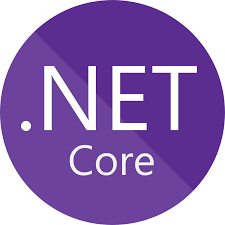
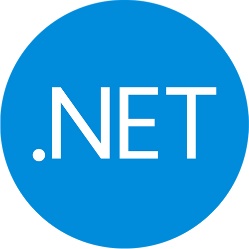
CCI Nîmes

Présentation .Net 

Présentation et introduction au .Net







Réalisé et enseigné par

Jérôme Ferratier



Table des matières

[Introduction 3](#_Toc115986757)

[Historique et LTS 6](#_Toc115986758)

[Présentation WPF 8](#_Toc115986759)

[Avantages du WPF 9](#_Toc115986760)

Introduction

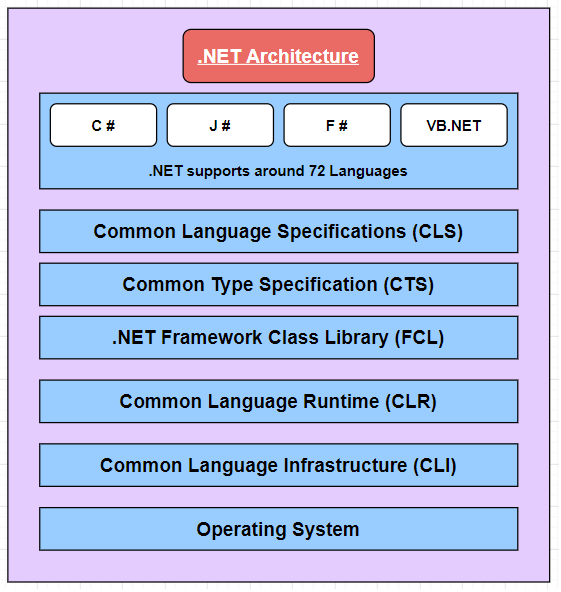
**Définition Wikipédia :**

***.NET Framework***[N 1](https://fr.wikipedia.org/wiki/.NET_Framework#cite_note-1) ([abr.](https://fr.wikipedia.org/wiki/Abr%C3%A9viation) **NetFx**) est un cadriciel ([*framework*](https://fr.wikipedia.org/wiki/Framework)) pouvant être utilisé par un [système d'exploitation](https://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27exploitation) [Microsoft Windows](https://fr.wikipedia.org/wiki/Microsoft_Windows) et [Microsoft Windows Mobile](https://fr.wikipedia.org/wiki/Microsoft_Windows_Mobile) depuis la version 5 ([*.NET Compact Framework*](https://fr.wikipedia.org/wiki/.NET_Compact_Framework)). Une version légère et limitée fournie avec un moteur d'exécution fonctionnant à l'intérieur d'un navigateur ou d'un périphérique mobile est disponible sous le nom de [*Silverlight*](https://fr.wikipedia.org/wiki/Microsoft_Silverlight). La version 3.0 du *framework* est intégrée à [Windows Vista](https://fr.wikipedia.org/wiki/Microsoft_Windows_Vista) et à la plupart des versions de [Windows Server 2008](https://fr.wikipedia.org/wiki/Windows_Server_2008) ou téléchargeable depuis le site de l'éditeur [Microsoft](https://fr.wikipedia.org/wiki/Microsoft). La version 4.5.1 du *framework* est intégrée à [Windows 8](https://fr.wikipedia.org/wiki/Windows_8). La version 4.6 est intégrée à [Windows 10](https://fr.wikipedia.org/wiki/Windows_10).

***.NET Core*** (prononcé « dot net core») est un cadriciel ([Framework](https://fr.wikipedia.org/wiki/Framework)) [Libre et Open Source](https://fr.wikipedia.org/wiki/Free/Libre_Open_Source_Software) pour les systèmes d'exploitation [Windows](https://fr.wikipedia.org/wiki/Windows), [macOS](https://fr.wikipedia.org/wiki/MacOS) et [Linux](https://fr.wikipedia.org/wiki/Linux)[4](https://fr.wikipedia.org/wiki/.NET_Core#cite_note-4). Il comprend CoreCLR, un [environnement d'exécution](https://fr.wikipedia.org/wiki/Environnement_d%27ex%C3%A9cution) complet de [CLR](https://fr.wikipedia.org/wiki/Common_Language_Runtime), la machine virtuelle qui gère l'exécution des programmes .NET.

CoreCLR est livré avec un [compilateur à la volée](https://fr.wikipedia.org/wiki/Compilateur_juste-%C3%A0-temps) amélioré, appelé RyuJIT. .NET Core inclut également CoreFX, qui est un fork partiel de [FCL](https://en.wikipedia.org/wiki/Framework_Class_Library). Bien que .NET Core partage un sous-ensemble d'API [.NET Framework](https://fr.wikipedia.org/wiki/.NET_Framework), il est livré avec sa propre [API](https://fr.wikipedia.org/wiki/Interface_de_programmation) qui ne fait pas partie de [.NET Framework](https://fr.wikipedia.org/wiki/.NET_Framework). En outre, .NET Core contient CoreRT, le runtime *.NET Native*, optimisé pour être intégré aux [fichiers exécutables](https://fr.wikipedia.org/wiki/Fichiers_ex%C3%A9cutables) natifs compilés [avant l'exécution](https://fr.wikipedia.org/wiki/Compilation_anticip%C3%A9e). Une variante de la bibliothèque .NET Core est utilisée pour [UWP](https://fr.wikipedia.org/wiki/Universal_Windows_Platform). L'interface de [ligne de commande](https://fr.wikipedia.org/wiki/Ligne_de_commande) .NET Core offre un point d'entrée d'exécution pour les [systèmes d'exploitation](https://fr.wikipedia.org/wiki/Syst%C3%A8mes_d%27exploitation) et fournit des services de développement, tels que la compilation et la gestion des packages.

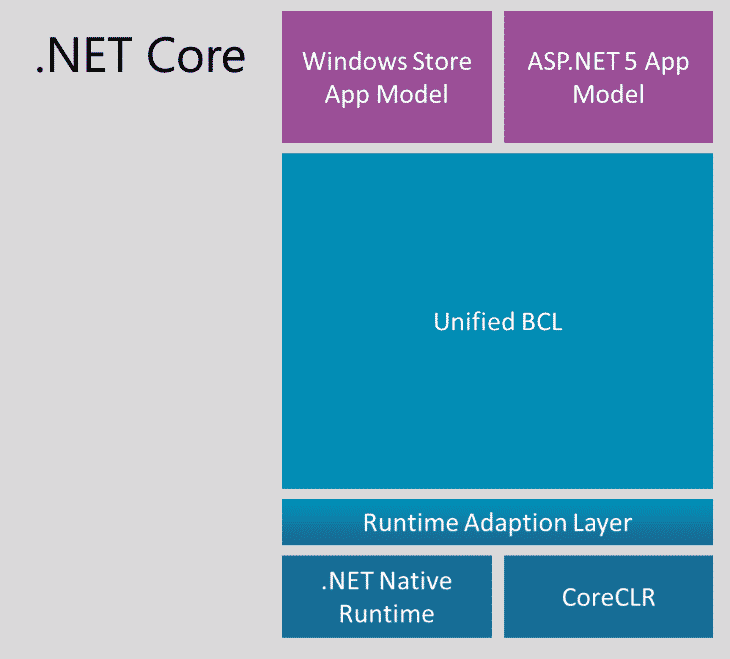
**Présentation framework .NET :**

* le framework .NET , étaient d’excellents outils. Ils sont faciles d’accès et peuvent en même temps se révéler particulièrement riches et complexes. Ils permettent d’élaborer des composants logiciels extrêmement performants.
* La richesse du framework .NET passe par le nombre important de terminaux pour lesquels nous pouvions produire une solution, les ordinateurs et serveurs utilisant un OS Microsoft avec le framework .NET, les appareils mobiles utilisant le framework .NET compact, les serveurs web en utilisant l’ASP.NET, la programmation pour mobile et les applications Windows Store, si on s’en tient à l’écosystème Microsoft. A cela il faut ajouter Mono et Xamarin qui nous ont permis de développer pour Android et les appareils de l’écosystème Apple (macOS et iOS). En faisant attention, il était même possible de passer une partie de son code d’un environnement vers un autre : après compilation, tout fonctionnait correctement.
* L’inconvénient majeur de ce modèle est que chacun de ces environnements dispose de son propre runtime et de sa propre implémentation du framework. C’est là que les choses se corsent. En effet, il faut pouvoir garantir une homogénéité à tout le monde, même si des équipes différentes sont en charge des évolutions. De la même façon il faut pouvoir satisfaire la demande de chacun de ces modèles applicatifs si on veut pouvoir continuer à tous les faire vivre.
* Chaque runtime permettaient d’accueillir des développeurs de tout bord ( VB.net, C#, J# …) pour traduire leur langage en langage compréhensible par le système.
* Dès lors on comprend rapidement à quel point cela peut être difficile d’apporter des évolutions dans le framework. D’autant plus que l’évolution de l’informatique tend à faire émerger de nouveaux modèles applicatifs (le Cloud par exemple) alors que certains n’étaient pas couverts (Linux).

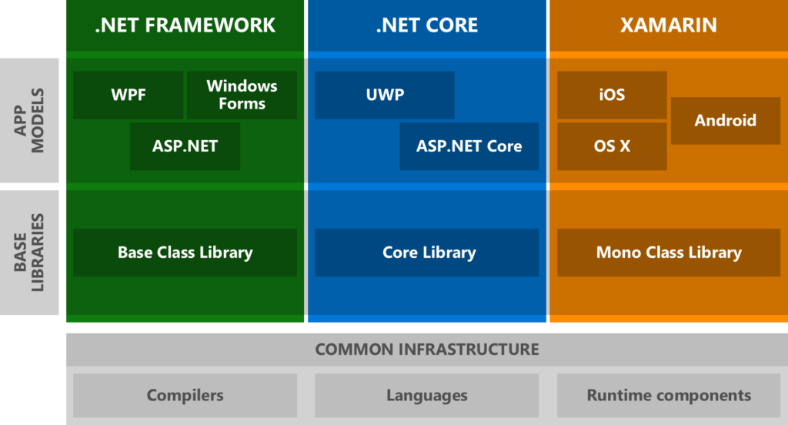
Une image contenant texte

Description générée automatiquement

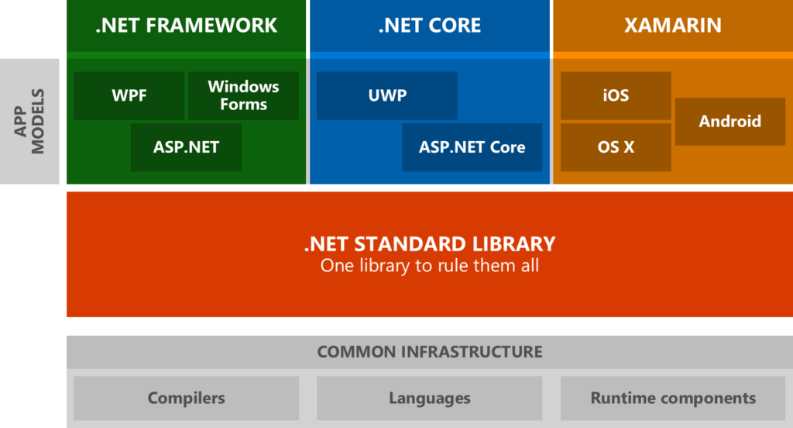
* Ce monde en silo a finalement commencé à bouger après l’arrivée du framework 4.5.1 avec [les annonces](https://blogs.msdn.microsoft.com/dotnet/2014/04/03/the-next-generation-of-net/) du nouveau Jit (RyuJit) et de Roslyn, la nouvelle plateforme de compilation. Le mouvement vers une meilleure unification a pu ainsi commencer. C’est d’ailleurs peu après que l’on a commencé à voir [apparaître le terme de .NET Core](https://blogs.msdn.microsoft.com/dotnet/2014/12/04/introducing-net-core/).

**Présentation .NET Core :**

* L’idée est d’abord de pouvoir fournir un écosystème identique quelle que soit la machine qui fait tourner une application. Évidemment cela passe par l’utilisation d’un seul et même compilateur.
* Il est bâti sur des composants nommés Base Class Library (BCL) qui sont le résultat du travail de factorisation nécessaire à ce travail d’homogénéisation
  + - * + En plus de cette construction, il a été choisi de rendre disponible l’intégralité des composants à travers NuGet. Désormais le scénario de déploiement est différent, puisque les packages sont inclus lors de la phase de développement. Plus besoin de se poser la question de la version du framework disponible sur les machines clientes.



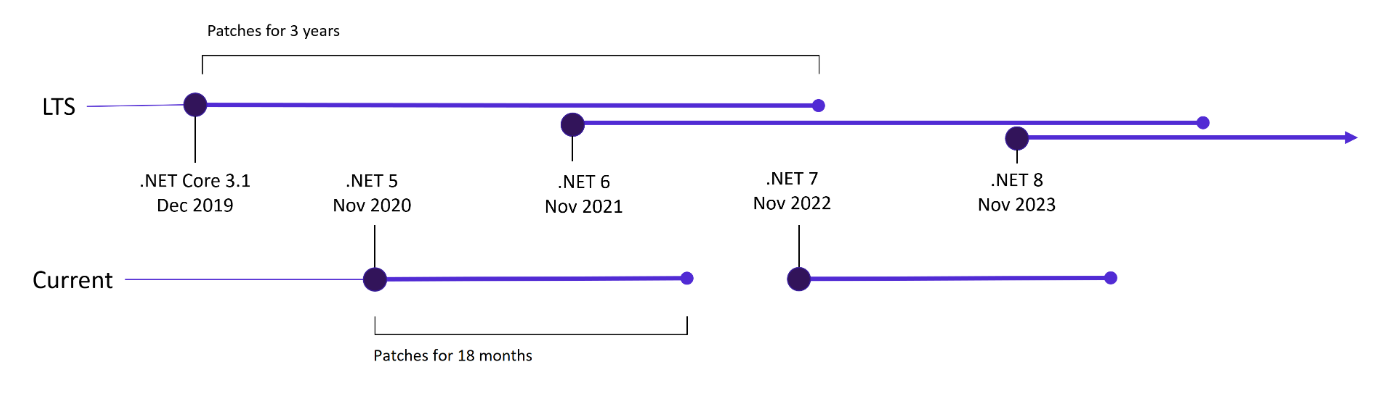
**Présentation .NET Standard ou .Net :**

* Le .NET Standard a poussé encore plus loin le raisonnement précédent. Dans les trois silos présentés ci-dessus, on note que chacun dispose de sa propre librairie de base (BCL pour le framework, Core library pour le .NET Core et Mono Class library pour Mono). Seulement, de nombreux composants sont communs à toutes ces librairies.
* Le .NET Standard constitue la nouvelle base sur laquelle nous pouvons bâtir les différents éléments constitutifs de nos applications car elle fédère les différents silos au sein d’une même librairie.
* Attention cela ne veut pas dire que WPF fonctionnera sur du linux. ( seuls les silos « core » le peuvent ).

Historique et LTS

Une image contenant texte, intérieur, capture d’écran

Description générée automatiquement



Une image contenant table

Description générée automatiquement

Présentation WPF

Windows Presentation Foundation est une infrastructure d’interface utilisateur qui crée des applications clientes de bureau. La plateforme de développement WPF prend en charge un large éventail de fonctionnalités de développement d’applications, notamment un modèle d’application, des ressources, des contrôles, des graphiques, la disposition, la liaison de données, les documents et la sécurité.

WPF fait partie de .NET. Par conséquent, si vous avez déjà créé des applications avec .NET à l’aide de ASP.NET ou de Windows Forms, l’expérience de programmation doit être familière. WPF utilise le langage XAML (Extensible Application Markup Language) pour fournir un modèle déclaratif utilisé pour la programmation d’applications.

Avantages du WPF

* C'est le plus récent comparé à winform et par conséquent, le plus en phase avec les standards actuels
* Microsoft l'utilise pour beaucoup de nouvelles applications, ex: Visual Studio
* C'est plus flexible, donc vous pouvez faire plus de choses sans à avoir à créer ou acheter de nouveaux contrôles
* Quand vous devez utiliser les contrôles d'un tiers, le développeur de ces contrôles aura de fortes chances de se concentrer sur WPF car ce dernier est plus récent
* XAML rend plus facile la création et l'édition de votre GUI, et permet de travailler séparément entre le designer (XAML) et le programmeur (C#, VB.NET...)
* La liaison de données (data binding), qui vous permet d'avoir une séparation plus nette entre les données (data) et l'interface (layout)
* Utilise l'accélération graphique matérielle (GPU) pour dessiner la GUI, pour de meilleures performances