

# **MICROSOFT DOTNET .NET FRAMEWORK .NET CORE**

François-Xavier BAILLET

version 2018





# Avant Visual Studio

- JAVA, multi OS
- Editeur avec bibliothèque
- AGL style Windev (1993)
- Delphi
- ...

# Pourquoi le .NET : Anders Hejlsberg



- Danemark – 1960
- Développeur d'outils de développement basés sur le langage Pascal
- Rencontre Philippe Kahn en 1986
- Débutent une nouvelle carrière avec

**Borland**



# Borland


- Développement de Turbo Pascal
  - Compilateur ultra performant
  - Prix dérisoire
- Années 90 : Idem avec Delphi et le monde Windows
  - Premières briques de réutilisabilité objets
- Microsoft réagit : débauche Hejlsberg et 30 développeurs de Borland en octobre 1996





# Chez Microsoft

- Hejlsberg créé WFC : les classes Java pour IHM Windows
  - Seul AWT existait
  - Pour Visual J++, machine virtuelle Java Microsoft
  - Litige Sun VS Microsoft
  - Abandon en 2000



# L'orientation objet

- Création du langage COOL
  - C-like Object Oriented Language
  - Annoncé au grand public en Juin 2000
  - Sous le nom de C#
- Arrivée de l'architecture .NET
- Sous la direction de Hejlsberg



# Avant Visual

- VS 97 – 1997
  - Visual Basic 5.0
  - Visual C++ 5.0
  - Visual J++ ...
- VS 6.0 – 1998
  - Idem v6
- VS .Net – 2002 . 2005 . 2008 . 2010 . 2012 . 2013
- ...


# Visual au cours des âges

- Outils de développement complet pour plateforme Windows Microsoft apparu en 2002
- Intégration du Framework .NET
- C# devient le langage de référence
- .NET 1.0 copie de JAVA
- .NET 3.5 beaucoup de nouveauté (LINQ,...)



# Versioning

- VS .Net 2002 - .Net 1
- VS .Net 2003 - .Net 1.1
- VS .Net 2005 - .Net 2
  - Deployment Designer
- VS .Net 2008 – Net 3.5 (Majeur)
  - Outils de métrologie
  - LINQ (requête « unique »)
  - Plateforme ALM
- VS .Net 2010 - .Net 4.0
  - Refonte graphique
  - Ajout de composants dans la plateforme
- VS .Net 2012 - .Net 4.5
  - Calé sur Windows 7 & 8
  - Environnement Windows RT
  - SDK Windows Phone 8.0 (non inclus)

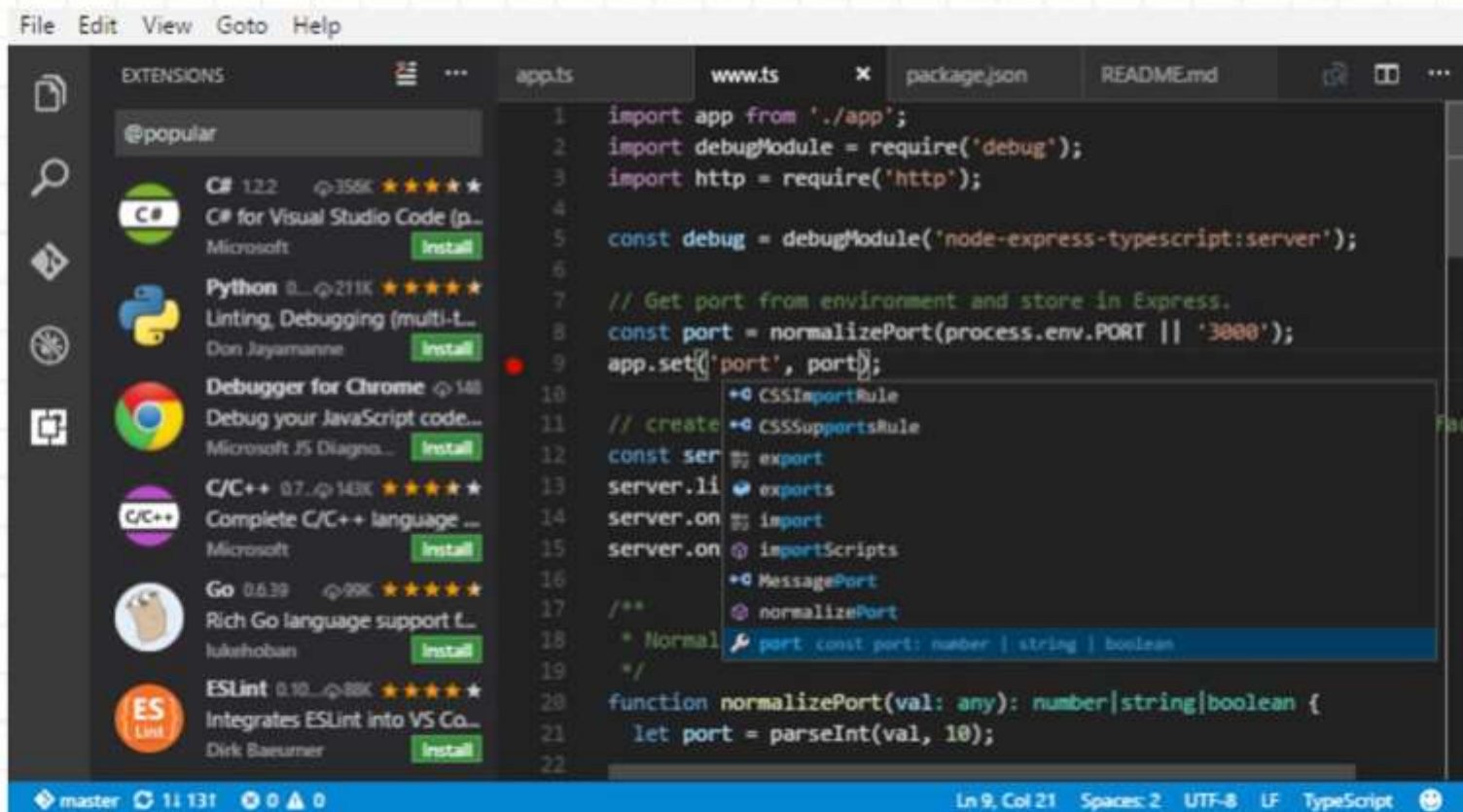


# Versioning

- VS .Net 2013 - .Net 4.5
  - Outils pour développeur (Code Lens, Peak Définition...)
  - XAML : IntelliSense (auto-complétion) couvre le Data Binding
- VS .Net 2015 - .Net 4.6
  - Multiplateforme mobile
  - Services (Azure, office 365)
  - Xamarin (C# => Android, iOS et Windows)
- VS .Net 2017 (15.5) - .Net 4.7.1 => Windows 10 v1709
  - Amélioration installation
  - Retour arrière pendant le débogage
  - Améliorations diverses et variées.

# Visual Studio Code

- Depuis avril 2015 (Free, Open Source)  
multi langages multi débbugger
- Version 1.19 depuis fin 2017







# Plateforme DotNet

- Plateforme .NET => exécution de  $\neq$  langages
- Compilateurs produisent du code IL (Intermediate Language) exécuté par la machine virtuelle .NET
- => Développement "simplifié"

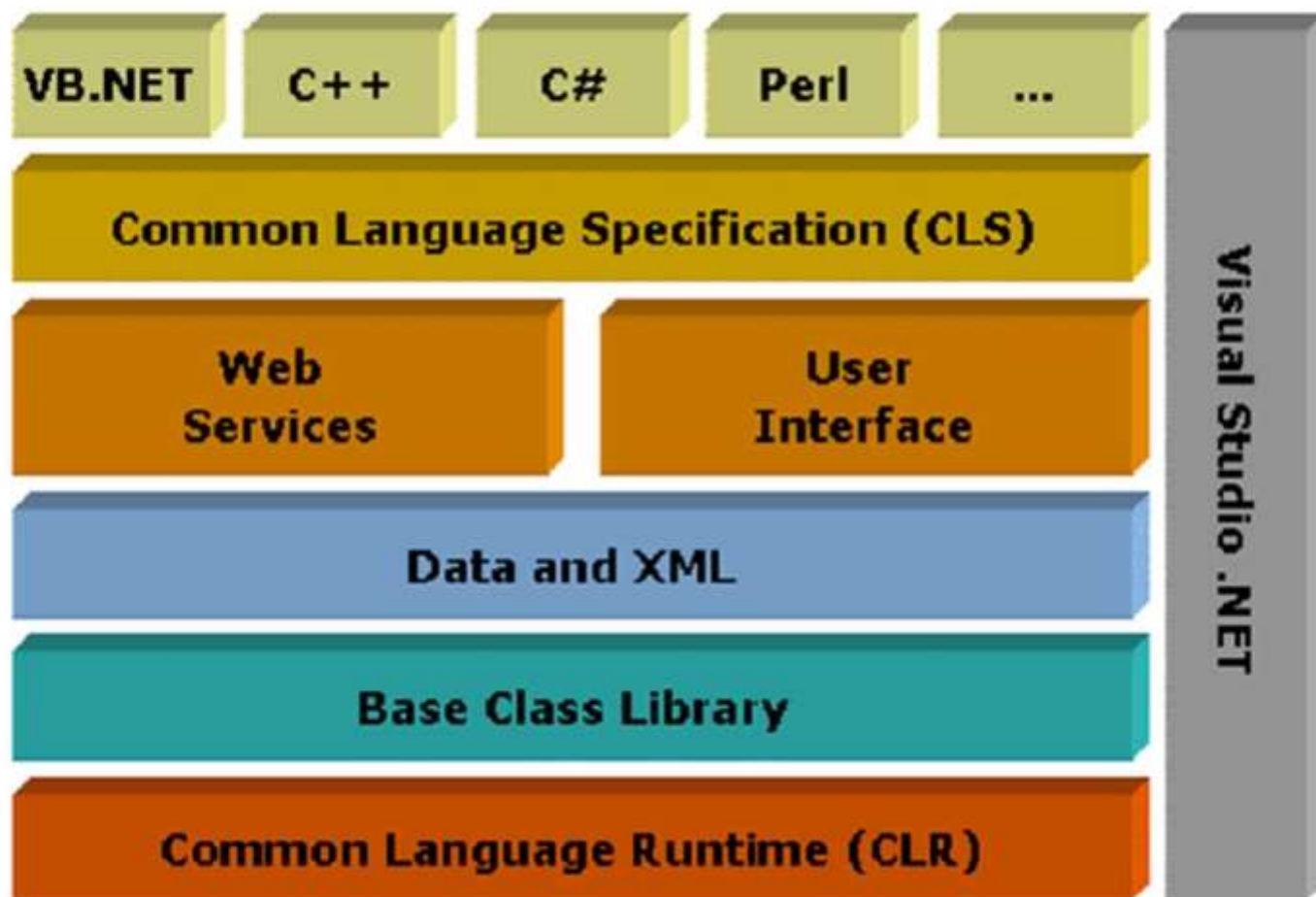




# Architecture DotNet

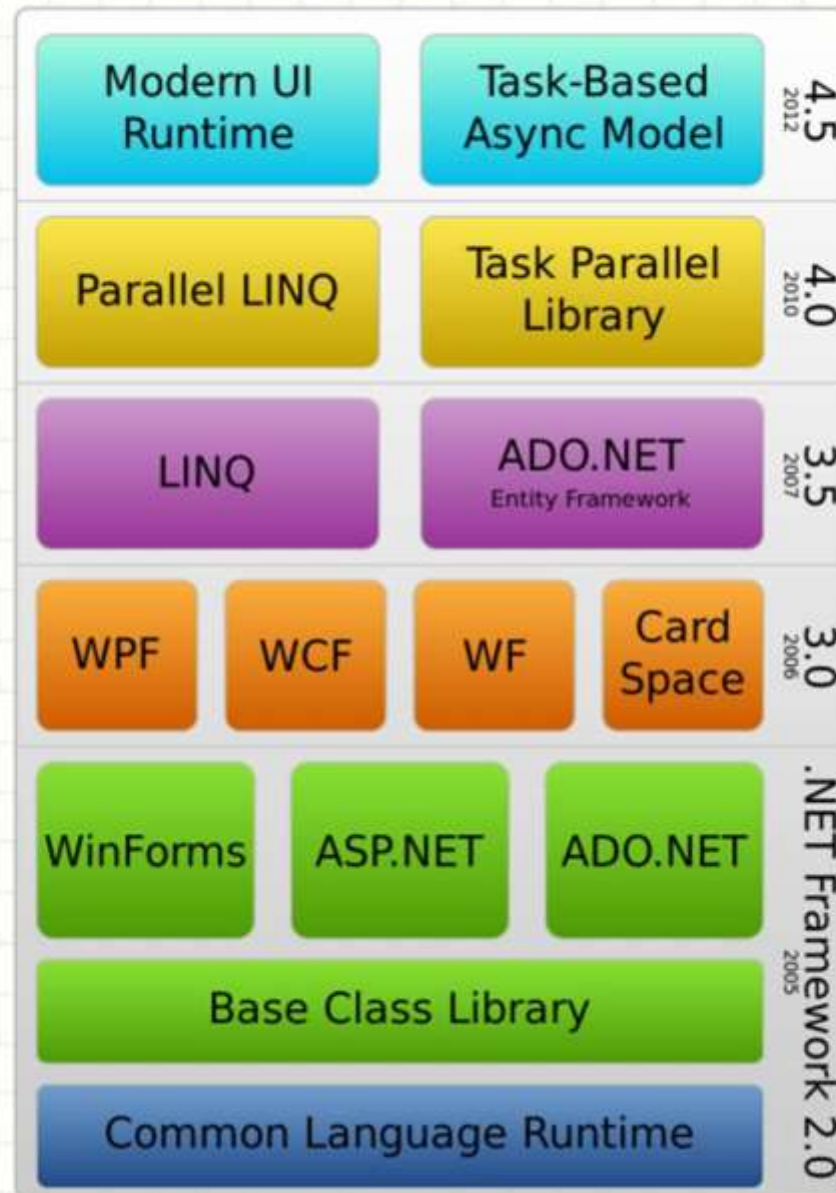
- Next Generation Web Services (NGWS) renommé en .NET
- Couche Windows (Collections de DLL)
- Intégré au Noyau de l'OS
- Milliers de classes (avec les namespaces)
- Moteur d'exécution (runtime) : CLR (Common Language Runtime) => machine virtuelle du framework .NET

# Architecture DotNET



© TechMetrix Research, 2001

# Le Framework .NET



The .NET Framework Stack



# Runtime .NET et le CLR

- Indépendance de l'architecture (MSIL ou IL)  
=> baisse des performance
- Ramasse-miettes (garbage collector) =>  
Récupération mémoire
- Sécurité du code : accès à la machine réelle  
réalisé par la machine virtuelle (droits)
- Support multi-langage



# Runtime .NET et le CLR

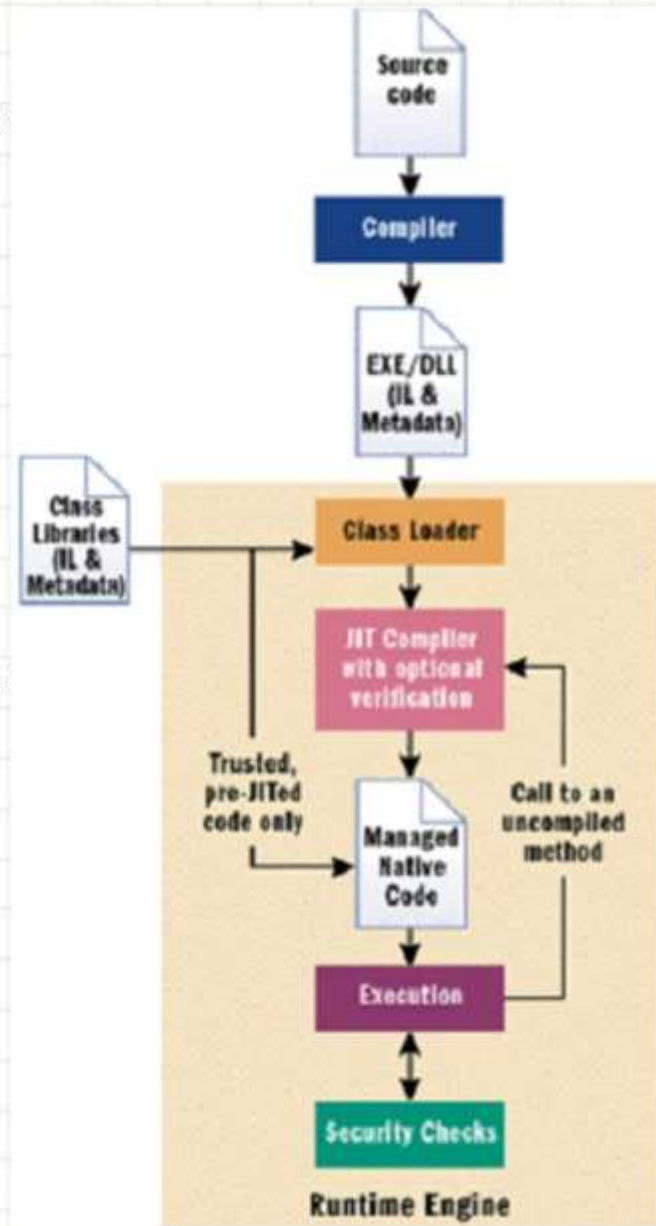
- Gestion intégrée des versions : notion assemblage (assemblies)  $\equiv$  paquetages machine virtuelle JAVA
- Sûreté du typage, évite Access Violation...
- Programmation simplifié : fin de l'API Win32
- Erreurs gérées par exception

# La "Compilation"

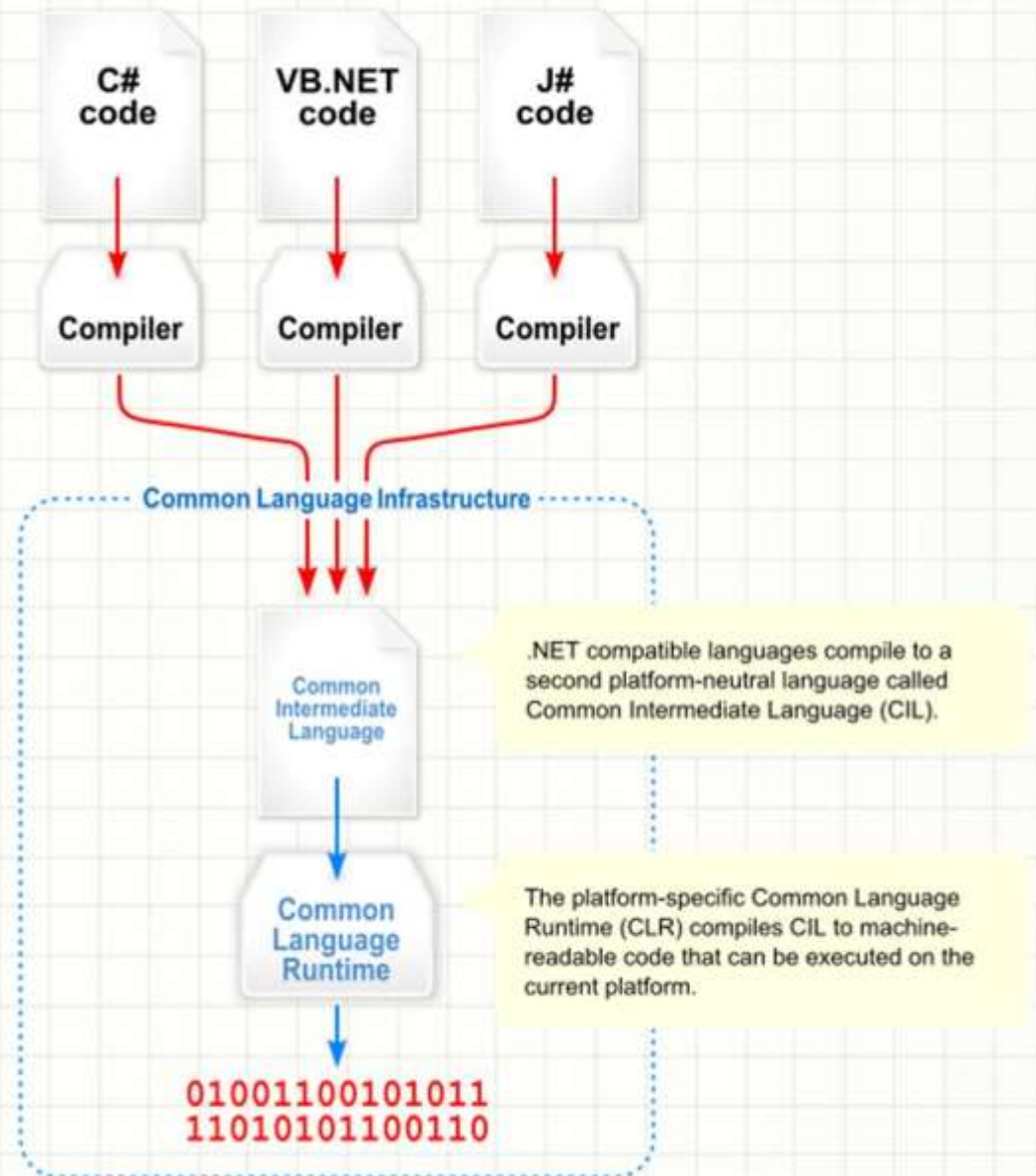
- Le code "source" IL est lisible (pseudo assembleur), un 'Hello Word' classique devient :

```
.assembly hello {}  
.assembly extern mscorlib {}  
.method static public void main( ) il managed {  
    .entrypoint  
    .maxstack 1  
    ldstr "Hello World"  
    call void [mscorlib]System.Console::WriteLine(class  
                                                    System.String)  
    ret  
}
```

- P-Code compilé par un compilateur Just-In-Time (JIT) → Compilation 'à la volée'
- C#, VB.NET, Visual C++,... doivent avoir les mêmes types de données :
- CTS (Common Type System)



# La "Compilation"







## Au final

- Exécution sécurisée
  - Le programme est contrôlé par .NET à l'exécution
  - Peu d'interaction direct avec le système, mais reste possible.
- ➔ A vous de le maîtriser sous peine de :



A problem has been detected and windows has been shut down to prevent damage to your computer.

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to be sure you have adequate disk space. If a driver is identified in the Stop message, disable the driver or check with the manufacturer for driver updates. Try changing video adapters.

Check with your hardware vendor for any BIOS updates. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

\*\*\* STOP: 0x0000007E (0xC0000005,0xF88FF190,0x0xF8975BA0,0xF89758A0)

\*\*\* EPUSBDISK.sys - Address F88FF190 base at FF88FE000, datestamp 3b9f3248

Beginning dump of physical memory



## Services

- ASP.NET Web API Cloud optimized
- ASP.NET SignalR Cloud optimized
- WCF
- WCF Data Services
- Windows Azure Services
- Entity Framework
- Service Bus for Windows Server
- Service Bus for Windows Azure

## Windows Store Apps

- Universal Windows App
- .NET Native
- WinRT XAML/.NET Windows Store SDK
- AppX Project for Windows Runtime
- AppX Project for Windows Store Apps

## Windows Phone Store Apps

- Universal Windows App
- .NET Native
- Windows Phone SDK

## Cloud Apps

- Microsoft Azure .NET SDKs
- Microsoft Azure Storage Lib
- Microsoft Azure Configuration Manager Lib
- Microsoft Azure Media Services .NET SDK
- Microsoft Azure Mobile Services
- AppX Project for Windows Store Apps
- AppX Project for Windows Phone
- AppX Project for Windows RT

## Web Apps

- ASP.NET MVC Cloud optimized
- ASP.NET Web Pages Cloud optimized
- ASP.NET Web Forms
- ASP.NET Web API
- ASP.NET Web API Client

## Core

Get the .NET  
technology guide



### .NET Runtimes

.NET Computer Platform ("Roslyn")

Languages (C#, VB, F#)

Base Class Library

## Partners Cross Device Apps

- Remix
- MS Security Framework
- Onix Mobile SDK for Windows App

## Internet of Things (IoT)

- .NET IoT Framework
- .NET IoT Gateway

## Desktop Apps

- Windows Presentation Foundation
- Windows Forms
- AppX Project for Desktop

## .NET Extension Libs

- Async
- WebClient
- Immutable Collections
- TPS, Synchronization
- Reactive Extensions
- Reactive Extensions in Async
- Parallel Extensions
- WP Activities Extensions
- AppX Services Logging App Block
- Portable Class Libraries
- AppX SDK - Validation App Block
- AppX SDK - Exception Handling App Block
- AppX SDK - Logging App Block
- Compression

## Data Access

- Entity Framework
- ADO.NET
- ASP.NET Universal Providers
- .NET App Project SDK for Windows
- .NET App Project SDK for Windows
- Using in Hubs
- Using in Web
- Using in Web
- AppX Data Access App Block

## DI and IoC Containers

- Composition .NET 4
- NET (Managed Extensibility Framework)
- AppX Unity

## Caching

- Microsoft Azure Caching
- Windows Server AppFabric Caching
- Microsoft Azure Caching Memcache Store
- ASP.NET Cache

## Security

- ASP.NET Identity
- OAuth2/OpenId Connect
- Windows Identity Foundation
- Authentication Manager (AuthM)
- Web Protection Library
- OAuth Authentication Middleware
- Microsoft Azure AD

## Emerging Application Patterns

## Established Application Patterns

## Cross-Cutting Patterns

## NuGet Package

## Open Source

## Support (MS Official)



Like it? Get it.

Microsoft

WPF

Windows  
Forms

ASP.NET (4 & 5)

ASP.NET 5

Universal  
Windows Apps

**.NET Framework 4.6**



*Fully-featured and integrated .NET runtime and libraries for Windows*

**.NET Core 5**

CoreCLR



.NET Native runtime



*Modular and optimized .NET runtimes and libraries*

**Shared**



**Runtime  
Components**

RyuJIT, GC, SIMD



**Compilers**


.NET Compiler Platform (Roslyn)  
Languages innovation



**NuGet packages**

.NET Core 5 Libraries  
.NET Framework 4.6 Libraries





# .Net Core

- Open Source
- C#, F# Visual Basic
- But : Sites Web, apps Serveurs & Console
- Portage difficile à partir de .NET Framework
- Microsoft : 20000 APIs
  - ➔ pack de compatibilité Windows
- Transition avec 50000 APIs
  - 50% apps Windows
    - ➔ Analyseur d'API



# Porting to .NET Core

## Reasons not to port

~~Not enough APIs~~

You're building desktop applications (WinForms, WPF)

You're building ASP.NET Web Form apps

You're simply happy with the status quo

→ Stay on .NET Framework!

## Reasons to port

You want to build highly scalable web apps

You want to run your web apps on Linux

You want self-contained deployments

→ Port to .NET Core!

# Windows Compatibility Pack

## Provided as a NuGet Package

Microsoft.Windows.Compatibility

Can be referenced from .NET Core & .NET Standard

Has ~20k APIs (Windows-only as well as cross-platform)

## Contents

ACLs

Code Pages

CodeDom

Configuration

Crypto

DirectoryServices

Drawing

EventLog

MEF

Odbc

Perf Counters

Permissions

Ports

Registry

Runtime Caching

WCF

Windows Services

...

# Too Easy ?

```
private static string GetLoggingPath()
{
    #pragma warning disable PC001
    if (RuntimeInformation.IsOSPlatform(OSPlatform.Windows))
    {
        using (var key = Registry.CurrentUser.OpenSubKey(@"Software\Fabrikam\AssetManagement"))
        {
            if (key?.GetValue("LoggingDirectoryPath") is string configuredPath)
                return configuredPath;
        }
    }
    #pragma warning restore PC001

    var appDataPath = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
    return Path.Combine(appDataPath, "Fabrikam", "AssetManagement", "Logging");
}
```

➔ Mise en place de Pragma pour le compilateur et de tests de plateformes dans le code



### Quelques sources :

Microsoft Developer Network : [msdn.microsoft.com](http://msdn.microsoft.com)

<https://msdn.microsoft.com/fr-fr/library/dn878908%28v=vs.110%29.aspx> pour .NET Core

"Apprentissage du langage C# 2008" de Serge Tahé

"ADO.NET – Base de données", "Création et consommation de services WCF" de Dotnet France

"Introduction à Open Data Protocol et WCF Data Services" de F.Casabianca - *Developpez.com* et de Wikipédia ;-)

Portage vers .Net Core

<https://www.youtube.com/watch?v=q0G5UFhUk2A&feature=youtu.be>