

Windows Presentation Foundation




Antonia Sacchitella

Analyst @icubedsrl

antonia.sacchitella@icubed.it



Week 8 - Agenda

- Introduzione a WPF
- Elementi di design
- Data Binding
- Eventi
- Pattern MVVM
-  Esercitazione



WPF = Windows Presentation Foundation

È la parte del .NET Framework per realizzare app per Windows

Basato su:

- XAML per la parte visuale
- C# per il codice

Motore vettoriale

XAML è un linguaggio simil-XML

Cos'è WPF?

Motore di presentazione vettoriale basato su DirectX

Insieme di classi per realizzare applicazioni di impatto

Sostituisce l'attuale motore di presentazione studiato 20+ anni fa

Unifica UI, Media, Grafica e Documenti

Componibile

- Ogni elemento può contenerne altri

Databinding

Stili

Animazioni

Perché WPF?

La semplice interfaccia Win32 non basta più

- Interfacce Office style
- Applicazioni portate sul Web (AJAX)
- SmartClient

Semplici animazioni ed effetti sono sempre piacevoli, anche per un gestionale
Sfruttare al massimo il client, anche la GPU

Win32 ha i suoi limiti

- Code dei messaggi
- Subclassing e codice unmanged per nuovi controlli

WPF 101

Contiene tutti gli elementi di accessibilità

- Access Key, keyboard navigation, focus

Nuova gestione Input più aperta

- Keyboard, mouse, stylus, speech, appCommand
- Future forme di input...

Eventi più completi (Preview / After)

DataBinding molto potente

- Automazione
- Master/details/details/details/details ails ails...
- Validazioni

WPF – Struttura Windows Application

Eventi

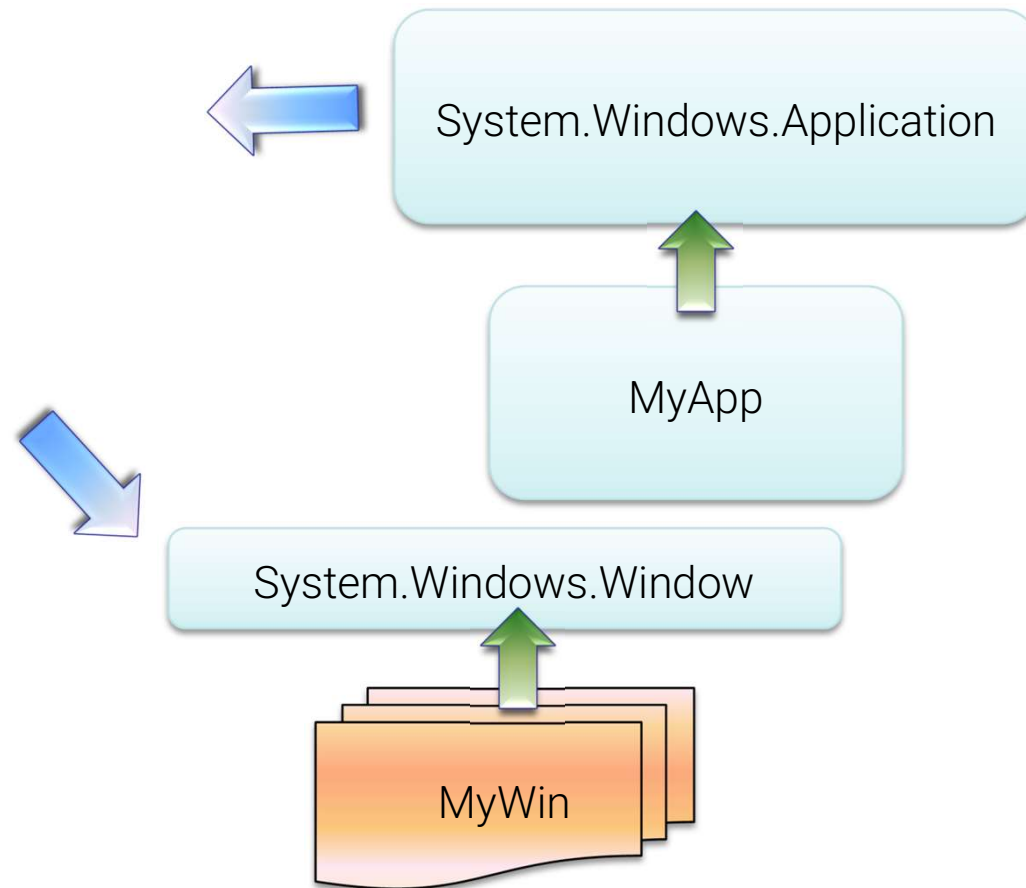
- *Startup, Exit*

Metodi

- *Run*

Proprietà

- *MainWindow*



Developer vs Designer

Developer

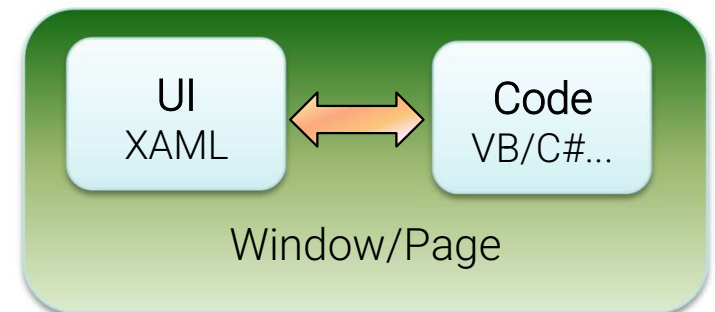
- Implementa le funzionalità

Designer

- Aspetti legati alla User Interface

Applications = Code (UI inclusa)

- Code behind
- Markup Language
- Interazione tra i due "mondi"
- Partial Class



XAML

Sintassi basata su XML

- Well formed and valid

Element = Classe

Attribute = Proprietà

Namespace XML -> Namespace CLR

- Mapping predefinito: <http://schemas.microsoft.com/winfx/2006/xaml/presentation>
<http://schemas.microsoft.com/winfx/2006/xaml>
- clr-namespace:namespace;assembly=
- [assembly: XmlnsDefinition("http://...", "CLR namespace")]

Commenti con <!-- -->

XAML

Extensible Application Markup Language

Based on XML

Used to initialize
objects

Used to create
User Interfaces

XAML

```
using System.Windows;

public partial class Window1
{
    public Window1()
    {
        this.Title = "MyWindow";
        this.AddChild(new Grid());
    }
}
```

Un NS XML può corrispondere ad un insieme di NS

- x:NS di supporto al compilatore
- **x:Class** =
RootNamespace.Windows1:Window

XAML

Property-Element

- Usato con oggetti complessi

```
<Button>  
  <Button.Content>  
    <StackPanel HorizontalAlignment="Center">  
      <!-- ... -->  
    </StackPanel>  
  </Button.Content>  
</Button>
```

Attached Properties

```
<Canvas>  
  <Button Canvas.Left="100" Canvas.Top="50">  
    Click Me!  
  </Button>  
</Canvas>
```

XAML

Custom Types

- XAML permette l'utilizzo di tipi custom

```
<Window x:Class="..."  
        xmlns="..."  
        xmlns:x="..."  
        xmlns:y="clr-namespace:System.Demo;assembly=foo" />
```

Gestione eventi

```
<Button Click="MyClickRoutine">  
    Click Me!  
</Button>
```

Mapping vs codice

```
<Button x:Name="button1">  
    Click Me!  
</Button>
```

XAML

I valori degli attributi sono convertiti tramite *TypeConverter*

Proprietà predefinite

- Attributo *ContentProperty*
- Usato anche per collezioni (es: *Panel.Children*)

Enumerazioni: stringhe separate da virgola

Eventi: nome handler del code behind

Property element syntax

- Per definire proprietà complesse
- NomeClasse.NomeProprietà

XAML

Collection syntax

- Per proprietà che implementano *IList* o *IDictionary*
- Permette di aggiungere elementi alla proprietà di tipo lista

Attached property

- Si qualificano con NomeClasse.NomeProprietà
 - Può essere omessa la classe e dedotta dall'elemento
 - Si può usare il nome del tipo base
- Supportano data binding, stili ed animazioni

Attached event

- Intercetta tutti gli eventi, anche dei figli

Demo

WPF - HelloWorld



Pannelli

Sono i principali elementi per la definizione del layout

Classe base *Panel*

- *Canvas*: rappresenta una tela
- *DockPanel*: ancora gli elementi
- *Grid*: griglia con righe e colonne
- *StackPanel*: impila gli elementi
- *UniformGrid*: griglia con celle di uguale dimensione
- *WrapPanel*: allinea gli elementi passando automaticamente alla linea/colonna successiva

Pannelli

Proprietà comuni a tutti gli elementi del layout

- *Background*
- *Width /Height*
- *Min – Max Width / Min – Max Height*
- *Margin – distanza minima tra i vari componenti*
- *Visibility: Visible / Hidden / Collapsed*
- *IsEnabled*

Tutte le proprietà possono essere settate sia da interfaccia sia da codice

Pannelli

VirtualizingPanel: usato dai controlli di lista per virtualizzare gli elementi

- Per risparmiare sull'UI

Utilizzano le attached property per informazioni inerenti agli elementi

- Es: *Canvas.Left*, *Grid.Row*

ClipToBounds: circoscrive l'area di rendering sui confini dell'elemento

Prendono in considerazione le informazioni degli elementi

- *Width*, *Height*, *HorizontalAlignment*, *VerticalAlignment*, *Margin*, *MaxWidth*, *MaxHeight*, *MinWidth*, *MinHeight*

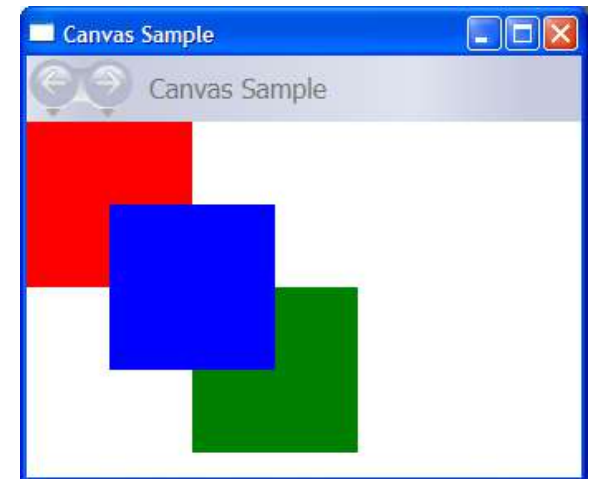
Canvas

Canvas.Left, Canvas.Top, Canvas.Bottom, Canvas.Right per la disposizione dell'elemento

L'ordine degli elementi dà lo z-index

- Forzabile con *Panel.Zindex*

Non ha una dimensione dedotta dai figli



DockPanel

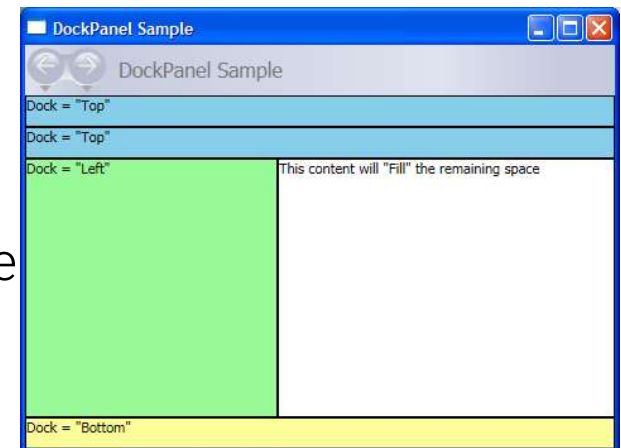
DockPanel.Dock per indicare l'ancoraggio

- *Left, Top, Right, Bottom*

L'ordine degli elementi determina la posizione

L'ultimo elemento se non ha il Dock è l'elemento che

- *LastChildFill* normalmente su *True*



Grid

RowDefinitions e *ColumnDefinition* per definire quante righe e quante colonne

- Possiamo definire l'altezza e la larghezza
 - Normalmente suddivide in spazi uguali
- *GridLength*
 - *Auto*: lo spazio è quello richiesto dai figli
 - *Numero*: lo spazio è esplicito
 - ***: lo spazio viene suddiviso in base alle altre righe
 - 2x, 3x ecc indicano il rapporto

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="1*" />
    <ColumnDefinition Width="1*" />
    <ColumnDefinition Width="1*" />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="200"/>
  </Grid.RowDefinitions>
</Grid>
```

Grid

Grid.Row e *Grid.Column* per indicare la posizione

- Se omesso è 0,0

Grid.RowSpan e *Grid.ColumnSpan* per occupare più righe e colonne

ShowGridLines a scopo di debug



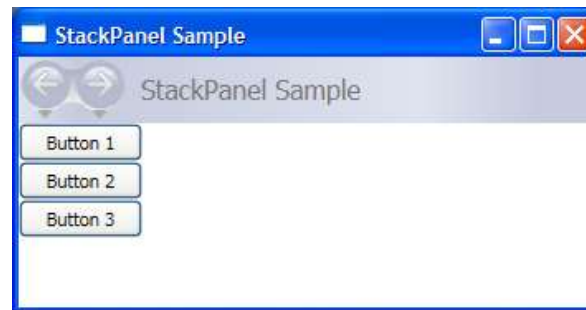
StackPanel

Pannello che distribuisce gli elementi in base ad un orientamento

Orientation per indicare l'orientamento verticale o orizzontale

Permette di andare oltre allo spazio a disposizione

- Diversamente da *DockPanel*

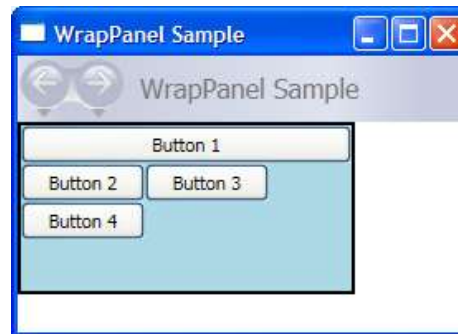


WrapPanel

Orientation per indicare l'orientamento verticale o orizzontale.
Comportamento simile al fluire del testo in un editor, va a capo non appena termina la linea

ItemWidth e *ItemHeight*

- Determinano lo spazio per ogni elemento ma l'elemento può forzare la dimensione



Layout system

Due fasi dei Panel

- *MeasureOverride*: vengono valutate le dimensioni richieste da ogni elemento
 - Valorizza *DesiredSize* (Margin inclusi)
- *ArrangeOverride*: vengono arrangiati elementi in base alle dimensioni imposte
 - Valorizza *RenderSize* (Margin esclusi)

InvalidateMeasure: invalida le fasi measure e arrange

InvalidateArrange: invalida la fase arrange

Entrambi non sono deterministici

- *UpdateLayout*

Layout system

Albero degli elementi logici e fisici

Logici: *Window* -> *Button*

Fisici: *Window* -> *Button* -> *Grid* -> *Border*

Proprietà *Parent* indica il padre logico

LogicalTreeHelper per l'accesso logico all'albero

VisualTreeHelper per l'accesso fisico all'albero

Dependency property

Particolari proprietà messe a disposizione dal sistema WPF

Estende le funzionalità di una normale proprietà e consente di ottenere valore basandosi su un altro input. Ad es.:

- Proprietà di sistema, come temi e preferenze dell'utente.
- Meccanismi di determinazione delle proprietà just-in-time (es. data binding)
- Modelli multiuso, come risorse e stili.
- Valori noti attraverso le relazioni padre-figlio con altri elementi nell'albero degli elementi.

Dependency property - Definizione

Da XAML

```
<Button Content="Bottone rosso" Width="200"/>
```

Da codice

```
Button myButton = new Button();  
myButton.Content = "Bottone Rosso";  
myButton.Width = 200.0;
```

Controlli

UIElement: classe base, eventi base

Shape: primitive di disegno

FrameworkElement: elemento visuale, dimensioni

Control: font, colori, padding, template

ContentControl: controllo con un contenuto

Albero e funzionalità logiche

ItemsControl: rappresenta una lista

Selector: lista con selezione

- *ComboBox, ListBox, DataGrid*

Controlli

HeaderedItemsControl: lista con intestazione

- *Menu, Toolbar*

RangeBase: range di valori

- *ProgressBar, Slider*

Decorator: decorano per aggiungere funzionalità

- *Border, ViewBox*

I controlli rappresentano funzionalità logiche

- La loro rappresentazione dipende dal template
- Si sceglie il controllo in base alla funzionalità, non ha all'aspetto
- Alcuni controlli sono quindi identici nell'uso

Demo

Layout complesso



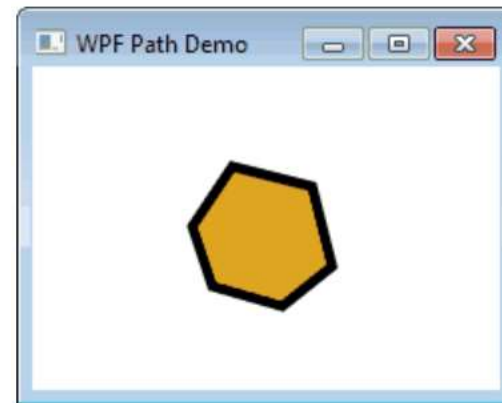
Geometry e Path

Rappresenta uno shape 2D

- *EllipseGeometry, RectangleGeometry, LineGeometry, CombinedGeometry, PathGeometry*

Usato

- Dagli shape
- Per il clip degli elementi
- Con il *GeometryDrawing*



Color

Rappresenta un colore

- ARGB 32 bit

Solitamente non viene usato direttamente ma solo tramite *Brush*

Metodi statici

- *FromRgb, FromArgb*

Classe *Colors*

- *Red, Blue, Black*

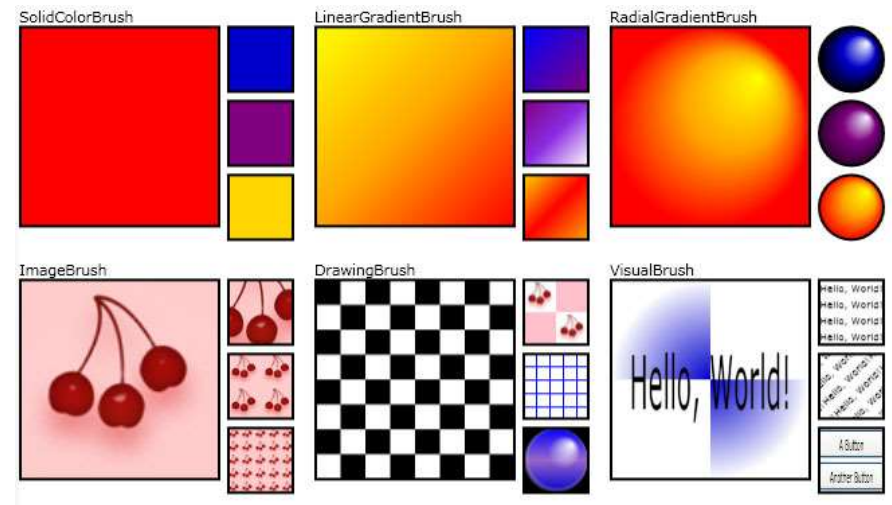
ColorConverter: trasformazione da stringa

- Esadecimale o known names

Brush

Rappresenta un pennello

- *SolidColorBrush*: tinta unita
- *LinearGradientBrush*: gradiente lineare
- *RadialGradientBrush*: gradiente radiale
- *ImageBrush*: brush basato su un'immagine
- *VisualBrush*: brush basato su un altro oggetto visuale
- *DrawingBrush*: brush che usa un *Drawing*



Trasformazioni

LayoutTransform: trasformazioni sul layout

RenderTransform: trasformazione sul rendering – non influisce sugli altri elementi

Tipologie

- *TranslateTransform*: translazione X,Y
- *ScaleTransform*: scalatura X,Y
- *SkewTransform*: sbieco
- *RotateTransform*: rotazione
- *MatrixTransform*: direttamente su matrice
- *TransformGroup*: aggrega più trasformazioni

Animazioni

Sono dependency property che variano nel tempo

Storyboard

- Timeline
- Specifico per un elemento
- Specifico per una proprietà

Trigger/EventTrigger/Action per controllare

Controllabile da codice

Garantisce il tempo e non il frame rate

Duration, RepeatBehavior, AutoReverse, BeginTime

Risorse

Dizionari di oggetti riutilizzabili

Chiave e valore di tipo object

- Normalmente si usa una stringa
- *ResourceKey*
- *ComponentResourceKey*

StaticResource per usare

DynamicResource per usare e monitorare

Definibili su ogni *FrameworkElement*

- Ricerca nel visual tree dall'elemento fino alla radice

Una sola istanza

- *x:Shared* a false per creare nuove istanze

VisualStateManager

Definisce degli stati logici visuali

Gruppi di stato

- Uno stato per gruppo

Transazioni automatiche tra stati

- In base alla durata
- *EasingFunction* applicabile

Storyboard per ogni transizione

Effetti applicabili

Utilizzabile su tutti gli elementi e template

Due gruppi di stato non possono lavorare sullo stesso elemento

Demo

App completa



Esercitazione n. 1

Realizzare un'applicazione desktop mediante WPF per un'e-commerce che consenta di visualizzare i dettagli dei prodotti del magazzino.

A	
B	C
	D

Specifiche implementative.

Il layout impostato secondo questa griglia, lo stile è lasciato a vostra discrezione.



Esercitazione n. 1

Inserire

nel riquadro A il nome del l'e.-commerce

nel riquadro B una list box contenente tutti i prodotti

nel riquadro C un pulsante che al click modifichi il contenuto del riquadro D. In particolare al click dovrà essere visualizzato il nome del prodotto selezionato

A	
B	C
	D

