

WIN802 - Introduzione a WinRT

Raffaele Rialdi



raffaeler@vevy.com



@raffaeler



<http://www.iamraf.net>

<http://blogs.ugidotnet.org/raffaele>



Grazie a

5Dlabs.it

Microsoft
Visual Studio
with MSDN Subscription

OverNet
EDUCATION

Sponsor

Windows Internet
Explorer 9

SOL Server
LIFESTYLE
Proactively. Reactively.

Powered by
Windows Azure

Windows Phone
Put people first.

icubed

HOEPLI
INFORMATICA

aspitalia.com

DomusDotNet

DotDotNet



DotNet
Liguria

DotNET

dotnet
MARCHE

DotNetSide

GUISA
Gruppo Utenti Italiani
Solution Architect

sharepointcommunity.it

UGIdotNET

Visual Basic tips&tricks
Dal 1996 la community su VB e VB.NET

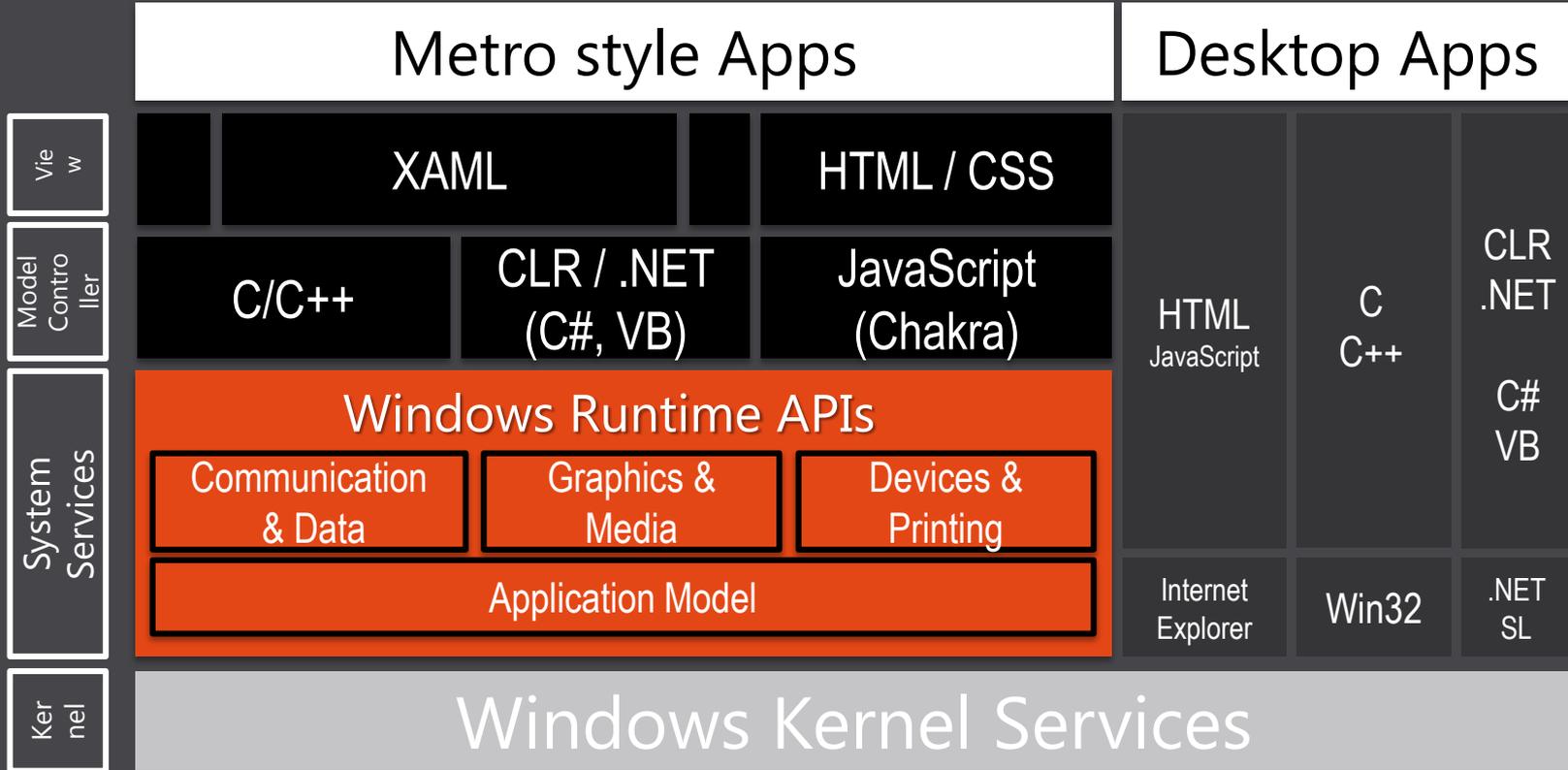
Le nuove spec Metro di WinRT

- Specifiche guidate dalla "tailored user experience"
 - *Immersive:* Le app sono full screen
 - *Engaging and Alive:* I tile sono l'anima dell'applicazione e animati
 - *Connected:* Condividere sui social network. Roaming su più devices
 - *Interactive/touch-first:* Pensate per il multi-touch
 - *Multiple form factors:* Più risoluzioni "standard"
 - *Inspiring confidence:* No virus, disinstallazione indolore
 - *Multitasking:* Le app lavorano in background o a lato dello schermo

I problemi delle applicazioni attuali

- Assenza di un marketplace (innocuità di una applicazione)
- Installazione critica (richiede admin)
- Interoperabilità (PInvoke / COM)
- Costo di esecuzione dei wrapper (memoria e performance)
- Complessità nel mixare linguaggi nativi/managed
 - Le API "C" pongono un problema di ciclo di vita
- Le I/O sono bloccanti (rete, disco, ...)
- Assenza di standard per scambiare dati tra applicazioni
 - postare su FB, cercare, etc.

Windows 8



Cos'è il Windows Runtime

- È l'evoluzione del Component Object Model (COM)
 - Sì, ci sono gli apartment, IUnknown e addref/release
 - Il Type System è totalmente differente
 - No BSTR, No Variant, No RegSvr32
 - Niente più IDispatch, Connection Points
- **WinRT non sostituisce il CLR**
 - Il CLR è sempre lì e deve starci (GC, metadati, etc. etc.)
- I metadati di WinRT sono ECMA-335 (.net Framework)

WinRT Type System

- Tipi base: bool, interi, floating point, enum, guid, type, object
- String: a livello binario è compatibile con STL (wstring) e .NET (string)
 - È immutabile
 - È un value-type (non-nullabile)
- Reference Types: tutti i tipi che implementano interfacce WinRT
- Value Types: gli "altri". Per esempio gli array e structure
 - Structure non possono contenere puntatori/reference (deep/shallow copy)
- Tipi 'evoluti' come Vector e Map (analoghi di collection e dictionary)
 - Sono Observable (Vector<T> implementa IObservableVector<T>)

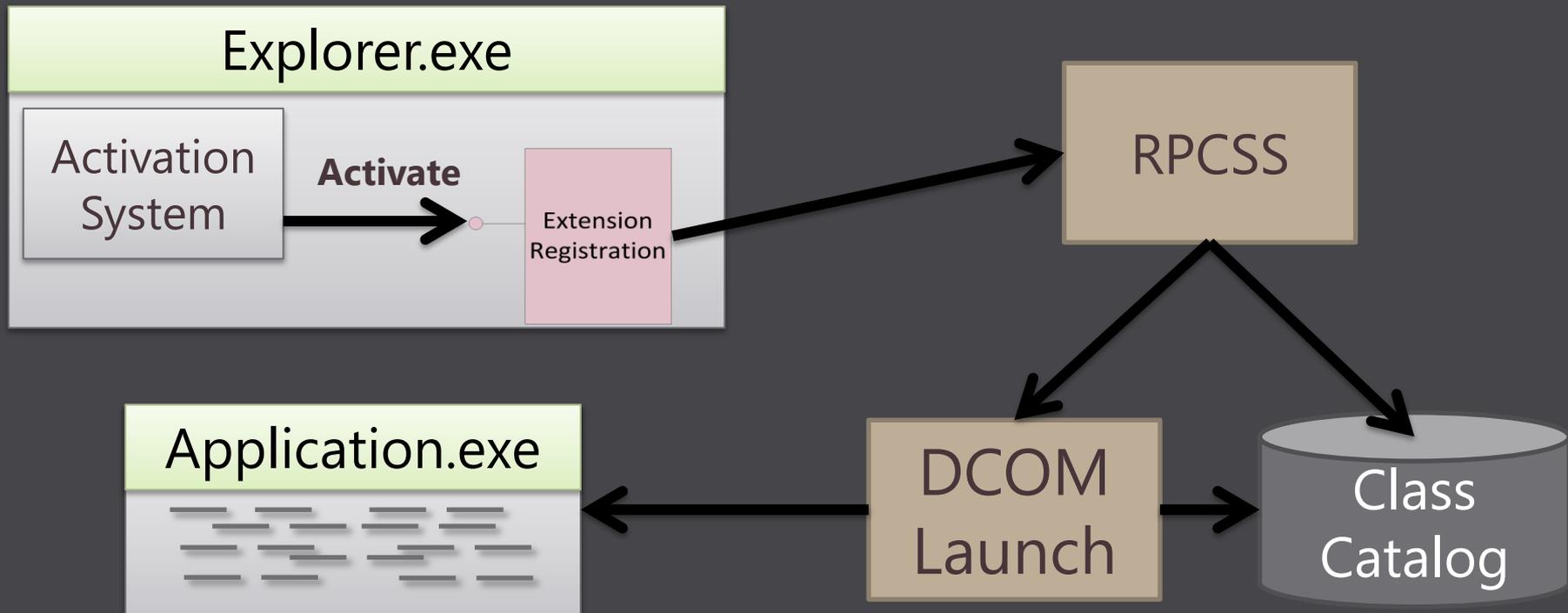
Le Projections

- Sono un layer sviluppato ad-hoc per ogni linguaggio
 - Projection MS: C++, javascript, .NET, ...
- Le projections rimappano/adattano i metadati delle librerie
 - Javascript usa camelCase per metodi/proprietà, PascalCase per i type e lowercase per gli eventi
 - C++ e il Framework.NET usano PascalCase
- I type del linguaggio vengono rimappati a WinRT
 - Javascript supporta solo i numeri IEEE-754 (floating point) quindi il massimo numero in javascript è a 53 bit di precisione
- Projection di terze parti? (Java?, Python?, D?, Delphi?, ...)

I Contracts « *il clipboard del XXI secolo* »

- Stabiliscono le modalità di dialogo tra applicazioni e/o OS
- Realizzano una relazione di publisher / subscriber
- I principali contratti sono:
 - Launch: attivazione dell'applicazione
 - Search: cerca i dati dell'applicazione
 - Share: condivide immagini, documenti, video, etc.
 - FilePicker
 - App to App picking: condivisione di file tra applicazioni
 - UI di sistema per far scegliere all'utente un file

Attivazione per Contratto



Asincrono è il default!

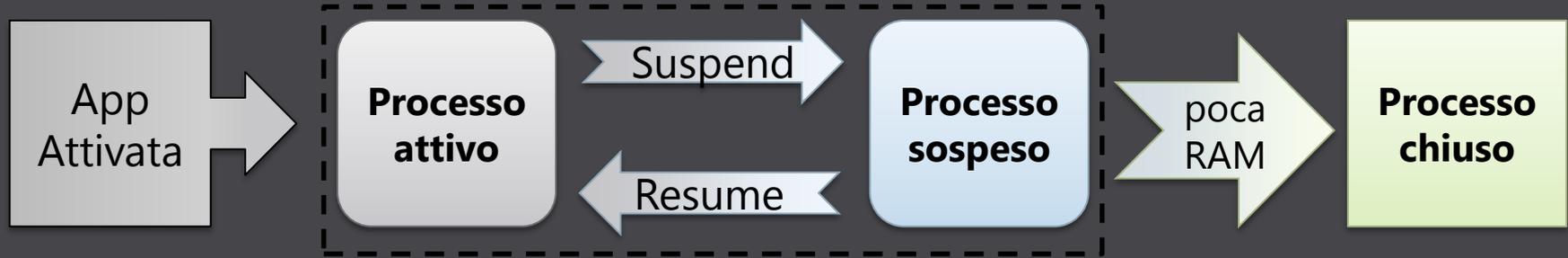
- *Problema*: le App non devono bloccare la UI
- *Soluzione*: le API ≥ 50 ms sono asincrone
- WinRT: `IAsyncOperation<T>`
 - » start manuale.
 - » Convertibile in .net con l'extension method `StartAsTask(...)`
- .net `Task<T>` (.net)
 - » start automatico.
 - » Convertibile in WinRT con `AsyncInfoFactory.Create(...)`
- Ogni linguaggio ha il suo pattern

C# 5.0 await / async

Javascript promises

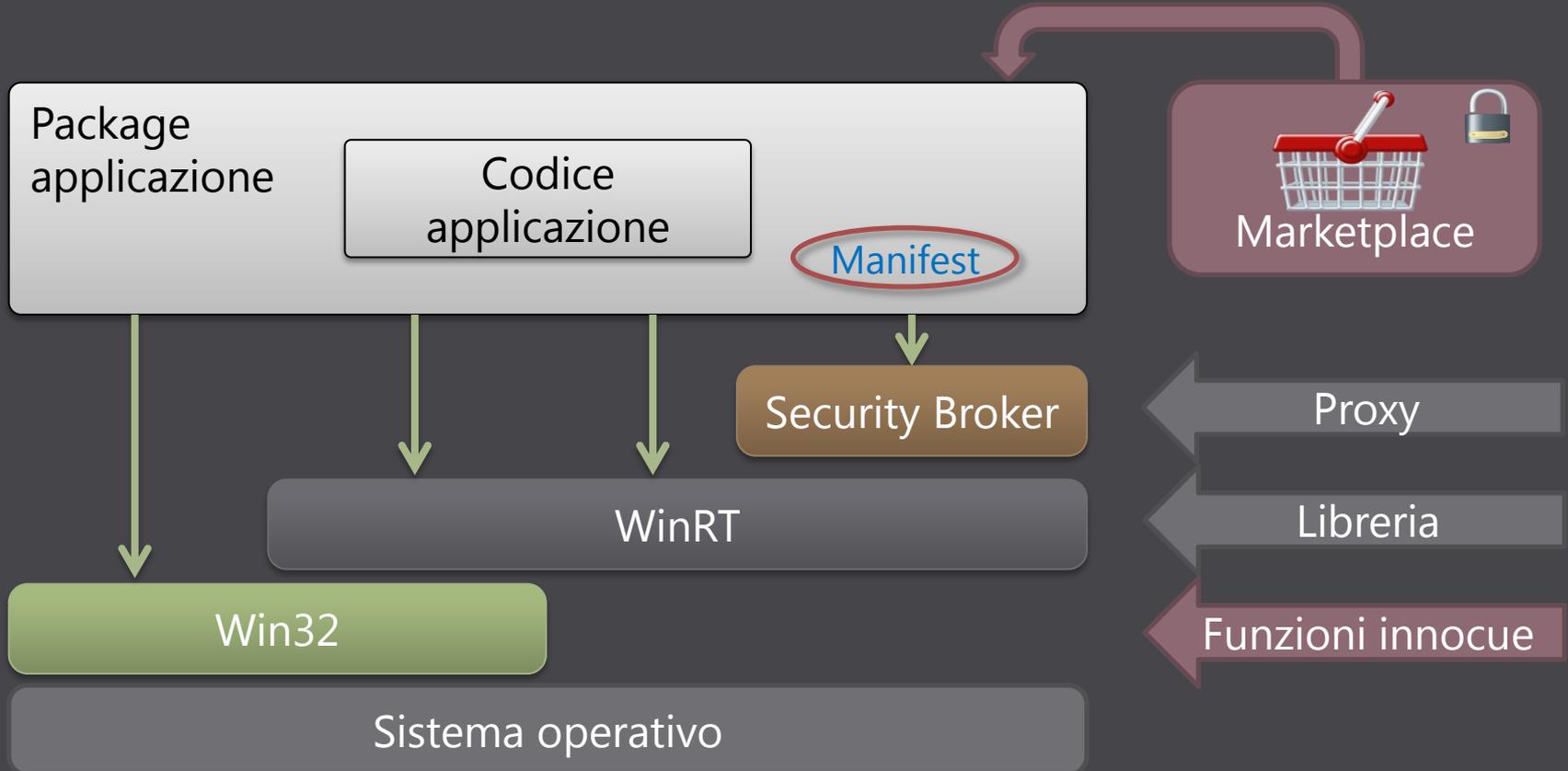
C++ promises / lambda

Il ciclo di vita dell'applicazione



- Dopo 5 secondi che l'app non è visibile viene sospesa
- L'App usa il messaggio di **sospensione** per salvare lo stato
 - **Non** esiste la notifica di **chiusura** del processo (sono già sospesi)
- Il messaggio di **resume** può essere usato dall'App
 - Aggiornamento dati web service, stato dei sensori, etc.

La sandbox



Componenti WinRT custom

- Sono librerie che espongono:
 - Tipi nativi WinRT
 - Metadati ECMA-335 (file WinMD)
- Possono essere realizzati con .net o C++
 - Rimarranno sempre e comunque privati al package
 - **Non** possono essere registrati in modo globale (stile regsvr32)
 - Utilizzano i type WinRT per le funzioni esposte
 - Utilizzano anche altri type (es: .net o STL) per il resto del codice

WinRT e la fine di PInvoke

- Le call a WinRT hanno costo zero – Nessun marshalling
 - Metadati e Type System rendono il layout di memoria compatibile
 - La chiamata ad una funzione WinRT è una call in una VTable
- Le applicazioni Desktop possono usare un subset di WinRT
 - WPF, Winform, etc.
- Benefici:
 - Performance nettamente superiori
 - Più librerie pronte a disposizione
 - Interoperabilità con linguaggi nativi più semplice

Il Profile Metro

	Metro	Fx 4.5	WP7
# assemblies	15	120	22
# namespaces	60	400	88
# types	~1'000	~14'000	~2'000
# members	~10'000	~110'000	~14'000

- Il profile elimina dalle API di .NET:
 - quelle già presenti in WinRT
 - le 'obsolete', 'pericolose', le 'non applicabili' (asp.net, wcf server, ...)
 - le liste non generiche (ArrayList)
 - le classi legacy che non rispettavano le 'design guidelines'
 - System.Data, Remoting, AppDomains, Private Reflection, Reflection.Emit, ...

Il Profile Metro

- Nuovo XAML implementato nativo per WinRT
 - WPF e Silverlight girano su Desktop e Web ma non in Metro
- Esiste una speciale "Portable Class Library"
 - Creazione di dll usabili in Metro, Desktop, Silverlight, WP7, XBox
- Si serializza conDataContractSerializer (più performante)
- Rimossa la dipendenza di System.Type da reflection
 - Ora si usa l'extension method GetTypeInfo
- Rimossi i thread a favore della Task Parallel Library
- DataContractSerializer è il serializzatore preferito
 - È il più performante

Rimozione "illustri"

- XML DOM
 - XElement, XDocument
- System.Security.IsolatedStorage
 - Windows.Storage.ApplicationData
- System.Resources
 - Windows.ApplicationModel.Resources
- System.Net.Sockets
 - Windows.Networking.Sockets
- System.Net.WebClient
 - Windows.Networking.BackgroundTransfer
 - System.Net.HttpClient

Conclusione

- WinRT è un passo fondamentale nell'evoluzione di Windows
- Interoperare tra linguaggi nativi e .net diventa triviale
- Utilizzare i servizi dell'OS è semplice quanto il Framework.NET
- Sviluppare per i device (cellulari, tablet, ... PC) è differente
 - Input da tastiera/mouse a touch/sensoristica
 - "Design for performance" → minore utilizzo di batteria
 - UI radicalmente differente
- WinRT risponde a questi ed altri problemi

Q&A

- Materiale su

<http://www.communitydays.it/>

Windows 8 Cheat Sheet



C

Charms Bar



F

File Search



H

Share charm



K

Connect charm



I

Settings charm



O

Locks device orientation



Q

Search pane



V

Cycles through toasts →



Shift

V

Cycles through toasts ←



W

Settings Search



Y

Temporarily peek at the desktop



Z

Application Bar



Space

Cycle input language, keyboard layout



Return

Narrator



Page Up

Tiles to the left



Page Down

Tiles to the right



Shift

.

Split to the left



.

Split to the right

WinRT.codeplex.com

- Controllo attivazione, suspend, resume, terminate dei package di WinRT

This project hosts tools/utilities targeted to WinRT development
WinRT is the new [Windows Runtime](#) (Windows 8 or above required) that exposes Operating System functionality in an object-oriented fashion.

Current Utilities

- » WinRTDebug is a console application (see also [command line examples](#))
 - » List all the packages installed on the Windows 8 machine
 - » Enable/Disable debugging the package activation
 - » Suspend/Resume a package loaded in memory
 - » Terminate all the processes that host the specified package
 - » Change the SessionId for package-debugging purposes
 - » **New in 1.1:** verbose error messages
 - » **New in 1.1:** support aliases for package names (\$index instead of full package name)

Please note that [Visual Studio 11 Ultimate](#) is required to re-compile the sample(s) as it installs also the latest Windows SDK (include files, libraries, ...).
[Visual Studio 11](#) can be installed over the Visual Studio Express edition that is pre-installed in the Windows 8 developer Preview with tools.

Feedbacks and suggestions in the Discussion pages are always welcome.

Project created and maintained by Raffaele Rialdi

English website: <http://www.iamraf.net>



Italian blog website: <http://blogs.ugidotnet.org/raffaele>

Member of the board of [UGIDOTNET](#) user group



Founder of [DotNetLiguria](#) user group

Proud member of the [MVP Community](#) since 2003

