespace ConsoleApplication1
{
    public partial class CategoryForm : Form
    {
        public CategoryForm()
        {
            InitializeComponent();
            Load += new EventHandler(CategoryForm_Load);
        }

        private void categoryDataGridView_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
        }

        private void CategoryForm_Load(object sender, System.EventArgs e)
        {
            ProdContext db = new ProdContext();
            db.Categories.Load();
            this.categoryBindingSource.DataSource = db.Categories.Local.ToBindingList();
        }
    }
}
```

- Aplikacja konsolowa

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ProdContext db = new ProdContext()){
                Console.WriteLine("Podaj nazwe kategorii: ");
                var category = Console.ReadLine();
                db.Categories.Add(new Category { Name = category });
                db.SaveChanges();

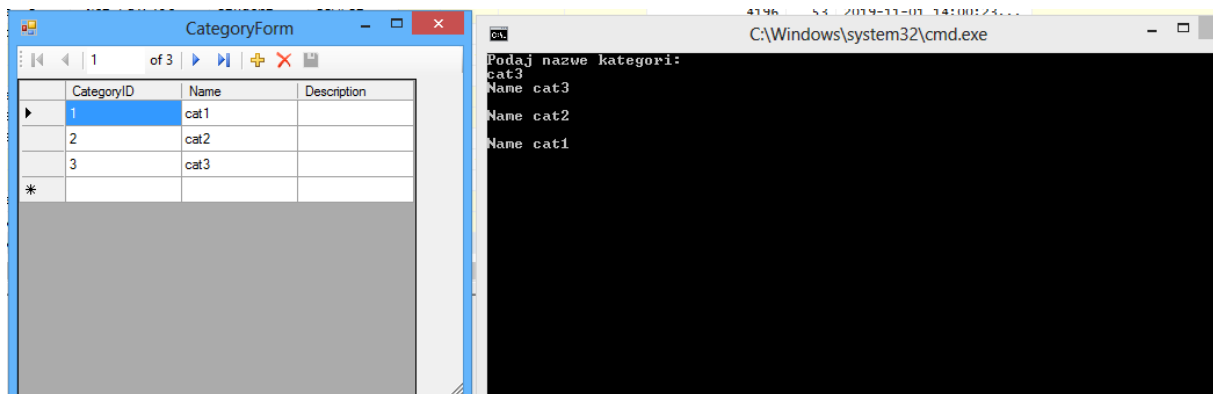
                var query = from cat in db.Categories
                            orderby cat.Name descending
                            select cat.Name;

                foreach (var item in query){
                    Console.WriteLine("Name {0}\n", item);
                }

                CategoryForm form = new CategoryForm();
                form.ShowDialog();
            }
        }
    }

    class Customer
    {
        [Key]
        public string CompanyName { get; set; }
        public string Description { get; set; }
    }
}
```

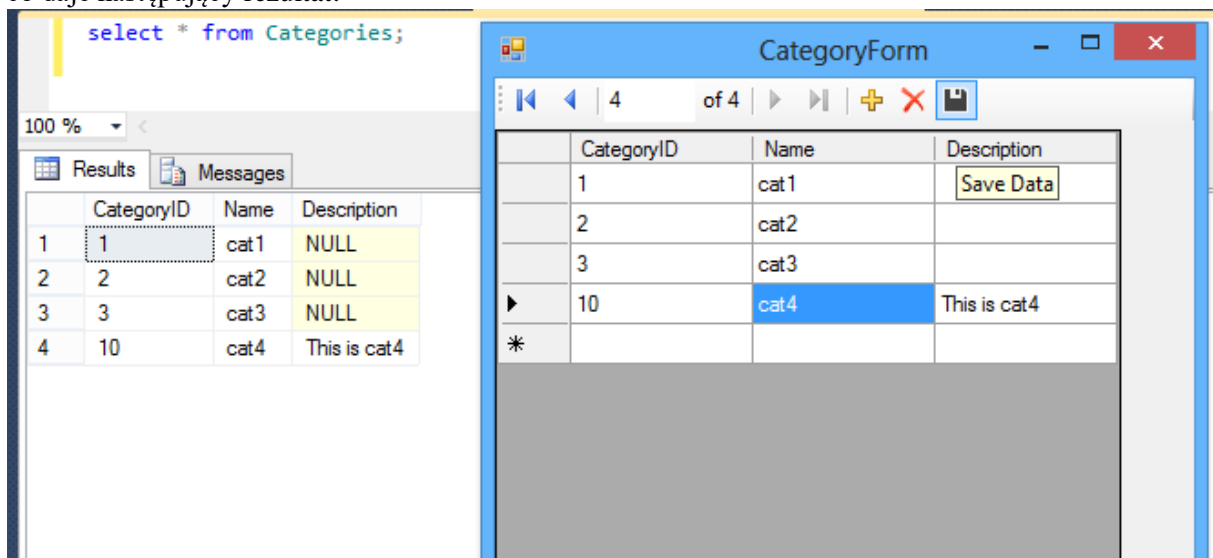
- Aktualny stan po odpaleniu aplikacji



Kolejnym zadaniem jest dodanie obsługi zmian oraz zapisu danych w formularzu. Robimy to w następujący sposób:

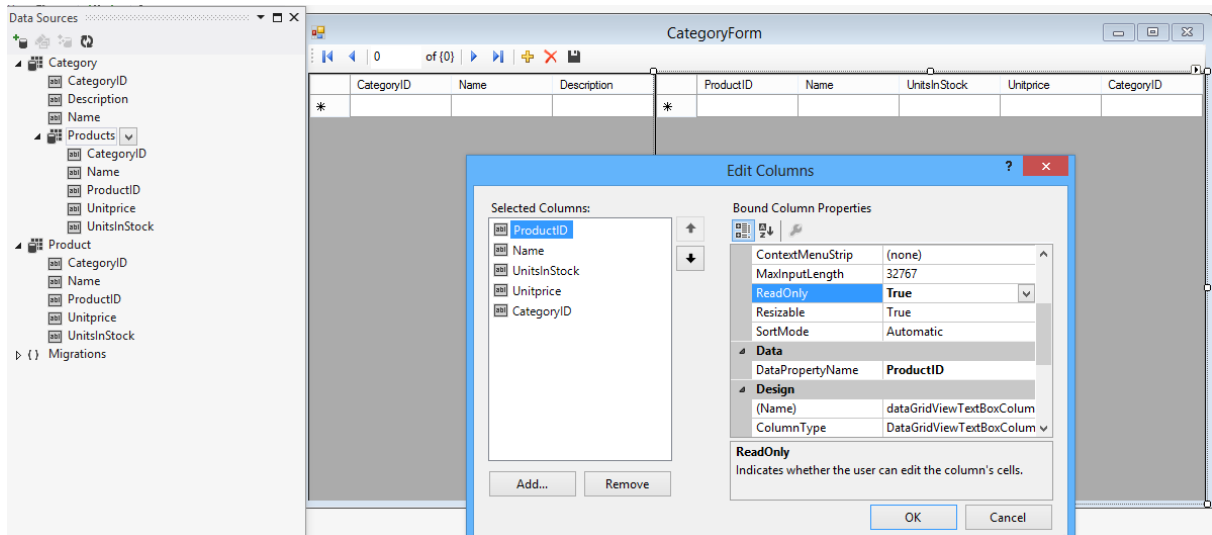
```
ProdContext db;  
  
public CategoryForm()  
{  
    InitializeComponent();  
    Load += new EventHandler(CategoryForm_Load);  
}  
  
private void categoryDataGridView_CellContentClick(object sender, DataGridViewCellEventArgs e)  
{  
  
}  
  
private void CategoryForm_Load(object sender, System.EventArgs e)  
{  
    db = new ProdContext();  
    db.Categories.Load();  
    this.categoryBindingSource.DataSource = db.Categories.Local.ToBindingList();  
}  
  
private void categoryBindingNavigatorSaveItem_Click(object sender, EventArgs e)  
{  
    db.SaveChanges();  
    this.categoryDataGridView.Refresh();  
}
```

co daje następujący rezultat:



Po dodaniu nowej kategorii, sprawdzamy, czy działa usuwanie usuwając nowo dodaną kategorię oraz ponownie zapisując dane do bazy, obserwując zachowanie zgodnie z założeniem.

Kolejnym krokiem jest dodanie produktów do naszej aplikacji konsolowej, robimy to w ten sam sposób co dodanie kategorii (przeciągamy źródło danych z „data source”), ustawiamy odpowiednie pola na do odczytu, oraz tworzymy obsługę eventu następującego po naciśnięciu odpowiedniej komórki z kategorią.



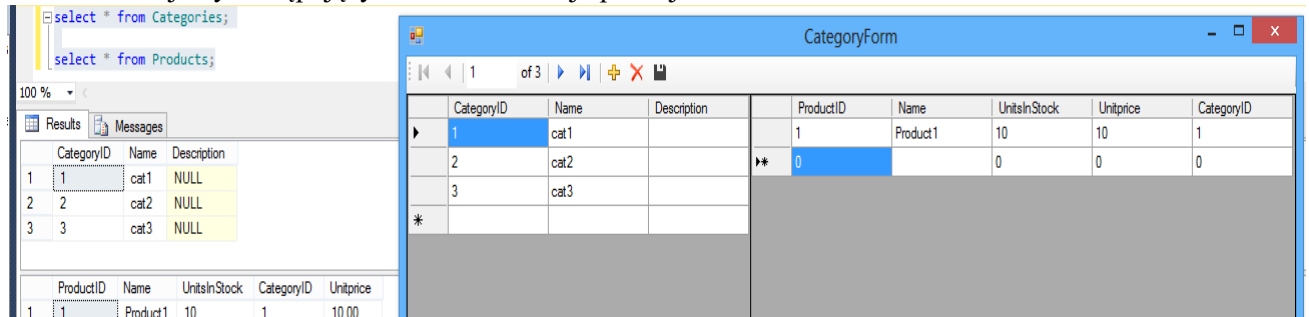
Zapytania zgodnie z poleceniem obsługę formularza tworzymy w obu notacjach.

```
private void categoryDataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
    DataGridView dgv = sender as DataGridView;

    if (dgv == null)
        return;
    if (dgv.CurrentRow.Selected)
    {
        int CategoryId = (int)dgv.CurrentRow.Cells[0].Value;
        //var query = from prod in db.Products
        //              where prod.CategoryID == CategoryId
        //              select prod;
        var query = db.Products.Where(prod => prod.CategoryID == CategoryId);

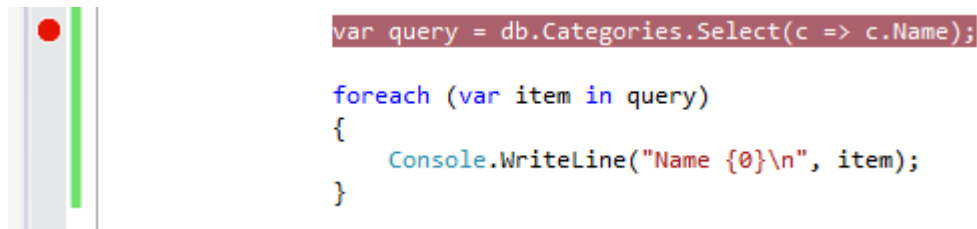
        this.productsBindingSource.DataSource = new BindingList<Product>(query.ToList());
    }
}
```

Oraz obserwujemy następujący rezultat w naszej aplikacji:



Wracamy do części konsolowej i dodajemy odpowiednie metody, które:

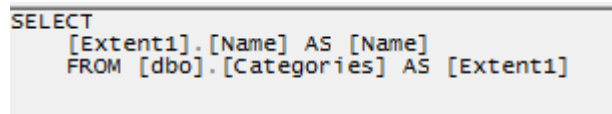
- i) Nazwy kategorii – method based syntax



```
var query = db.Categories.Select(c => c.Name);

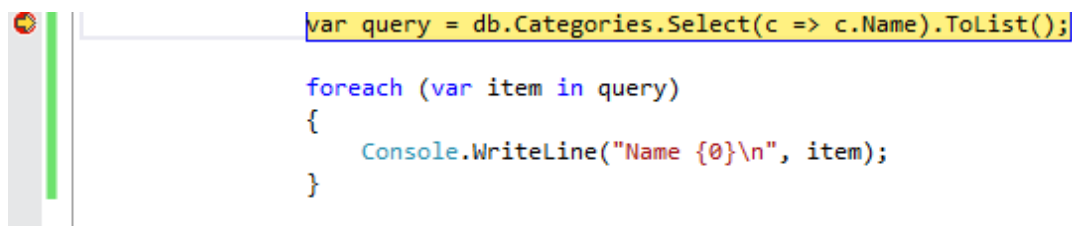
foreach (var item in query)
{
    Console.WriteLine("Name {0}\n", item);
}
```

Po krokowym przejściu, obserwujemy na profilerze egzekucję następującego zapytania, dopiero w momencie wywołania pętli foreach, czyli zgodnie z przewidywaniem, gdyż nie wymusiliśmy natychmiastowej egzekucji zapytania.



```
SELECT
    [Extent1].[Name] AS [Name]
FROM [dbo].[Categories] AS [Extent1]
```

Dokonujemy drobnej zmiany, wymuszając natychmiastową egzekucję i obserwujemy w profilerze to samo zapytanie, ale wykonane w momencie deklaracji zapytania, a nie przy egzekucji pętli foreach.

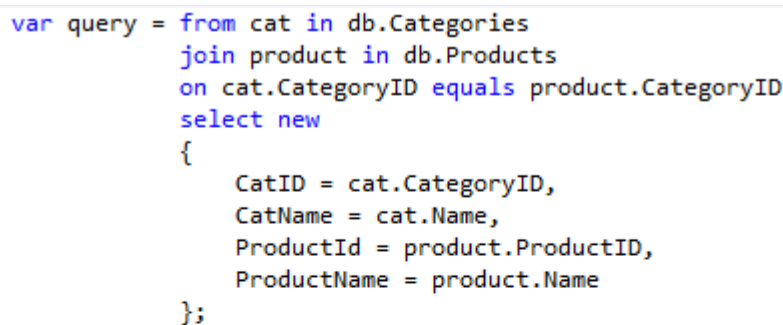


```
var query = db.Categories.Select(c => c.Name).ToList();

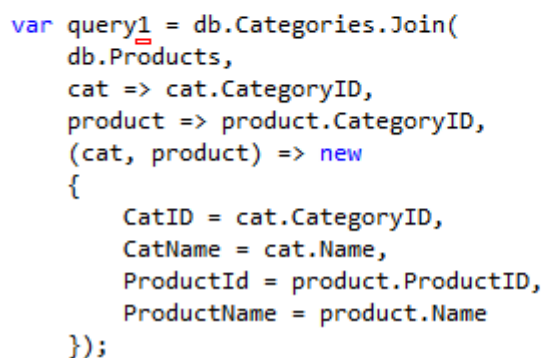
foreach (var item in query)
{
    Console.WriteLine("Name {0}\n", item);
}
```

- ii) Wszystkie kategorie i produkty (w obu notacjach)

- Joigny (Obie notacje)



```
var query = from cat in db.Categories
            join product in db.Products
            on cat.CategoryID equals product.CategoryID
            select new
            {
                CatID = cat.CategoryID,
                CatName = cat.Name,
                ProductId = product.ProductID,
                ProductName = product.Name
            };
};
```



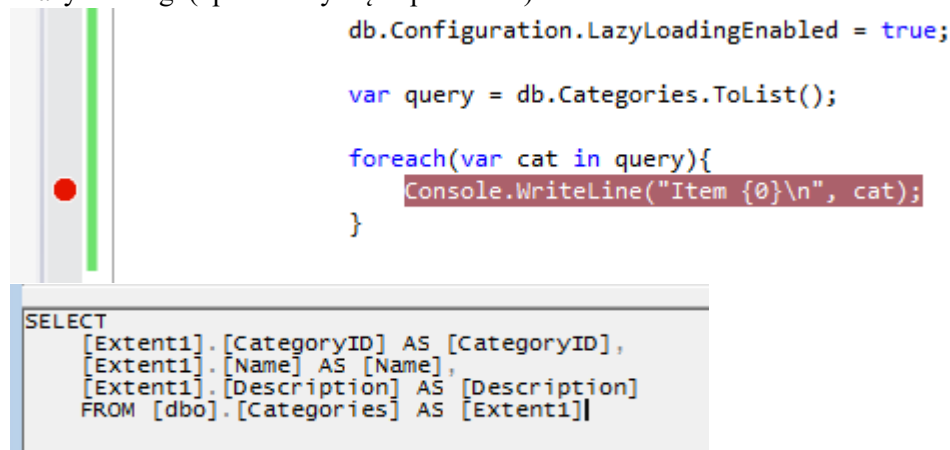
```
var query1 = db.Categories.Join(
    db.Products,
    cat => cat.CategoryID,
    product => product.CategoryID,
    (cat, product) => new
    {
        CatID = cat.CategoryID,
        CatName = cat.Name,
        ProductId = product.ProductID,
        ProductName = product.Name
    });
};
```

- Navigation property

```
var query = from cat in db.Categories
             select new
             {
                 catId = cat.CategoryID,
                 catName = cat.Name,
                 prods = cat.Products
             };

foreach (var cat in query)
{
    foreach (var prod in cat.prods)
    {
        Console.WriteLine("Item {0} {1} {2}\n", prod.ProductID, prod.Name, cat.catName);
    }
}
```

-Lazy loading (upewniamy się w profilerze)



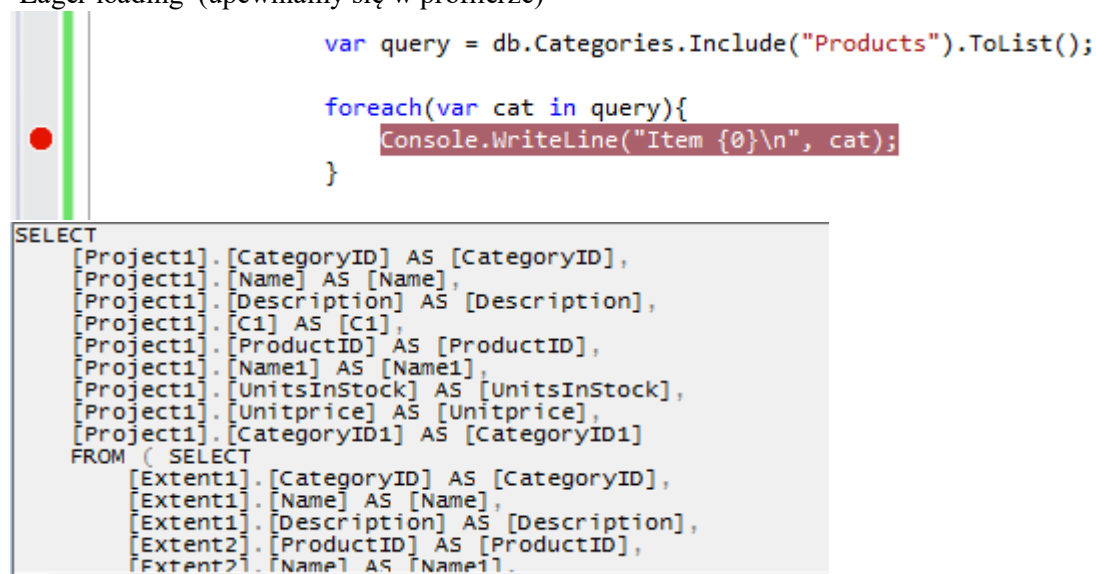
```
db.Configuration.LazyLoadingEnabled = true;

var query = db.Categories.ToList();

foreach(var cat in query){
    Console.WriteLine("Item {0}\n", cat);
}
```

```
SELECT
    [Extent1].[CategoryID] AS [CategoryID],
    [Extent1].[Name] AS [Name],
    [Extent1].[Description] AS [Description]
FROM [dbo].[Categories] AS [Extent1]
```

-Eager loading (upewniamy się w profilerze)



```
var query = db.Categories.Include("Products").ToList();

foreach(var cat in query){
    Console.WriteLine("Item {0}\n", cat);
}
```

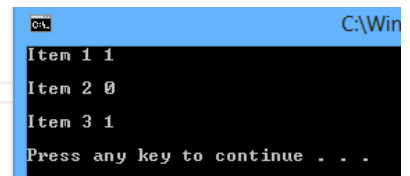
```
SELECT
    [Project1].[CategoryID] AS [CategoryID],
    [Project1].[Name] AS [Name],
    [Project1].[Description] AS [Description],
    [Project1].[C1] AS [C1],
    [Project1].[ProductID] AS [ProductID],
    [Project1].[Name1] AS [Name1],
    [Project1].[UnitsInStock] AS [UnitsInStock],
    [Project1].[Unitprice] AS [Unitprice],
    [Project1].[CategoryID1] AS [CategoryID1]
FROM ( SELECT
        [Extent1].[CategoryID] AS [CategoryID],
        [Extent1].[Name] AS [Name],
        [Extent1].[Description] AS [Description],
        [Extent2].[ProductID] AS [ProductID],
        [Extent2].[Name] AS [Name1],
        [Extent1].[UnitsInStock] AS [UnitsInStock],
        [Extent1].[Unitprice] AS [Unitprice],
        [Extent1].[CategoryID] AS [CategoryID1]
    FROM [dbo].[Categories] AS [Extent1]
    INNER JOIN [dbo].[Products] AS [Extent2] ON [Extent1].[CategoryID] = [Extent2].[CategoryID]
    ) AS [Project1]
```

Wszystkie powyższe zapytania dają oczekiwane rezultaty.

iii) Dla każdej kategorii pokaż liczbę produktów (jeśli dla kategorii brak produktu - wyświetl 0)

- Method syntax

```
var query = db.Categories.Include("Products");
foreach (var cat in query)
{
    Console.WriteLine("Item {0} {1}\n", cat.CategoryID, cat.Products.Count());
}
```



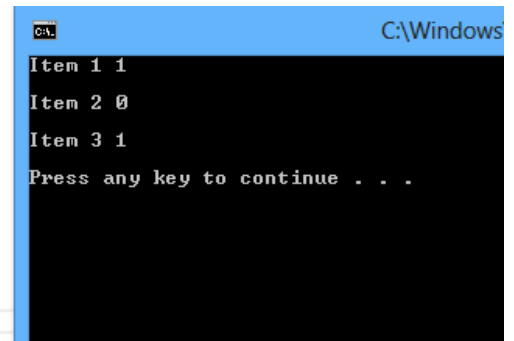
C:\Win

```
Item 1 1
Item 2 0
Item 3 1
Press any key to continue . . .
```

- Query syntax

```
var query = from cat in db.Categories.Include("Products")
            select new
            {
                catId = cat.CategoryID,
                count = cat.Products.Count()
            };

foreach (var cat in query)
{
    Console.WriteLine("Item {0} {1}\n", cat.catId, cat.count);
}
```



C:\Windows

```
Item 1 1
Item 2 0
Item 3 1
Press any key to continue . . .
```

Tym samym kończymy zadania laboratoryjne, jako zadanie domowe tworzymy aplikację webową z użyciem Asp.Net Core Web API oraz frameworka Angular.