 **INSTYTUT PODSTAW INFORMATYKI POLSKIEJ AKADEMII NAUK**

***PROGRAMOWANIE NA PLATFORMIE .NET***

**Programowanie aplikacji samodzielnych (WPF) - Warsztat 4**

### Materiały do zajęć

**Entity Framework Code First to a New Database**

<https://msdn.microsoft.com/en-us/library/jj193542(v=vs.113).aspx>

**Entity Framework Code First Migrations**

<https://msdn.microsoft.com/en-us/library/jj591621(v=vs.113).aspx>

**Entity Framework Code First In 15 Minutes**

<https://www.developerhandbook.com/entity-framework/entity-framework-code-first-in-15-minutes/>

1. Nowy projekt:  
     
   **Aplikacja WPF (.NET Framework)**  
   Nazwa: **SuperQuick**
2. Dodać do projektu pakiet NuGet: **Entity Framework**  
     
   a) Zarządzaj pakietami NuGet > Przeglądaj  
   lub:  
   b) Narzędzia > Menedżer Pakietów NuGet > Konsola Menedżera Pakietów:  
    Install-Package EntityFramework
3. Dodać następujące klasy do projektu:  
   WAŻNE! W każdym pliku dodać  **public**  przed  **class**  
   * Address.cs
   * Order.cs
   * Customer.cs
   * SuperQuickContext.cs
   * SuperQuickInitializer.cs
   * MainWindowViewModel.cs
4. Klasa **Address**:

**public** class Address

{

public string Line1 { get; set; }

public string Line2 { get; set; }

public string Town { get; set; }

public string PostCode { get; set; }

public override string ToString()

{

return string.Format("{0}\n{1}\n{2}\n{3}",

Line1, Line2, Town, PostCode);

}

}

1. Klasa **Order**

**public** class Order

{

public int OrderId { get; set; }

public string Item { get; set; }

public double Amount { get; set; }

public int CustomerId { get; set; }

public virtual Customer Customer { get; set; }

}

1. Klasa **Customer**

**public** class Customer

{

public Customer()

{

Orders = new List<Order>();

}

public int CustomerId { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public Address Address { get; set; }

public virtual List<Order> Orders { get; set; }

public override string ToString()

{

return string.Format("{0}, {1}", LastName.ToUpper(), FirstName);

}

}

1. Klasa **SuperQuickContext** - uzupełnić brakujące wskazanie przestrzeni nazw   
   (na przykład używając szybkich akcji z podpowiedziami, “żółta żarówka”)

using **System.Data.Entity;**

namespace SuperQuick

{

**public** class SuperQuickContext : **DbContext**

{

//public SuperQuickContext(): base("Name=SuperQuickContext")

//{

//}

public **DbSet**<Customer> Customers { get; set; }

public **DbSet**<Order> Orders { get; set; }

}

}

1. Klasa **SuperQuickInitializer** - uzupełnić brakujące wskazanie przestrzeni nazw   
   (na przykład używając szybkich akcji z podpowiedziami, “żółta żarówka”)  
     
   Użyty poniżej dla inicjalizatora jest interfejs **DropCreateDatabaseIfModelChanges**. Określa strategię sposobu inicjalizacji bazy danych, wykonywaną przez biblioteki. Dostępne są ponadto interfejsy z innymi strategiami: **CreateDatabaseIfNotExists** oraz **DropCreateDatabaseAlways**.

**public** class SuperQuickInitializer  
 **: DropCreateDatabaseIfModelChanges<SuperQuickContext>**

{

protected override void Seed(SuperQuickContext context)

{

//Create some dummy data

Address addressOne = new Address

{

Line1 = "Address Line 1",

Line2 = "Address Line 2",

PostCode = "AB1 ABC",

Town = "The Town"

};

Address addressTwo = new Address

{

Line1 = "Second Address 1",

Line2 = "Second Address 2",

PostCode = "DE2 DEF",

Town = "Second Town"

};

Customer customerOne = new Customer

{

Address = addressOne,

FirstName = "Jon",

LastName = "Preece",

};

Customer customerTwo = new Customer

{

Address = addressTwo,

FirstName = "Mike",

LastName = "Smith"

};

// Orders below ...

Order order = new Order {

Amount = 10,

Item = "Mouse"

};

Order orderTwo = new Order {

Amount = 20,

Item = "Keyboard"

};

Order orderThree = new Order {

Item = "Monitor",

Amount = 100

};

customerOne.Orders.Add(order);

customerTwo.Orders.AddRange(new[] { orderTwo, orderThree });

//Add to the context

context.Customers.Add(customerOne);

context.Customers.Add(customerTwo);

//Save changes

context.SaveChanges();

}

}

1. Odszukać w drzewie projektu plik **App.xaml.cs** i dodać kod metody **OnStartup**.  
   Jeśli trzeba, uzupełnić również brakujące wskazanie przestrzeni nazw   
   (na przykład używając szybkich akcji z podpowiedziami, “żółta żarówka”)

using **System.Data.Entity;**

using System.Windows;

namespace SuperQuick

{

public partial class App : Application

{

protected override void **OnStartup**(StartupEventArgs e)

{

base.OnStartup(e);

Database.SetInitializer(**new SuperQuickInitializer()**);

}

}

}

1. Klasa **MainWindowViewModel** - uzupełnić brakujące wskazania przestrzeni nazw   
   (na przykład używając szybkich akcji z podpowiedziami, “żółta żarówka”)

using **System.Collections.ObjectModel;**

using **System.ComponentModel;**

namespace SuperQuick

{

public class MainWindowViewModel : **INotifyPropertyChanged**

{

protected readonly SuperQuickContext context;

public MainWindowViewModel()

{

context = new SuperQuickContext();

Customers = new **ObservableCollection**<Customer>(context.Customers);

}

public **ObservableCollection**<Customer> Customers { get; set; }

public event PropertyChangedEventHandler **PropertyChanged**;

private void **OnPropertyChanged**(

[System.Runtime.CompilerServices.CallerMemberName]

string propertyName = "")

{

PropertyChanged?.Invoke(this,

new PropertyChangedEventArgs(propertyName));

}

}

}

1. Plik **MainWindow.xaml**

<Window x:Class="SuperQuick.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

**xmlns:local="clr-namespace:SuperQuick"**

mc:Ignorable="d"

**Title="SuperQuick"** Height="350" Width="525">

<!--

<Window.DataContext>

<**local:MainWindowViewModel**/>

</Window.DataContext>

-->

<Grid>

<ScrollViewer>

<ItemsControl ItemsSource="{Binding Customers}"

AlternationCount="2"

ScrollViewer.CanContentScroll="True">

<ItemsControl.ItemTemplate>

<DataTemplate>

<StackPanel x:Name="Main">

<TextBlock Text="{Binding}"

FontWeight="Bold"

FontSize="14"

Padding="10,10,0,0" />

<TextBlock Text="{Binding Address}"

FontSize="12"

Padding="10,0,0,10" />

<StackPanel Orientation="Horizontal">

<TextBlock Text="Orders:"

Margin="0,0,5,0"

Padding="10,0,0,10"

FontWeight="Bold" />

<TextBlock Text="{Binding Orders.Count}"

Padding="0,0,0,10"/>

</StackPanel>

</StackPanel>

<DataTemplate.Triggers>

<Trigger Property="ItemsControl.AlternationIndex"

Value="0">

<Setter TargetName="Main"

Property="Background"

Value="#220000FF" />

</Trigger>

<Trigger Property="ItemsControl.AlternationIndex"

Value="1">

<Setter TargetName="Main"

Property="Background"

Value="White" />

</Trigger>

</DataTemplate.Triggers>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</ScrollViewer>

</Grid>

</Window>

1. Zbudować projekt, włączyć aplikację.  
   Nie powinno być widać jeszcze żadnych danych.  
     
   Pierwszy start aplikacji może być dość długi, ponieważ będzie tworzona baza danych oraz inicjalizowana początkowymi danymi.
2. W pliku **MainWindow.xaml** odblokować ustawienie **DataContext** jak poniżej, usuwając zaznaczone na niebiesko komentarze <!-- oraz -->

<!--

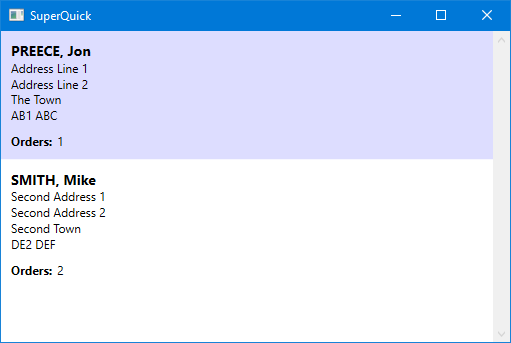
<Window.DataContext>

<**local:MainWindowViewModel**>

</Window.DataContext>

-->

1. Zbudować projekt, włączyć aplikację ponownie.





Gotowa aplikacja po rozbudowie:

<http://bit.ly/IPI-PAN-2018-V-superquick2>

### Dodatkowe materiały do zajęć

**Entity Framework Code First to a New Database**

<https://msdn.microsoft.com/en-us/library/jj193542(v=vs.113).aspx>

**Entity Framework Code First Migrations**

<https://msdn.microsoft.com/en-us/library/jj591621(v=vs.113).aspx>

**Database Connection in Entity Framework**

<https://www.tektutorialshub.com/database-connection-in-entity-framework/>

**Entity Framework Config File Settings**

<https://msdn.microsoft.com/en-us/data/jj556606.aspx>



<http://www.entityframeworktutorial.net/>

**Entity Framework Databinding with WPF**

<https://msdn.microsoft.com/en-us/library/jj574514(v=vs.113).aspx>

**Entity Framework MVVM Walk Through 1**

<https://code.msdn.microsoft.com/windowsdesktop/Entity-Framework-MVVM-Walk-959bb749>

**Create a simple data application with WPF and Entity Framework 6**

<https://msdn.microsoft.com/en-us/library/mt715492.aspx>