

Windows Presentation Foundation (WPF) Лекція 06

- 1. Поняття системи WPF
- 2. Архітектура WPF
- 3. Створення проекту з графічним інтерфейсом
- 4. Елементи керування WPF
- 5. Режим дизайну
- 6. Властивості
- 7. Приклад

WPF (Windows Presentation Foundation) - це потужний інструментарій для розробки візуально привабливих та функціональних застосунків для Windows. Він надає можливість створювати візуально привабливі та функціональні інтерфейси користувача з 3D-анімацією, різноманітними кольорами та ефектами, при цьому зменшуючи складність коду.

Переваги:

- WPF використовує векторну графіку, що дозволяє створювати інтерфейси, які масштабуються без втрати якості на різних екранах.
- WPF використовує апаратне прискорення сучасних графічних карт.
- WPF використовує мову розмітки XAML для опису інтерфейсу користувача.
- WPF надає широкі можливості для налаштування та розширення функціональності

WPF є невід'ємною частиною .NET Framework та служить для створення сучасних графічних інтерфейсів. На відміну від традиційних WinFormsдодатків, де за відображення елементів інтерфейсу та графіки відповідали компоненти операційної системи (User32, GDI), WPF використовує DirectX для рендерингу.

Ключова відмінність WPF полягає в тому, що значна частина роботи з візуалізації, від простих кнопок до складних 3D-моделей, перекладається на графічний процесор, що забезпечує апаратне прискорення графіки та підвищує продуктивність. Розробка інтерфейсу в WPF базується на декларативній мові розмітки XAML (Extensible Application Markup Language), використовується для опису елементів інтерфейсу. XAML базується на XML(Extensible Markup Language) та дозволяє створювати складні UI-структури декларативно.

XML використовується для зберігання та передачі даних. XAML є одним із варіантів використання XML, адаптованим спеціально для опису інтерфейсу WPF.

Архітектура WPF



Компоненти архітектури WPF



Cuesta a new nucleat			_	
Create a new project		for templates (Alt+S)		
Recent project templates	C#	 All platforms All proj 	ject types	Ŧ
IN WPF Custom Control Library C#		WPF User Control Library A project for creating a user control library for .NET WPF Applications		•
Console App (.NET Framework) C#		C# Windows Desktop Library		
WPF App(.NET Framework)		WPF App (.NET Framework) Wind (yes Presentation Foundation client application C# XAML Windows Desktop		١,
	Ľ٦	NUnit 3 Test Project A project that contains NUnit tests that can run on .NET on Windows, macOS	Linux and	
	 c:\	C# LINUX macOs Windows NUNIT Test Console App (.NET Framework) A project for creating a command-line application C# Windows Console		
		Class Library (.NET Framework) A project for creating a C# class library (.dll) C# Windows Library		
	「「」	Unit Test Project (.NET Framework) A project that contains MSTest unit tests.		Ŧ
		Back	N	lext

Першим кроком є створення нового проекту. Для цього потрібно обрати WPFApp(.NET Framework)



Visual Studio за замовчуванням створює два файли

Файл XAML (MainWindow.xaml) Файл CS (MainWindow.xaml.cs)

У вікнах ХАМL такі теги записуються за замовчуванням.

Сітка є першим елементом за замовчуванням.

ХАМL використовує простий синтаксис для визначення елементів інтерфейсу користувача та їхніх властивостей.

Кожен елемент, схожий на елементи XML, укладений у початковий і кінцевий теги (наприклад, <Window>...</Window>) або може бути представлений у самозакриваючій формі (наприклад, <Window />).

Елемент XAML відповідає певному класу С#. Наприклад, елемент XAML < Button > асоціюється з System. Windows. Controls. Button класом у C#.

Атрибути елемента XAML відображають властивості відповідного класу С#.

Файл MainWindow.xaml. Простір імен XAML

<Window x:Class="WpfApp1.MainWindow"</pre>

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

Базовий простір імен, який охоплює всі класи WPF, включаючи елементи управління, які застосовуються при побудові інтерфейсу користувача. Так як цей простір імен оголошено без префікса, то він поширюється на весь ХАМL-документ.

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

- це простір імен ХАМІ. Він включає різні властивості утиліт ХАМІ, які дозволяють впливати на те, як ХАМІ-документ слід інтерпретувати.

```
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:local="clr-namespace:WpfApp1"
mc:Ignorable="d"
Title="MainWindow" Height="450" Width="800">
<Grid>
```

</Grid>

</Window>

Префікс x, що використовується, у визначенні xmlns:x означає, що ті властивості елементів, які укладені в цьому просторі імен, будуть використовуватися з префіксом x - x:Name або x:Key.

x:Class="XamlApp.MainWindow" - тут створюється новий клас MainWindow і відповідний файл коду, куди буде прописуватися логіка для даного вікна програми.

!!! Ці простори імен не еквівалентні тим просторам імен, які підключаються за допомогою директиви using с#.

MainW	indow.xan	nl	MainWindow.xaml.cs 👳 🗙 📼 👻	Solution Explorer
💷 Wpf	App1		- 😚 WpfApp1.MainWindow - 😚 MainWindow() - 💠	/0 · ≒ () 🗐 🗗 ↔
	1 😨) ~	using System;	Search Solution Explorer (Ctrl+1)
	2		using System.Collections.Generic;	
	3		using System.Linq;	Solution 'Lec_6' (1 of 1 project
	4		using System.Text;	▲ I WptApp1
	5		using System.Threading.Tasks;	Properties
	6		using System.Windows;	▷ Patr References
	7		using System.Windows.Controls;	App.config
	8		using System.Windows.Data;	App.xaml
	9		using System.Windows.Documents;	App.xaml.cs
	10		using System.Windows.Input;	A 🔝 MainWindow.xaml
	11		using System.Windows.Media;	MainWindow.xaml
	12		using System.Windows.Media.Imaging;	
	13		using System.Windows.Navigation;	
	14		using System.Windows.Shapes;	
	15	L		
	16	\sim	namespace WpfApp1	
	17		{	
	18	- ×	/// <summary></summary>	
	19		<pre>/// Interaction logic for MainWindow.xaml</pre>	
	20		///	
		-	2 references	
81	21	Ý	public partial class MainWindow : Window	
	22		{	
			0 references	
	23	Ň	public Mainwindow()	
	24			
	25		InitializeComponent();	
	26	-	3	
	27	-	3	
110 %	78		} No issues found 🛛 😽 🔹 🕨 Ln: 1 Ch: 1 SPC CRIF	

MainWindow.xaml основне вікно дизайну . Для опису дизайну використовуємо xaml

MainWindow.xaml.cs містить відповідний код, який описує функціонал.



У панелі інструментів обрати TextBlock.

Перетягнути обраний елемент до вікна дизайну.

TextBlock з'явиться у вікні дизайну.

В коді XAML можна побачити доданий TextBlock.



Натисніть кнопку Пуск. З'явиться вікно, де знаходиться надпис: «Перша програма» Змініть текст на «Перша програма».

MainWindow		-	×
	_		
	Перша програма		

Створення кнопки

Створення кнопки в XAML:

- Додайте елемент < Button > в потрібне місце.
- Задайте атрибути кнопки:
 - ✓ Content -текст, який відображається на кнопці.
 - ✓ Розміри кнопки.
 - ✓ Click обробник події натискання на кнопку.

<Button Content="Клацніть по мені" Margin="208,165,208,217" Click="Button_Click"/>

Створення обробника події Click в C#:

- У файлі С# створіть метод-обробник події Click. Цей метод буде викликатися при натисканні на кнопку.
- Усередині методу напишіть код, який буде виконуватися при натисканні, наприклад, виведення повідомлення.

MessageBox.Show("Привіт, мій перший Windows Presentation Foundation!");

Властивості

Режим дизайну в WPF дозволяє візуально створювати та редагувати інтерфейс користувача за допомогою WYSIWYG-редактора (What You See Is What You Get).

Кнопка має багато властивостей, які дозволяють налаштувати її зовнішній вигляд та поведінку: Content- текст або інший вміст кнопки. Background- колір фону кнопки. Foreground- колір тексту кнопки. FontFamily- шрифт тексту кнопки. FontSize- розмір тексту кнопки. Width- Ширина кнопки. Height- висота кнопки. Margin- відстань від кнопки до інших елементів. Padding- внутрішній відступ кнопки. IsEnabled - визначає, чи доступна кнопка для натискання. Visibility - визначає, чи відображається кнопка.



Створення кнопки



Режим дизайну

Режим дизайну в WPF надає візуальний інтерфейс для створення інтерфейсів користувача. Ключові аспекти:



Дизайн користувальницького інтерфейсу



Організація елементів керування на екрані. Grid, StackPanel, Canvas

Створення стилів для уніфікованого вигляду.



ູ້ເອີ້ Прив'язка даних

Прив'язка властивостей елементів керування до ViewModel для оновлень, для динамічного оновлення інтерфейсу

Режим дизайну

Основні теги XAML:

<Window> - кореневий елемент, що представляє вікно програми;

<Grid> - контейнер для розміщення елементів у вигляді сітки;

<<u>StackPanel></u> - контейнер для розміщення елементів один за одним (вертикально або горизонтально);

<DockPanel> - контейнер для розміщення елементів з прив'язкою до країв контейнера;

<Canvas> - контейнер для розміщення елементів з абсолютним позиціонуванням;

<Button> - кнопка; <Label> - текстовий напис; <TextBox> - поле для введення тексту; <Image> - зображення; <ListBox> - список елементів; <Grid x:Name="lecture_6" Height="434" VerticalAlignment="Center"
Background="#FFEDEDE3">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center"
VerticalAlignment="Center">

<Button Content="Button 1" Margin="5" Width="131"/>
<Button Content="Button 2" Margin="5" Width="131"/>
<Button Content="Button 3" Margin="5" Width="131"/>
</StackPanel>

</Grid>



Режим дизайну у С#

```
public MainWindow(){
    InitializeComponent(); CreateButtons();
}
private void CreateButtons(){
    StackPanel stackPanel = new StackPanel();
    stackPanel.Orientation = Orientation.Vertical;
    stackPanel.HorizontalAlignment = HorizontalAlignment.Center;
    stackPanel.VerticalAlignment = VerticalAlignment.Center;
```

```
Button button1 = new Button();
button1.Content = "Button 1";
button1.Margin = new Thickness(5);
Button button2 = new Button();
button2.Content = "Button 2";
button2.Margin = new Thickness(5);
```

stackPanel.Children.Add(button1);
stackPanel.Children.Add(button2);
lecture_6.Children.Add(stackPanel);

Button 1

Button 2

Режим дизайну

DockPanel

<DockPanel>

<Button DockPanel.Dock="Top" Height="30" Click="Button_Click">Верхня
панель</Button>

<Button DockPanel.Dock="Bottom" Height="30" Click="Button_Click_3">Нижня
панель</Button>

<Button DockPanel.Dock="Left" Width="100" Click="Button_Click_1">Ліва панель</Button>

<Button DockPanel.Dock="Right" Width="100" Click="Button_Click_2">Права панель</Button>

<TextBox Text="DockPanel- це контейнер макету у WPF, який дозволяє poзташовувати дочірні елементи вдовж країв панелі. Він також може заповнювати простір, що залишився, останнім дочірнім елементом. Це робить його корисним для створення макетів з такими елементами, як панелі інструментів, бічні панелі та рядки стану."

```
TextAlignment="Center"
VerticalContentAlignment="Center"
FontSize="16"
Foreground="Blue"
TextWrapping="Wrap"/>
</DockPanel>
```

Результат використання DockPanel

Iecture_6			×
	Верхня панель		
Ліва панель	DockPanel- це контейнер макету у WPF, який дозволяє розташовувати дочірні елементи вдоль країв панелі (зверху, знизу, ліворуч чи праворуч). Він також може заповнювати простір, що залишився, останнім дочірнім елементом. Це робить його корисним для створення макетів з такими елементами, як панелі інструментів, бічні панелі та рядки стану.	Права п	анель
	Нижня панель		

WrapPanel— це елемент керування макету у WPF, який має дочірні елементи і розташовує їх у послідовному порядку, зліва направо, з перенесенням на новий рядок, коли простір у поточному рядку закінчується.

Розмір дочірніх елементів WrapPanel не змінює. Він просто розташовує їх у доступному просторі.

WrapPanel не додає додаткових пробілів між дочірніми елементами. Якщо ви хочете додати пробіли, потрібно встановити поля або відступи для дочірніх елементів.

Це корисно для створення гнучких макетів, які адаптуються до різних розмірів вікна.

Режим дизайну

<WrapPanel>

<TextBlock Text="..." Margin="5" Padding="5" Background="LightBlue" FontSize="16"
TextWrapping="Wrap"/>

<TextBlock Text="..." Margin="5" Padding="5" Background="LightGreen" FontSize="16"
TextWrapping="Wrap" />

<TextBlock Text="..." Margin="5" Padding="5" Background="LightCoral" FontSize="16"
TextWrapping="Wrap"/>

<TextBlock Text="..." Margin="5" Padding="5" Background="LightYellow"
FontSize="16" TextWrapping="Wrap"/>

</WrapPanel>

Результат використання WrapPanel

100 I I	I + · · · 6	
	lecture (

5 DI 🔽 🛛 🐺 | 🐼 🛞 | 🖉 <

Windows Presentation Foundation — це платформа для створення клієнтських програм Windows із візуальним інтерфейсом користувача. WPF підтримує широкий спектр функцій розробки додатків, включаючи інтерфейс користувача, 2D/3D-графіку, фіксовані та адаптивні документи, типографіку, векторну графіку, анімацію, прив'язку даних, аудіо та відео.

WPF використовує розширювану мову розмітки додатків (XAML) для визначення інтерфейсів користувача. XAML — це декларативна мова, яка дозволяє розробникам описувати елементи інтерфейсу користувача та їх властивості у структурованому форматі на основі XML. XAML спрощує створення та підтримку складних інтерфейсів користувача.

WPF надає потужну систему макета, яка дозволяє розробникам створювати адаптивні інтерфейси користувача, які можуть адаптуватися до різних розмірів екрану та роздільної здатності. WPF також пропонує широкий спектр вбудованих елементів управління, таких як кнопки, текстові поля та списки, які можна використовувати для створення інтерактивних інтерфейсів користувача.

Прив'язка даних — це потужна функція WPF, що дозволяє інтерфейс користувача автоматично оновлюватись, коли змінюються дані.



Canvas- це панель компонування WPF, яка дозволяє розміщувати дочірні елементи з використанням абсолютних координат. Це означає, що ви вказуєте точне положення кожного елемента всередині Canvas від його верхнього лівого кута.

<Canvas>

<Rectangle Fill="Red" Width="50" Height="50" Canvas.Left="10"
Canvas.Top="20" />
 <Ellipse Fill="Blue" Width="50" Height="50" Canvas.Left="700"
Canvas.Top="20" />
 <TextBlock Text="Привіт, Canvas!" Canvas.Left="450" Canvas.Top="300"
FontSize="26" />
 <Polygon Points="100,150 150,250 50,250" Fill="Green" />
 <Ellipse Fill="Yellow" Width="180" Height="150" Canvas.Left="400"
Canvas.Top="100" />
 </Canvas>

Результат використання Canvas



Приклад

Створити віконний додаток для підрахунку суми двох значень, наприклад а+в =?

Для цього потрібно:

- Створити вікно з текстом і текстовими полями для введення чисел А та В, кнопку "Обчислити" і поле для відображення результату.
- Використовуємо Grid для організації елементів інтерфейсу.
- Призначимо обробник події Button_Click для кнопки.

Число А:			
5			
Число В:			
-8			
Результат:			
-3			
	Обчислити		

Приклад

```
<Grid HorizontalAlignment="Center" Width="800">
     <Label Content="Число A:" HorizontalAlignment="Left" Margin="25,25,0,0"
VerticalAlignment="Top"/>
     <TextBox x:Name="A" HorizontalAlignment="Left" Margin="25,50,0,0"</pre>
```

```
<TextBox x:Name="A" HorizontalAlignment="Left" Margin="25,50,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="200"/>
```

<Label Content="Число B:" HorizontalAlignment="Left" Margin="25,100,0,0"
VerticalAlignment="Top"/>

```
<TextBox x:Name="B" HorizontalAlignment="Left" Margin="25,125,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="200"/>
```

<Label Content="Peзультат:" HorizontalAlignment="Left" Margin="25,175,0,0"
VerticalAlignment="Top"/>

<TextBlock x:Name="txtResult" HorizontalAlignment="Left" Margin="25,200,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="200"/>

```
<Button Content="0бчислити" Click="Button_Click" Margin="25,246,0,130"
HorizontalAlignment="Left" Width="260"/>
</Grid>
```

Windows Presentation Foundation

Easy WPF in C# Windows Presentation Foundation for Beginners -

https://www.udemy.com/course/easy-wpf-in-c-windows-presentation-foundation-forbeginners/?utm_source=adwords&utm_medium=udemyads&utm_campaign=Search_DSA_Bet a_Prof_la.EN_cc.ROW-English&campaigntype=Search&portfolio=ROW-English&language=EN&product=Course&test=&audience=DSA&topic=&priority=Beta&utm_ content=deal4584&utm_term=_.ag_162511579404_.ad_696197165418_.kw_.de_c_.dm_._ pl_._ti_dsa-

1677053911088_._li_9194581_._pd__._&matchtype=&gad_source=1&gclid=CjwKCAiAqrG9BhA VEiwAaPu5zvdJnWD-tnCGKeN3I-xRgIP4OPVnCC4r3pR8z8DDyCtcypZGzmK-6BoCDX8QAvD_BwE&couponCode=2021PM25

Easy WPF in C# Windows Presentation Foundation for Beginners -

https://www.udemy.com/course/easy-wpf-in-c-windows-presentation-foundation-forbeginners/?couponCode=2021PM25