



.Net (DotNet)

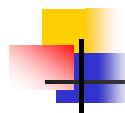
Ch A – La plateforme .Net



B. Chervy
2023

© 3iL Limoges - 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

Sommaire



1. - La plateforme et l'architecture .Net
2. - Les classes, namespaces et APIs .Net
3. - Les langages VB.Net et C#
4. - Visual-Studio .Net et les autres IDE



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL



1. - La plateforme et l'architecture .Net



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

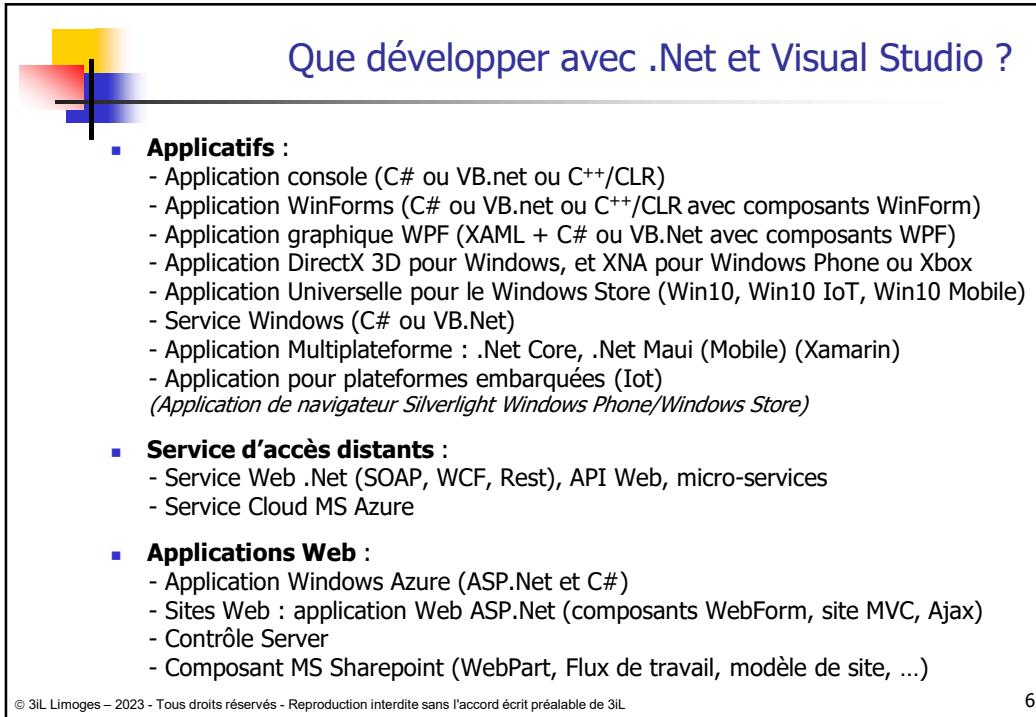
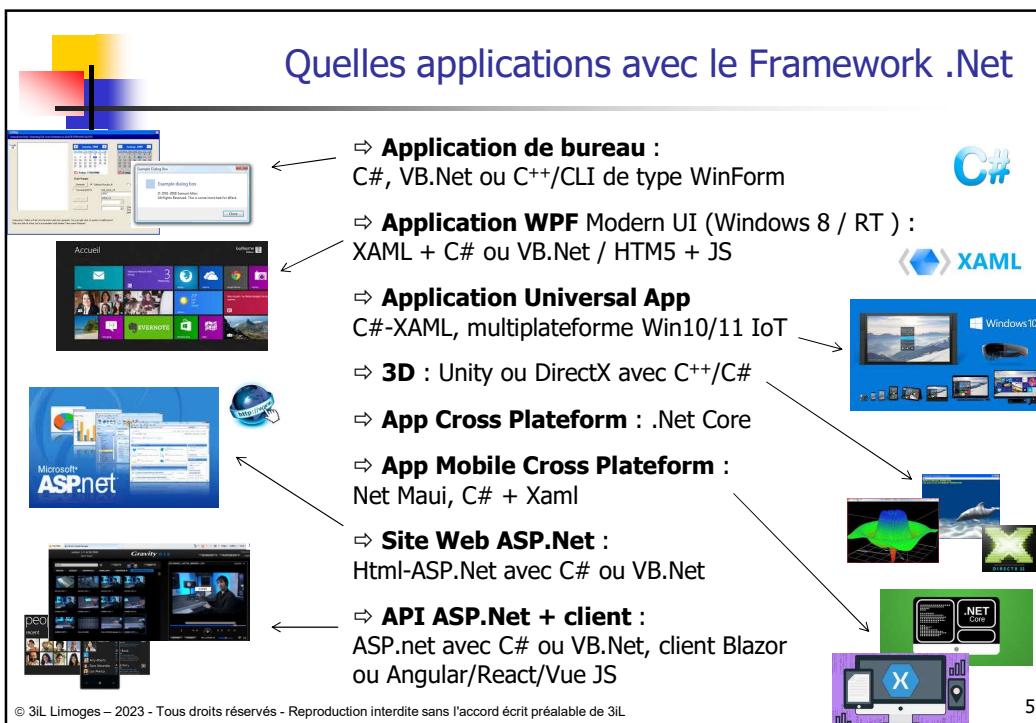
3

La plate-forme .NET

- **.NET (Microsoft)** : plate-forme de conception d'applications et de services Web
 - ⇒ initiée au début des années 2000 par Microsoft, elle n'a depuis cessé d'évoluer et de s'étendre pour devenir au fil des ans une technologie « phare »
 - ⇒ **plateforme de développement** gratuite, multiplateforme, open source qui permet de créer de nombreux types d'applications
 - ⇒ repose sur une **runtime** hautes performances utilisée en production par de nombreuses applications à grande échelle.
 - ⇒ applications clientes Windows, multiplateformes, cloud, Web, services, ...
 - ⇒ interopérabilité entre les langages supportés
 - ⇒ possibilité de développer avec le Framework .NET sur des Os non-MS
- ⇒ Les programmes peuvent s'exécuter sur toutes les plates-formes possédant un **framework .NET**, le code étant géré par un runtime appelé **CLR** (Common Language Runtime) (dans l'esprit, similaire au concept de machine virtuelle Java).
- ⇒ Framework 100% objet qui intègre les langages **C#, VB.Net, C++, ASP.Net, ...**
- ⇒ **Visual Studio .NET** constitue un environnement de développement destiné à la création d'applications sur **.NET Framework**.

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

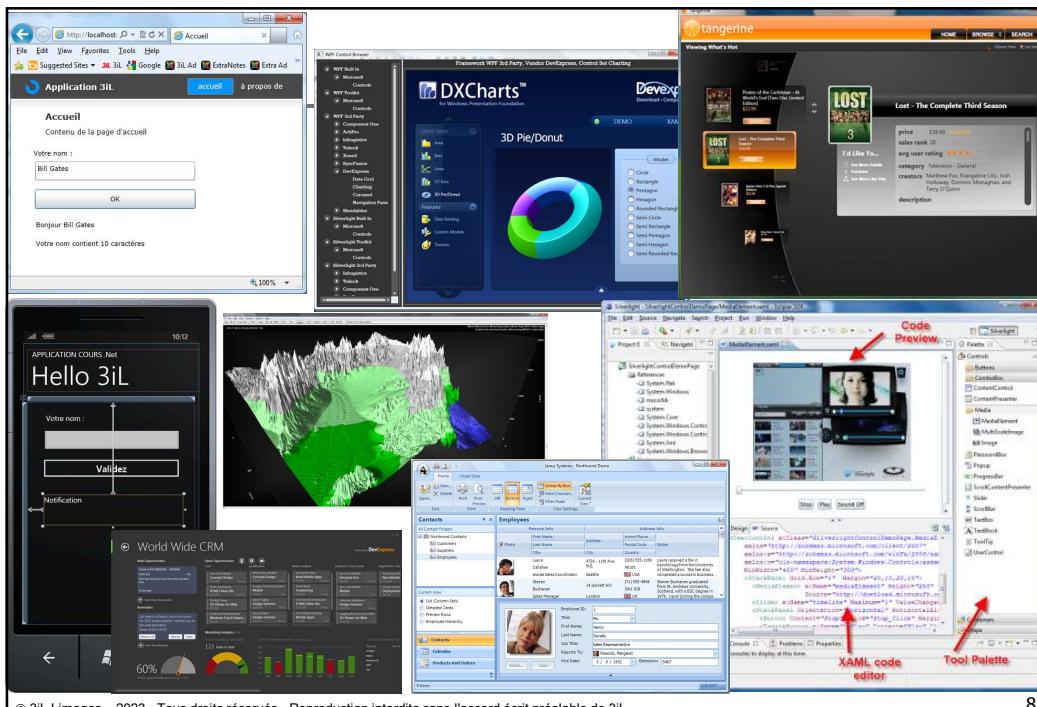
4



- Le **Framework .Net** permet de réaliser :
 - des applications orientées Serveur (Services Windows, Applications Web (ASP.Net), Web Services, Cloud, ...)
 - des applications classiques pour postes clients (WinForms)
- ⇒ **Avec une seule et même plate-forme**
 - ⇒ Approche client riche : **WinForms, WPF, 3D**
 - ⇒ Approche client léger : **WebForms** ou **MVC** en **ASP.NET** pour applications web
 - ⇒ Approche Universelle multi-plateformes : **UWP, .Net Native, .Net Maui**
 - ⇒ Accès aux données : **ADO.Net, Linq, SW, EF**, ... accès local ou à distance
- ⇒ support de **toutes les plate-formes Microsoft**, du Windows 98 à Windows 11
- ⇒ versions pour **plate-formes mobiles**
- ⇒ versions pour **plate-formes embarquées** (Windows 10 IoT, µFramework.Net, Compact Framework .Net, Silverlight embedded)
- ⇒ **interopérabilité** entre les langages supportés
- ⇒ permet une cohérence des technologies du système d'information de l'entreprise
- ⇒ technologies Internet basées sur les normes **http, XML, SOAP, WCF, REST**
- ⇒ permet la conception d'**applications Web** et de services Web

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

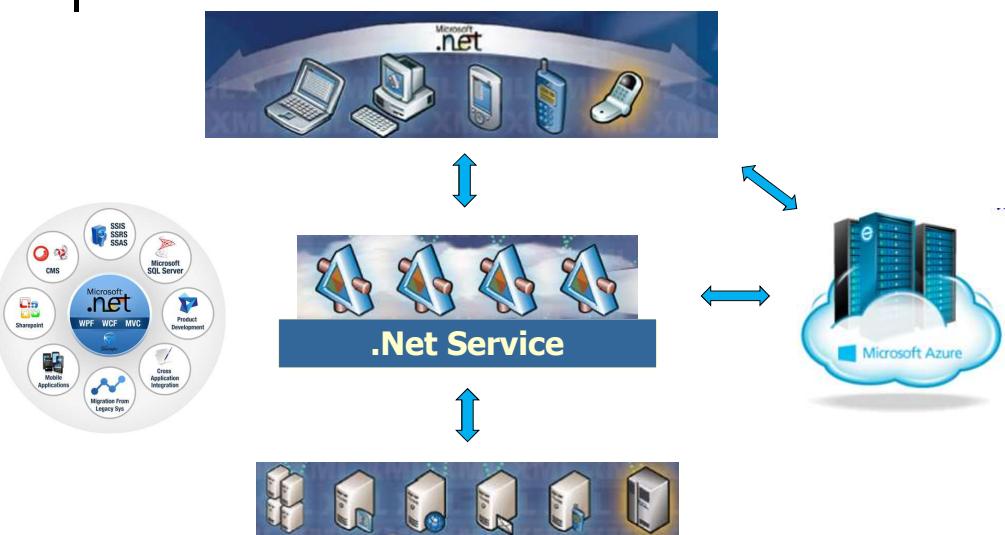
7



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

8

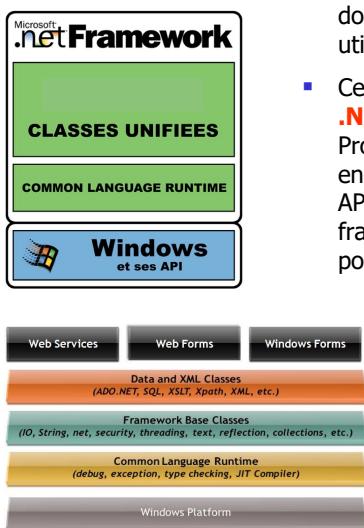
La communication sous .Net



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

9

Code et classes unifiées du Framework .NET

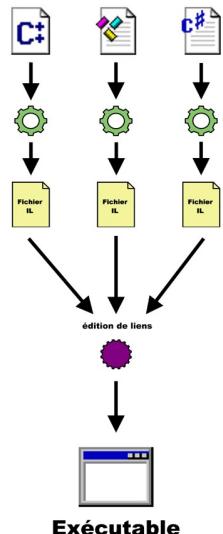


- Le Framework fournit un grand nombre de classes de base dont les fonctionnalités sont accessibles à tous les langages qui utilisent le **CLR**.
- Cela permet, sur tous les systèmes disposant d'un **Framework .NET**, de ne plus avoir à passer par des API (Application Programming Interface) spécifiques à l'OS et de programmer en utilisant les **classes unifiées de .NET** développées sur ces API (principe voisin de celui des langages interprétés : un framework pour chaque système et des programmes portables).
- Bibliothèques de classes très importantes ⇒ boîte à outils complète pour le développeur (28600 classes dans le Framework .NET 2.0)
- Intègre en natif la gestion de la sécurité du code
- .NET Framework 4.7 intégré au système w10 (4,5 Go, version 32 et 64 bit, compatible w7 & w8) (Vista limité à v4.6, XP v4)

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

10

Code managé et langage intermédiaire

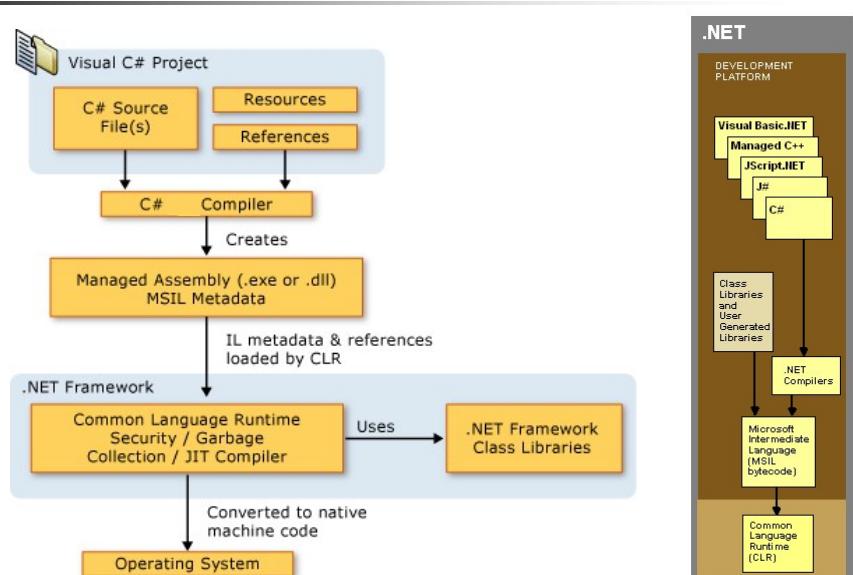


- Le code généré à partir d'un langage .NET sur la plate-forme .NET recevra le nom de **code managé** (vient du fait qu'il est pris en charge par la CLR tout au long de sa vie).
- Attention donc à ne pas confondre le C++ (classique) et le Managed C++ utilisant les possibilités des classes unifiées du framework .NET.
- Tous les langages supportés par la plate-forme sont compilés dans un **langage intermédiaire**, appelé **IL** ou **MSIL** (pour Langage Intermédiaire) : permet de lier ensemble des composants écrits dans des langages différents
 - ⇒ Permet de développer un composant pour un langage, et transmettre le binaire IL qui pourra s'utiliser dans d'autres langages et sur d'autres systèmes d'exploitation.
 - ⇒ Permet d'hériter de l'implémentation de composants entre divers langages et de déboguer plusieurs langages à l'aide d'une seule bibliothèque.

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

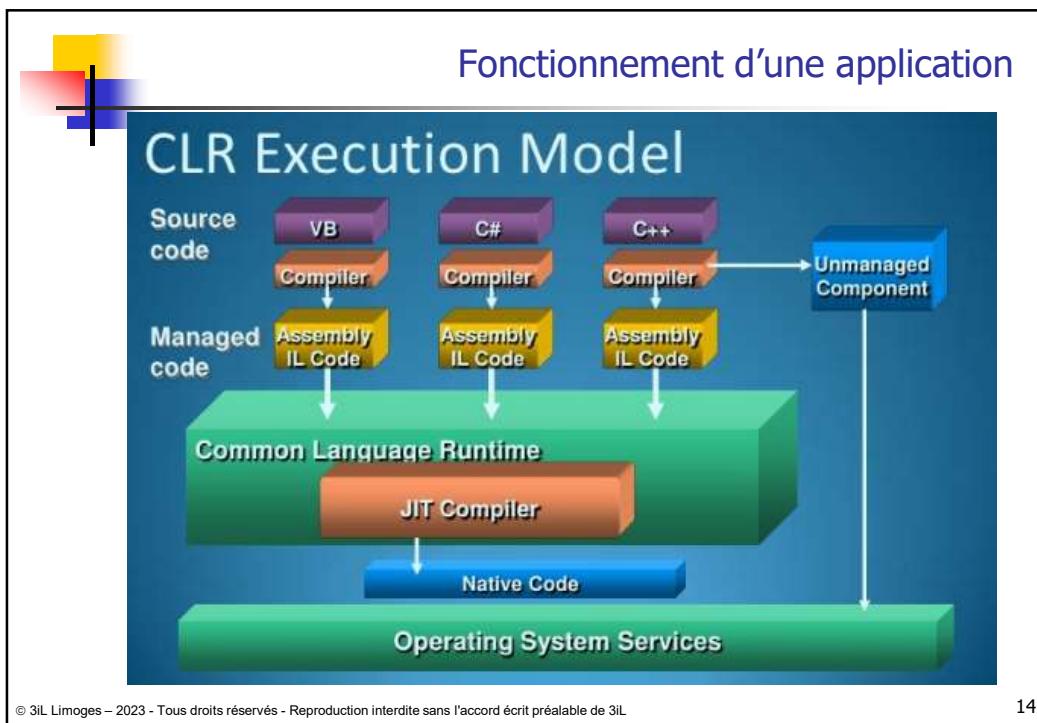
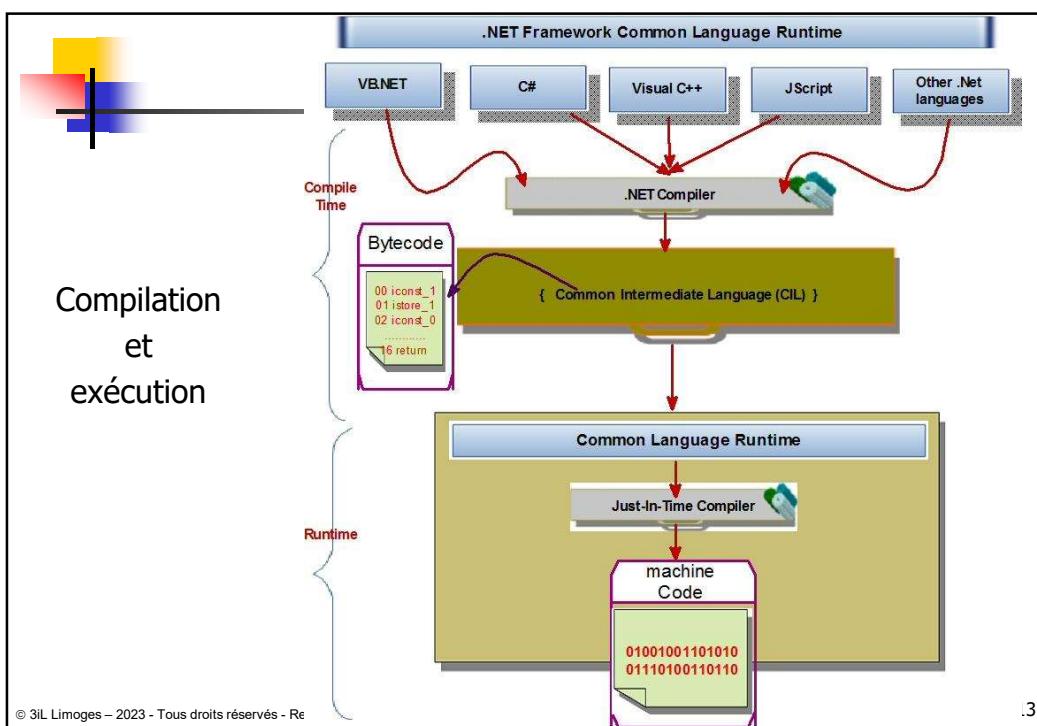
11

Compilation et exécution

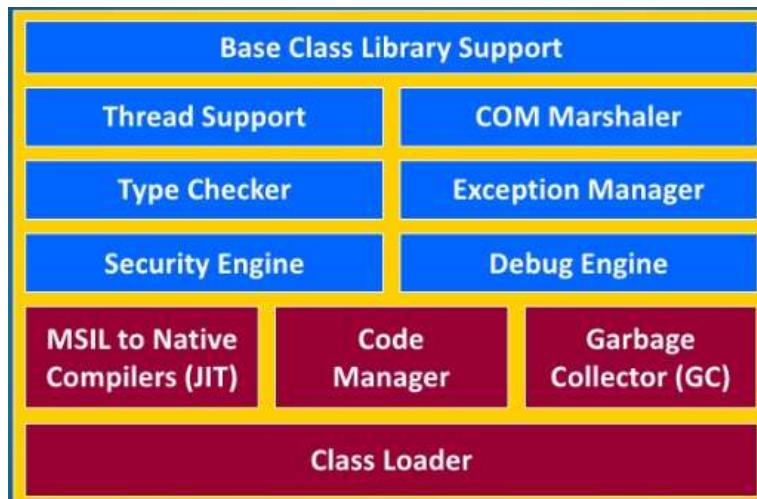


© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

12



Architecture de la CLR



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

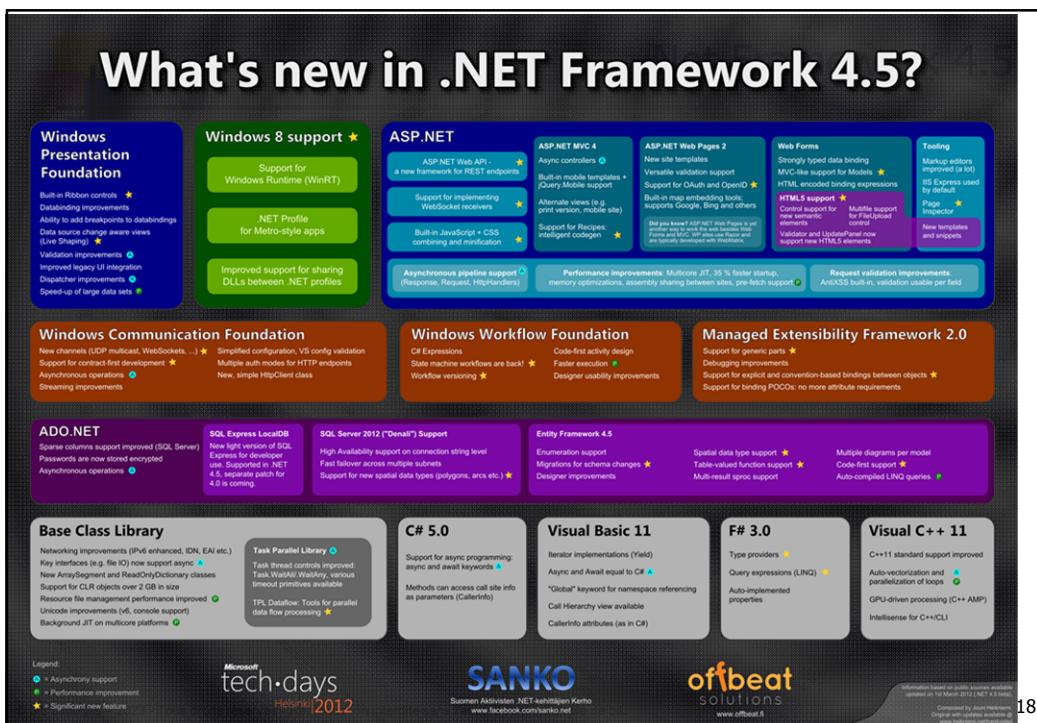
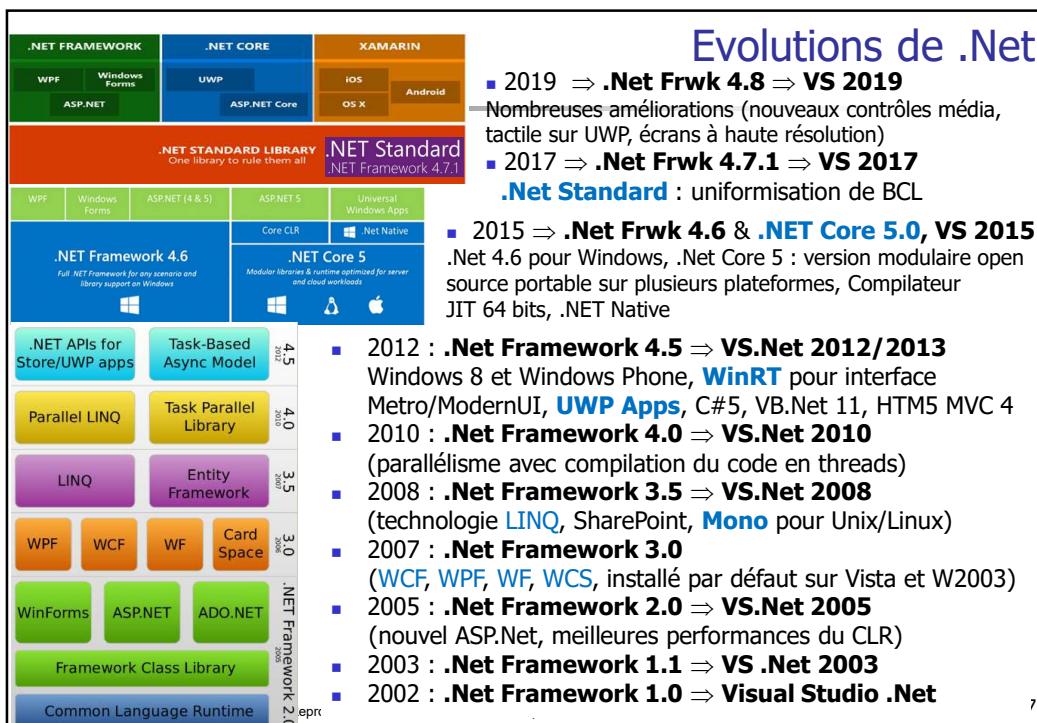
15

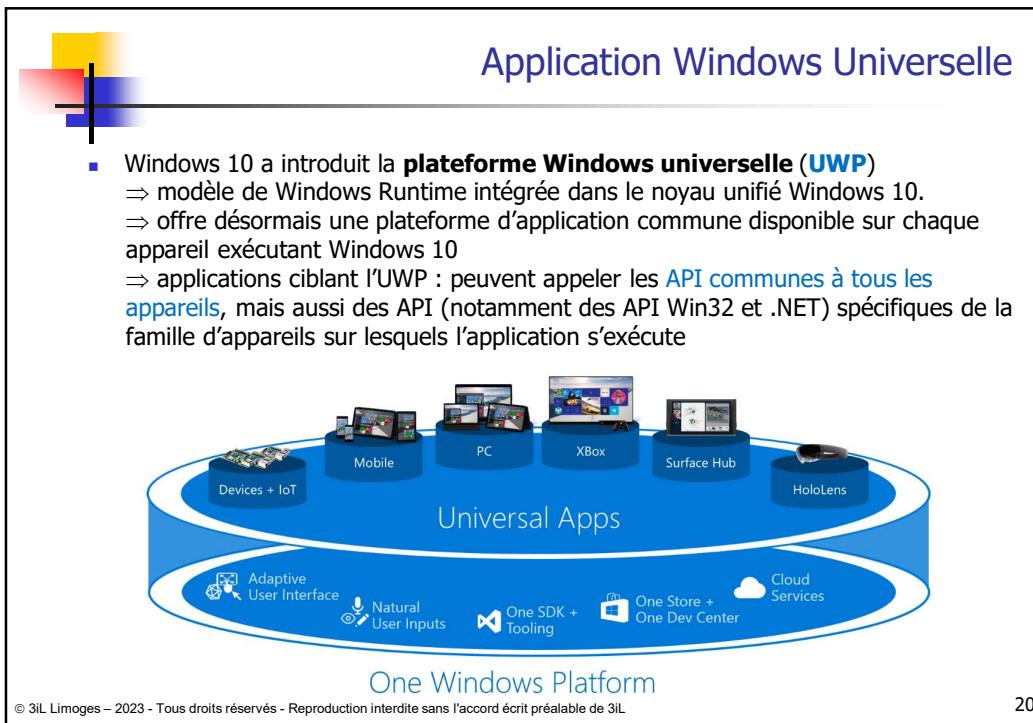
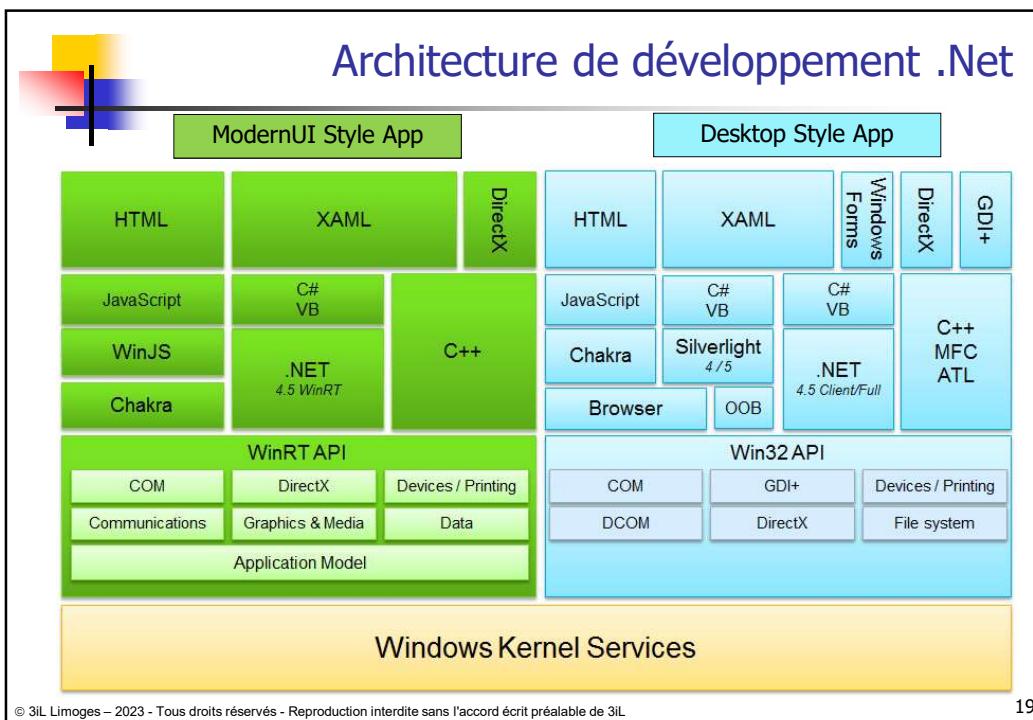
Définitions

- **CLS** (Common Language Specification) : ensemble de fonctionnalités de langage appelé **spécifications** permettant l'interaction entre objets managés
⇒ respect du CLS augmente la sécurité de votre programme
- **CTS** (Common Type System) : **ensemble de types de données communs** que la Runtime .NET comprend et que les applications .NET peuvent utiliser, afin que des classes définies dans plusieurs langages puissent communiquer
- **CLR** (Common Language Runtime ou runtime .NET) : **moteur d'exécution** du noyau (gère la sécurité d'accès de code, la gestion de la vie d'un objet, la gestion des ressources, la sûreté des types, etc ...)
- **MSIL** (Microsoft Intermediate Language) : **langage compilé** avant ou pendant l'exécution du programme par le **VES** (Virtual Execution System) qui fait partie intégrante du CLR

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

16





Runtime/Bibliothèques disponibles

⇒ .NET Framework

⇒ Windows

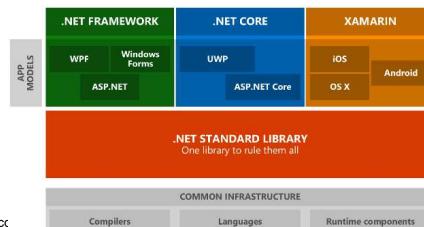
- ⇒ .NET Core multiplateforme modulaire et opensource
 - Windows, Linux, MacOS
 - UWP et ASP.Net

⇒ .Net Maui (Xamarin)

Windows, Android et iOS

.NET Framework	.NET Core	Xamarin
	  MacOS	 iOS 
Platform for .NET applications on Windows	Cross-platform and open source framework optimized for modern app needs and developer workflows	Cross-platform and open source Mono-based runtime for iOS, MacOS, Android and Windows devices
Distributed with Windows	Distributed with app	Distributed with app

.Net Standard :
uniformisation de l'API (BCL)

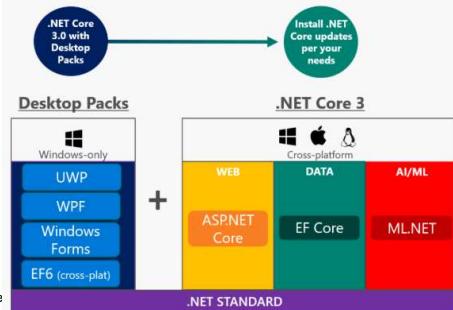


© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable

21

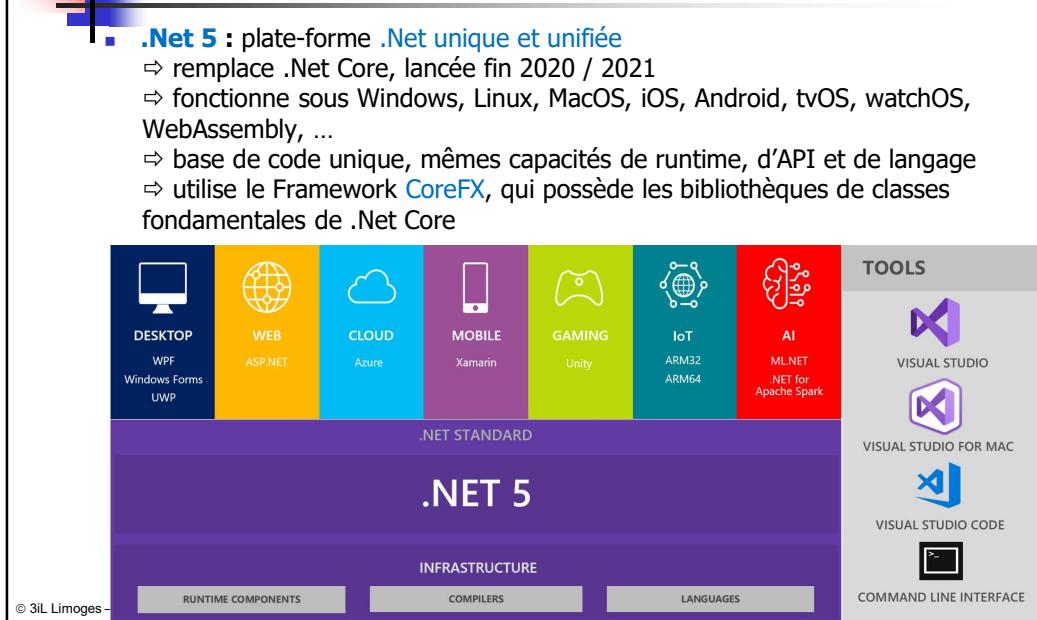
.Net Core

- **.Net Core** : publié en juin 2016, version 3, open-source, .Net 5
 - ⇒ Cross-plateforme, permet de développer pour Windows, MacOS et Linux
 - ⇒ **Runtime .NET Native** embarquée aux fichiers exécutables natifs
 - ⇒ composants modulaires (Nuget), utilisables à la demande,
 - ⇒ 2 parties : (sources disponibles sur GitHub)
 - **CoreFx** : librairie sous forme d'assemblées (dll)
 - **CoreCLR** : Runtime (Garbage collector, compilateur JIT, types de base .Net)
 - ⇒ langages C# et F# et partiellement en charge Visual Basic.Net
 - ⇒ Dev avec Visual Studio ou Visual Studio Code (Windows, Linux)
 - ⇒ SDK .Net Core



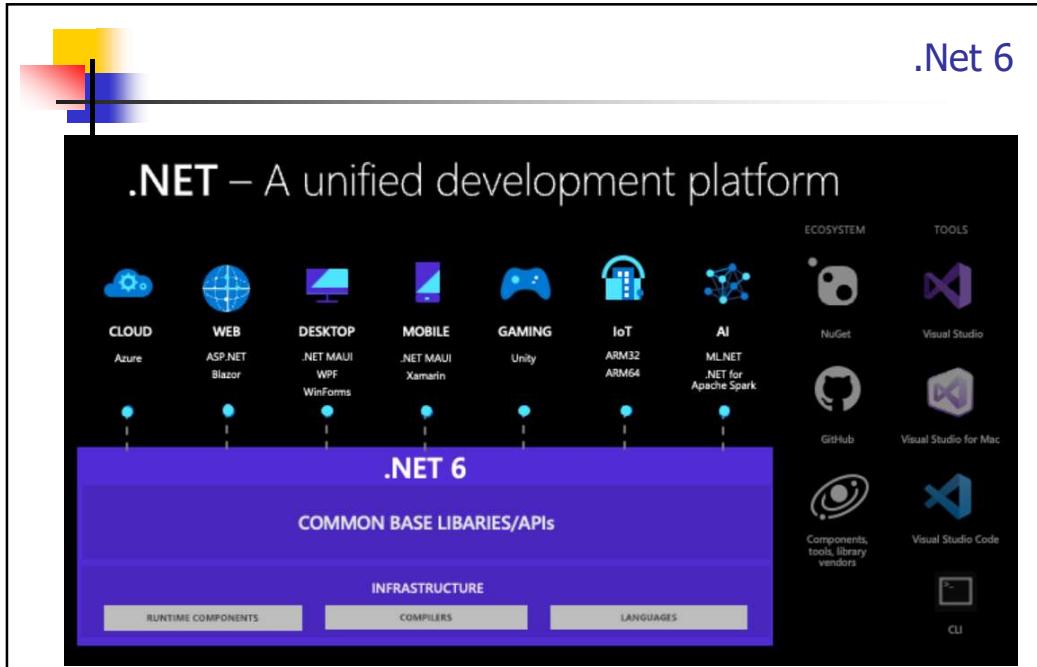
© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable

2020 : Plateforme .Net unifiée .Net 5

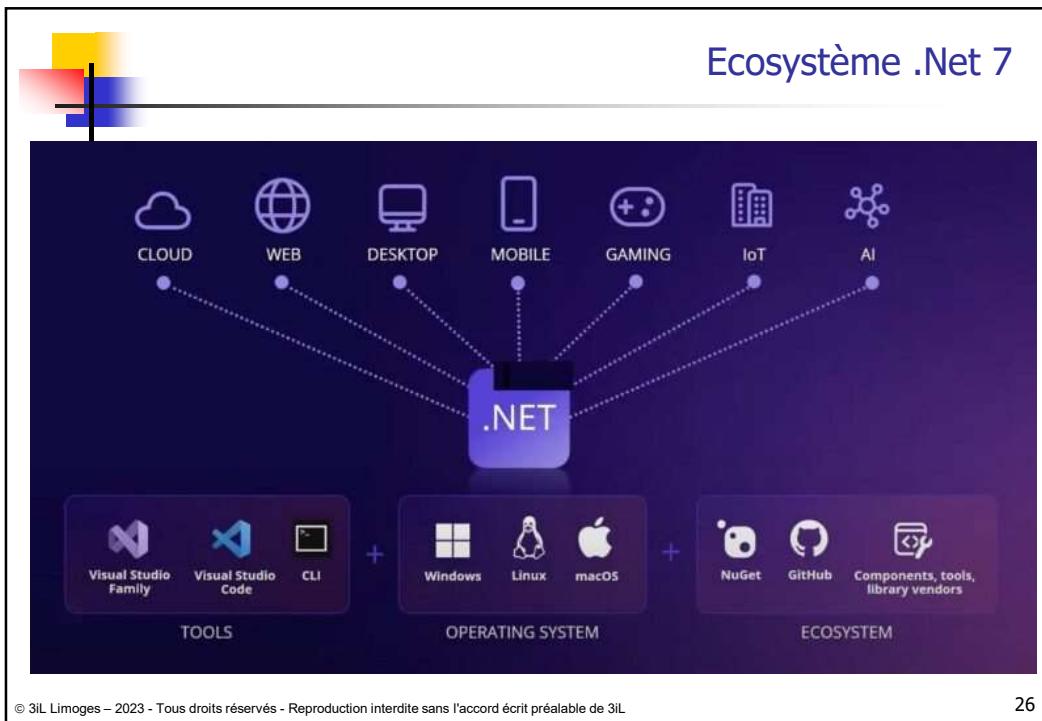
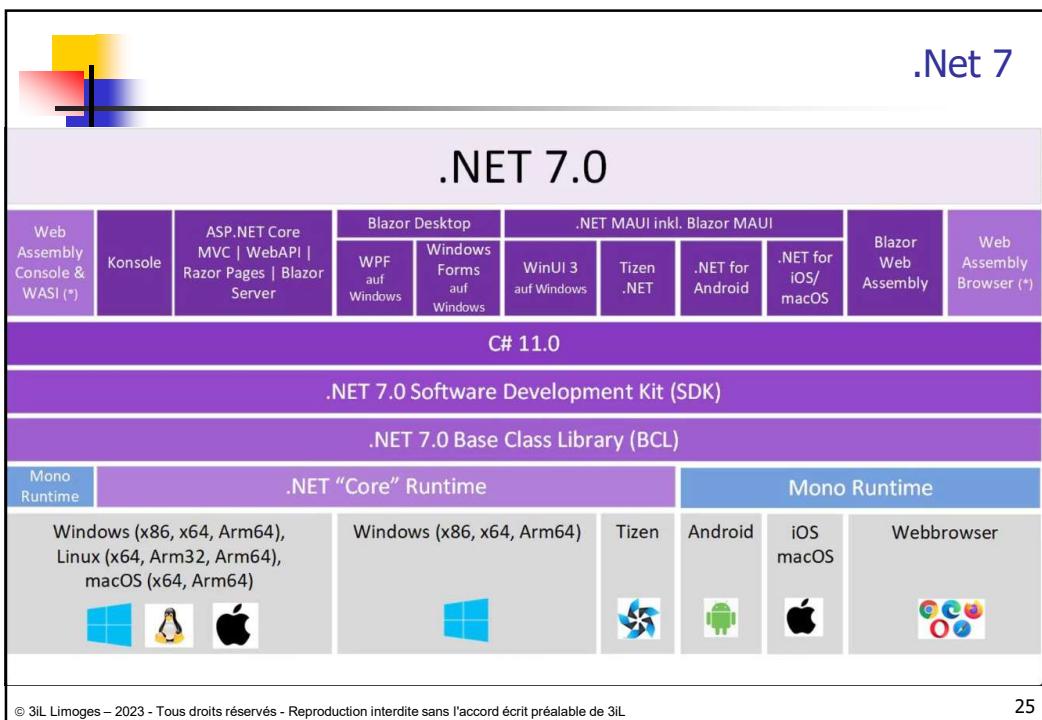


© 3iL Limoges -

.Net 6



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL



.Net Framework vs .Net Core

■ .NET Core vs .NET Framework

- ⇒ Microsoft maintient les deux moteurs d'exécution pour la création d'appli.
- ⇒ partagent un grand nombre des mêmes API (nommée .NET Standard)
- ⇒ .Net Framework (v4.8.1 09/2023) : plateforme de développement logiciels pour applications Windows, sites Web, services Web et jeux
- ⇒ .Net Core (v7 09/2023) : Framework modulaire, léger, rapide, multiplateforme et open-source, pour créer des applications multiplateforme (Windows, Linux, Mac)

Quelques différences :

- .Net Core fonctionne bien sur Docker, .Net Framework est plus lourd
- .Net Core ne dispose pas de toute les fonctionnalités de .Net : bibliothèques et extensions, certains packages NuGet non disponibles, services liés au Workflow, ASP.NET Web Forms et Web Pages non disponibles
- Entity Framework Core ≠ Entity Framework v6
- .Net Framework : service WCF, .Net Core : API REST avec ASP.NET Core MVC
- .Net Core plus éloigné du SE et n'accède pas au registre Windows ni à WMI
- ...

Autres « runtime » .Net

- **Silverlight** : machine virtuelle/plugin pour navigateur internet (noyau CLR propre), librairie légère, sous-ensemble de base du Framework .NET développement d'applications web riches dans 1 moteur de rendu vectoriel, compatibilité limitée à Windows, Visual Studio ou Expression Blend



- **Mono** : mise en œuvre open source de la machine virtuelle .NET 4.0 pour Linux, Solaris, Mac OS X, Windows et Unix (2016: v4.6), compilateur C# 3.0, machine virtuelle, de nombreuses classes de base, développé par Xamarin puis par MS, sous licence MIT (usage libre), <http://www.mono-project.com>



- **Micro Framework. NET ou .Net µFramework** : sous ensemble du Frwrk .NET destiné aux systèmes embarqués ultra légers dont les processeurs ne disposent pas de MMU et qui ne requiert pas d'OS pour fonctionner, non temps réel, boot et s'exécute en mémoire flash, plus 1,5 million d'équipements utilisent le µF (télécommandes, robots, périphériques audio / vidéo, GPS, ...)



- **Compact Framework .Net** : Framework .Net allégé (2 à ~ 10 Mo pour 3.5) destiné aux systèmes Windows Embedded CE, codes fonctionnent sur tous les processeurs embarqués WEC sans recompilation



Versions .Net Framework

- Taille du Framework .Net
 - ⇒ environ 54 Mo pour le .NET 3.0 et 197 Mo pour la version 3.5
 - ⇒ .Net 4.8 nécessite 4,5 Go
 - ⇒ version allégée .Net Client Profile et qui ne pèse que 26,5 Mo
- Systèmes 32/64 bit :
 - ⇒ Systèmes 32 bits : .Net version 32 bits natif
 - ⇒ Systèmes 64 bits : .Net Framework 64 bits natif
 - ou 32 bits natif grâce au système WOW64
- Versions installées :
 - ⇒ Seul Windows 11 et 10 (Mise à jour mai 2019) inclut le [.Net Framework 4.8](#)
 - ⇒ Windows Server 2022 inclut .Net Framework 4.8
 - ⇒ Windows 10 (Mise à jour d'avril 2018) et Windows server 2019 inclut 4.7.2

Voir : <https://docs.microsoft.com/fr-fr/dotnet/framework/migration-guide/versions-and-dependencies?redirectedfrom=MSDN>

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

29

2. - Les langages VB.Net et C#



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

30

Langages de développement pour .Net

- De nombreuses possibilités :
 - ⇒ **C#** : créé pour cette plateforme
 - ⇒ **VB.Net** : permet aux développeurs VB6 ou VB2005 de passer à .Net (VS)
 - ⇒ **C++ / CLI** : pour les développeurs C++
 - ⇒ **J#** pour les développeurs Java (dérivé non officiel de Java créé par Microsoft)
 - ⇒ **JavaScript**
 - ⇒ **C#, VB.Net, J# et C++ CLI** sont inclus dans **Visual Studio .Net** de Microsoft
 - ⇒ Existe Delphi.Net, Fortran.Net, COBOL.NET, Python.Net, ...
 - ⇒ WinDev à partir de la version 10
- ⇒ tout langage si le compilateur est capable de générer du MSIL
(code compilé lors de l'exécution)
- Interopérabilité avec l'existant
 - ⇒ Permet de continuer à utiliser les composants COM, les DLL non-managées au sein des applications .Net
 - ⇒ Possibilité d'utiliser des composants .Net dans les applications non-managées

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

31

Exemple en C++/CLI

```
Form1(void) { InitializeComponent(); ... }

void InitializeComponent(void)
{
    this->button1 = (gcnew System::Windows::Forms::Button());
    this->button1->Location = System::Drawing::Point(48, 29);
    this->button1->Name = L"button1";
    this->button1->Size = System::Drawing::Size(195, 52);
    this->button1->Click += gcnew System::EventHandler(this, &Form1::button1_Click);
}

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
    System::String ^ chaine;
    System::DateTime ^ date = System::DateTime::Now;
    chaine = "Hello World, nous sommes le : " + date->ToString();
    MessageBox::Show(chaine,"Info");
}
```



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

32

C++ managed et non managed

- Le compilateur propose différentes options de compilation (non managed, managed prioritaire, pure managed, et safe managed) (⇒ Voir Annexe)
⇒ possibilité en C++ de forcer l'état du code compilé
- Cas du **C++/CLI** (C++ managé)
 - ⇒ **classe managée** ⇒ objet pris en charge par le CLR (allocation/désallocation)

```
String ^ str = gcnew String("chaîne managée allouée sur le heap managé");  
ref class CMaClasse { CMaClasse(void); ~CMaClasse(void); }  
value class MonTypeDeValeur { String^ uneChaine; } (pas de constructeur)
```
 - ⇒ **classe non managée ou native** ⇒ non gérée par le CLR : (new et delete)

```
class CClassNoManaged {CClassNoManaged(void); ~CClassNoManaged(void);}
```
- Possibilité de forcer le compilateur à générer du langage machine pour certaines portions de code

```
#pragma unmanaged  
#include <windows.h>  
#pragma comment(lib, "User32.lib")  
class CNonManagee {public: CNonManagee(void)  
void Show() { MessageBoxW(.....); } };  
#pragma managed
```

VB.Net et C#

- **VisualBasic.Net** et **C#** (C sharp) sont les deux langages « phares » de Microsoft pour le Framework .Net
 - Ce sont des langages entièrement objets
⇒ Chaque programme est composé de plusieurs classes
 - Ils s'appuient sur le même runtime : la **CLR**
⇒ La syntaxe changera mais les classes .NET resteront les mêmes
- **C#** est apparu avec et a été développé pour le Framework .Net
La version 4 est avec le Framework 4.0, la version 5 avec le Framework 4.5
- **VB.Net** a été créé pour permettre aux développeurs VB6 de passer à .Net
Contrairement aux apparences, le passage de VB à VB.NET est difficile.
VB6 n'est pas un langage orienté objets alors que VB.NET l'est complètement
On considère que le VB.Net du framework 4.5 est la version 11

C# / VB.Net

- Les deux langages sont très proches

- ⇒ C# est plus un langage de puriste plus objet et permet du code plus poussés
- ⇒ VB.Net est un langage beaucoup plus verbeux avec un code moins aéré mais plus lisible et plus facile à maintenir
- ⇒ VB.Net est plus facile d'approche (?)

Ex : l'héritage se définit avec le mot clé « Inherits » (<< : >> en C#)

C# :

```
using System;
class HelloWorld
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

VB.Net :

```
Imports System
Module Module1
Sub Main()
    Console.WriteLine("Hello VB.NET World!")
End Sub
End Module
```

Visual-Basic .Net : Exemple

```
Private Function MakeStr(ByVal strI As String, ByVal strH As String, ByVal strTxt As String) As String
    Dim str As String
    If Len(strI) < 2 Then strI = " " & strI
    str = strI & "#" & strH & "#" & strTxt
    MakeStr = str
End Function

Private Sub MenuDel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MenuDel.Click
    If (IsNothing(TV1.SelectedNode)) Then Exit Sub
    If TV1.SelectedNode.Level = 0 Then
        MsgBox("Suppression impossible sur le 1e niveau", MsgBoxStyle.Exclamation, "Refus")
        Exit Sub
    End If
    TCat(CInt(TV1.SelectedNode.Name)) = ""
    TV1.Nodes.Remove(TV1.SelectedNode)
    'détruire les entrées du dataset et du tableau
End Sub

Private Sub MenuAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MenuAdd.Click
    Dim Name As String = "Nouveau"
    Dim NewNode As TreeNode
    If (IsNothing(TV1.SelectedNode)) Then Exit Sub
    If TV1.SelectedNode.Level = 0 Then
        MsgBox("Ajout impossible sur le 1e niveau", MsgBoxStyle.Exclamation, "Refus")
        Exit Sub
    End If
    NewNode = TV1.SelectedNode.Parent.Nodes.Add(NewKey.ToString, Name)
    NewNode.BeginEdit()
    NewKey = NewKey + 1
End Sub

Private Sub MenuAddEn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MenuAddEn.Click
```

C# : Exemple

The screenshot shows the Microsoft Visual Studio interface. On the left is the code editor for 'Form1.cs' with the title 'TP2_VS2008.Form1'. The code implements a Windows application with a database creation feature. On the right is the 'Object Browser' window, which lists various methods and components available in the current project.

```
Form1.cs [ Page de démarrage ]
```

```
TP2_VS2008.Form1
```

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlServerCe;

namespace TP2_VS2008
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private string MaBase = "\\\\My Documents\\\\BDTP.sdf";
        private void bCreerDB_Click(object sender, EventArgs e)
        {
            if (System.IO.File.Exists(MaBase) == true)
            {
                System.IO.File.Delete(MaBase);
            }
            SqlCeEngine engine =
                new SqlCeEngine("Data source = " + MaBase);
            engine.CreateDatabase();
            engine.Dispose();
            MessageBox.Show("Base créée", "Info");
        }
    }
}
```

```
Form1_Load(object sender, EventArgs e)
```

```
bCreerDB
bCreerDB_Click(object sender, EventArgs e)
bDInsert(int Id, string Nom, string Tel)
bInsert
bInsert_Click(object sender, EventArgs e)
bPull
bPull_Click(object sender, EventArgs e)
bPush
bPush_Click(object sender, EventArgs e)
bRefresh
bRefresh_Click(object sender, EventArgs e)
bTable
bTable_Click(object sender, EventArgs e)
bUpdate
bUpdate_Click(object sender, EventArgs e)
components
Dispose(bool disposing)
dSet
Form1
Form1_Load(object sender, EventArgs e)
InitializeComponent()
label1
lb
lb_SelectedIndexChanged(object sender, EventArgs e)
MaBase
mainMenu1
rdlInternetURL
rdaOleDbConnectionString
RemplirDataSet()
Tel
```

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

37

Application .Net (C#)

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

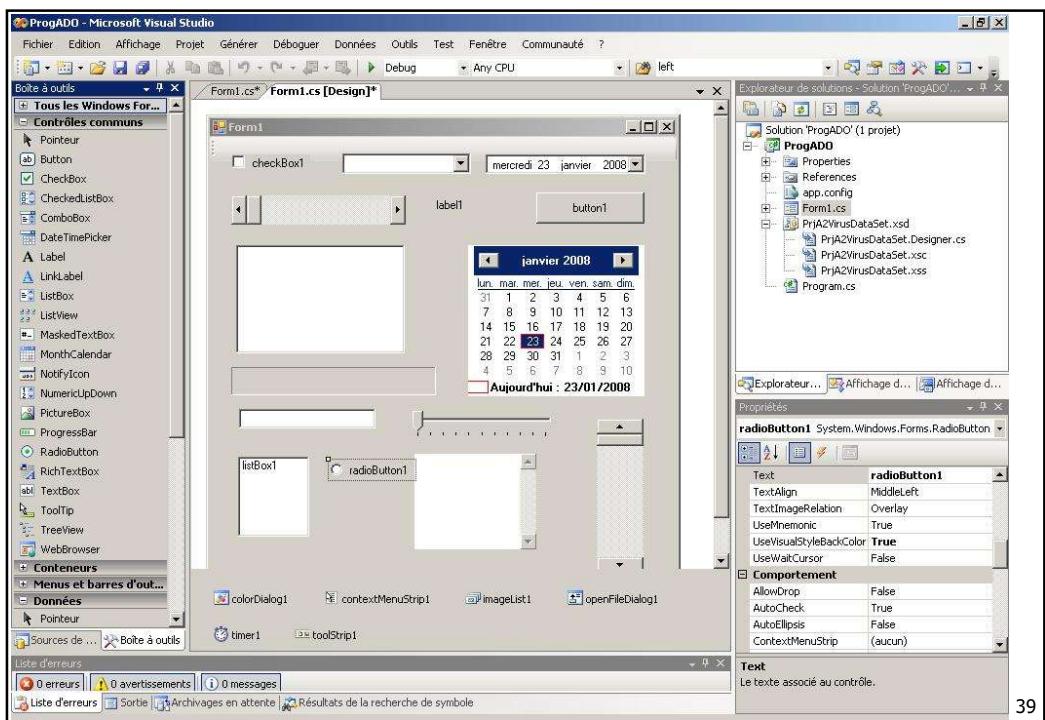
namespace WinApp
{
    public class Form1 : System.Windows.Forms.Form
    {
        .....
        public Form1()
        {
            InitializeComponent();
        }

        static void Main()
        {
            Application.Run(new Form1());
        }
    }
}
```

- Boucle des messages encapsulée dans la fonction « Run » de la classe application (Cette classe application gère le cycle de vie de l'application en fournissant les services généraux.)

- « CreateWindowEx(...) » intégrée à la classe « Form », les attributs de la fonction devenant des propriétés de la classe.

- La fonction DoEvent peut être appelée pour forcer le traitement de message en attente.



39

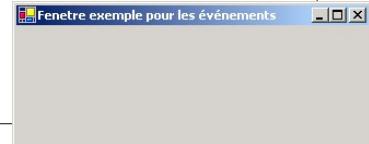
C#

- **Caractéristiques du C# :** (version 11 en 2022)
 - ⇒ entièrement orienté objet
 - ⇒ langage fortement typé (statique, fort, nominatif)
 - ⇒ dispose de types de valeurs intrinsèques définis à partir des types de base du CLS
 - ⇒ majorité des mécanismes de classes, interfaces, exceptions proviennent des fonctionnalités de la bibliothèque .NET.
 - ⇒ instructions de base assez proches syntaxiquement et sémantiquement à celles de Java et de C++ (beaucoup plus typé)
- **Le C# est le langage de programmation qui exploite le mieux l'architecture et les fonctionnalités Microsoft .NET**
 - ⇒ types natifs correspondent à ceux de .NET
 - ⇒ classes, interfaces, délégués, exceptions, typage dynamique des variables, délégués, classes partielles, surcharge d'opérateurs
 - ⇒ mots-clés select, from et where (C# 3.0)
 - ⇒ nettoyage automatique de la mémoire par un ramasse-miettes

C#11 .NET 7

C# : Exemple avec WinForm

```
namespace WindowsApplication1
{
    public class Form1 : System.Windows.Forms.Form
    {
        private System.ComponentModel.Container components = null;
        public Form1()
        {
            InitializeComponent();
        }
        #region
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.Size = new System.Drawing.Size(300,300);
            this.Name = "FormExemple";
            this.Text = "Fenetre exemple pour les événements";
        }
        #endregion
        [STAThread]
        static void Main()
        {
            Application.Run(new Form1());
        }
    }
}
```



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

41

Exemple C# en mode console

```
using System; // imports
public class console1 {
    public static void Main(String[] args) {
        System.Console.Write("Nom : "); // On demande le nom
        String nom=System.Console.ReadLine(); // lecture réponse
        int age=0;
        Boolean ageOK=false;
        while ( ! ageOK){
            Console.Out.Write("âge : "); // question
            try{
                age=int.Parse(System.Console.ReadLine());
                ageOK=true;
            }
            catch {
                Console.Error.WriteLine("Age incorrect, recommencez..."); //try-catch
            }
        }
        Console.Out.WriteLine("Vous vous appelez " + nom + " et vous avez " + age + " ans");
        Console.ReadLine();
    }
}
```

```
E:\data\serge\MSNET\c#\bases\1>console1
Nom : dupont
âge : xx
Age incorrect, recommencez...
âge : 12
Vous vous appelez dupont et vous avez 12 ans
```

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

42

```

// espaces de noms importés
using System;

// la classe de test
public class conv1{
    public static void Main(){
        String S;
        const int i=10;
        const long l=100000;
        const float f=45.78F;
        double d=-14.98;

        // nombre --> chaîne
        S="" + i;
        affiche(S);
        S="" + l;
        affiche(S);
        S="" + f;
        affiche(S);
        S="" + d;
        affiche(S);

        //boolean --> chaîne
        const bool b=false;
        S="" + b;
        affiche(S);

        // chaîne --> int
        int i1;
        i1=int.Parse("10");
        affiche(""+i1);
        try{
            i1=int.Parse("10.67");
            affiche(""+i1);
        } catch (Exception e){
            affiche("Erreur "+e.Message);
        }

        // chaîne --> long
        long l1;
        l1=long.Parse("100");
        affiche(""+l1);
        try{
            l1=long.Parse("10.675");
            affiche(""+l1);
        } catch (Exception e){
            affiche("Erreur "+e.Message);
        }

        // chaîne --> float
        float f1;
        f1=float.Parse("100,87");
        affiche(""+f1);
        try{
            f1=float.Parse("abcd");
            affiche(""+f1);
        } catch (Exception e){
            affiche("Erreur "+e.Message);
        }
    }
}

// fin main
public static void affiche(String S){
    Console.Out.WriteLine("S=" + S);
}

// fin classe

```

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

C# : Exemple

43

```

// lecture d'un fichier texte

using System;
using System.IO;

public class listes1{
    public static void Main(string[] args){
        string ligne=null;
        StreamReader fluxInfos=null;
        // lecture contenu du fichier
        try{
            fluxInfos=new StreamReader("infos.txt");
            ligne=fluxInfos.ReadLine();
            while (ligne != null){
                System.Console.Out.WriteLine(ligne);
                ligne=fluxInfos.ReadLine();
            }
        }catch (Exception e) {
            System.Console.Error.WriteLine("L'erreur suivante s'est produite : " + e);
        }finally{
            try{
                fluxInfos.Close();
            }catch {}
            // attente
            Console.ReadLine();
        }///try-catch
    }///main
}

switch(expression) {
    case v1:
        actions1;
        break;
    case v2:
        actions2;
        break;
    . . . . .
    default:
        actions_salon;
        break;
}

public static void Main(){
    int n=5;
    if(n>1)
        if(n>6)
            Console.Out.WriteLine(">6");
        else Console.Out.WriteLine ("<=6");
}

```

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

C# : Exemple

44

C# : Classes

```

using System;

public class personne{
    // attributs
    private string prenom;
    private string nom;
    private int age;

    public class C1{
        type1 p1; // propriété p1
        type2 p2; // propriété p2
        ...
        type3 m3(...); // méthode m3
        ...
        type4 m4(...); // méthode m4
        ...
    }
}

// méthode
public void initialise(string P, string N, int age){
    this.prenom=P;
    this.nom=N;
    this.age=age;
}

// méthode
public void identifie(){
    Console.Out.WriteLine(prenom+","+nom+","+age);
}

public class test1{
    public static void Main(){
        personne p1=new personne();
        p1.initialise("Jean","Dupont",30);
        p1.identifie();
    }
}

```

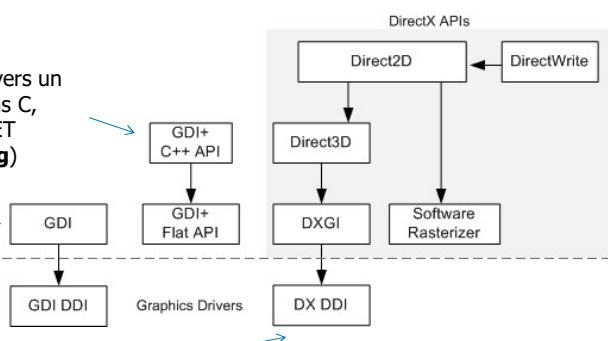
© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

45

Librairies Graphiques Windows

2 ⇒ GDI+ : successeur de GDI introduit dans Windows XP librairie accessible à travers un jeu de classe C++ qui « wrap » les fonctions C, présente une version managée dans le .NET Framework (namespace **System.Drawing**)

1 ⇒ Graphics Device Interface (GDI) : API graphique originale de Windows (32 et 64 bits) utilisant le processeur (La majorité des prog. avant 2009 utilisent GDI)



3 ⇒ WPF/Aero/DirectX :

- **Aero** : moteur graphique de Windows (7/Vista/8/10) qui utilise la carte graphique pour améliorer la vitesse et les fonctions de l'UI, passe par **DirectX**, (carte compatible DirectX est obligatoire).
- **WPF** : nouvelle API de programmation d'Interface Windows
- **Direct2D** : nouvel API 2-D graphique, succède à GDI/GDI+, bâtie sur Direct3D
- **DirectWrite** : API permettant d'afficher du texte avec un haut niveau de qualité

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

46

C# : exemple avec GDI+

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace TestCSharp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void bTest_Click(object sender, EventArgs e)
        {
            Graphics g = CreateGraphics();
            Pen blackPen = new Pen(Color.Black, 3);
            Rectangle rect = new Rectangle(0, 0, 200, 100);
            g.DrawEllipse(blackPen, rect);
            g.DrawString("Hello 3iL", new Font("Arial Black", 40), new SolidBrush(Color.Red), new Point(10,10));
            g.Dispose();
        }
    }
}
```



Utilisation de System.Drawing

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

47

C# : exemple avec GDI+

- Affichage avec un objet *Graphics* (ex contexte de périphérique)

```
private void bTest_Click(object sender, EventArgs e)
{
    Graphics g = CreateGraphics();
    Image newImage = Image.FromFile("Logo3iL.gif");
    g.DrawImage(newImage, new Point(10, 10));
    g.Dispose();
}
```



- Affichage permanent (*Form1_Paint* équivalent à *WM_PAINT*)

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Image newImage = Image.FromFile("Logo3iL.gif");
    e.Graphics.DrawImage(newImage, new Point(10, 10));
}
```



Se déclenche par la méthode *Invalidate* (...)

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

48

Exemple de classe : Classe TcpClient

- Espace de noms : **System.Net.Sockets.TcpClient**
Fournit des connexions client pour des services réseau TCP.
- Présente des méthodes simples de connexion, d'envoi et de réception de flux de données sur un réseau en mode blocage synchrone.
- Pour se connecter et échanger des données :
 - Créer **TcpClient** et appeler une des trois méthodes **Connect** disponibles.
 - Créer **TcpClient** à l'aide du nom d'hôte et du numéro de port de l'hôte distant. Ce constructeur tentera automatiquement d'établir une connexion.
- Pour envoyer et recevoir des données :
 - méthode **GetStream** pour obtenir un objet **NetworkStream**
 - Pour envoyer et recevoir des données avec l'hôte distant :
 - méthodes **Write** et **Read** de l'objet **NetworkStream**
 - Pour libérer toutes les ressources associées à **TcpClient** :
 - méthode **Close**

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

49

C# : Exemple classe TcpClient

```
static void Connect(string server, string message)
{
    try
    {
        Int32 port = 13000;
        TcpClient client = new TcpClient(server, port);
        Byte[] data = System.Text.Encoding.ASCII.GetBytes(message);
        NetworkStream stream = client.GetStream();
        stream.Write(data, 0, data.Length);
        Console.WriteLine("Sent: {0}", message);
        data = new Byte[256];
        String responseData = String.Empty;
        Int32 bytes = stream.Read(data, 0, data.Length);
        responseData = System.Text.Encoding.ASCII.GetString(data, 0, bytes);
        Console.WriteLine("Received: {0}", responseData);
        client.Close();
    }
    catch (ArgumentNullException e)
    {
        Console.WriteLine("ArgumentNullException: {0}", e);
    }
    catch (SocketException e)
    {
        Console.WriteLine("SocketException: {0}", e);
    }
}
```

© 3iL Limoges - 25

50

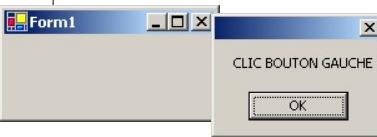
Application .Net : surcharge de WndProc

```
public class Win32
{
    [DllImport("user32.dll")]
    public static extern int PostQuitMessage(int ret);
}

public class Form1 : System.Windows.Forms.Form
{
    private const int WM_LBUTTONDOWN = 0x0202;
    protected override void WndProc(ref Message m)
    {
        switch (m.Msg)
        {
            case WM_LBUTTONDOWN:
                MessageBox.Show("CLIC BOUTON GAUCHE");
                Win32.PostQuitMessage(0);
                break;
        }
        base.WndProc(ref m);
    }
    .....
}
```

Ex :

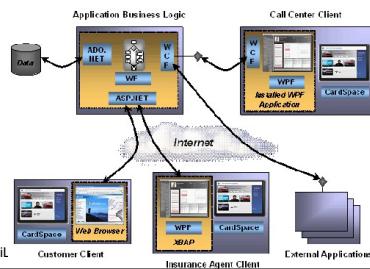
- Classe « Win32 » fait appel directement à la fonction de l'API « PostQuitMessage(...) »
- Classe « Form1 » gère la fenêtre de l'application
- « WndProc » est surchargée (override) dans la classe « Form1 » (WndProc de Form1 est rappelée ensuite)
- m est une instance de structure de System.Structure.Form.Message



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

51

3. - Les classes et les APIs .Net



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

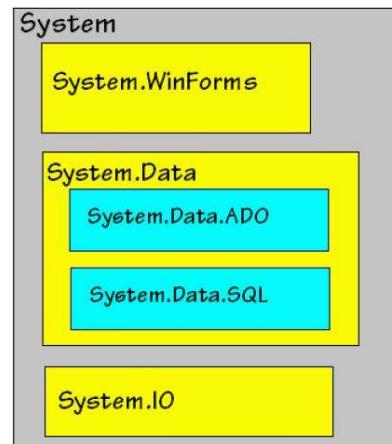
52

Hiérarchie de classes

- La librairie de classes de **.NET Framework** est organisée en noms d'espace hiérarchisés

⇒ Exemple :

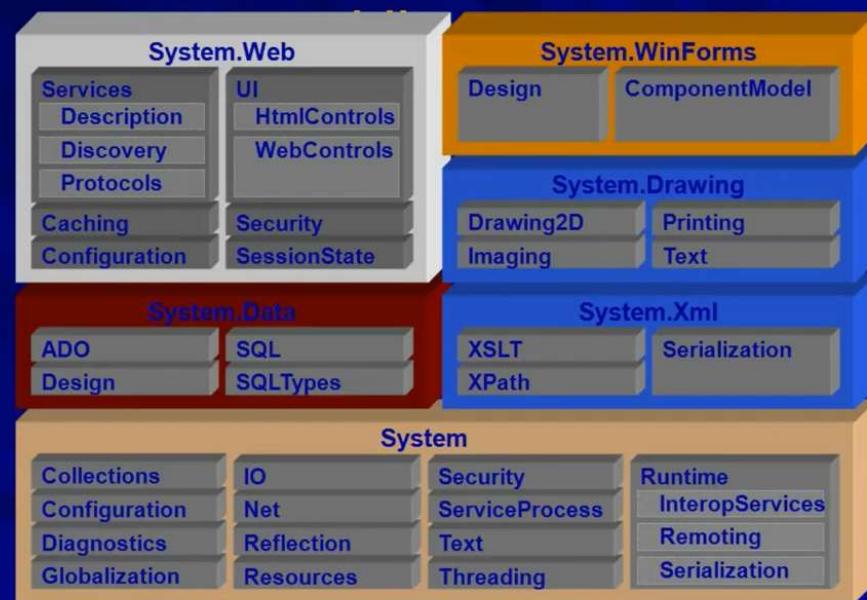
La classe **DataSet** se trouve dans l'espace de noms **"System.Data.ADO"** se déclare comme **"System.Data.ADO.Dataset"**.



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

53

The .NET Framework Class



© 3iL L

54

Les espaces de noms sous .NET

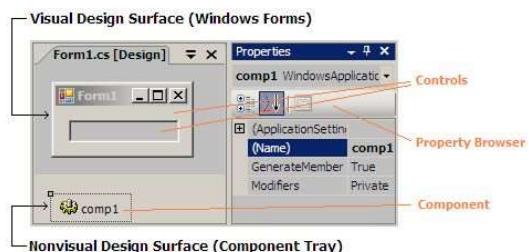
Exemples d'espaces de noms sous .NET

- ⇒ **System.Windows.Forms** : espace de nom regroupant les classes permettant la réalisations d'applications Windows (System.Windows.Forms.dll)
- ⇒ **System.ComponentModel** : espace de nom contenant les classes et interfaces des composants (code composant et contrôle)
- ⇒ **System.Data** : comprend les classes qui constituent l'architecture ADO.NET (méthode d'accès aux données pour les applications managées)
- ⇒ **System.Drawing** : fournit l'accès aux fonctionnalités graphiques de base GDI+ (fonctionnalités plus avancées dans System.Drawing.Drawing2D, System.Drawing.Imaging et System.Drawing.Text)
- ⇒ **System.ServiceProcess** : fournit des classes qui permettent d'installer et d'exécuter des fichiers exécutables sans interface utilisateur
- ⇒ **System.Collections** : contient des interfaces et des classes qui définissent différentes collections d'objets (listes, files d'attente, tableaux, ...)

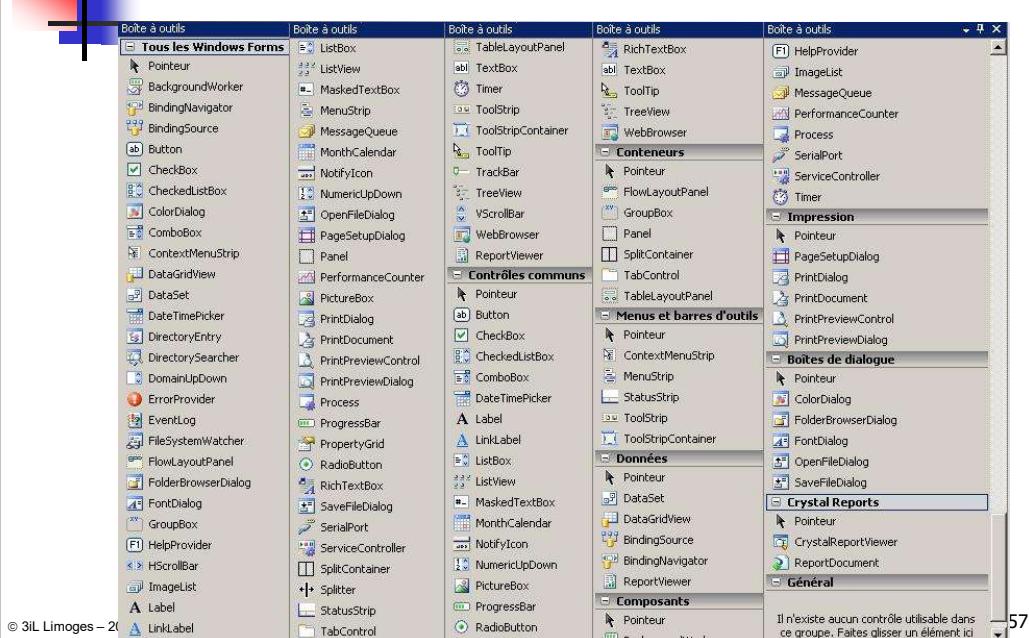
Les Windows Forms (ou Winforms)

■ **Les Windows Forms (ou Winforms) :**

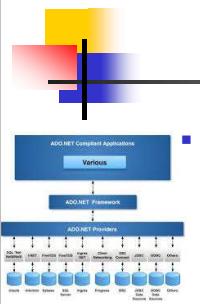
- ⇒ nom donné à la partie du framework .net responsable de l'interface graphique (GUI), donnant accès aux contrôles windows managés
- ⇒ basées sur **GDI+**, un ensemble de classes et de fonctionnalités qui permettent de créer et de gérer les IHMs
- ⇒ classes disponibles pour utiliser les Windows Forms se trouvent dans le CLR (Common Language Runtime)
 - ⇒ classe fondamentale est System::Windows::Forms::Form.
 - ⇒ héritage ⇒ construction d'une hiérarchie de classes dérivées de Winforms
- ⇒ Très simple à prendre en main, grâce à l'outil de design (similaires aux Forms VB6 mais avec une orientation très orienté objet)
- ⇒ Code plus simple à faire et plus propre qu'avec les MFC, orienté objet avec un très haut niveau d'abstraction



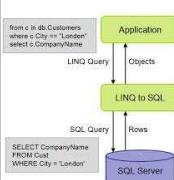
Les composants WinForm disponibles



Accès aux bases de données

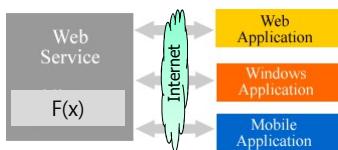


- **ADO.NET** (ADO pour Activex DataBase Object) :
 - ⇒ couche d'accès aux DB fournie par la plate-forme .NET, tous les objets sont gérés par la CLR, entièrement conçu avec XML
 - ⇒ fonctionne sur le principe de fournisseurs managés ou codés
 - ⇒ mode connecté ou mode déconnecté (objet **DataSet** ⇒ données en mémoire, connexion libérée, indépendant d'un fournisseur de données)
 - **LINQ** (Language Integrated Query) :
 - ⇒ composant du Framework qui ajoute des capacités d'interrogation sur des données aux langages .NET en utilisant une yntaxe proche de celle de SQL
 - ⇒ opérateurs de requêtes pour filtrer et projeter des données dans des collections, classes énumérables, structures XML, SGBD, sources tierces
 - ⇒ résultat renvoyé sous forme d'une collection d'objets en mémoire
 - **Entity Framework** :
 - ⇒ couche mappeur objet-relationnel (ORM)
 - ⇒ niveau supérieur d'abstraction avec moins de code que dans les applications traditionnelles



Services Web ou WebServices

- **Services Web dis « WebServices » :**
 - ⇒ Services applicatifs interrogeable via Internet sur le port 80 (http)
 - ⇒ exécute des **WebMethod** (= méthodes d'un WebService) demandées par le client (appli, page web, autre services) puis renvoie les réponses sous forme de fichiers XML ou Json
 - ⇒ formats supportés : SOAP, WCF, REST
 - ⇒ mise en place d'architectures distribuées, d'accéder à Azure
- L'utilisation des WebServices avec les bases de données permet de :
 - ⇒ travailler en mode déconnecté : les WS ne renvoient que des **DataSet** qui sont des « copies locales » d'une partie de la base de données
 - ⇒ accéder aux données à partir d'Internet (fonctionne avec le protocole http)
 - ⇒ accéder à de multiples sources de donnée de différents SGBD
 - ⇒ de réutiliser des Web Service pour d'autres applications



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

59

Les nouvelles APIs du .Net 3.0

Nouvelles APIs dans le .Net 3.0 (livrée par défaut sur Vista et Seven)

- **WPF** : Windows Presentation Foundation
nouvelle couche de présentation gérant les animations, le rendu vectoriel et la 3D doit remplacer WindowsForms pour la création d'application
- **WCS** : Windows CardSpace
prend en charge les authentications par carte d'identité numérique sur Internet
- **WCF** : Windows Communication Foundation
nouvelle couche de communication simplifiée et unifiée
- **WF** : Windows Workflow Foundation
comporte les outils et un moteur de gestion de workflows



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

60

WPF et XAML

- **Windows Presentation Foundation (WPF)** : nouveau système d'affichage graphique intégré à .Net 3.0 (WinFX) avec API entièrement managées pour IHM
 - ⇒ **surcouche logicielle à DirectX**, entièrement vectorielle (dessin et texte)
 - ⇒ approche unifiée des GUI, des documents et des animations : **Framework de présentation unifié**, qui intègre et gère toutes les parties de l'interface utilisateur (animations, images, contrôles, etc..).
 - ⇒ moteur de composition basé sur des vecteurs intégrés (Runtime) qui expédie les requêtes de rendu au bon composant logiciel ou matériel (APIs Windows, carte vidéo compatible DirectX) ⇒ utilisation Direct3D pour l'affichage vectoriel
 - ⇒ Un modèle de programmation déclarative : **langage XAML** ⇒ séparation distincte entre le designer et le développeur

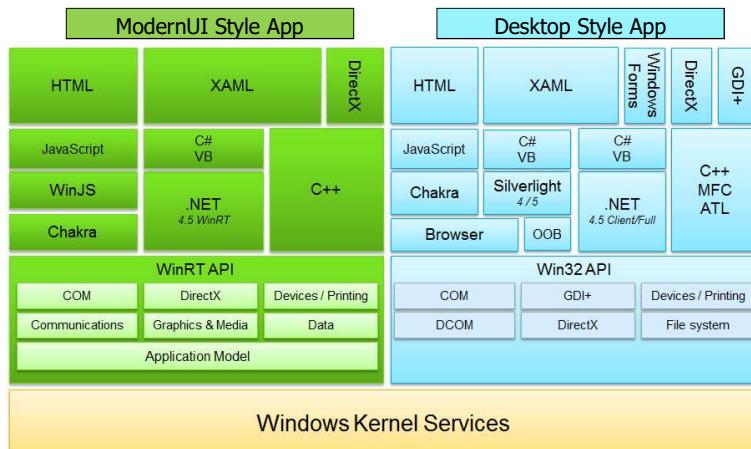


© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

61

Architecture de dév .Net 2015

- ⇒ Permet d'écrire les applications en XAML avec C#-VB, ou HTML/CSS avec JS
- ⇒ HTML 5 devient natif dans Metro
- ⇒ Permet d'écrire des composants universels instanciable dans n'importe quel langage



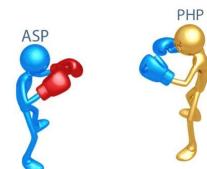
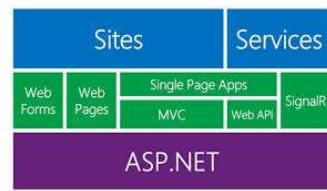
© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

62

ASP.NET

ASP.NET

- ⇒ technologie Microsoft dédiée à la création d'**applications Web dynamiques** intégrée et basé sur le **Framework .NET**
- ⇒ ensemble de fonctionnalités dédiées à la création et à la gestion de sites Web : Webs dynamiques, des applications Web ou des Web Services
- ⇒ nécessite un serveur Web **Microsoft-IIS** (ou encore apache/mono/linux) technologie typique : serveur Web **IIS** sur **Windows Server** avec **Sql-Server**
- ⇒ s'utilise aussi sur le Cloud Windows Azure
- ⇒ ASP.NET est à IIS ce que PHP est à apache ⇒ moyen de coder la partie logicielle
- ⇒ Exemples : Microsoft Sharepoint (Windows SharePoint Services ou WSS, Microsoft Office SharePoint Server ou MOSS), et certains CMS (DotNetNuke, Umbraco, ...)



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

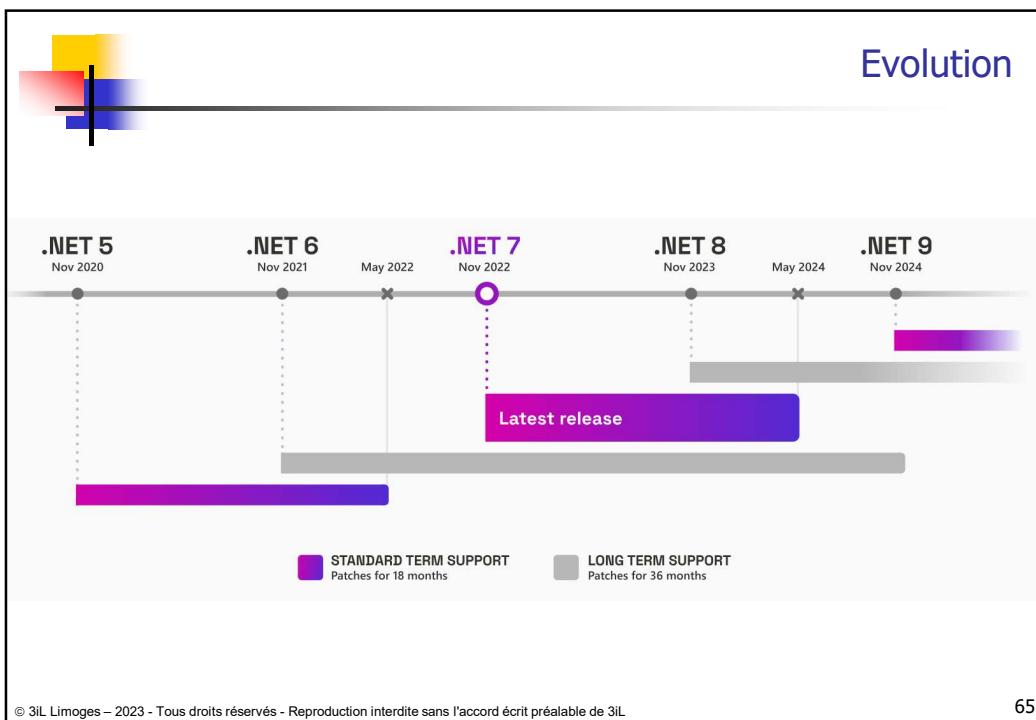
63

UWP

- **Universal Windows Platform (UWP)** : plateforme d'application universelle Windows
 - ⇒ architecture homogène pour développer des applications universelles qui fonctionnent sur **plusieurs types de périphériques** (Windows 10, Windows 10 Mobile, Xbox One et Hololens) avec le même code source pour chacun de ces systèmes
 - ⇒ prend en charge C++, C#, VB.NET et XAML, F# et JavaScript
 - ⇒ extension de la plate-forme Windows Runtime introduite pour la première fois dans Windows Server 2012 et Windows 8

The diagram illustrates the Universal Windows Platform (UWP) architecture. It shows the UWP runtime layer at the top, which integrates with various development frameworks and operating systems. Key components include:

- Universal Windows Platform (UWP):** The core runtime layer.
- Windows 10 operating system:** The underlying OS.
- Bridging technologies:** Includes Hosted Web Apps (HTML5/Javascript), Obj-C, and iOS.
- Development frameworks:** WPF, WinForms, MFC, XAML, DirectX, HTML, C/C++, and JavaScript.
- .NET Framework and Win32 API:** Integrated into the UWP runtime.
- Visual Studio IDE:** Shows the project structure for a HelloUWP app, including App.xaml, MainPage.xaml, and project.json files.
- Solution Explorer:** Shows the project files and dependencies.
- Taskbar:** Displays icons for the application and other running tasks.



Visual Studio .Net

- **Visual Studio .Net** (versions majeures : 2003, 2005, 2008, 2010, 2015, 2019, 2022)
 - ⇒ Solution complète : IDE & compilateurs et éditeurs de liens
 - ⇒ supporte la dernière version du .Net Framework (4.8.1) et de .Net Core (7)
 - ⇒ supporte un grand nombre de technologies de développement
(C#, VB.Net, C++, C-API, MFC, F#, ASP.Net, WPF, WCF, Service Web, Site ASP.Net, Service Système, projet Office ou Sharepoint, Silverlight, Azure, ...)
 - ⇒ prise en charge de Windows 10/11
 - ⇒ nombreux SDK et Framework additionnels disponibles (.Net Maui, Nuget, ...)
 - ⇒ intégration au Cloud
 - ⇒ look basé désormais sur la spécification graphique WPF
 - ⇒ gestion du multicore, compilation 32 et 64 bits, mode debugage
 - ⇒ possibilité de recompiler les librairies externes (Qt pour VS, ...)
 - ⇒ existe en plusieurs versions (professionnelles ou Express gratuite)
 - ⇒ déploiement d'application automatisé, Cloud, Conteneurs, ...

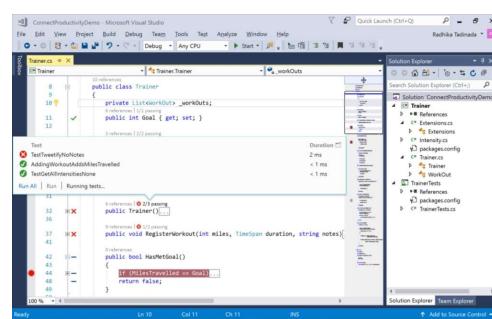
© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

67

Visual Studio 2022

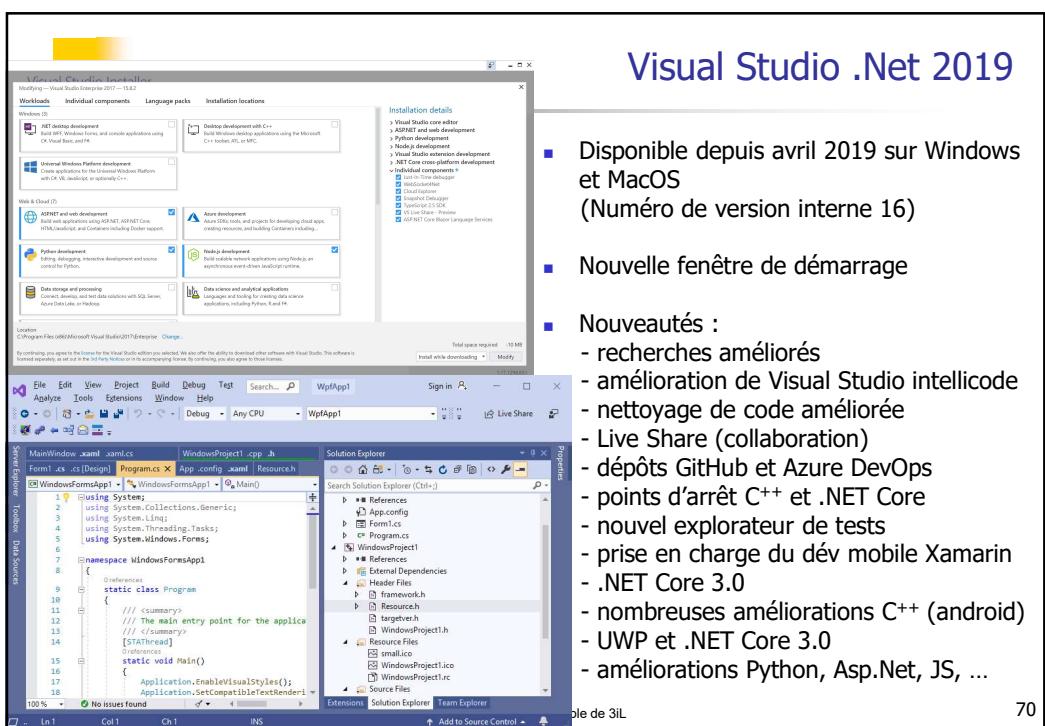
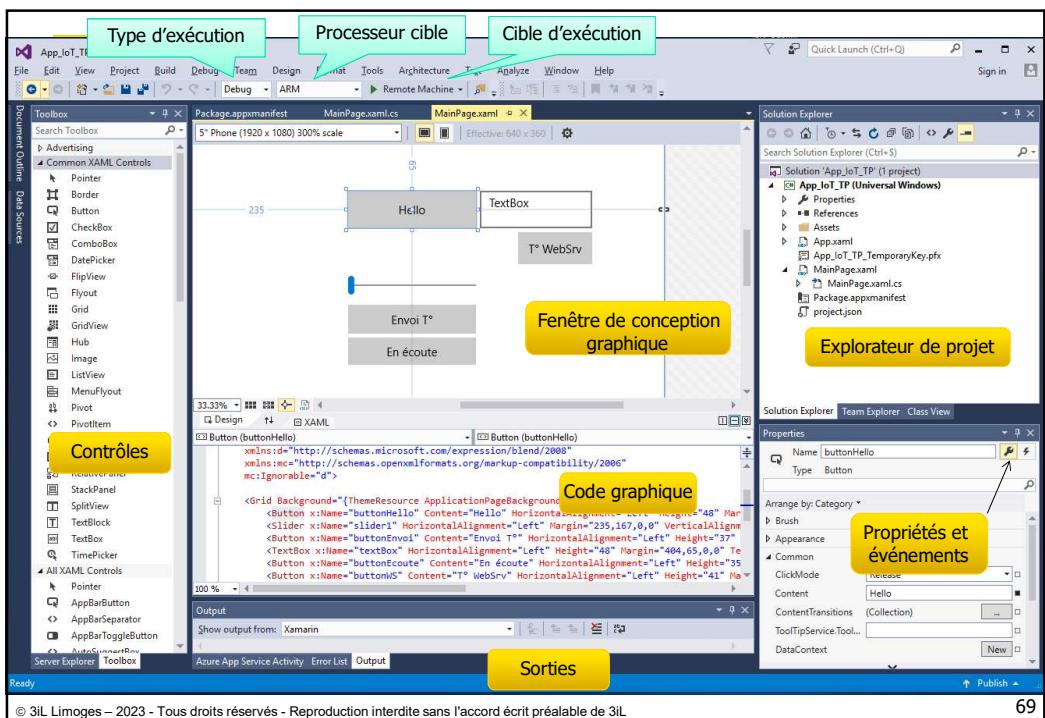
■ Visual Studio .Net 2022

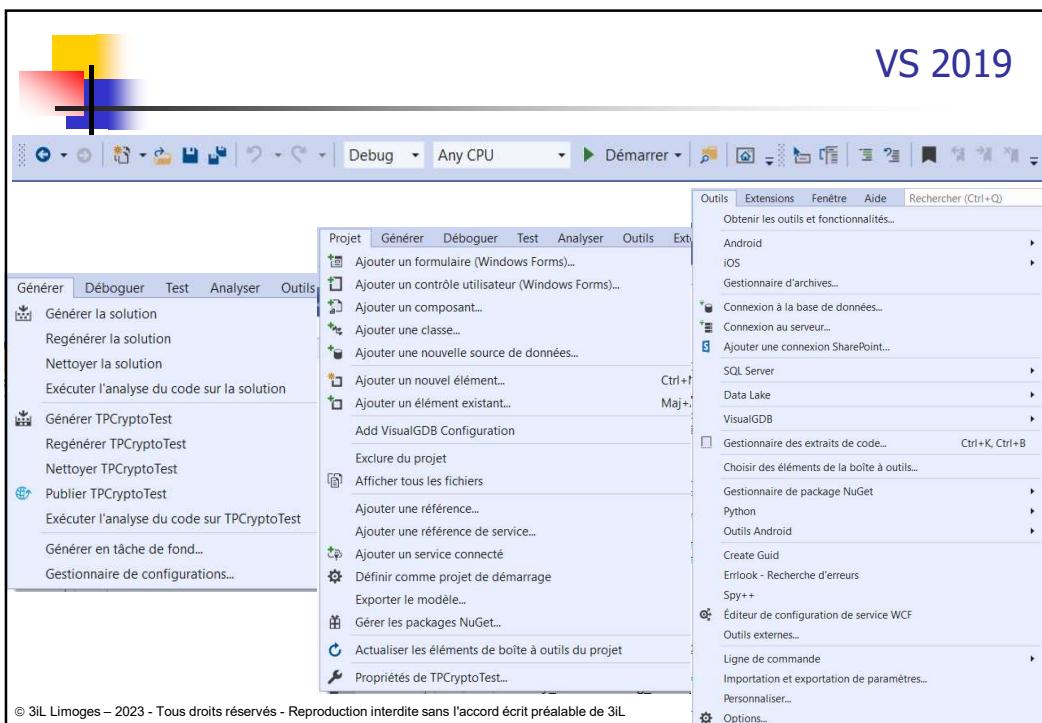
- ⇒ Solution complète : IDE & compilateurs, éditeurs de liens, SDK, ...
- ⇒ Développement .Net Framework, .Net Core, .Net Maui, Azure, Sharepoint, ASP.Net, UWP, ...
- ⇒ compatibles Windows 10/11
- ⇒ IntelliCode (complétion de code automatique),
CodeLens (recherche d'informations importantes),
Live Share (collaboration en temps réel),
Débogage intégré, Intégration au cloud,
Tests, ...



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

68





Visual Studio Code

- **Visual Studio Code**
 - ⇒ éditeur de code extensible développé par MS, cross-platform, pour Windows, Linux et macOS
 - ⇒ open source et gratuit (code source sous licence libre MIT)
 - ⇒ supporte plusieurs dizaines de langages (C, C++, C#, Java, JS, CSS, XML, Python, Perl, PHP, Swit, Sql, ..)

The screenshot shows the Visual Studio Code interface. The title bar says "Visual Studio Code". The left sidebar has "DEBUG .NET Core 1", "VARIABLES", "WATCH", "CALL STACK PAUSED ON BREAKPOINT", "External Code Unknown Source", and "BREAKPOINTS". The main editor shows a C# file with the following code:

```

1 using System;
2
3 namespace HelloWorld
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             Console.WriteLine("Hello World!");
10        }
11    }
12}

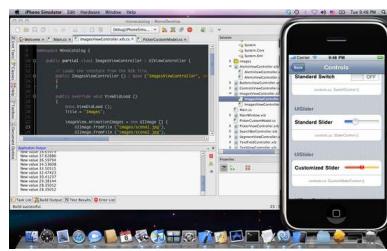
```

The bottom status bar shows "In 0 Col 87 (4 selected) Space 4 - 1118 with ROM CR LF CR". The right sidebar shows "EXTENSIONS" with several installed extensions: "@popular", "C# for Visual Studio Code", "Python", "Linting, Debugging", "Debugger for Chrome", "C/C++", "Go", and "ESLint". The bottom status bar also shows "File Edit View Goto Help".

MonoDevelop

- **MonoDevelop** : IDE Cross Platform pour C#, F#, VB.Net (Windows, Linux, Mac)
 - ⇒ API compatibles avec le Framework .Net (Mono), applications natives
 - ⇒ serveur web léger entièrement compatible avec ASP.NET, GTK# Visual Designer

<http://www.monodevelop.com/>



```
Xwt.Gtk - Xwt.GtkBackend/FrameBackend.cs - MonoDevelop
File Edit View Search Project Build Run Version Control Tools Window Help
Debug | Mixed Plat Default
Xwt.Gtk - Xwt.GtkBackend/FrameBackend.cs - MonoDevelop
Solution
FrameBackend.cs
GdkDrawingArea.cs
GdkExtensions.cs
GdkPopoverWindow.cs
GdkViewPort.cs
GdkAlertDialog.cs
GdkDesktopBackend.cs
GdkEngine.cs
GdkImage.cs
GdkKeyboardHandler.cs
GdkFactory.cs
GdkPlatformBackend.cs
GdkWebViewGtk.cs
GdkWorkarounds.cs
HeaderBox.cs
HeaderBoxGtk2.cs
ImageBuilderBackend.cs
ImageHandler.cs
ImagePatternBackendHandler.cs
ImageViewBackend.cs
LabelBackend.cs
LabelBackendGtk2.cs
LinkLabelBackend.cs
ListStoreBackend.cs
ListStoreBackend.cs
Properties
Build
Build action Compile
Copy to output Do not copy
Custom Tool
Custom Tool
Resource ID
Msc
Name FrameBackend
Path /home/jedidj/Pri...
Type C# source code
Deployment
File attributes
Has path ref?
Include in dep?
Relative target FrameBackend
Target direct Program files
Use project re?
Source Changes Blame Log Merge
Errors Tasks
164     if (borderColor == null)
165         return WidgetStyle.Dark (Gdk.StateType);
166     else
167         return borderColor.value;
168 }
169 set {
170     borderColor = value;
171     HeaderBox hb = widget as HeaderBox;
172     if (hb != null)
173         hb.OnBorderColorChanged (value);
174 }
175
176 public override Color BackgroundColor {
177     get {
178         return base.BackgroundColor;
179     }
180     set {
181         base.BackgroundColor = value;
182         if (Widget is HeaderBox) {
183             ((HeaderBox)Widget).BackgroundColor = value;
184         }
185     }
186 }
187
188 public string Label {
189     get {
190         return Label;
191     }
192     set {
193         Label = value;
194         if (Widget is Gtk.Label)
195             ((Gtk.Label)Widget).Label = value;
196     }
197 }
198 }
199 }
200 }
201 }
```

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

73

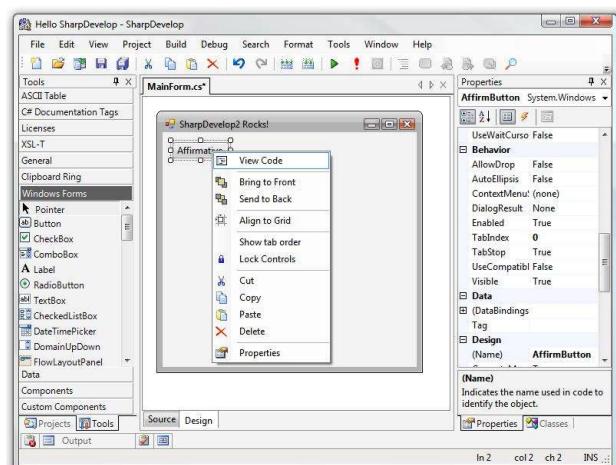
SharpDevelop

- IDE libre (open source) pour C#, VB.NET et C++ sur plateforme .NET.
 - www.icsharpcode.net

⇒ Ergonomie très proche de celle de VisualStudio
 ⇒ N'a pas toutes les possibilités de Visual Studio
 ⇒ Très prometteur

⇒ .Net 2.0 à 4.0
 et Compact Framework

⇒ Peu de documentation
 ⇒ Intégration de mono

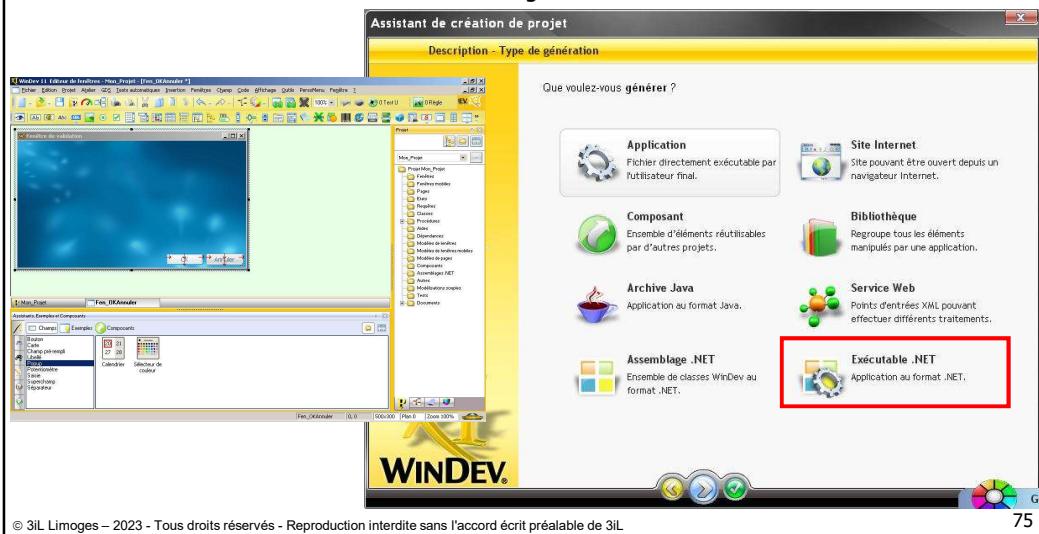


© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

74

WinDev et .Net

- Permet de créer des programmes .Net à partir des classes WinDev
- Permet d'utiliser des assemblages .Net



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

75

Que retenir

- Comprendre le Framework .Net, sa composition, les processus de compilation et d'exécution, sa runtime, ce qu'est du code MSIL et un assembly
- Connaître les technologies disponibles avec le Framework .Net (WinForm, WPF, 3D, ASP.Net, UWP, ...)
- Comprendre l'architecture de développement
- Savoir nommer les 3 langages de base du .Net Framework
- Comprendre les différences entre les 3 technologies que sont .NET Framework, .NET Core et Xamarin
- Nommer les logiciels de développement qui peuvent générer des Assembly .Net

© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL

76

Annexes

Liens pour aller plus loin

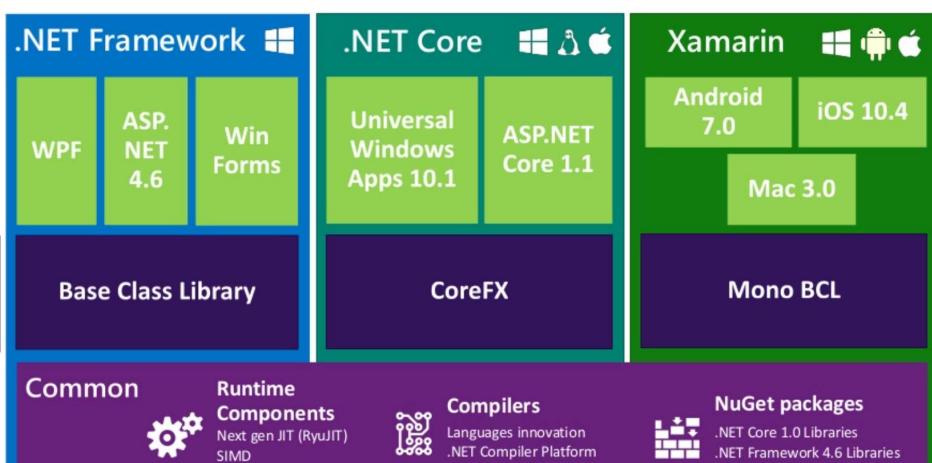
- dotNet
<http://dotnet.developpez.com/>
<http://msdn.microsoft.com/fr-fr/library/>
- C#
<http://tahe.developpez.com/dotnet/csharp/>
<https://learn.microsoft.com/fr-fr/dotnet/csharp/>
<https://msdn.microsoft.com/fr-fr/library/618ayhy6.aspx>
<http://dotnet.developpez.com/faq/csharp/>
<https://www.w3schools.com/cs/index.php>
<https://stackoverflow.com/questions/tagged/c%23>
- Option de compilation
<http://nico-pyright.developpez.com/tutoriel/vc2005/managedworld/>

Librairie des espaces de noms de classes



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

79



© 3iL Limoges – 2023 - Tous droits réservés - Reproduction interdite sans l'accord écrit préalable de 3iL.

80

Options de compilation managed et unmanaged

- Le compilateur nous propose différentes options de compilation (non managed, managed prioritaire, pure managed, et safe managed)
 - ⇒ **No Common Language Runtime Support** (langage machine, pas extensions managées, exécutable Win32 classique, pas d'accès à la bibliothèque .Net)
 - ⇒ **Common Language Runtime Support** (/clr) (génère prioritairement du MSIL, puis du langage machine lorsqu'il ne sait pas générer de MSIL)
 - ⇒ **Pure MSIL Common Language Runtime Support** (/clr:pure) : empêche la compilation de code natif mais autorise l'écriture de code qui dérogerait aux règles du CLS
 - ⇒ **Safe MSIL Common Language Runtime Support** (/clr:safe) : génère des assemblies vérifiables, en se conformant aux exigences du CLS garantir que le code ne viole aucun paramètre de sécurité
 - ⇒ **Common Language Runtime Support, Old Syntax** (/clr:oldSyntax) : utilisé pour avoir une compatibilité avec les extensions managées du framework 1.x

Exemple avec point d'entrée (C#)

```
#include "stdafx.h"
#include "Form1.h"
#include <windows.h>

using namespace test1;

int APIENTRY _tWinMain(HINSTANCE hInstance,
                      HINSTANCE hPrevInstance,
                      LPTSTR lpCmdLine, int nCmdShow)
{
    System::Threading::Thread::CurrentThread->ApartmentState =
        System::Threading::ApartmentState::STA;
    Application::Run(new Form1());
    return 0;
}
```



Interception d'un message

```
namespace WindowsApplication1
{
    public class Form1 : System.Windows.Forms.Form
    {
        .....
        private void InitializeComponent()
        {
            .....
            this.MouseDown += new
                System.Windows.Forms.MouseEventHandler(this.FormExemple_MouseDown);
        }
        private void FormExemple_MouseDown(object sender,
                                         System.Windows.Forms.MouseEventArgs e)
        {
            MessageBox.Show("Le bouton de la souris a été appuyé");
        }
        .....
        static void Main()
        {
            Application.Run(new Form1());
        }
    }
}
```

