



.net ?



.net c'est quoi ?

- La plus grande évolution dans le monde du développement depuis Java.
- La plus grande évolution chez Microsoft depuis le passage de DOS à Windows.
- Des nouveaux langages (CIL, C#, VB.NET...).
- Des spécifications techniques normalisées par l'ECMA et le W3C.
- Des méthodes de programmation « Agile ».
- Une bibliothèque de fonctions reprenant la totalité de l'API Windows.
- Une programmation **orientée métier** plutôt que matériel.



mais aussi...

- Simplification du développement.
- Simplification du déploiement et de la maintenance des applications.
- Mise à disposition d'un environnement d'exécution plus robuste et sécurisé.
- Support d'un grand nombre de langages.
- Nouvelle terminologie et de nouvelles habitudes.



Multi langages

- Chaque programmeur utilise le langage qu'il préfère.
- Toutes les fonctionnalités de la plateforme .NET sont disponibles dans tous les langages
- Les différents composants d'une application peuvent être écrits dans différents langages
- Debuggers, Profilers, Analyseurs de code sont disponibles pour tous les langages



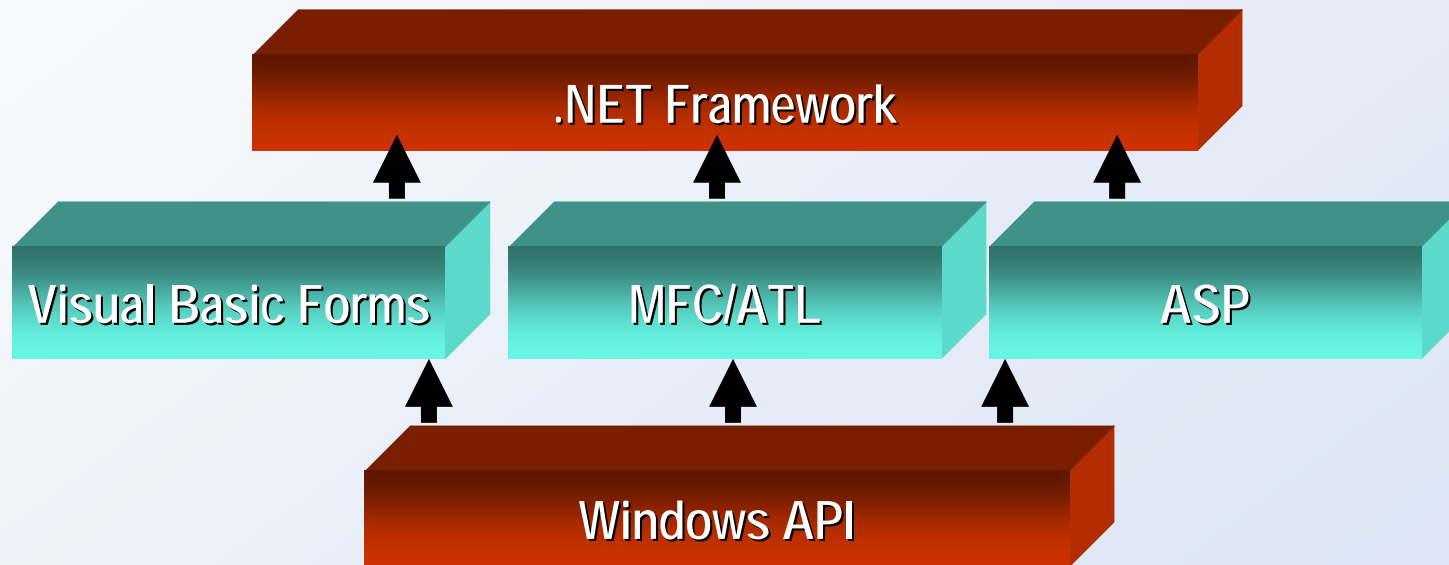
Quelques langages disponibles

- A# (Ada)
- Abstract IL (IL+OCaml)
- Active Oberon
- ActiveState Python
- ASNA Visual RPG
- BETA
- Boo (python)
- **C#**
- C!
- Component Pascal
- **Delphi 2005**
- Delta Forth .NET
- DotLisp
- Dyalog APL
- Eiffel
- F# (ML+CamI)
- Glasgow Haskell
- Haskell.NET
- Hugs98 (Haskell)
- HotDog Scheme
- IL (a.k.a.MSIL, CIL)
- ILX (functional IL)
- IronPython
- JScript.NET
- (ECMAScript)
- Lahey Fortran
- Lexico (educational)
- Mercury
- (Prolog, kinda)
- Mondrian
- MonoLOGO
- Nemerle (functional C#)
- NetCOBOL
- Net Express (MicroFocus COBOL)
- Oberon
- PerINET
- Python
- Salford FTN 95 (Fortran)
- Scheme.NET
- S#
- (Smalltalk 98)
- #Smalltalk
- SML.NET (Standard ML)
- Tachy (Scheme-like)
- TMT .NET Pascal
- **Visual Basic**
- **Visual C++**
- **Visual J# (Java)**
- Zonnon (Oberon trend)



Connectivité et facilités de .NET

- Basé sur les standards web et la pratique.
- Facilité de développement.
- Classes extensibles.





Exemple de langage 1

Exemple: Visual C++ (Managed)

```
#using <mscorlib.dll>
using namespace System;
__gc public class HelloWorldCPP
{
    public:
    void SayHelloCPP()
    {
        Console::WriteLine("Hello World from C++!");
    }
};
```



Exemple de langage 2

Exemple: Visual Basic

```
Imports System
Imports HelloWorldCPP

Public Class HelloWorldVB
    Inherits HelloWorldCPP
        Sub SayHelloVB()
            Console.WriteLine ("Hello World from Visual Basic!")
        End Sub
    End Class
```



Exemple de langage 3

Exemple: Cobol

```
CLASS-ID. HelloWorldCOB INHERITS HelloWorldVB.  
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
REPOSITORY.  
        CLASS HelloWorldVB AS "HelloWorldVB"  
OBJECT.  
PROCEDURE DIVISION.  
METHOD-ID. SayHelloCOB.  
PROCEDURE DIVISION.  
        DISPLAY "Hello World from COBOL!".  
END METHOD SayHelloCOB.  
END OBJECT.  
END CLASS HelloWorldCOB.
```



Exemple de langage 3

Exemple: C#

```
using System;

class HelloWorldCS: HelloWorldCOB
{
    public void SayHelloCS()
    {
        String message = "Hello World from C#!";
        Console.WriteLine(message);
    }

    public static int Main()
    {
        HelloWorldCS h = new HelloWorldCS();
        h.SayHelloCPP();
        h.SayHelloVB();
        h.SayHelloCOB();
        h.SayHelloCS();
        return 0;
    }
}
```



Simplification du développement

Windows API (C++)

```
HWND hwndMain = CreateWindowEx(  
    0, "MainWinClass", "Main Window",  
    WS_OVERLAPPEDWINDOW | WS_HSCROLL | WS_VSCROLL,  
    CW_USEDEFAULT, CW_USEDEFAULT,  
    CW_USEDEFAULT, CW_USEDEFAULT,  
    (HWND)NULL, (HMENU)NULL, hInstance, NULL);  
ShowWindow(hwndMain, SW_SHOWDEFAULT);  
UpdateWindow(hwndMain);
```

.NET Framework (C#)

```
Form form = new Form();  
form.Text = "Main Window";  
form.Show();
```

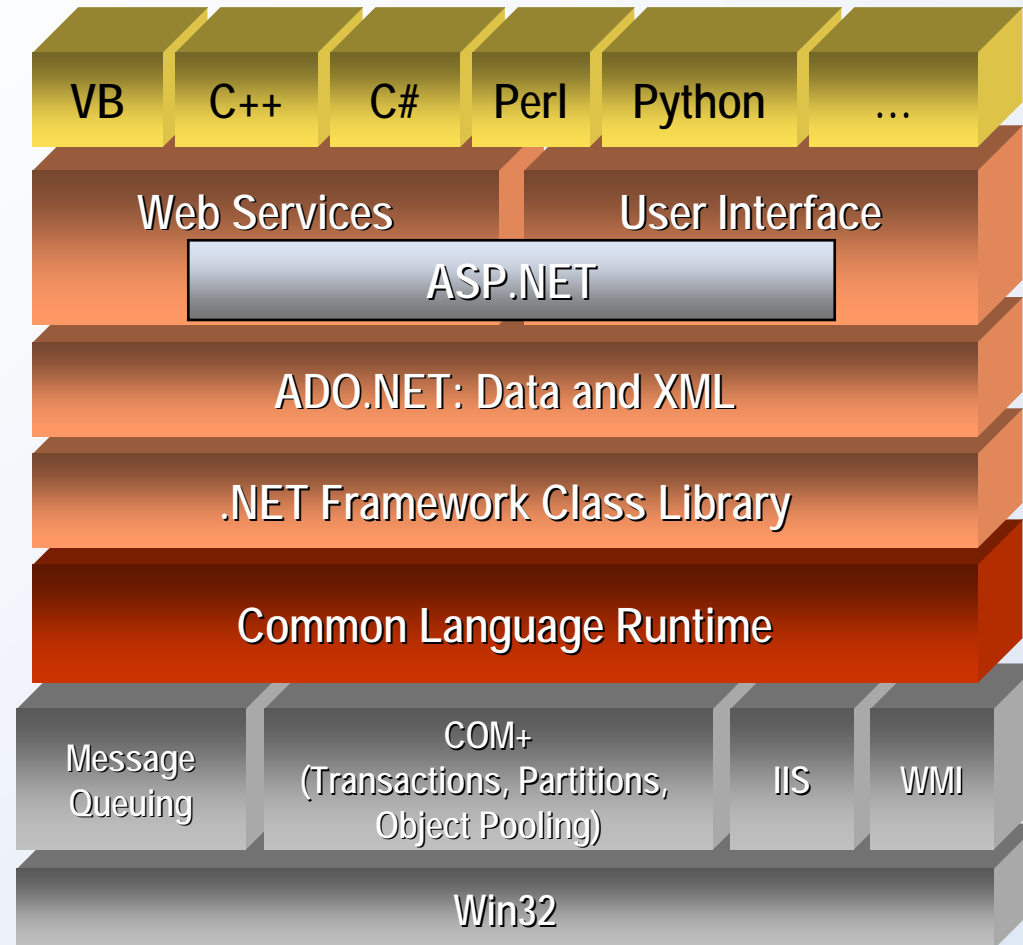


Robuste et sécurisé

- Tous les objets de .NET sont gérés par un **garbage collector** (ramasse-miettes). **Pas de pointeurs perdus**, pas de références circulaires.
- Types protégés et vérifications strictes. IL vérifie les types, pas de “casts” non protégés, pas de variables non initialisées, pas de débordement de tableaux.
- **Sécurité intégrée** : basée sur l’origine du code et de l’utilisateur. Extension possible des permissions.



Architecture de .NET





Common Language Runtime (CLR)

Le **CLR** prend en charge essentiellement :

- Le chargement des classes.
- Les vérifications de types.
- La gestion de la mémoire, des exceptions, de la sécurité.
- La traduction à la volée du code MSIL en code natif (compilateur interne JIT), à travers le CTS (Common Type System) qui implémente le CLS (Common Language Specification)
- Le CLR assure la sécurité de compatibilité des types connus, mais syntaxiquement différents selon les langages utilisés.



Common Language Runtime (CLR)

