Fantin Francesco – IV E

Windows Presentation Foundation

Che Cos'è WPF?

WPF (acronimo di Windows Presentation Foundation) è un **framework** usato per lo sviluppo grafico di applicazioni client desktop di Windows. L'ambiente di sviluppo WPF, incluso all'interno del .NET Framework, offre un'ampia gamma di funzionalità per la creazione di applicazioni e un'interfaccia composta da 3D, animazione, colori intensi con una complessità minima del codice. WPF è preinstallato in Windows Vista, 7, 8 e 10, e può essere installato anche su Windows XP e Server 2003.

Nel 2018, Microsoft ha reso open source sia Windows Presentation Foundation sia Windows Form, provvedendo a caricare su GitHub il codice sorgente.

Breve Storia

La prima versione di WPF (versione 3.0) è stata lanciata nel 2006 e ha cambiato radicalmente l'approccio alla creazione di applicazioni Windows.

Prima di WPF, infatti, gli sviluppatori Windows hanno utilizzato per oltre 15 anni la stessa tecnologia di visualizzazione. Questo perché ogni tradizionale applicazione realizzata in Windows Form o altre librerie grafiche del .NET si basa su due componenti messi a disposizione dal sistema operativo per creare l'interfaccia utente: User 32 e GDI/GDI+. User 32 è in grado di fornire l'aspetto tradizionale di Windows per elementi come finestre, pulsanti, caselle di testo e così via. Invece, GDI / GDI + fornisce supporto per il disegno di rendering di forme, testo e immagini.

Con lo scopo di aggirare i limiti delle due librerie, Microsoft ha introdotto Direct X. Direct X è un sistema grafico ad alte prestazioni, rivolto inizialmente ai giochi e agli ambienti grafici. Per la sua realizzazione, Microsoft ha lavorato a stretto contatto con i fornitori di schede video per fornire a DirectX l'accelerazione hardware necessaria per trame complesse ed effetti speciali come trasparenza parziale e grafica tridimensionale.

A partire dal suo lancio nel 1995, DirectX è maturato e attualmente rappresenta una parte integrante di Windows, con supporto per tutte le schede video moderne. A causa delle performance migliori, le librerie DirectX sono state scelte come tecnologia grafica per WPF. Di conseguenza, ogni applicazione WPF usa DirectX, indipendentemente dal tipo di interfaccia utente creata. Questo significa che anche le banali applicazioni aziendali possono utilizzare ricchi effetti come trasparenza e anti-aliasing.

DirectX risulta più efficiente di GDI perché comprende proprietà di livello superiore come trame e sfumature che vengono processate direttamente dalla scheda video. Al contrario, GDI è costretto a convertirle pixel per pixel tramite istruzioni che vengono eseguite molto più lentamente dalle moderne schede video.

Un componente che è ancora usato è User32: infatti, WPF fa ancora affidamento su User32 per determinati servizi, come la gestione dell'input e dello spazio occupato sullo schermo da ogni applicazione, oltre allo scambio di messaggi tra applicazioni.

WPF vs. Windows Forms

Windows Forms è l'interfaccia grafica del .NET Framework ed è stato per anni il punto di riferimento per la creazione di applicazioni Windows. I vantaggi derivanti dall'uso di Windows Forms erano legati alla sua facilità d'uso, alla gestione dei controlli e alla scrittura del codice. Tuttavia, con l'avvento di WPF, Windows Forms è passato in secondo piano e non riceve più aggiornamenti dal 2005.

In particolare, molte aziende hanno iniziato ad abbandonarlo a causa di diversi svantaggi, tra i quali:

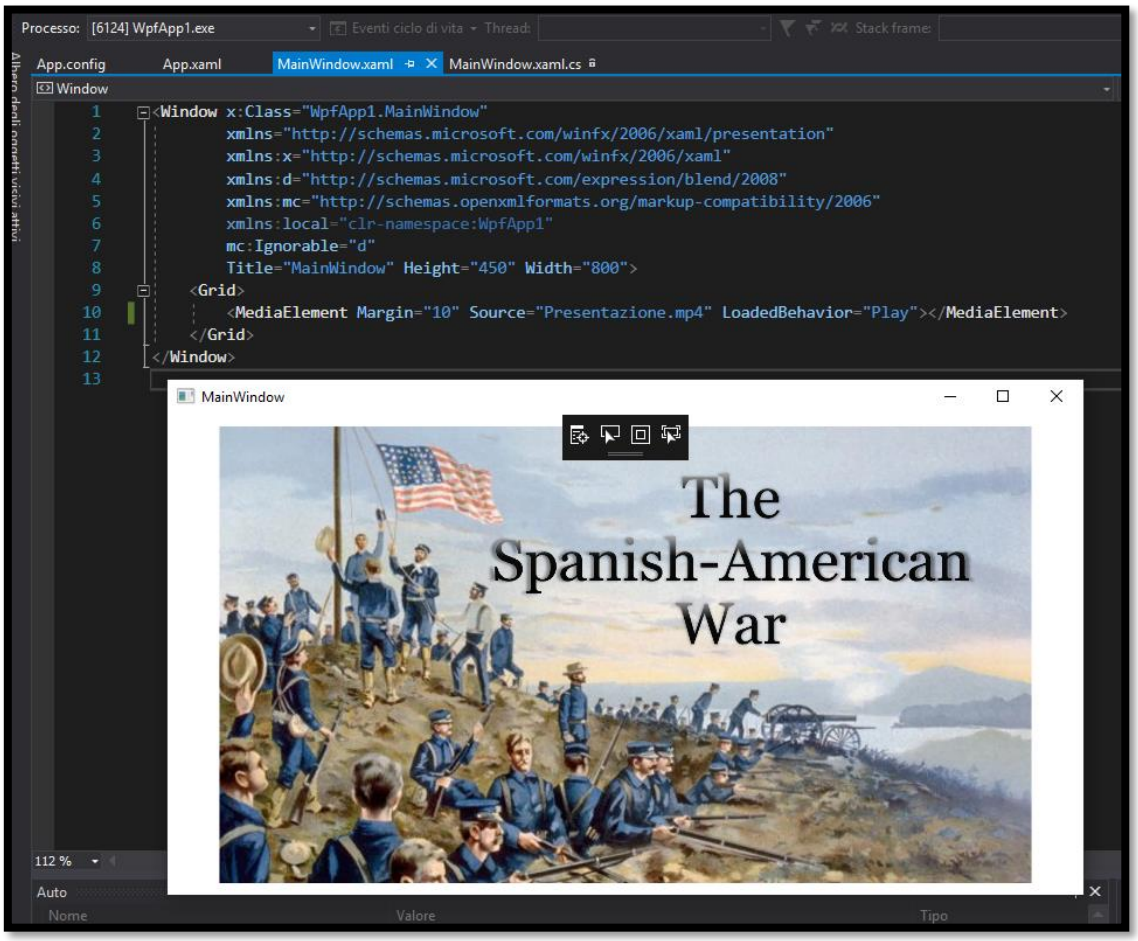
- La presenza di interfacce utente poco personalizzabili e meno accattivanti rispetto a WPF;
- Difficoltà nell'inserimento e nella gestione di contenuti audio e multimediali;
- Interfacce utente difficili da adattare (non scalabili) a seconda della dimensione occupata dalla finestra sullo schermo (a causa della tecnologia grafica utilizzata);
- La mancanza di supporto per creare interfacce per applicazioni desktop e web;
- Data binding difficile da applicare.

- Figura 1.1: Differenza tra un'interfaccia creata con Windows Forms (a sinistra) e un'interfaccia creata con WPF (a destra). Con WPF è possibile creare interfacce altamente personalizzabili e accattivanti.

Al contrario, WPF risolve tutti questi problemi e facilità così lo sviluppo di applicazioni; tra le sue principali qualità vi sono:

- ✓ Un modello di layout simile al Web, che permette all'interfaccia utente di adattarsi per mostrare contenuti altamente dinamici o lingue differenti all'interno dello stesso controllo.
- ✓ Una tecnologia grafica performante: anziché definire ciascun pixel, WPF mette a disposizione tipi primitivi— forme di base, blocchi di testo e altri strumenti grafici.
- ✓ Utilizzo dell'accelerazione hardware per tracciare la GUI, il che comporta un miglioramento delle prestazioni.

- ✓ Supporto per supporti audio e video: WPF include il supporto per la riproduzione di qualsiasi file audio o video supportato da Windows Media Player e consente di riprodurre più file multimediali contemporaneamente.
- ✓ Stili e modelli: gli stili consentono di standardizzare la formattazione e riutilizzarla in tutta l'applicazione. I modelli consentono di cambiare il modo in cui è rappresentato qualsiasi elemento, anche un controllo di base come un pulsante.
- ✓ Interfaccia utente dichiarativa: sebbene sia possibile costruire una finestra WPF con codice, Visual Studio adotta un approccio diverso. Serializza il contenuto di ogni finestra in un set di tag XML in un documento XAML. Il vantaggio è che **l'interfaccia utente è completamente separata dal codice** e i grafici possono modificare i file XAML e perfezionare il front-end dell'applicazione da soli. Di conseguenza, il lavoro ad un progetto può essere comodamente suddiviso tra designer e sviluppatore.
- ✓ Applicazioni basate su una pagina Web: utilizzando WPF, è possibile creare un'applicazione simile a un browser che consente di spostarsi in una raccolta di pagine, completa di pulsanti di navigazione.

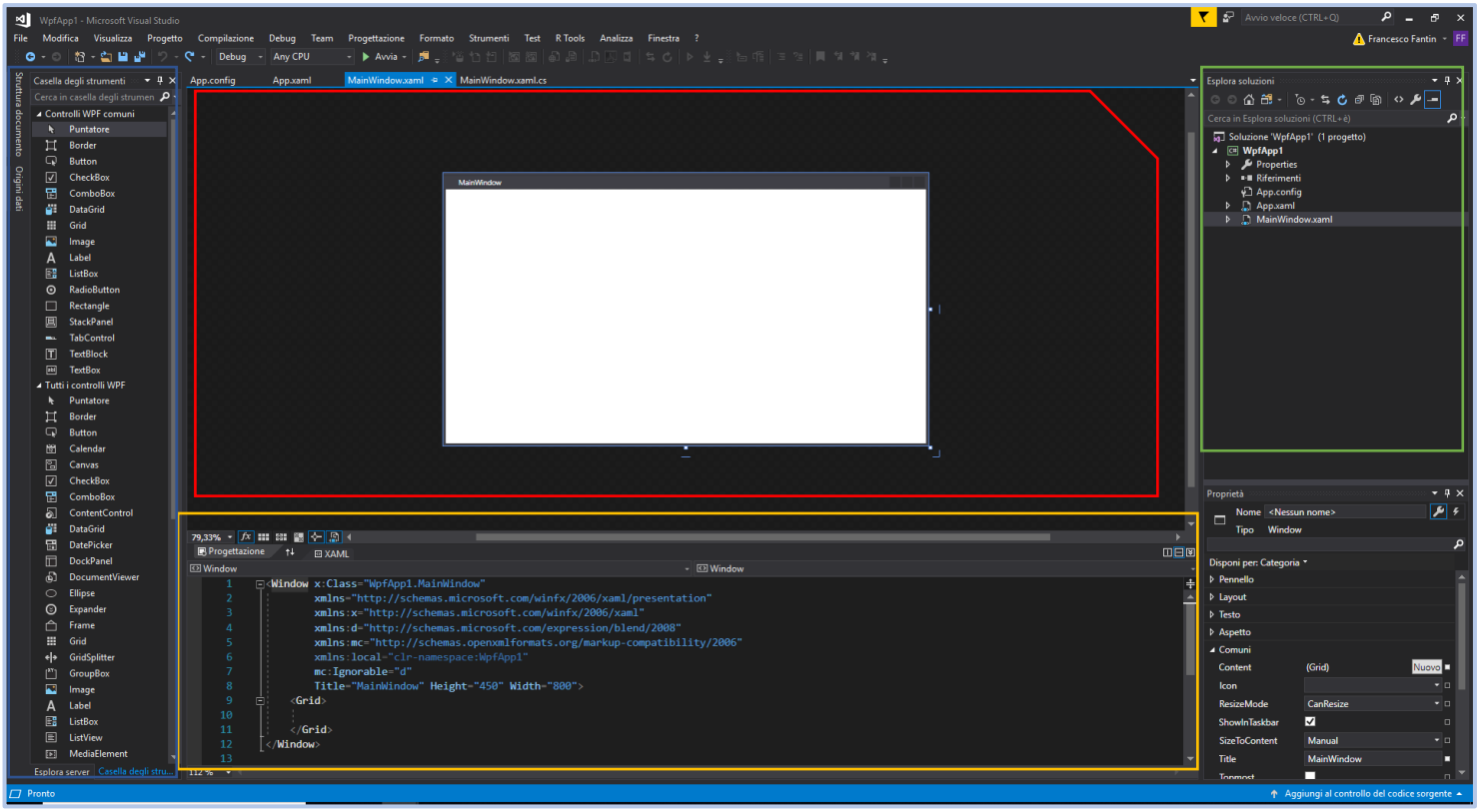


- Figura 1.3: Esempio di come sia facile incorporare contenuti multimediali all'interno della finestra dell'applicazione: per aggiungere un video è infatti necessaria una sola riga di codice XAML.

Un’applicazione in WPF

L’ambiente di sviluppo che utilizza WPF è Visual Studio. Basato sul .NET Framework, Visual Studio mette a disposizione del programmatore quattro modelli di progetto WPF differenti:

- WPF Application, utilizzato per creare un’applicazione desktop WPF per sistemi operativi Windows;
- WPF Browser Application, che consente di creare un’applicazione WPF eseguibile direttamente all’interno di un browser Web;
- WPF Custom Control Library, utilizzato per ridefinire alcuni dei controlli WPF preesistenti;
- WPF User Control Library, che permette di creare controlli utente completamente personalizzati.

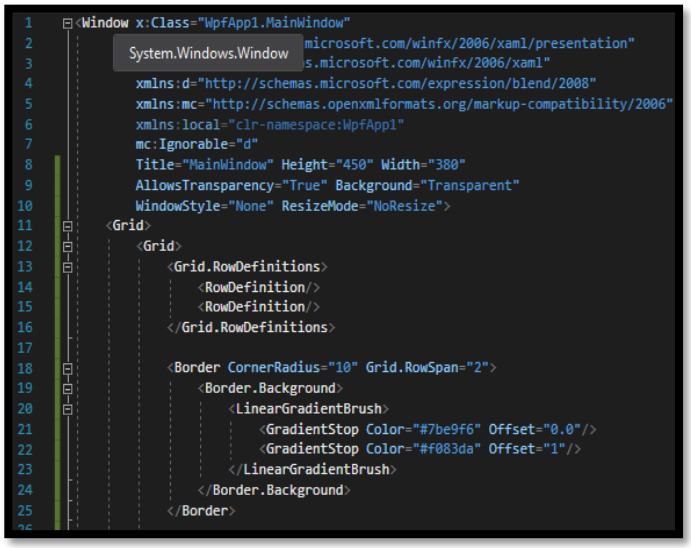


- Figura 1.4: Modello WPF Application. Attualmente, l’ultima versione disponibile di WPF è la 4.6, uscita nel 2015.

Nella figura 1.4 è possibile distinguere le varie parti che formano il modello WPF Application e comprendere alcune delle differenze rispetto a Windows Forms. A sinistra è presente la Casella Degli Strumenti, contenente i controlli WPF comuni e di terze parti (se aggiunti dal programmatore). Il funzionamento della Casella degli Strumenti è fondamentalmente simile a quella di Windows Forms: permette infatti di trascinare nuovi controlli all’interno della finestra dell’applicazione e comporta la modifica del codice XAML contenuto nell’editor (rettangolo giallo).

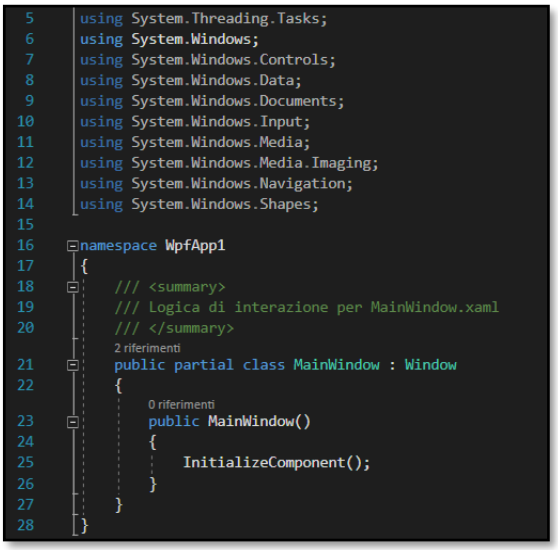
Per istanziare oggetti all’interno del .NET framework si utilizza **XAML**, un linguaggio di markup che consente al programmatore di costruire interfacce grafiche, cioè di definire la disposizione degli strumenti quali pannelli, bottoni e controlli. Il rettangolo rosso contiene invece il designer, da cui vengono definiti le finestre, i controlli, e le pagine. La sezione Esplora Soluzioni è invece molto diversa rispetto a Windows Forms: all’interno della scheda sono infatti presenti file in formato .xaml, ognuno dei quali identifica una finestra

oppure un controllo. Ciascun file è associato ad un file di code-behind, che contiene a sua volta il codice (nel nostro caso scritto in C#) che gestisce i controlli e ne indica le funzionalità.




```
<Window x:Class="WpfApp1.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:WpfApp1"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="380"
        AllowsTransparency="True" Background="Transparent"
        WindowStyle="None" ResizeMode="NoResize">
    <Grid>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition/>
                <RowDefinition/>
            </Grid.RowDefinitions>

            <Border CornerRadius="10" Grid.RowSpan="2">
                <Border.Background>
                    <LinearGradientBrush>
                        <GradientStop Color="#7be9f6" Offset="0.0"/>
                        <GradientStop Color="#f083da" Offset="1"/>
                    </LinearGradientBrush>
                </Border.Background>
            </Border>
```



```
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfApp1
{
    /// <summary>
    /// Logica di interazione per MainWindow.xaml
    /// </summary>
    2 riferimenti
    public partial class MainWindow : Window
    {
        0 riferimenti
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

- Figura 1.5: Codice XAML relativo alla finestra di Login (Figura 1.1)
- Figura 1.6: Codice operativo (C#) associato alla finestra principale dell'applicazione

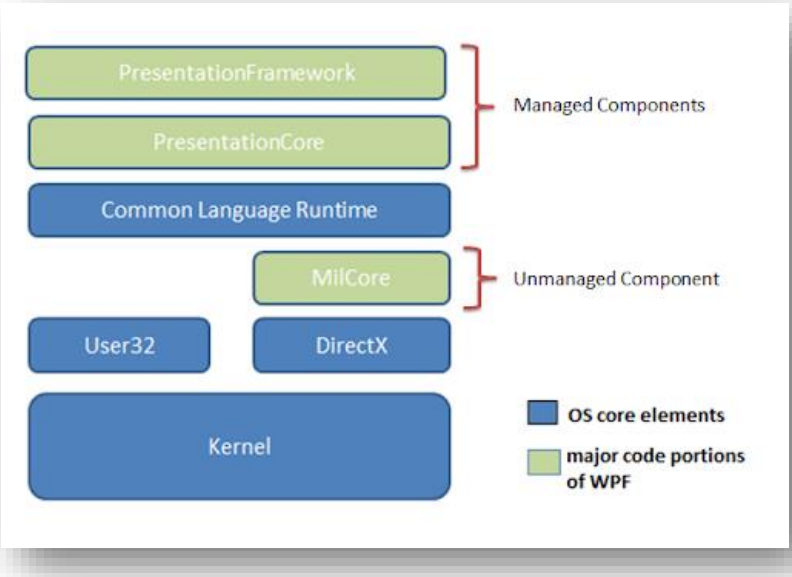
Struttura e Architettura

WPF presenta un'architettura a tre strati, detti rispettivamente **WPF Managed Layer**, **WPF Unmanaged Layer** e gli **elementi** del sistema operativo.

In cima, nella sezione del codice gestito (**Managed Components**) sono presenti gli assembly che contengono i tipi e che implementano le astrazioni di alto livello come gli stili.

L'assembly **PresentationFramework.dll** contiene tutti i namespaces e le classi utilizzabili dal programmatore per scrivere l'applicazione. **PresentationCore.dll**, invece, espone il linguaggio XAML e contiene i tipi base da cui derivano tutte le forme che costituiscono i grafici vettoriali. Nella libreria **Windowsbase.dll** (non presente nella figura) sono contenuti i tipi utilizzati anche al di fuori di WPF.

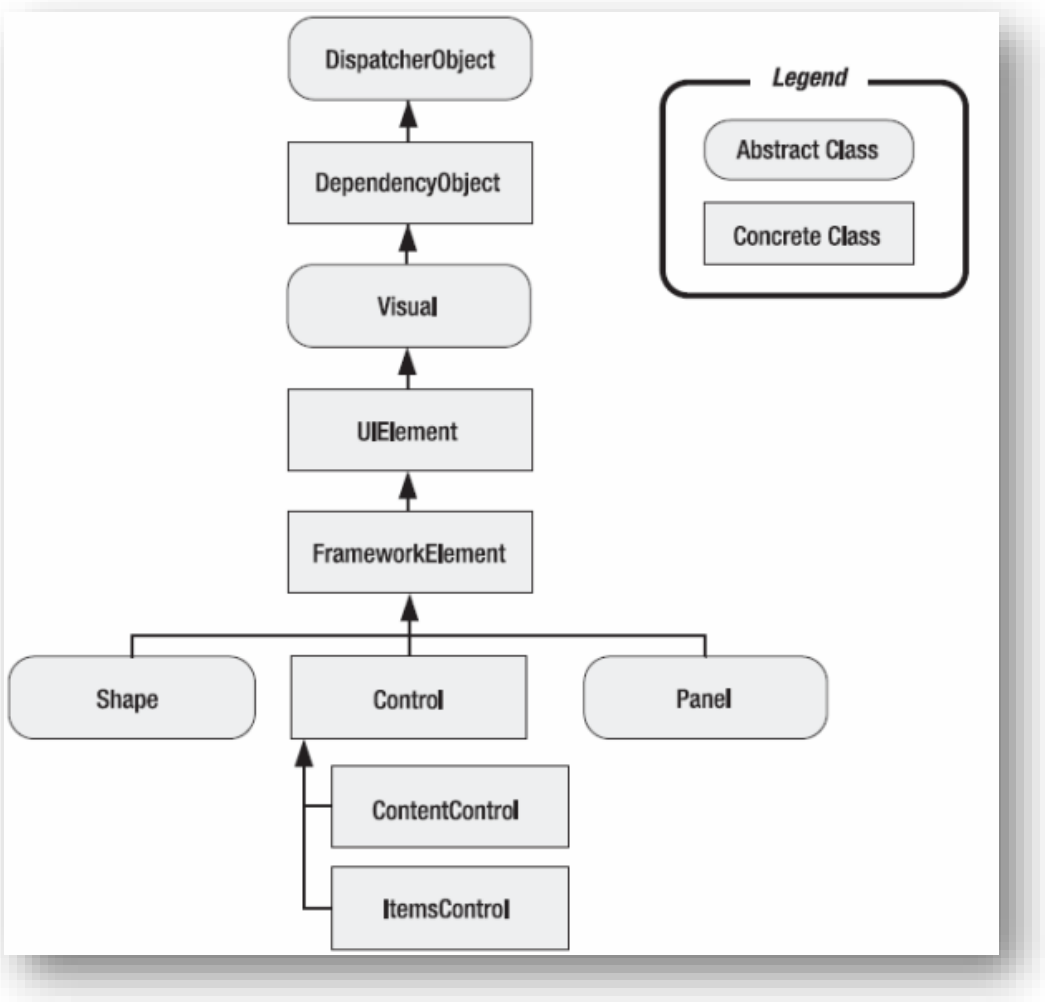
Nel livello sottostante è presente lo stato di codice non gestito, occupato dalla libreria **milCore.dll**. Lo scopo di milCore è di interfacciarsi direttamente con DirectX e fornire supporto di base per le superfici 2D e 3D. Questa sezione è un codice non gestito perché funge da ponte tra lo strato di codice gestito superiore (scritto in codice nativo) e lo strato formato da User32 e DirectX.



Nell'ultimo livello, sono invece presenti i due componenti precedentemente introdotti: **User32** e **DirectX**. Come già detto, User32 è usato per la gestione dell'input e dello spazio occupato sullo schermo da ogni applicazione, oltre allo

scambio di messaggi tra applicazioni. Dall'altra parte, le librerie DirectX forniscono supporto per il rendering grafico.
Al di sotto dell'architettura, si trova il **kernel** di Windows, ovvero il software che fornisce un accesso sicuro e controllato dell'hardware ai processi in esecuzione sul computer.

Gerarchia delle Classi



System.Threading.DispatcherObject
Le applicazioni WPF utilizzano il familiare modello di affinità a thread singolo (STA), ovvero l'intera interfaccia utente è di proprietà di un singolo thread.

System.Windows.DependencyObject
Questa classe costituisce il modo principale interagire con gli elementi su schermo attraverso le proprietà. Derivando da DependencyObject, le classi WPF ottengono il supporto per le proprietà di dipendenza.

System.Windows.Media.Visual
Ogni elemento che appare in una finestra di WPF è, in sostanza, un Visual. La classe Visual fornisce anche il collegamento tra le librerie WPF gestite e il milcore.dll che esegue il rendering del display. Qualsiasi classe che derivi da Visual ha la capacità di essere visualizzata su una finestra

System.Windows.UIElement
UIElement aggiunge il supporto per gli elementi essenziali di WPF come layout, input, focus ed eventi.

System.Windows.FrameworkElement
FrameworkElement è l'ultima parte dell'album ereditario WPF di base. Implementa alcuni dei membri che sono semplicemente definiti da UIElement. Ad esempio, UIElement pone le basi per il sistema di layout WPF, ma FrameworkElement include le proprietà chiave (come HorizontalAlignment e Margin) che lo supportano. UIElement implementa anche il supporto per l'associazione, l'animazione e gli stili degli oggetti.

System.Windows.Shapes.Shape
Le classi di forme di base, come Rettangolo, Poligono, Ellisse, Linea e Tracciato, derivano da questa classe. Queste forme possono essere utilizzate insieme a controlli di Windows più tradizionali come pulsanti e caselle di testo.

System.Windows.Controls.Control
Un controllo è un elemento che può interagire con l'utente. Include ovviamente classi come TextBox, Pulsanti e ListBox. La classe Control aggiunge proprietà aggiuntive per l'impostazione del carattere e del primo piano e colori di sfondo.

System.Windows.Controls.ContentControl
Questa è la classe di base per tutti i controlli che hanno un singolo contenuto.

System.Windows.Controls.ItemsControl
Questa è la classe base per tutti i controlli che mostrano una raccolta di elementi, come ListBox e TreeView.

System.Windows.Controls.Panel
Questa è la classe base per tutti i contenitori di layout, elementi che possono contenere uno o più figli e disporli secondo specifiche regole di layout. Questi contenitori sono la base del sistema di layout WPF e utilizzarli è la chiave per organizzare i contenuti nel modo più flessibile possibile.