 **INSTYTUT PODSTAW INFORMATYKI POLSKIEJ AKADEMII NAUK**

***PROGRAMOWANIE NA PLATFORMIE .NET***

**Programowanie aplikacji samodzielnych (WPF) - Warsztat 1**

# **Windows Presentation Foundation (WPF) Application Development**

<https://channel9.msdn.com/Series/ConnectOn-Demand/208>

# **Visual Studio - Features - Windows Presentation Foundation**

[PL] <https://www.visualstudio.com/pl/vs/wpf/>

[EN] <https://www.visualstudio.com/vs/wpf/>

**What's new in C# 6.0**

<https://channel9.msdn.com/Series/ConnectOn-Demand/211>

**What's new in C# 7.0 - Connect(); // 2016**

<https://channel9.msdn.com/Events/Connect/2016/105>

### Zasoby na temat WPF

Introduction to WPF: <https://msdn.microsoft.com/en-us/library/mt149842.aspx>

Getting started WPF (MSDN): <https://msdn.microsoft.com/en-us/library/ms752299(v=vs.100).aspx>

WPF Walkthroughs (MSDN): <https://msdn.microsoft.com/en-us/library/ee649089(v=vs.100).aspx>

WPF resources (MSDN): <http://msdn.microsoft.com/en-us/library/ms754130.aspx>

Introduction to WPF in VS 2015 (MSDN): <http://msdn.microsoft.com/en-us/library/aa970268.aspx>

Walkthrough: My First WPF Desktop Application: <https://msdn.microsoft.com/en-us/library/ms752299(v=vs.110).aspx>

### Przewodniki na temat WPF

WPF Tutorial: <http://www.wpftutorial.net/>

The complete WPF tutorial: <http://www.wpf-tutorial.com/>

TutorialsPoint - WPF Tutorial: <https://www.tutorialspoint.com/wpf/>

TutorialsPoint - C# Tutorial: <https://www.tutorialspoint.com/csharp/>

### Model aplikacji samodzielnej w WPF

Introduction to WPF: <https://msdn.microsoft.com/en-us/library/mt149842.aspx>

* XAML markup - definicja w XML dla wyglądu widoku (strony) aplikacji:
  + elementy → obiekty (controls)
  + atrybuty → właściwości obiektów (properties)
* Code-Behind - zachowanie widoku, reakcja na zdarzenia
* Nawigacja między stronami
* Stylowanie klas kontrolek i wybranych elementów widoku
* Logika biznesowa obsługi danych - najlepiej w oddzielnych klasach, a nie w widoku
* Elegancki wygląd - środowisko Blend, animacje, definicje styli

#### Cechy wyróżniające WPF

* Data Binding - wiązania danych w widoku do źródeł danych
* Drzewo komponentów w pamięci - Live Visual Tree  
  <https://msdn.microsoft.com/pl-pl/library/mt270227.aspx>
* Attached Events - zdarzenia kontrolek zagnieżdżonych
* Wzorzec projektowy MVVM
* Routed Events, Routed Commands

### Wbudowane komponenty interfejsu użytkownika

**Kontrolki WPF według funkcji**

* **Buttons**: Button, ToggleButton and RepeatButton.
* **Data Display**: DataGrid, ListView,and TreeView.
* **Date Display and Selection**: Calendar and DatePicker.
* **Dialog Boxes**: OpenFileDialog, PrintDialog, and SaveFileDialog.
* **Digital Ink**: InkCanvas and InkPresenter.
* **Documents**: DocumentViewer, FlowDocumentPageViewer, FlowDocumentReader, FlowDocumentScrollViewer, and StickyNoteControl.
* **Input**: TextBox, RichTextBox, and PasswordBox.
* **Layout**: Border, BulletDecorator, Canvas, DockPanel, Expander, Grid, GridView, GridSplitter, GroupBox, Panel, ResizeGrip, Separator, ScrollBar, ScrollViewer, StackPanel, Thumb, Viewbox, VirtualizingStackPanel, Window, and WrapPanel.
* **Media**: Image, MediaElement, and SoundPlayerAction.
* **Menus**: ContextMenu, Menu, and ToolBar.
* **Navigation**: Frame, Hyperlink, Page, NavigationWindow, and TabControl.
* **Selection**: CheckBox, ComboBox, ListBox, RadioButton, and Slider.
* **User Information**: AccessText, Label, Popup, ProgressBar, StatusBar, TextBlock, and ToolTip.

### Model obsługi zdarzeń w WPF

Zdarzenia:

* obsługiwane w Code Behind widoku (strony)
* zdarzenia proste - bezpośrednio z danej kontrolki
* zdarzenia kontrolek zagnieżdżonych - Attached Events

**How to: Create a Simple Event Handler (MSDN):** <https://msdn.microsoft.com/en-us/library/bb675300(v=vs.100).aspx>

**How to: Use Attached Events (MSDN):** <https://msdn.microsoft.com/en-us/library/bb514616(v=vs.100).aspx>

Polecenia:

* standardowe: ApplicationCommands
* wydzielone w innych klasach niż Code Behind
* automatyczne wyłączanie i włączanie kontrolek, zależnie od dostępności polecenia
* specyfikowanie sposobów wywołania polecenia: skróty, gesty, itp.

**Understanding Routed Events and Commands In WPF:** <https://web.archive.org/web/20080905222741/http://msdn.microsoft.com/en-us/magazine/cc785480.aspx>

**WPF Tutorial - Routed Events:** <http://www.wpftutorial.net/RoutedEvents.html>

[wiktor.wandachowicz@p.lodz.pl](mailto:wiktor.wandachowicz@p.lodz.pl)

[wiktor.wandachowicz@gmail.com](mailto:wiktor.wandachowicz@gmail.com)

### Zasoby Google Drive

Folder **[IPI-PAN-2018-V]**

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

<https://drive.google.com/drive/folders/1ROG-JC0eRruS5j6YlXCuJNDDAH-tzx4Q?usp=sharing>

**Ćwiczenia - Warsztaty 1**

Folder **[wpf1]**

plik: ShopLib.zip

1. Utworzyć nowy projekt C# / WPF Application o nazwie **ShopApp**
2. Połączyć z projektem ShopApp istniejący projekt **ShopLib** w ramach tego samego solution

* ShopLib: definicje klas danych Client i Item oraz kontener danych Shop zawierający logikę biznesową
* ShopApp: dodać referencję do projektu ShopLib

1. Dodać nowy projekt C# / Console Application o nazwie **ShopConsole**

* ShopConsole: dodać referencję do projektu ShopLib

1. W programie głównym umieścić kod:

class Program

{

static void Main(string[] args)

{

Shop shop = new Shop();

Client client1 = shop.RegisterClient("Jan", "Kowalski",

"jk@example.com");

shop.AddItem(client1, new Item {

ProductName = "Wireless Keyboard",

UnitPrice = 89.90m,

Amount = 2 });

shop.AddItem(client1, new Item("Wireless Mouse", 44.90m));

Display(shop, client1);

shop.RemoveItemById(client1, 1);

Display(shop, client1);

Console.ReadLine();

}

private static void Display(Shop shop, Client client)

{

Console.WriteLine("Client: " + client.Name);

foreach (Item item in client.Items)

{

Console.WriteLine(item.ToString());

}

Console.WriteLine("Total: " + shop.TotalPriceFor(client));

Console.WriteLine();

}

}

1. Ustawić projekt jako główny ShopConsole > Set as StartUp Project i uruchomić
2. Dodać definicje kolumn i wierszy siatki w MainWindow.xaml - jak poniżej:

<Grid.ColumnDefinitions>

<ColumnDefinition Width="50\*"/>

<ColumnDefinition Width="50\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

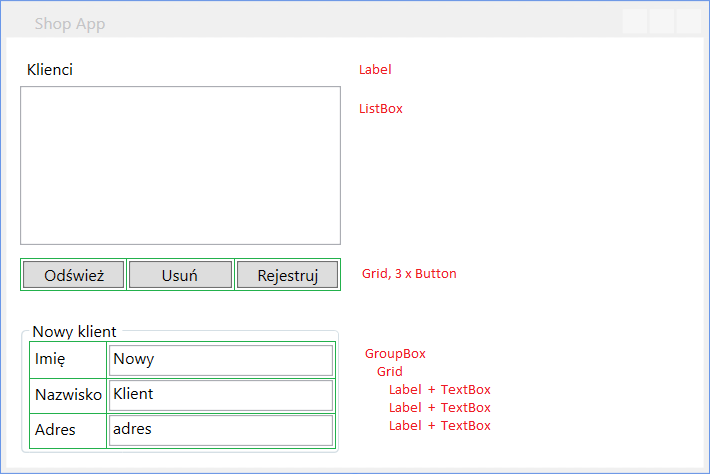
<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

1. Przygotować interfejs użytkownika do ręcznego wyświetlania i edycji klientów:

* labelKlient - kontrolka Label (etykieta)
* listBoxKlienci - kontrolka ListBox (lista klientów)
* zagnieżdżona siatka Grid z trzema kolumnami (miejsce na przyciski)
* buttonKlienci, buttonUsunKlienta, buttonRejestrujKlienta   
  - przyciski odświeżania, usuwania i rejestracji (klientów)
* groupBoxKlient - kontrolka GroupBox (dane nowego klienta)
* zagnieżdżona siatka - trzy kolumny x dwa wiersze (formularz)
* labelImie, labelNazwisko, labelAdres - kontrolki Label (etykiety)
* textBoxImie, textBoxNazwisko, textBoxAdres - kontrolki TextBox (pola edycyjne)



1. Ustawić projekt jako główny ShopApp > Set as StartUp Project i uruchomić
2. Sprawdzić jak zachowują się kontrolki przy zmianie rozmiaru okna
3. Jeśli trzeba, ustawić właściwości kontrolek:

* Grid.Column="*pozycja*" oraz Grid.Row="*pozycja*"
* zamiast *pozycja* podać właściwe wartości (numerowane od 0)
* Margin="*lewy*,*górny*,*prawy*,*dolny*" lub Margin="*margines*"

1. Jeśli trzeba, usunąć właściwości kontrolek:

* HorizontalAlignment
* VerticalAlignment
* Width
* Height

1. Dodać pole dla obiektu Shop w Code-Behind widoku (wskazana dyrektywa: using ShopLib), np.

private Shop shop = new Shop();

1. Dodać obsługę odświeżania listy klientów z kolekcji obiektu Shop, np.

* czyścić ListBox.Items
* iterować po elementach kolekcji klientów, dodając ich obiekty do ListBox.Items

private void OdswiezKlientow()

{

listBoxKlienci.Items.Clear();

foreach (Client client in shop.Clients)

{

listBoxKlienci.Items.Add(client);

}

}

1. Dodać szablon elementu listy ListBox, np.

<ListBox x:Name="listBoxKlienci" Margin="10,0,10,0"

Grid.Column="0" Grid.Row="1">

**<ListBox.ItemTemplate>**

**<DataTemplate>**

**<StackPanel Margin="0">**

**<Label Content="{Binding Name}" FontWeight="Bold"/>**

**<Label Content="{Binding Address}"/>**

**</StackPanel>**

**</DataTemplate>**

**</ListBox.ItemTemplate>**

</ListBox>

Patrz także: <http://www.ikriv.com/dev/wpf/TextStyle/>

1. Dodać obsługę zdarzenia Click dla przycisku odświeżania:

* wywoływać OdswiezKlientow();

1. Dodać obsługę zdarzenia Click dla przycisku rejestracji klientów:

* tworzyć obiekt klienta korzystając z metod klasy Shop
* wywoływać OdswiezKlientow();

1. Dodać obsługę zdarzenia Click dla przycisku usuwania klientów:

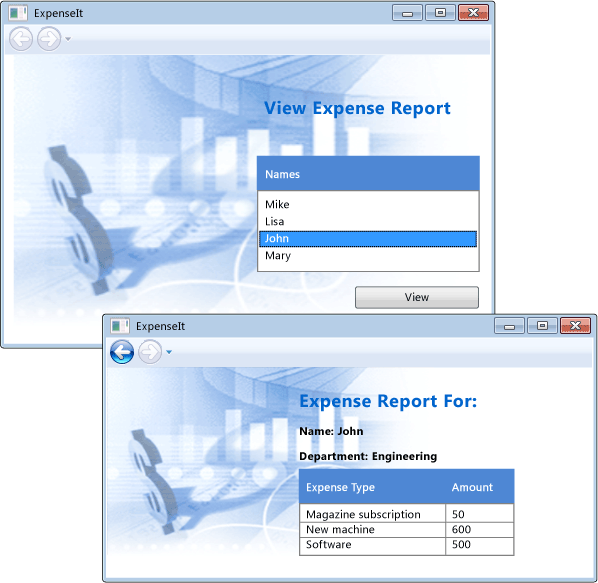
* odczytywać i rzutować obiekt klienta korzystając z ListBox.SelectedItem
* usuwać obiekt klienta korzystając z metod klasy Shop
* wywoływać OdswiezKlientow();

**Przewodnik - zajęcia 2**

Walkthrough: Getting Started with WPF

Projekt: **ExpenseIt**

<https://msdn.microsoft.com/en-us/library/ms752299(v=vs.100).aspx>



Folder **[wpf1]**

plik: watermark.png



#### Sortowanie danych w XmlDataProvider

Use XML Files with WPF: <https://weblogs.asp.net/psheriff/use-xml-files-with-wpf>