



# .NET Plateforme Overview

.NET Framework, CLR, MSIL, Assemblies, CTS, etc.

---

Abdallah MOUJAHID  
PMP®, COBIT® V5, ITIL® V3, ISO 27002



# Table de Contenu

## 1. C'est quoi .NET?

- Architecture de la plateforme Ms.NET

## 2. C'est quoi le Framework .NET?

- Architecture du Framework .NET

## 3. Common Language Runtime (CLR)

## 4. Managed Code

## 5. Le langage intermédiaire MSIL

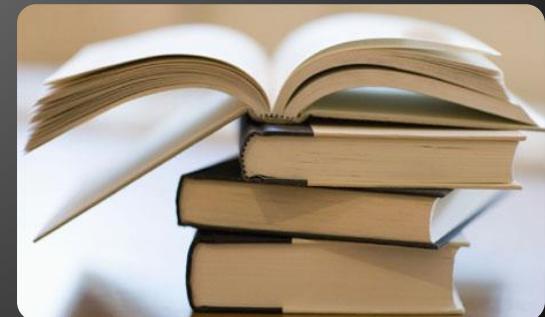
## 6. Assemblies & Metadata

## 7. Applications .NET



# Table de Contenu (2)

8. Common Language Infrastructure (CLI) et integration des différents langages
  - ◆ Common Language Specification (CLS)
  - ◆ Common Type System (CTS)
9. Framework Class Library
10. L'environnement de développement intégré - Visual Studio





# .NET Framework

Microsoft's Platform for  
Application Development

# C'est quoi la plateforme .NET?

- ◆ La plateforme .NET

- ◆ Plateforme Microsoft pour le développement logiciel
- ◆ Technologie unifiée pour le développement de miltitude de type d'applications
  - ◆ GUI / Web / mobile / serveur / cloud / etc.

- ◆ Versions de la plateforme .NET

- ◆ .NET Framework
- ◆ .NET Compact Framework



# C'est quoi .NET Framework?

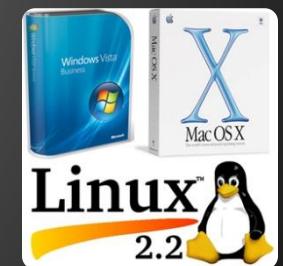
## ◆ .NET Framework

- ◆ Un environnement de développement et d'exécution des applications .NET
- ◆ Un modèle de programmation unié, plusieurs langages, librairies de classe, infrastructure, composants et outils pour le développement des applications.
- ◆ En general, on considère que:
  - ◆ Plateforme .NET == Framework .NET



# Architecture du Framework .NET

- Le système d'exploitation gère les ressources, les processus et les utilisateurs de la machine
- Fournit plusieurs services aux applications (Threads, I / O, GDI +, DirectX, COM, COM +, MSMQ, IIS, WMI, ...)
- CLR est un processus distinct dans le système d'exploitation



Système d'Exploitation (SE)

# Architecture du Framework .NET (2)

- CLR gère l'exécution du code .NET
- Gère la mémoire, l'accès concurrentiel, la sécurité, ...

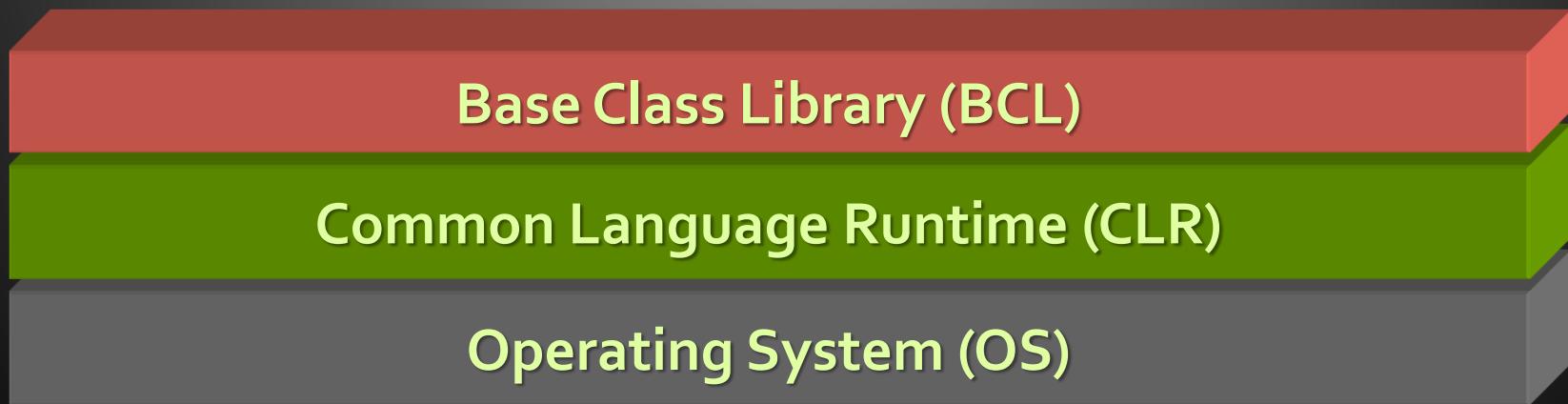


Common Language Runtime (CLR)

Operating System (OS)

# Architecture du Framework .NET (3)

- Bibliothèque orientée objet riche en classes fondamentales
- Entrée-sortie, collections, traitement de texte, mise en réseau, sécurité, multithreading, ...



# Architecture du Framework .NET (4)

- Accès aux bases de données
- ADO.NET, LINQ, LINQ-to-SQL et Entity Framework
- Prise en charge renforcé de XML



ADO.NET, LINQ and XML (Data Tier)

Base Class Library (BCL)

Common Language Runtime (CLR)

Operating System (OS)

# Architecture du Framework .NET (5)

- Windows Communication Foundation (WCF) et Windows Workflow Foundation (WWF) pour le monde SOA.



**WCF and WWF (Communication and Workflow Tier)**

**ADO.NET, LINQ and XML (Data Tier)**

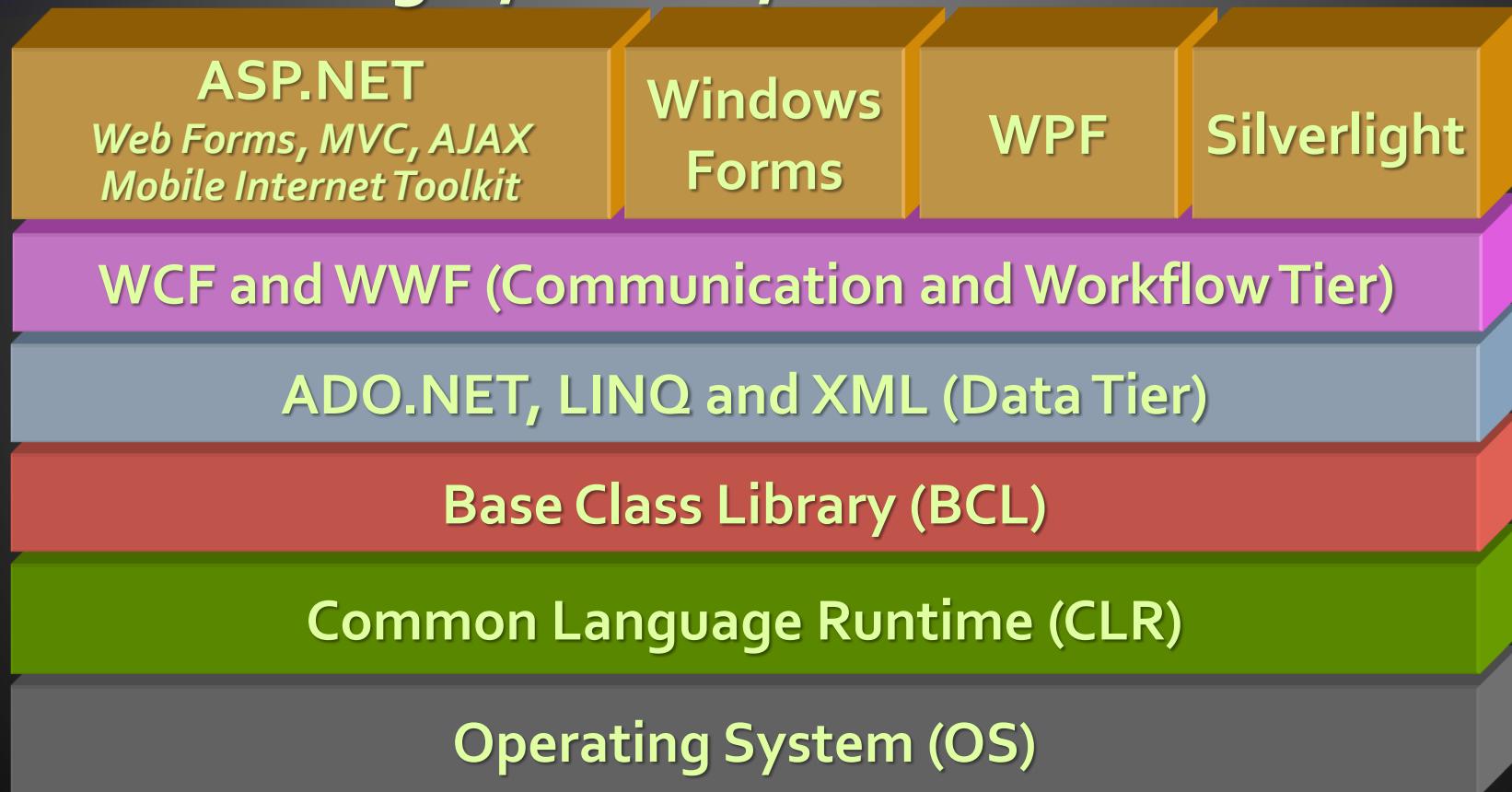
**Base Class Library (BCL)**

**Common Language Runtime (CLR)**

**Operating System (OS)**

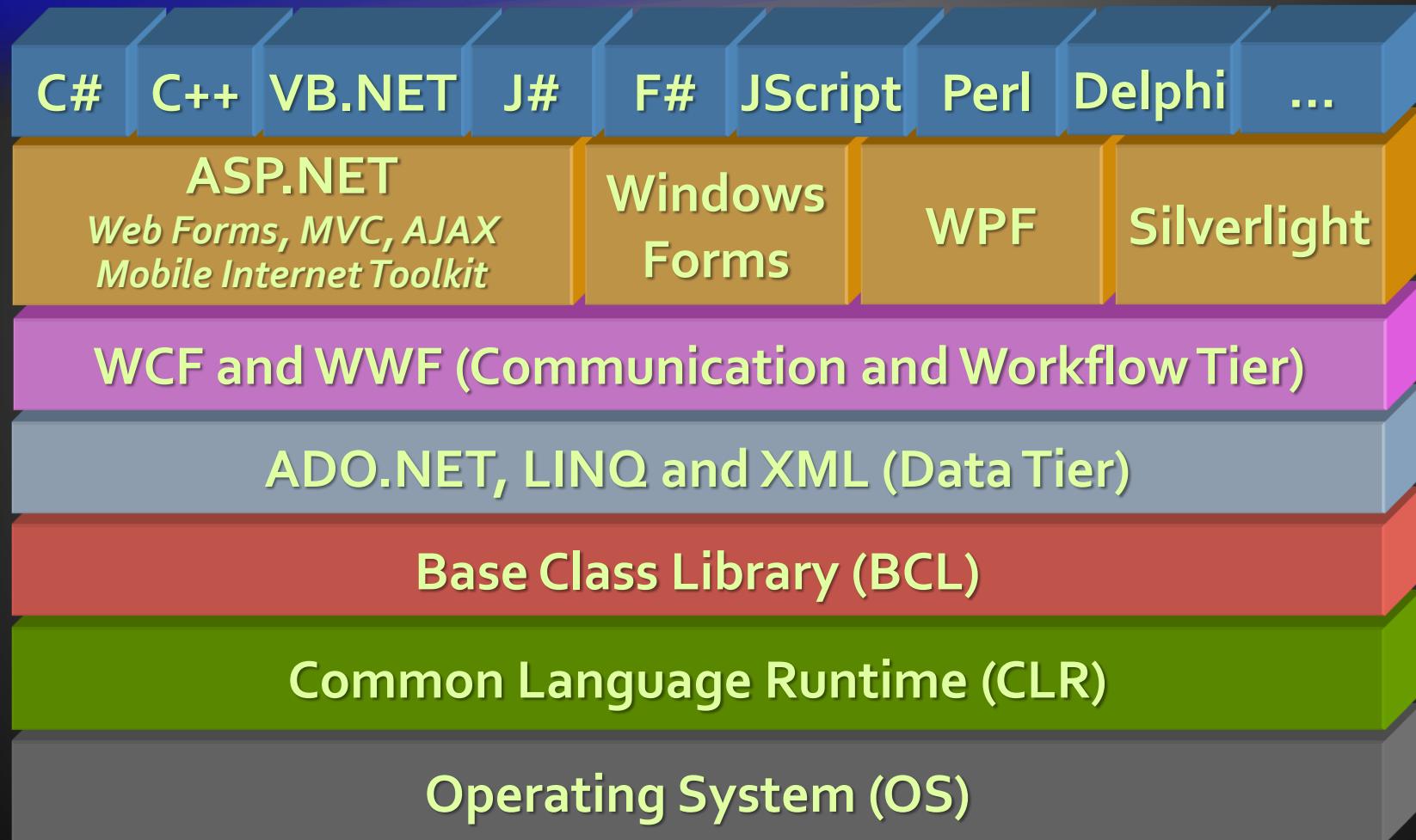
# Architecture du Framework .NET(6)

- Technologies pour les interfaces utilisateurs: Web, Windows GUI, WPF, Silverlight, mobile, ...



# Architecture du Framework .NET (7)

Langage de programmation pour tous les goûts!



# Framework .NET 4.0

## .NET Framework 4.0

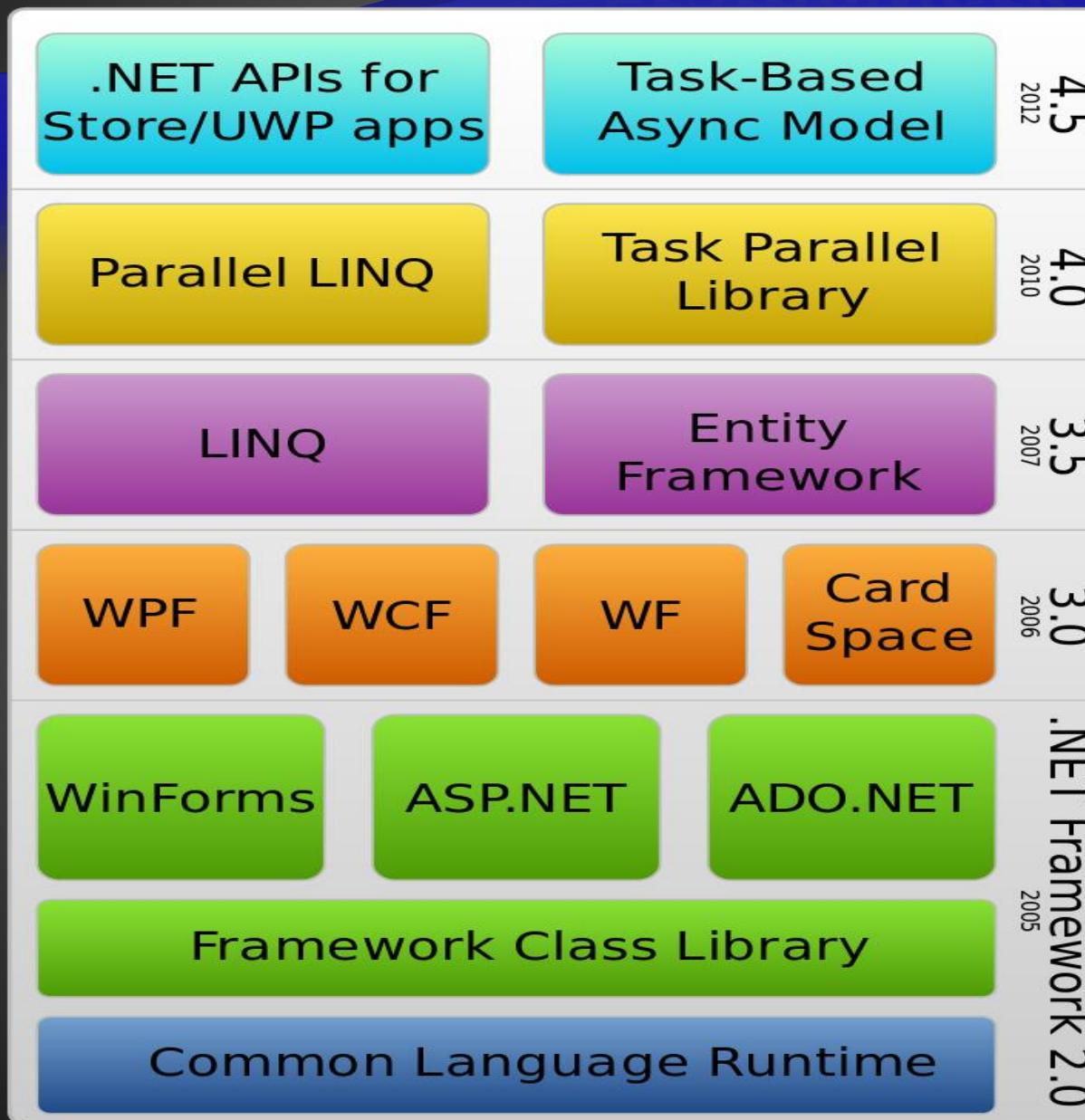


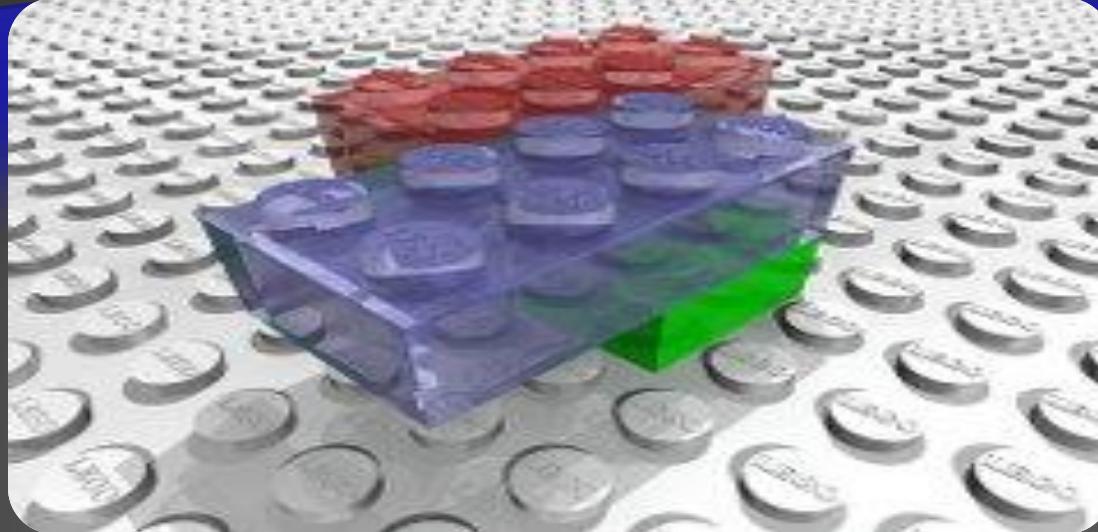
# Versions du Framework .NET

Overview of .NET Framework release history

Version number	CLR version	Release date	Development tool	Included in		Replaces
				Windows	Windows Server	
1.0	1.0	2002-02-13	Visual Studio .NET <sup>[22]</sup>	XP <sup>[a]</sup>	N/A	N/A
1.1	1.1	2003-04-24	Visual Studio .NET 2003 <sup>[22]</sup>	N/A	2003	1.0 <sup>[23]</sup>
2.0	2.0	2005-11-07	Visual Studio 2005 <sup>[24]</sup>	N/A	2003, 2003 R2, <sup>[25]</sup> 2008 SP2, 2008 R2 SP1	N/A
3.0	2.0	2006-11-06	Expression Blend <sup>[26][b]</sup>	Vista	2008 SP2, 2008 R2 SP1	2.0
3.5	2.0	2007-11-19	Visual Studio 2008 <sup>[27]</sup>	7, 8 <sup>[c]</sup> , 8.1 <sup>[c]</sup> , 10 <sup>[c]</sup>	2008 R2 SP1	2.0, 3.0
4.0	4	2010-04-12	Visual Studio 2010 <sup>[28]</sup>	N/A	N/A	N/A
4.5	4	2012-08-15	Visual Studio 2012 <sup>[29]</sup>	8	2012	4.0
4.5.1	4	2013-10-17	Visual Studio 2013 <sup>[30]</sup>	8.1	2012 R2	4.0, 4.5
4.5.2	4	2014-05-05	N/A	N/A	N/A	4.0–4.5.1
4.6	4	2015-07-20	Visual Studio 2015 <sup>[31]</sup>	10	N/A	4.0–4.5.2
4.6.1	4	2015-11-30 <sup>[32]</sup>	Visual Studio 2015 Update 1	10 v1511	N/A	4.0–4.6
4.6.2	4	2016-08-02 <sup>[33]</sup>		10 v1607	N/A	4.0–4.6

# Framework .NET 4.5





# Common Language Runtime (CLR)

Le Coeur du Framework .NET

# Common Language Runtime (CLR)

- ◆ Environnement d'exécution géré (Managed)
  - ◆ Contrôle l'exécution du code « managé »
- ◆ Peut être considéré comme machine virtuelle.
  - ◆ Pareille à la machine virtuelle Java/JVM.
- ◆ Une Compilation à la demande est utilisé  
(il n'interprète pas le code)
  - ◆ Connue comme une compilation Juste à Temps  
(Just In Time ou JIT compilation)



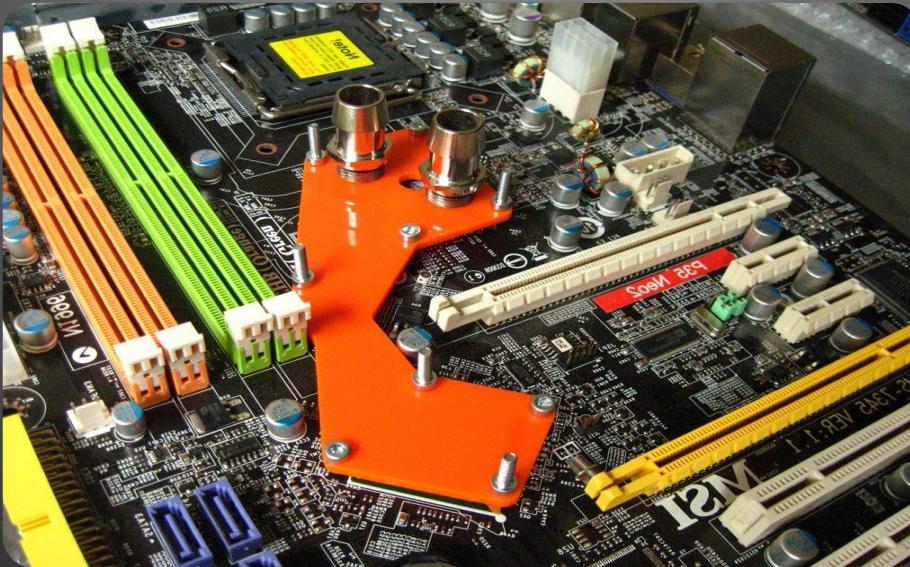
# Responsabilités de la CLR

- ◆ L'exécution du code IL et la compilation JIT
- ◆ La gestion des ressources de mémoire et d'application
- ◆ L'interaction avec le système d'exploitation
- ◆ Gestion de la sécurité:
  - ◆ La sécurité d'accès au code
  - ◆ La sécurité basée sur les rôles
- ◆ Gestion des exceptions



# Code Managé & Non-Managé

Quelle est la Difference?



- ◆ Le code exécuté par la CLR est appelé code managé
- ◆ Représente le code de programmation dans le niveau le plus bas - Langage MSIL
- ◆ Contient les metadatas:
  - Description des classes, interfaces, propriétés, champ, méthodes, paramètres, etc.
- ◆ Les programs, écrits dans n'importe quelle langage .NET sont:
  - Compilés en un code managé (MSIL)
  - Packagé entant qu' assemblies (fichiers .exe ou .dll)



- ◆ Orienté-Objet
- ◆ Sécurisé
- ◆ Permet l'intégration entre les composants et les types de données des différents langages de programmation.
- ◆ Portable entre les différentes plateforme:
  - ◆ Windows, Linux, Max OS X, etc.

# Code Non-Managé (Win32)

- ◆ Pas de protection de mémoire
- ◆ Ne contient pas les metadatas
- ◆ Compilé en un code dépendant de la machine:
  - ◆ Besoin de version spécifique pour chaque plateform
  - ◆ Difficile à porter à d'autres plateforms

# Gestion de la Mémoire

- ◆ CLR gère la mémoire automatiquement
  - ◆ Les objets dynamique sont stockés dans le “managed heap”
  - ◆ Les objets non utilisés sont automatiquement nettoyés par le “garbage collector”
- ◆ Certains problèmes ont été résolus:
  - ◆ Les fuites de mémoire
  - ◆ L'accès à la mémoire libérée ou non allouée
- ◆ Les objets sont accessibles via une référence

# Langage Intermédiaire (MSIL)



```
    ldi    0000
    ldi    20013
    ldi    20013
    cli
    jsr    register
    lda    #<normstart
    lda    #>normstart
    sta    P2
    sty    P2+1
    jmp    input

    pla    p1          ; dem aufruf folgt
    sta    p1          ; den text folgt
    pla    p1+1        ; brk=ende ausgetragen
```

# Langage Intermédiaire (MSIL, IL, CIL)

- ◆ Langage bas niveau (langage machine) pour la CLR .NET
- ◆ A ces propres instructions CPU instructions
  - ◆ Chargement et stockage des données, appel des méthodes
  - ◆ Opérations arithmétiques et logiques
  - ◆ Gestion des exceptions
  - ◆ Etc.
- ◆ MSIL est converti en instructions dédiées au CPU physique par le compilateur JIT

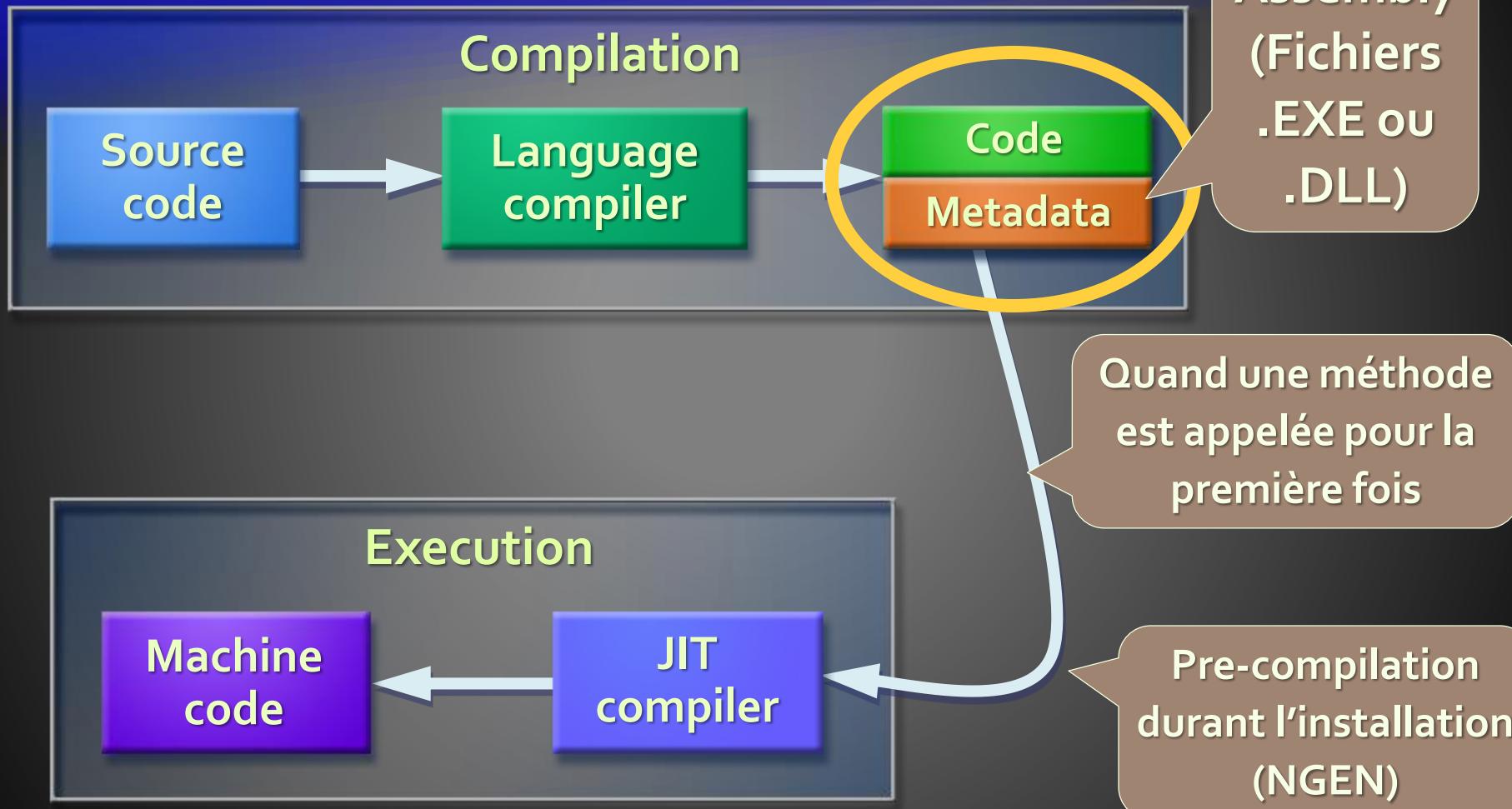


# Exemple de Programme MSIL

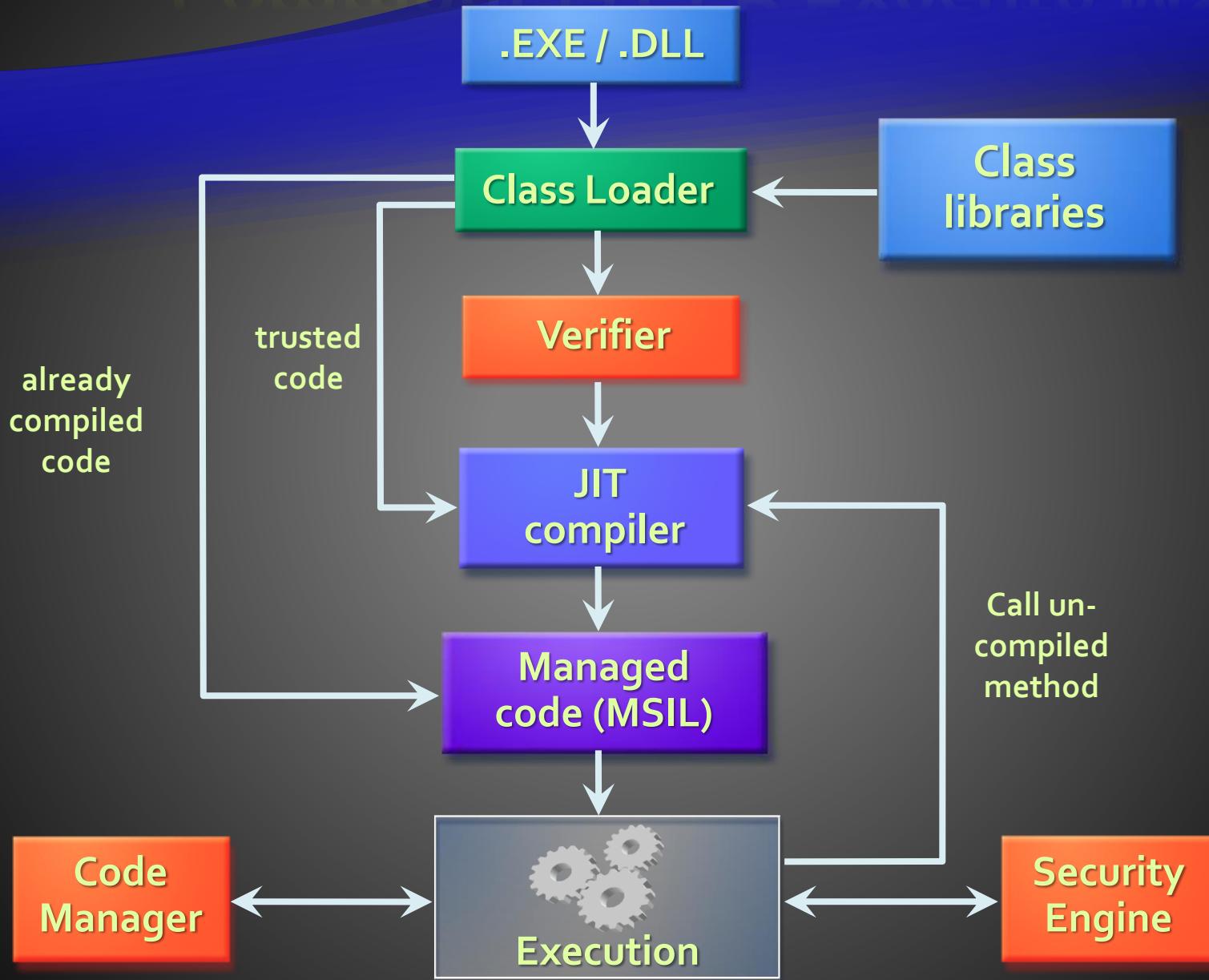
```
.method private hidebysig static void Main() cil managed
{
    .entrypoint
    // Code size      11 (0xb)
    .maxstack  8
    ldstr     "Hello, world!"
    call      void
        [mscorlib]System.Console::WriteLine(string)
    ret
} // end of method HelloWorld::Main
```



# Compilation & Exécution



# Comment la CLR Exécute MSIL?





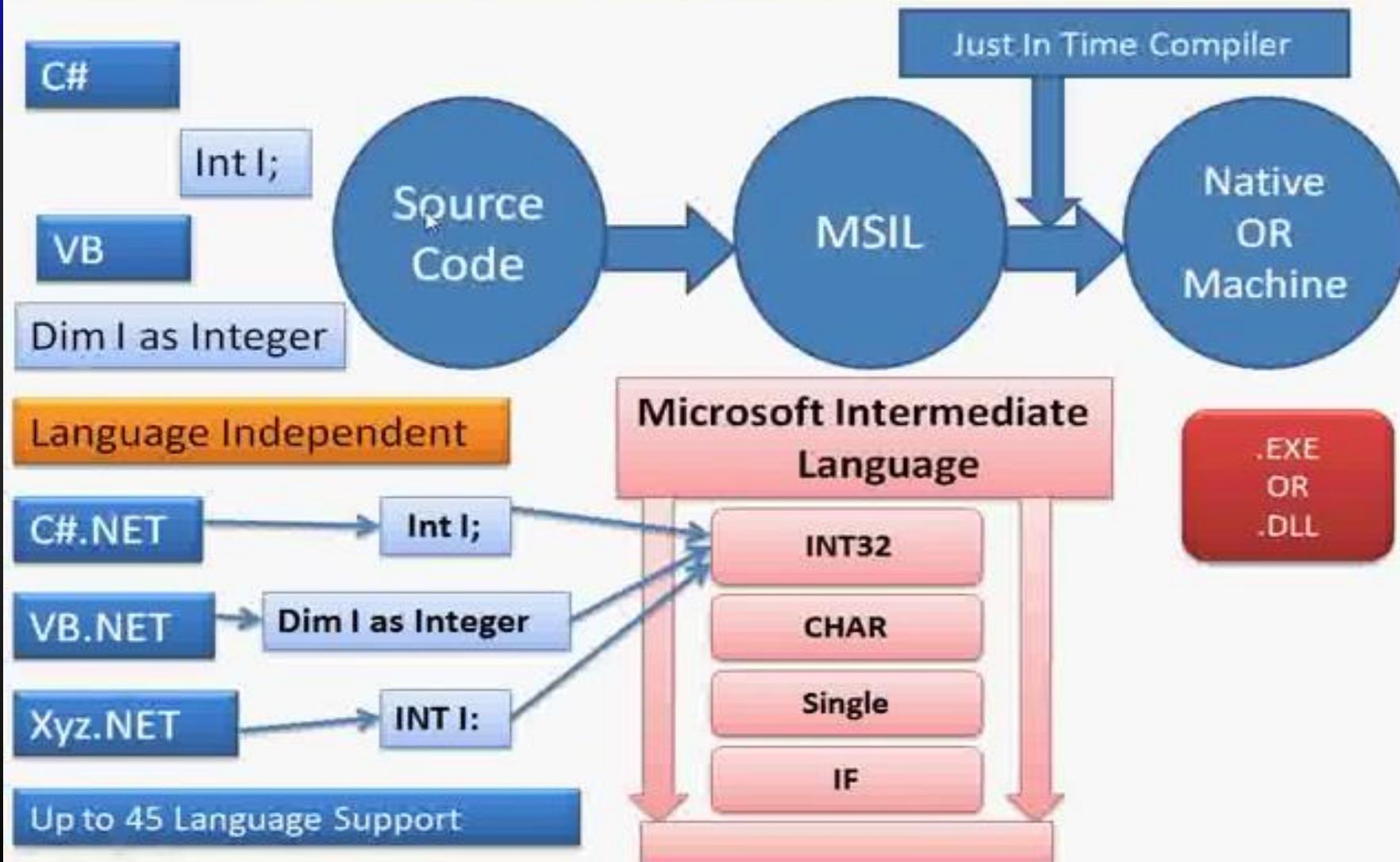
# .NET Applications

## Assemblies, Metadata & Applications

# .NET Assemblies

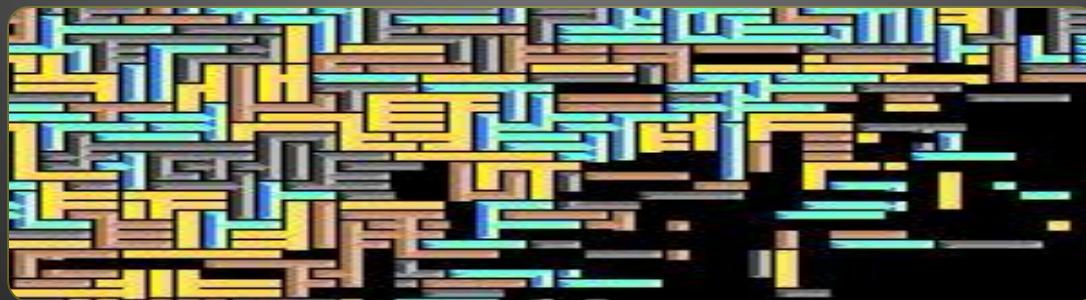
- ◆ Stockées dans les fichiers DLL et EXE
- ◆ Contient une liste des classes, des types et des ressources
- ◆ La plus petite unité de déploiement dans CLR
- ◆ Un assembly a un numéro de version unique
- ◆ C'est un modèle de déploiement .NET
- ◆ Pas de conflits de version (enfer DLL ou DLL hell).
- ◆ Prise en charge de l'exécution côte-à-côte de différentes versions d'un même ensemble

# Common Language Runtime



# Métadonnées dans les assemblies

- ◆ Données sur les données contenues dans l'assembly
- ◆ Partie intégrante de l'assembly
- ◆ Généré par le compilateur de langages .NET
- ◆ Décrit toutes les classes, les membres de leur classe, les versions, les ressources, etc



# Métadonnées dans les assemblies

## Type Description

Classes, interfaces, inner types, base classes, implemented interfaces, member fields, properties, methods, method parameters, return value, attributes, etc.

## Assembly Description

Name

Version

Localization

[digital  
signature]



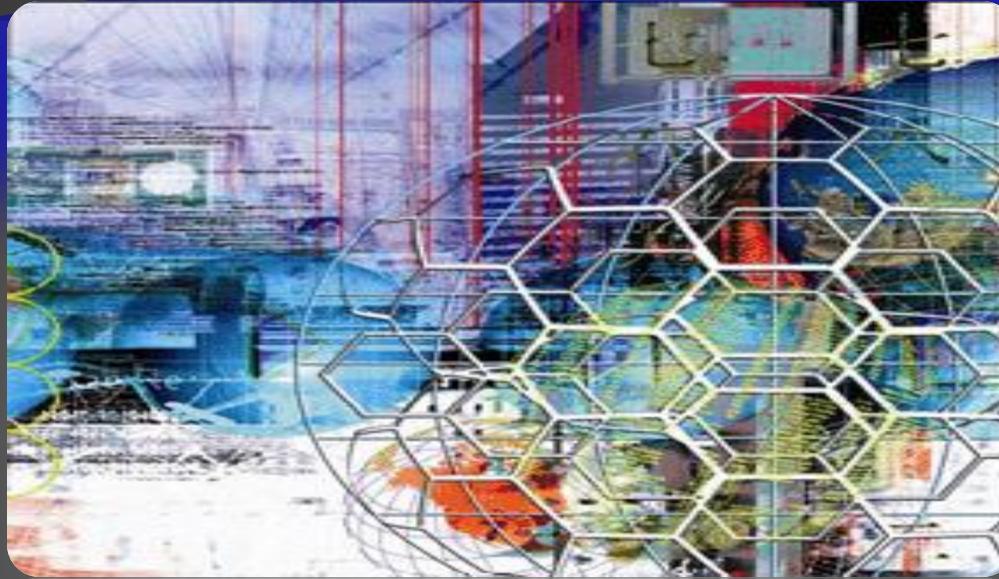
Dependencies on other assemblies

Security permissions

Exported types

# Applications .NET

- ◆ Consiste en un ou plusieurs assemblies
- ◆ Installées par "copier / coller"
  - ◆ Pas d'enregistrement complexe des components
- ◆ Différentes applications utilisent différentes versions des assemblies communs
  - ◆ Pas de conflits en raison de leurs "Nom fort"
- ◆ Installation facile, désinstallation et mise à jour.



# Common Language Infrastructure

Comment le .NET Support Multiple Langages?

# Common Language Infrastructure

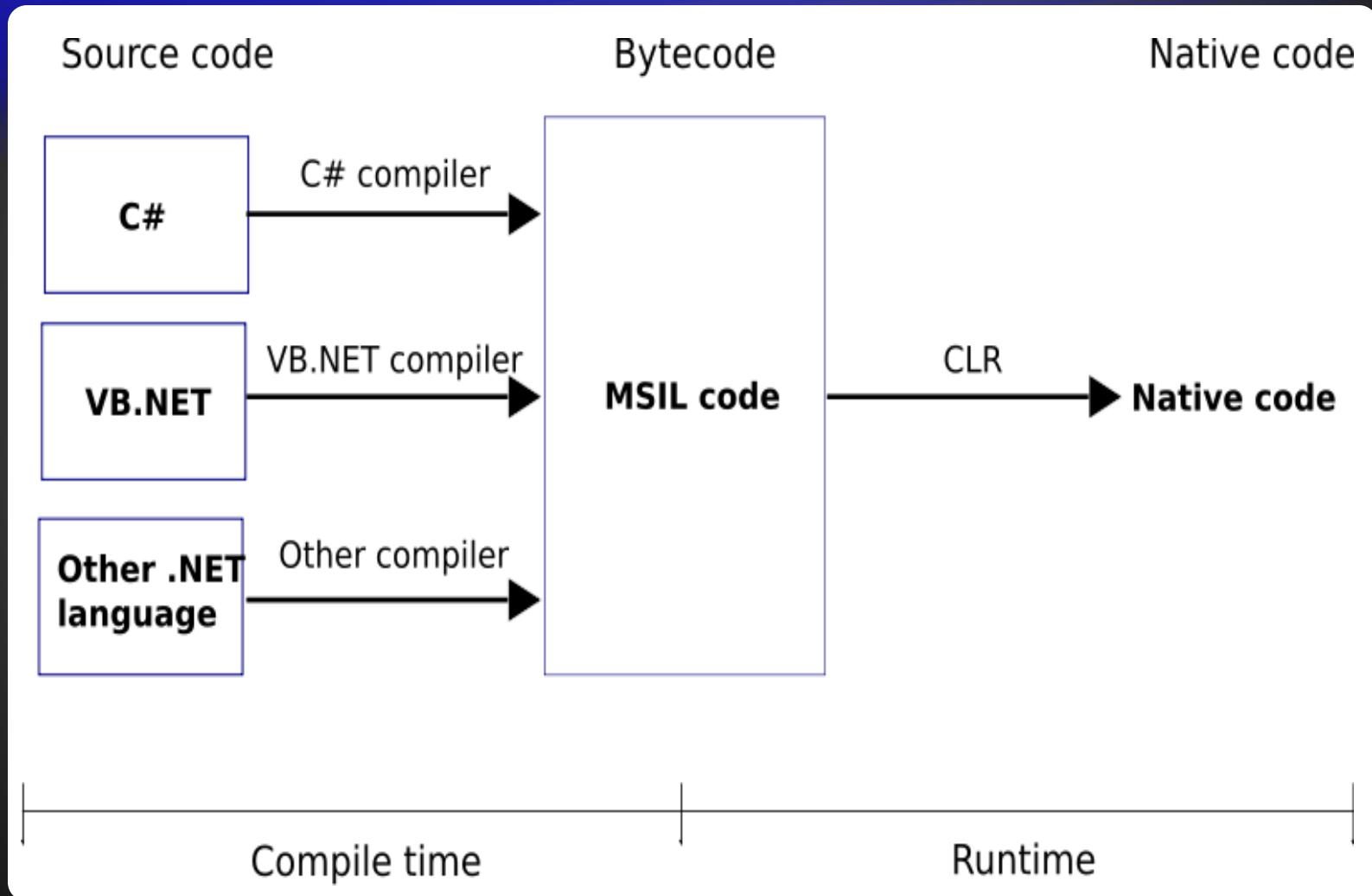
- ◆ Infrastructure Commune de langage
  - ◆ Specification ouverte développée par Microsoft (ECMA – 335)
  - ◆ Multiple langages de haut-niveau qui s'excutent dans différentes platforms sans changement dans le code source
  - ◆ Partie Standardisée du CLR
  - ◆ Le framework .NET est l'implémentation Windows de la CLI
  - ◆ Mono est l'implementation Linux de la CLI

# Common Language Infrastructure (2)

- ◆ CLI décrit les aspects suivants:
  - ◆ Le système Commun des Types (CTS)
  - ◆ Assemblies and métadonnées
  - ◆ Spécification Commune de Langage (CLS)



# Compilation et Exécution du code .NET



# Système Commun de Type (CTS)

- ◆ CTS définit les types de données supportés par le CLR ainsi que les opérations pouvant être effectuées sur ces types.
- ◆ Assure la compatibilité au niveau des données entre les différents langages .NET
  - Exemple: **string** dans C# est le même que le type **String** dans VB.NET et J#
- ◆ Tous les types héritent de **System.Object**

# Spécification Commune de Langage (CLS)

- ◆ CLS est un système de règles et d'obligations, que tous les langages .NET doivent respecter
- ◆ Assure la compatibilité et la facilité d'interaction entre les langages .NET
- ◆ Exemple: CLS impose que tous les langages .NET doivent être orienté objet
- ◆ Lors de l'utilisation des techniques de programmation non-conforme CLS on perd la compatibilité avec les autres langages .NET



# Les langages .NET

C#, VB.NET, C++, J#, etc.

- ◆ Langages .NET proposés par Microsoft
  - C#, VB.NET, Managed C++, J#, F#, JScript
- ◆ Langages proposés par des tiers
  - Object Pascal, Perl, Python, COBOL, Haskell, Oberon, Scheme, Smalltalk...
- ◆ Différents langages peuvent être utilisés dans la même application
- ◆ Héritage de types et gestion des exceptions inter-langages

- ◆ C# est mélange entre C++, Java
  - ◆ Entièrement Orienté-Objet
- ◆ Modèle de programmation orienté-composants
  - ◆ Composants, propriétés et événements
  - ◆ Pas d'entêtes de fichiers comme C/C++
  - ◆ Convient pour des applications graphiques et Web
  - ◆ Documentation basé sur XML
- ◆ En C# tous les types de données sont des objets
  - ◆ Exemple: 5.ToString() est un appel valide

# Langage C# – Exemple

- ◆ C# est standardisé par l'ECMA et l'ISO
- ◆ Exemple d'un programme C#:

```
using System;

class NumbersFrom1to100
{
    static void Main()
    {
        for (int i=1; i<=100; i++)
        {
            Console.WriteLine(i);
        }
    }
}
```



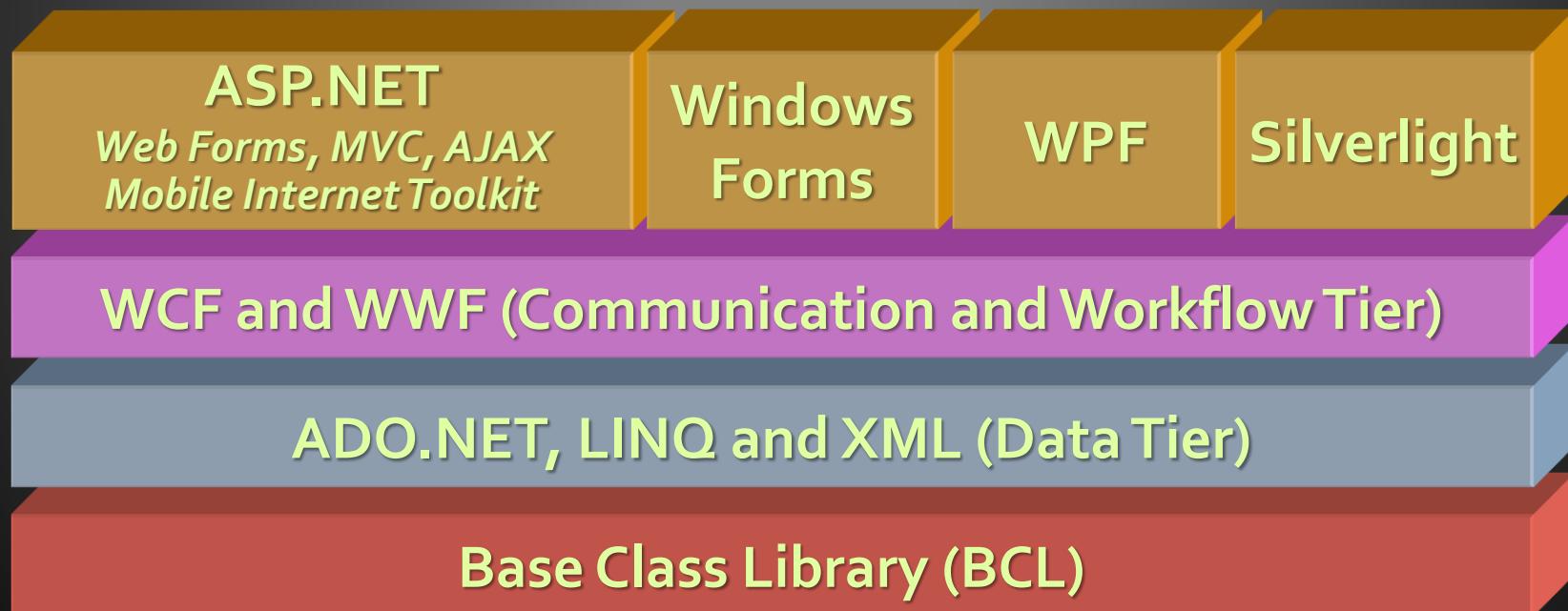


# Framework Class Library (FCL)

Standard Out-of-the-box .NET APIs

# Framework Class Library (FCL)

- ◆ La librairie de Classe du Framework est la librairie standard du Framework .NET qui contient les classes et les composant réutilisables (APIs)



# FCL Namespaces

## ASP.NET

*Web Forms, MVC, AJAX  
Mobile Internet Toolkit*

System.Web

System.Web.Mvc

## Windows Forms

System.Windows.Forms

System.Drawing

## WPF & Silverlight

System.Windows

System.Windows.Media

System.Windows.Markup

## WCF and WWF (Communication and Workflow Tier)

System.ServiceModel

System.Activities

System.Workflow

## ADO.NET, LINQ and XML (Data Tier)

System.Data

System.Linq

System.Xml

System.Data.Linq

System.Xml.Linq

System.Data.Entity

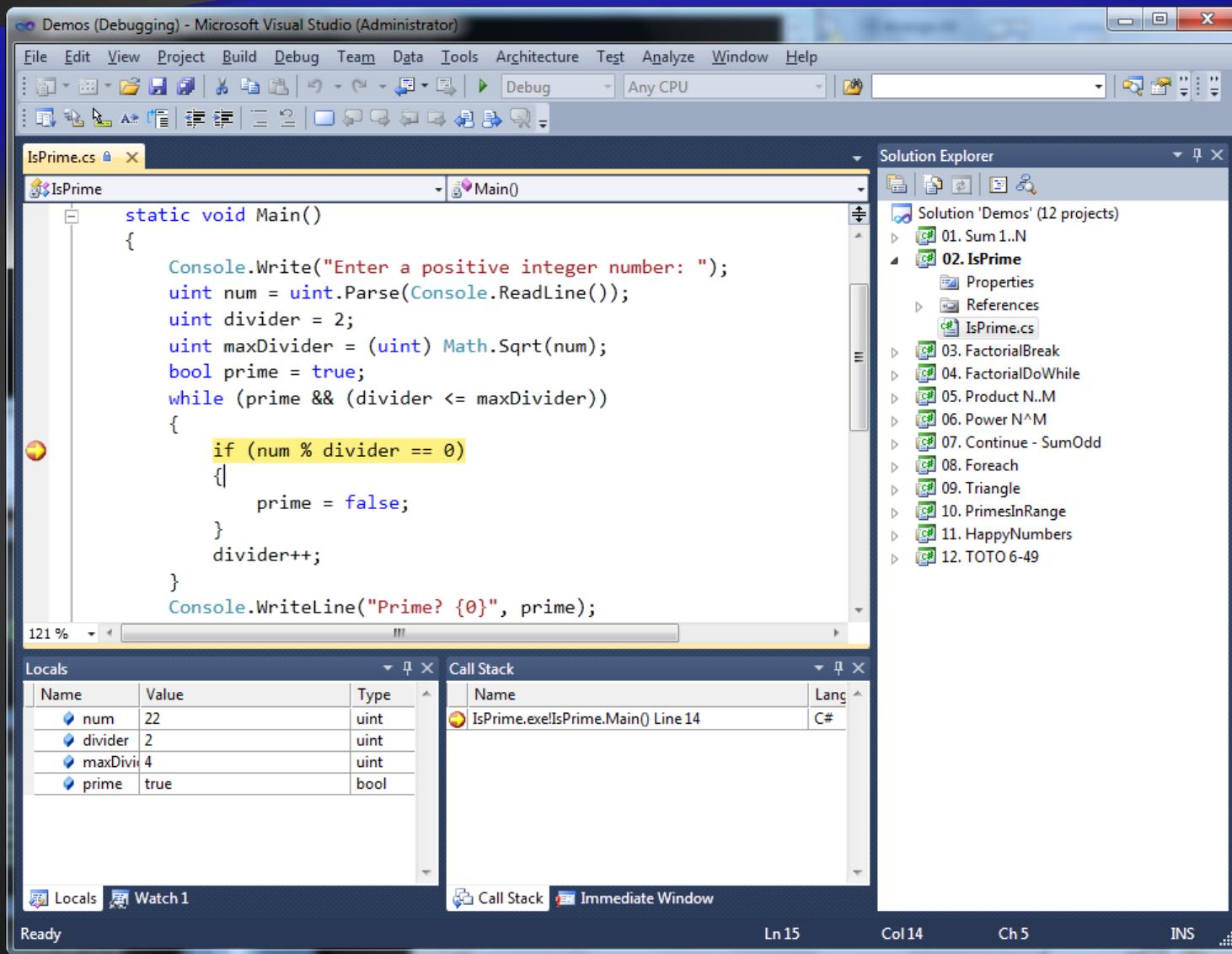
- Visual Studio est un puissant Environnement de développement Intégré (IDE) pour les développeurs .NET
  - Créer, éditer, compiler et exécuter les applications .NET
  - Différents langages – C#, C++, VB.NET, J#, ...
  - Editeur de code flexible
  - Puissant débogueur
  - Intégré avec SQL Server and IIS
  - Puissant support des Web services, WCF et WWF



- ◆ **Programmation Visuelle**
  - ◆ Orienté-composant, basé sur des événements
- ◆ **Code managé et non-managé**
- ◆ **Assistants et éditeurs utiles**
  - ◆ Windows Forms Designer
  - ◆ WCF / Silverlight Designer
  - ◆ ASP.NET Web Forms Designer
  - ◆ ADO.NET / LINQ-to-SQL / XML Data Designer
- ◆ **Plusieurs extensions tierces**



# Visual Studio IDE





# Comparaison entre C# & C++

# Introduction

- ◆ Le C ++ a été écrit par Bjarne Stroustrup à Bell Labs durant la période 1983 - 1985.
- ◆ Le C ++ est une extension de C.
- ◆ Anders Hejlsberg est le principal concepteur et architecte en chef de Microsoft qui était à l'origine de la création du Langage C#.

# Concepts OO dans C++ & C#

- ◆ Les concepts de base de la POO utilisés dans C++ et C# sont les suivants:
  - ◆ Objets
  - ◆ Classes
  - ◆ Abstraction de données
  - ◆ Encapsulation
  - ◆ Héritage
  - ◆ Polymorphisme

# Différences C# & C++ (1)

- ◆ L'Héritage:

- Contrairement à C++, le C# n'autorise pas l'héritage multiple et se contente de l'héritage simple.
- L'implémentation de plusieurs interface (nombre illimité) est possible avec C#.

- ◆ Les Includes:

- Le C# remplace les includes, utilisés dans le C++, par la directive « using » permettant de référencer les autres classes (namespaces)

# Différences C# & C++ (2)

- ◆ Garbage Collector:

- ◆ La CLR, libère automatiquement les objets en mémoire
- ◆ Pas besoin de “free” comme en C/C++