




CppWinRT e il futuro dello sviluppo desktop in Windows

Raffaele Rialdi
Senior Software Architect
Microsoft MVP
Consultant - Speaker - Teacher

 [@raffaeler](https://twitter.com/raffaeler)
 <https://github.com/raffaeler>
 <http://iamraf.net>
 raffaeler@vevy.com

Chi sono

@raffaeler



- Raffaele Rialdi, Senior Software Architect in Vevy Europe – Italy
- Consulente in diversi ambiti:
 - Banking/finanza, healthcare, manufacturing, racing, ...
- Parte del grande gruppo dei Microsoft MVP fin dal 2003
- Speaker & consulente in giro per il globo (sviluppo e sicurezza)
 - Italy, Romania, Bulgaria, Russia , USA, ...



Il progetto CppWinRT

- Nasce da un'idea di Kenny Kerr, oggi nel team di Windows
- In ItalianCpp, ho introdotto il progetto a Firenze, nel 2016
 - Video disponibile sul sito
- Evoluzione esponenziale
 - Adottato da tutti i team di Windows per le app native
 - Repository GitHub 'archiviato': <https://github.com/Microsoft/cppwinrt>
 - Documentazione disponibile su docs.microsoft.com
 - Estensione VSIX per Visual Studio (templates) chiamata "C++/WinRT"

Che cos'è CppWinRT

- Un generatore di codice C++ aderente allo standard ISO C++17
 - Ha reso obsoleta e deprecata l'estensione nota come C++/CX
- Il codice generato è una "projection" dei componenti WinRT / UWP
- cppwinrt.exe legge i file ".winmd" e genera la projection (.h)
 - I file winmd sono metadati dei componenti WinRT in formato Ecma-335
- La projection può essere usata per applicazioni desktop o UWP
- Finalmente supportati: XAML e Binding (in preview)

Requisiti

- Ultimo Windows 10 (1803) e relativo SDK (10.0.17134.0)
 - cppwinrt.exe fa parte di questo sdk
- Visual Studio 2017 (versione 15.7)
- Estensione VSIX "C++/WinRT"
 - Visualizer
 - Template per progetti e 'items'
 - Estensioni a MSBuild che invocano il generatore di codice cppwinrt.exe
- Per UWP, abilitare il 'sideload' nel developer mode di Windows 10
- Abilitare **/await** nella Command Line delle proprietà C/C++
- Nota: Win e SDK sono sufficienti per lavorare manualmente

Projection di quali classi?

- API disponibili su nuget. Per esempio: Win2D
- Le API esposte ad UWP: <https://docs.microsoft.com/en-us/uwp/api/>



LearningModelPreview Class

Namespace: [Windows.AI.MachineLearning.Preview](#)

Assemblies: Windows.AI.MachineLearning.Preview.dll, Windows.dll

Represents a machine learning model.

Edit

C++/WinRT

Copy

```
struct winrt::Windows::AI::MachineLearning::Preview::LearningModelPreview  
: ILearningModelPreview
```

Attributes [ContractVersionAttribute](#), [StaticAttribute](#)

Feedback

Share

Dark

Language

C++/WinRT ▼

C++/CX

C++/WinRT

C#

VB

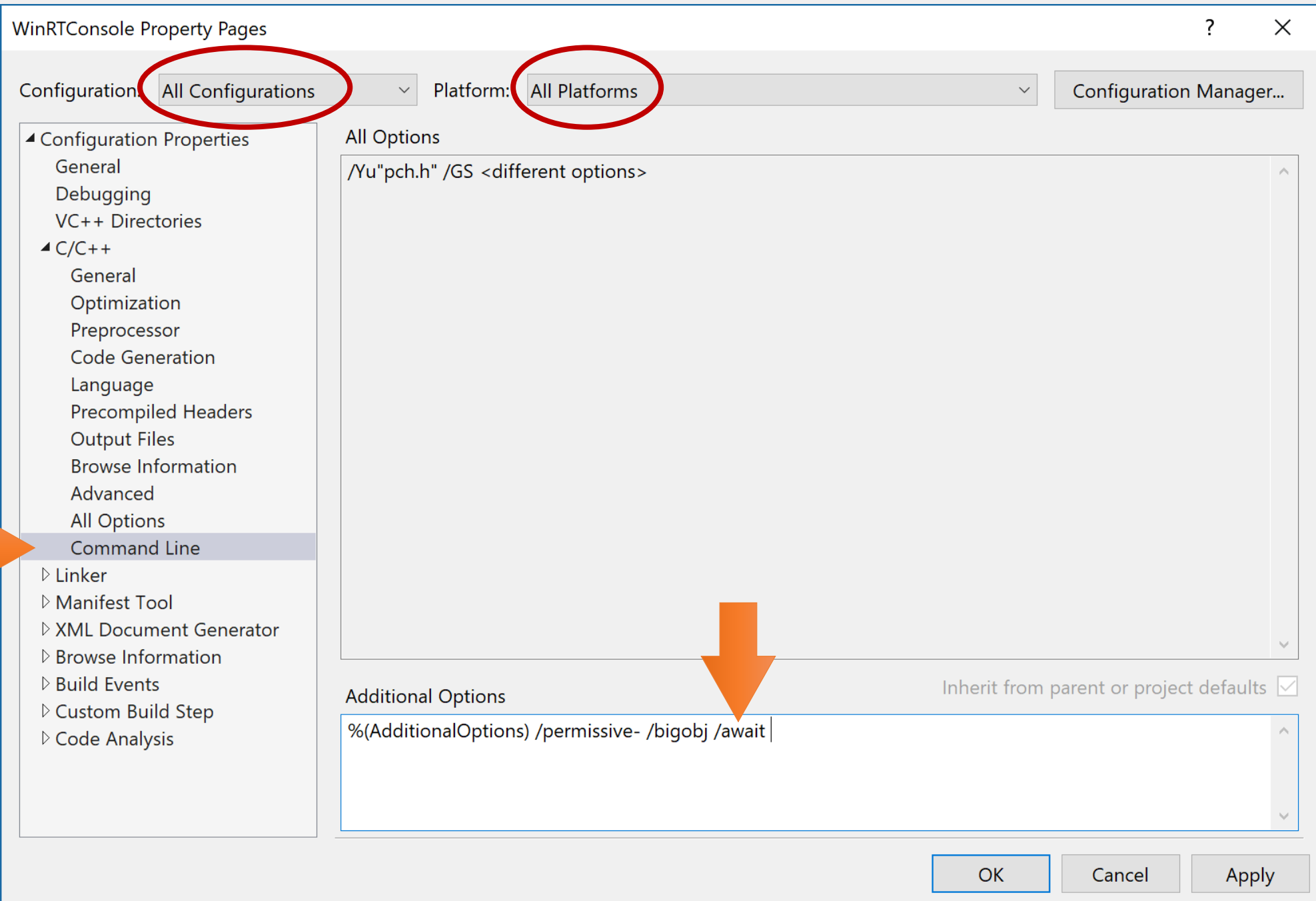
JavaScript

[Examples](#)

[Properties](#)

[Methods](#)

Projections
disponibili



cppwinrt.exe

```
C:\>cppwinrt
```

```
C++/WinRT v1.0.180227.3
```

```
Copyright (c) 2018 Microsoft Corporation. All rights reserved.
```

```
cppwinrt.exe [options...]
```

Options:

-in	<spec>	windows metadata to include in projection
-ref	<spec>	Windows metadata to reference from projection
-out	<path>	Location of generated projection and component templates
-component	[<path>]	Generate component templates, and optional implementation
-filter	<prefix>	One or more prefixes to include in component
-name	<name>	Specify explicit name for component files
-verbose		Show detailed progress information
-help		Show detailed help with examples
@<path>		Response file containing command line options

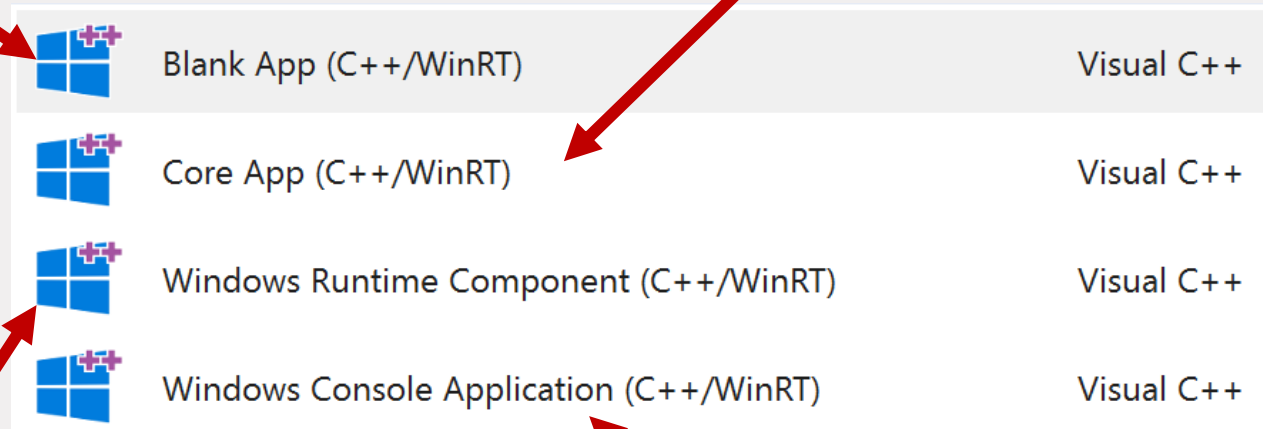
where <spec> is one or more of:

path	Path to winmd file or recursively scanned folder
local	Local %WinDir%\System32\WinMetadata folder
sdk[+]	Current version of windows SDK [with extensions]
10.0.15063.0[+]	Specific version of windows SDK [with extensions]





Templates di progetto

I setting di progetto
indispensabili

Applicazione desktop (UWP)
XAML + cppwinrt



The diagram shows a list of four project templates. Red arrows point from external text to specific templates: one from 'I setting di progetto indispensabili' to 'Blank App', one from 'Applicazione desktop (UWP)' to 'Core App', one from 'Un componente WinRT riusabile in altri progetti' to 'Windows Runtime Component', and one from 'Una applicazione Console usabile server side o per test' to 'Windows Console Application'.

	Blank App (C++/WinRT)	Visual C++
	Core App (C++/WinRT)	Visual C++
	Windows Runtime Component (C++/WinRT)	Visual C++
	Windows Console Application (C++/WinRT)	Visual C++

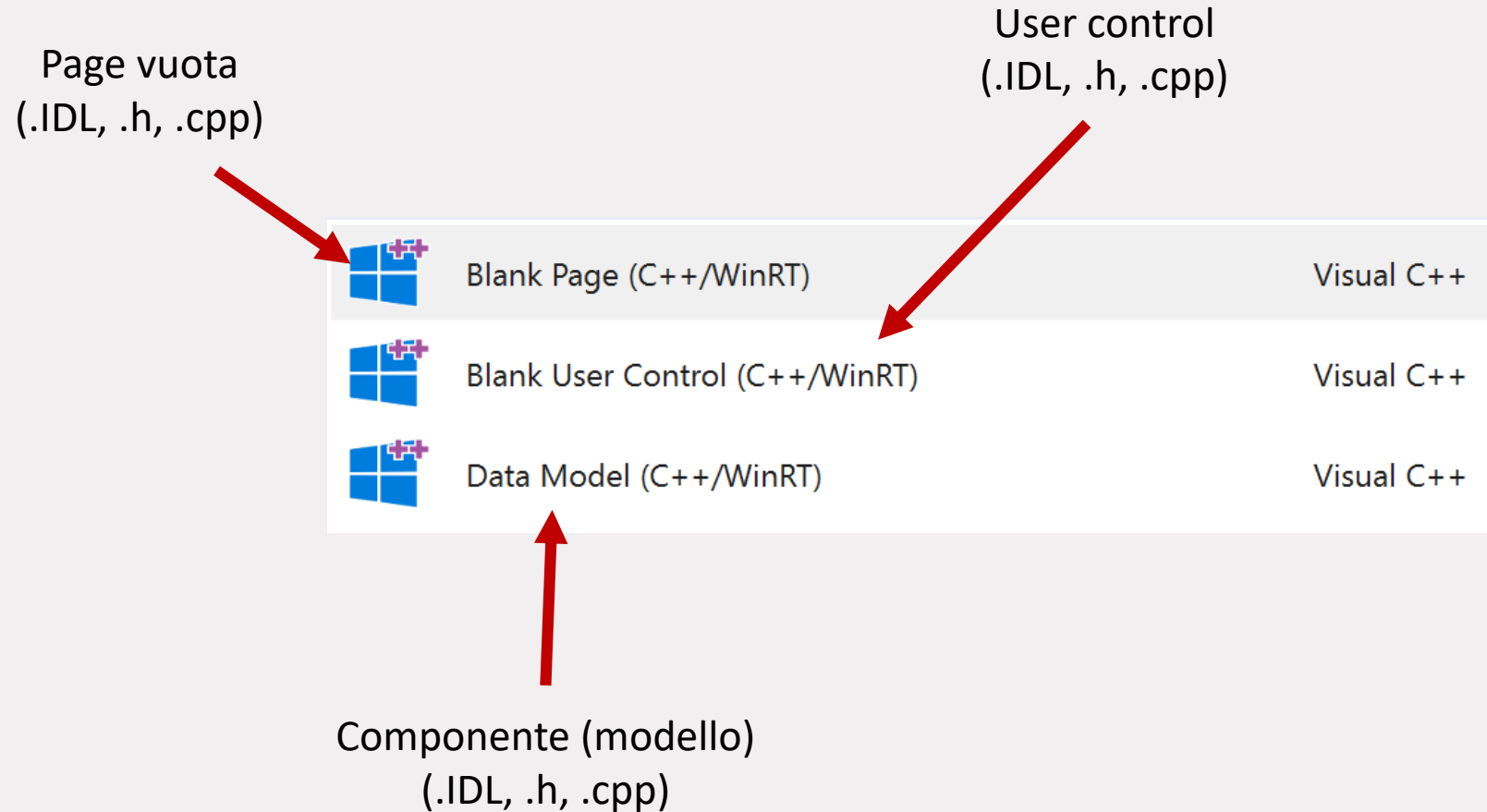
Un componente WinRT
riusabile in altri progetti
(produce winmd e dll)




Una applicazione Console
usabile 'server side'
o per test

Templates per XAML e Modelli

Page vuota
(.IDL, .h, .cpp)

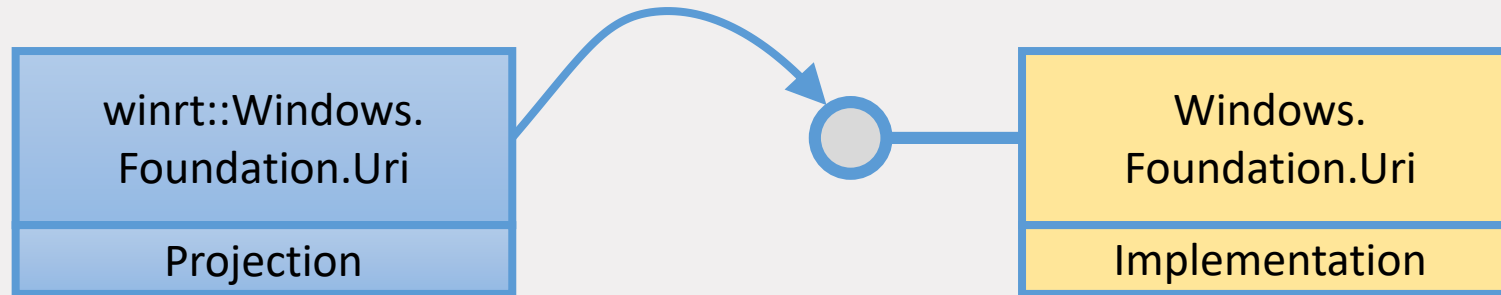
User control
(.IDL, .h, .cpp)



	Blank Page (C++/WinRT)	Visual C++
	Blank User Control (C++/WinRT)	Visual C++
	Data Model (C++/WinRT)	Visual C++

Componente (modello)
(.IDL, .h, .cpp)

Projections and Implementations



- Le Projections sono smart pointers / proxy
 - Leggeri, possono essere passata by value
 - Nei nostri progetti: `NamespaceProgetto::Person`
- Le Implementazioni sono gli oggetti 'veri'
 - Ogni istanza deve essere gestita con attenzione
 - Nei nostri progetti: `NamespaceProgetto::implementation::Person`

Projections & Implementations: Tips

- Il ctor della projection crea una implementazione automaticamente



- Per evitarlo inizializzarle con nullptr

```
BindingViewModels::Person _currentPerson { nullptr };
```

- Si usa make<T> per creare entrambi:

```
auto person = winrt::make<BindingViewModels::implementation::Person>(
    L"John", L"Doe", 35);
```

- Infine si può inizializzare la projection ancora non inizializzata

```
_currentPerson = person;
```

com_ptr<T>

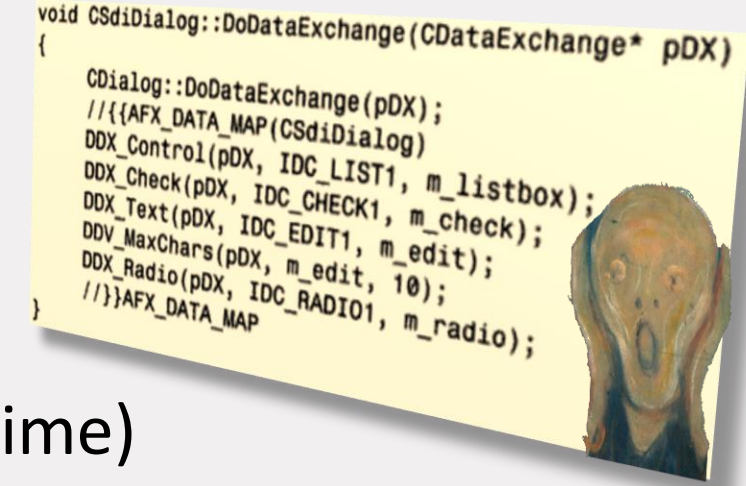
- Ottenere uno smart pointer diretto all'implementazione
 - `com_ptr<T> sp = winrt::make_self<T>(...);`
- Per passare l'oggetto by pointer → `com_ptr<T>.get()`
- Per cambiare il puntatore → `com_ptr<T>.put()`

```
com_ptr<ID3D11Texture2D> backBuffer;  
com_ptr<ID3D11RenderTargetView> view;  
swapChain->GetBuffer(0, IID_PPV_ARGS(backBuffer.put()));  
device->CreateRenderTargetView(backBuffer.get(), nullptr,  
                               view.put());
```

Binding

- XAML supporta x:Bind (codice generato a compile-time)
 - "Binding" si appoggia a reflection e non può funzionare in C++
 - x:Bind è più performante ma ha alcune limitazioni
- Binding per scrivere da XAML al modello
 - Proprietà: x:Bind sulle proprietà del modello
 - Eventi: x:Bind su un metodo del modello
- Binding per aggiornare XAML dal modello
 - Implementare INotifyPropertyChanged nella pagina e nei modelli
 - Sollevare l'evento per aggiornare il valore

Nota importante: il default di 'Mode' in x:Bind è 'OneTime' e non 'OneWay'



Binding - esempi

XAML

IDL

C++

```
<TextBlock Text="{x:Bind Title}" />
```

```
String Title{ get; };
```

```
hstring Title() { return _title; }
```

```
<Button Content="New"  
Click="{x:Bind ViewModel.NewPerson}" />
```

```
void NewPerson();
```

```
void NewPerson() { ... }
```

```
<TextBox Text="{x:Bind Path=ViewModel.  
CurrentPerson.FirstName,  
Mode=TwoWay}" />
```

```
runtimeclass Person :  
    Windows.UI.Xaml.Data.  
        INotifyPropertyChanged  
{  
    String FirstName;  
}
```


```
hstring FirstName()  
{  
    return _firstName;  
}  
  
void FirstName(hstring const&  
    firstName)  
{  
    _firstName = firstName;  
    RaisePropertyChanged(L"FirstName");  
}
```

Stringhe!

- La projection su HSTRING è hstring (conforme a (w)string_view)

Da	A	Funzione	Note
std::wstring std::string	UTF8	to_string()	usa WideCharToMultiByte
<code>inline std::string to_string(std::wstring_view value);</code>			
tipi base e std::string	hstring	to_hstring()	non usare per convertire wstring
<code>template <typename T> winrt::hstring to_hstring(T const& value);</code>			
std::wstring std::wstring_view	hstring	hstring()	costruttore con overload multipli
<code>hstring(std::wchar_t const* c);</code>			

Custom Controls

- Aggiungere un 'Data Mode' (IDL + h + cpp)
 - Nel file IDL derivare il controllo di base
 - runtimeclass MyPanel : Windows.UI.Xaml.Controls.SwapChainPanel
 - Nel file .h derivare un eventuale helper
 - struct MyPanel : MyPanelT<MyPanel>, public virtual SwapChainPanelBase
 - Recuperare la projection al controllo di base
 - `SwapChainPanel` panel(*this);
 - Inizializzare l'helper con la projection
 - Initialize(panel, false);
 - Includere il .h del controllo nel pch.h o prima dell'include della pagina
- 

Links

- Demo binding:
 - https://github.com/raffaeler/cppwinrt_binding
- Microsoft PhotoEditor:
 - <https://github.com/Microsoft/Windows-appsample-photo-editor>
- Repo archiviato (contiene anche samples e issues risolte):
 - <https://github.com/Microsoft/cppwinrt>
- Documentazione
 - <https://docs.microsoft.com/en-us/windows/uwp/cpp-and-winrt-apis/>

Domande?



Grazie!

@raffaeler

raffaeler@vevy.com

Slides su <https://italiancpp.org>

Powered by:

