2.1 Microsoft .NET

Microsoft .NET est un Framework qui supporte le développement logiciel orienté objet en fournissant une approche modulaire d'organiser les fichiers (les classes) en composants. Un composant .Net qui s'appelle assembly représente une collection de ressources (fichiers) qui travaillent ensemble et fournirent des services aux applications clientes.

En Anglais, un composant .NET se nomme "assembly". Ce dernier est défini à partir la documentation offerte par Microsoft comme "Assemblies form the fundamental units of deployment, version control, reuse, activation scoping, and security permissions for .NET-based applications. An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality. Assemblies take the form of executable (**.exe**) or dynamic link library (**.dll**) files, and are the building blocks of .NET applications".

2.2 Les composants .NET

Dans le cadre de ce cours, on distingue deux types de composants .Net, à savoir :

2.3 Les composants DLL

Abréviation pour Dynamic Link Library, ce qu'on pourrait traduire par bibliothèque de liens dynamiques, est un type de fichiers qui existe dans le système Windows depuis 1985[1]. Une DLL est tout simplement représente une bibliothèque qui contient du code et de données pour assurer le bon fonctionnement des autres applications sur Windows.

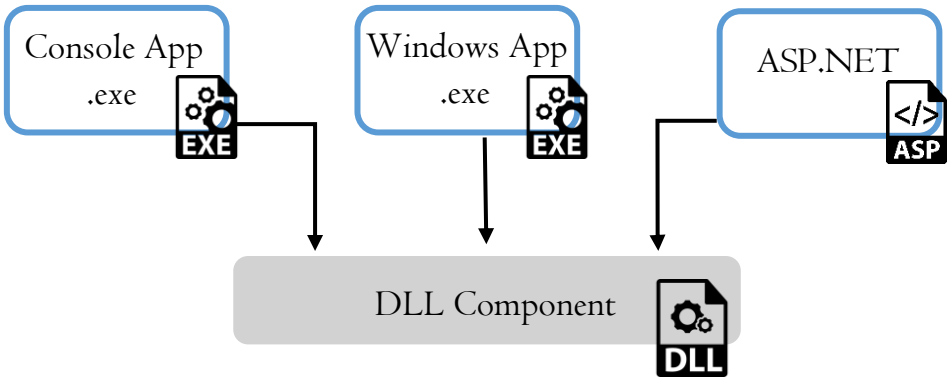
Des répertoires tel que c:\windows\system32 incluent plusieurs DLLs qui offrent des services aux programmes Windows afin de fonctionner correctement dans la platform Windows.

Comme tous les composants logiciels, une DLL est une entité qui, seul, ne sert à rien, mais qui permet aux applications de fonctionner. On peut trouver une DLL intégrée dans plusieurs extensions à savoir les boites de dialogue, les widgets,

[1] https://fr.wikipedia.org/wiki/Dynamic_Link_Library

les correcteurs orthographique, les générateurs de polices de caractères, les drivers, etc.

L'avantage majeur des DLL est que les applications qui sollicitent leurs services peuvent réduire l'espace mémoire qu'ils occupe sur la mémoire vive. Une seule Dll peut servir simultanément à plusieurs applications en même temps. Par exemple, MS Word et MS Access peuvent partager une collection de DLLs qui assurent le fonctionnement de plusieurs outils dans les rubriques de leurs rubbons telle que les assistants de l'impression, la correction orthographique, la mise en page, etc.



FigI : Réutilisabilité des DLL.

Note

Les composants DLL sont reconnus grâce à leur extension ".dll", et pour la sécurité de Windows, ils ne sont pas visibles dans les explorateurs de fichiers.

2.4 Les composants EXE

C'est un autre type de composant offerts par la platform .NET. Ce composant n'est rien d'autre qu'un fichier qui porte un ensemble d'instructions et même des éléments nécessaires à l'exécution d'un programme.
De tel composant possède généralement une extension de fichier se terminant par ".exe" à l'instar de l'extension ".dll" des composants DLL.

Dans le système Windows, tous les fichiers compilés ont l'extension ".exe" est sont appelés "executable".

Notes

- Un fichier ".exe" ne s'exécute que sur une plateforme MS Windows.
- Les scripts qui sont des fichiers no-compilés (ex : les fichiers Batch) sont souvent considérés en tant que des exécutables.
- Les composants ".exe" sont dits " Out-of-process components" dont ils ont un espace mémoire réservé pour assurer leurs exécutions, au contraire des composants ".dll" qui sont dits "In-process" qui partagent le même espace mémoire avec les composants ".exe" qui sollicitent leurs services.

2.5 MS Visual Studio

C'est un IDE (pour Integrated Development Environment) complet qui propose une collection d'outils aux développeurs pour concevoir, déboguer, tester et déployer leurs applications. La version actuelle de cet IDE s'appelle Visual Studio 2019 qui est gratuite pour les étudiants, les contributeurs open source et les particuliers.
Grâce à Visual Studio, on peut créer différents types d'applications, à savoir les applications Bureau, web (avec ASP.NET), bureautiques, mobiles, etc.
Il est considéré comme un éditeur de code multiplateformes qui support une large gamme de langage de programmation, à savoir le langage C, C++, C#, Python, JavaScript, TypeScript, Visual Basic, HTML, CSS, F#, pour n'en nommer que quelques-uns.

Il existe trois éditions pour Visual Studio : Community, Professionnel et Enterprise, dont on a opté pour la version gratuite de l'édition Community 2019.

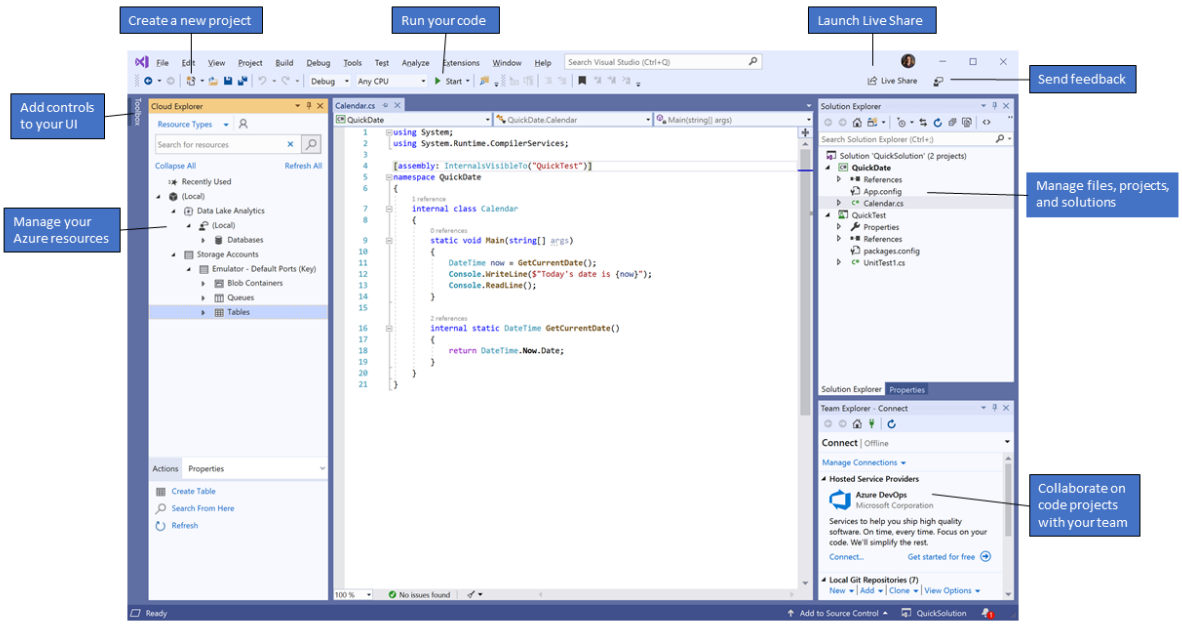


Fig2 : L'IDE Visual Studio 2019

2.6 TP 01

Dans cet exemple, on va créer des composants par la line de commande en utilisant le langage C#.

Les éléments de cet exemple sont les suivants :

- MathLibrary.DLL : le composant DLL qui joue le rôle d'une bibliothèque de méthodes invocables par des composants EXE. Nous avons défini deux méthodes qui sont Add et Mult.
- Add : retourne la somme de deux entiers. Cette méthode existe dans une class qui se nomme AddClass.cs.
- Mult : retourne le produit scalaire de deux entiers. Cette méthode existe dans une classe qui se nomme MultClass.cs.
- Test : c'est la classe qui va nous donne le composant EXE. Celle-ci contient la méthode Main () fait appel aux deux méthodes précédentes.

```
namespace UtilityMethods
{
    public class AddClass
    {
        public static long Add(long i, long j)
        {
         return (i + j);
        }
    }
}
```

Fig3 : AddClass.cs

```
namespace UtilityMethods
    {
        public class MultiplyClass
          {
             public static long Multiply(long x, long y)
              {
                return (x * y);
              }
          }
     }
```

Fig4 : MultiplyClass.cs

```
using UtilityMethods;
 class Test
  {
    static void Main(string[] args)
    {
     System.Console.WriteLine("Calling methods from MathLibrary.DLL:");
      if (args.Length != 2)
       {
         System.Console.WriteLine("Usage: TestCode <num1> <num2>");
         return;
       }
      long num1 = long.Parse(args[0]);
      long num2 = long.Parse(args[1]);
      long sum = AddClass.Add(num1, num2);
      long product = MultiplyClass.Multiply(num1, num2);
      System.Console.WriteLine("{0} + {1} = {2}", num1, num2, sum);
      System.Console.WriteLine("{0} * {1} = {2}", num1, num2, product);
    }
  }

```

Fig4 : MultiplyClass.cs

Pour la compilation, on utilise la commande suivante pour générer le fichier MathLibrary.DLL :

```
>csc /target:library /out:MathLibrary.DLL AddClass.cs Mult.cs
```

Pour le fichier Test.EXE, on utilise également la commande suivante :

```
>csc /out:Test.exe /reference:MathLibrary.DLL Test.cs
```

Note

Le chemin qui nous amène au compilateur CLR doit être ajouté à la variable d'environnement pour que les commandes précédentes soient reconnues.

2.7 TP 02

Dans cet exemple, on va utiliser Visual Studio pour implémenter un scénario simple dans lequel un composant EXE invoque une méthode qui affiche "Hello World" définie dans un composant DLL.

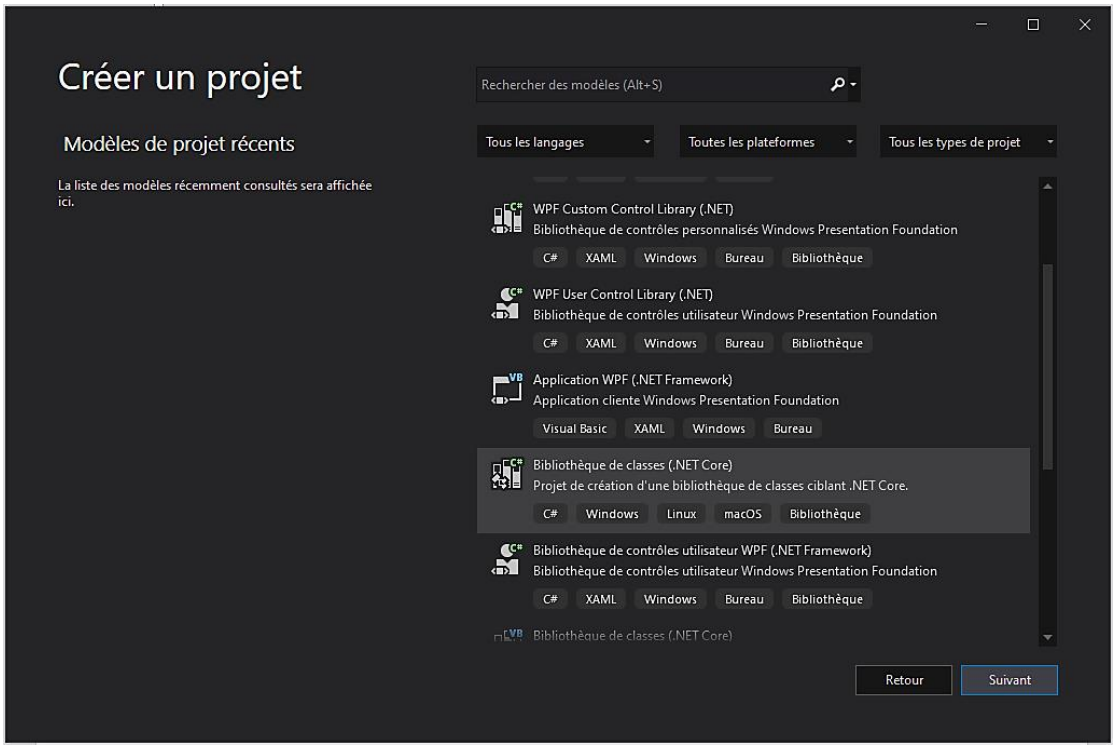


Fig5 : Le choix d'un projet Bibliothèque de classes.

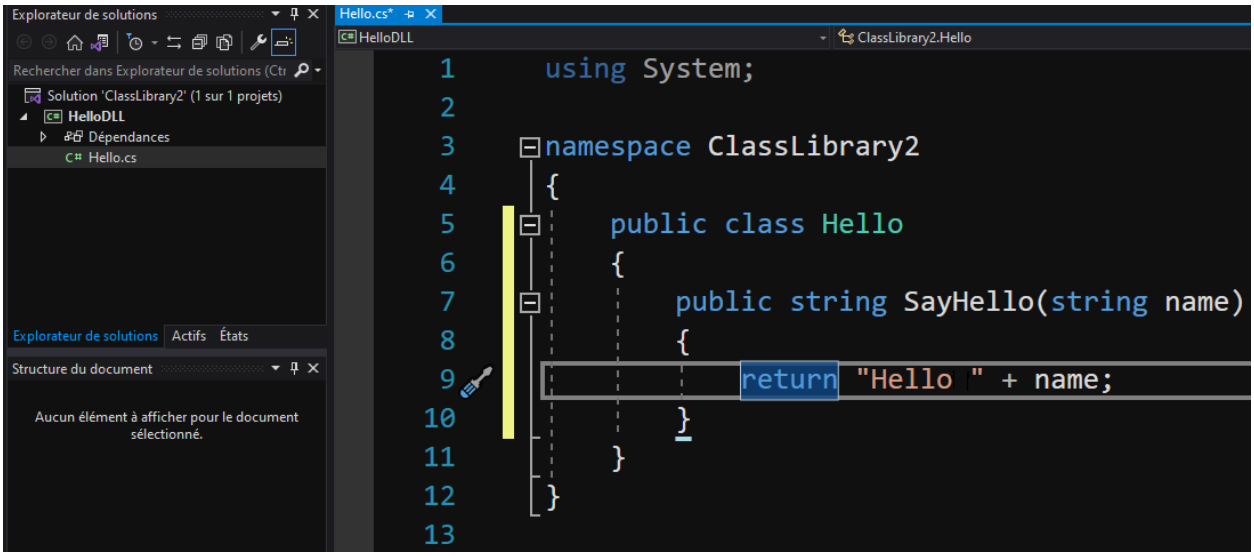


Fig6 : La création d'un composant DLL.

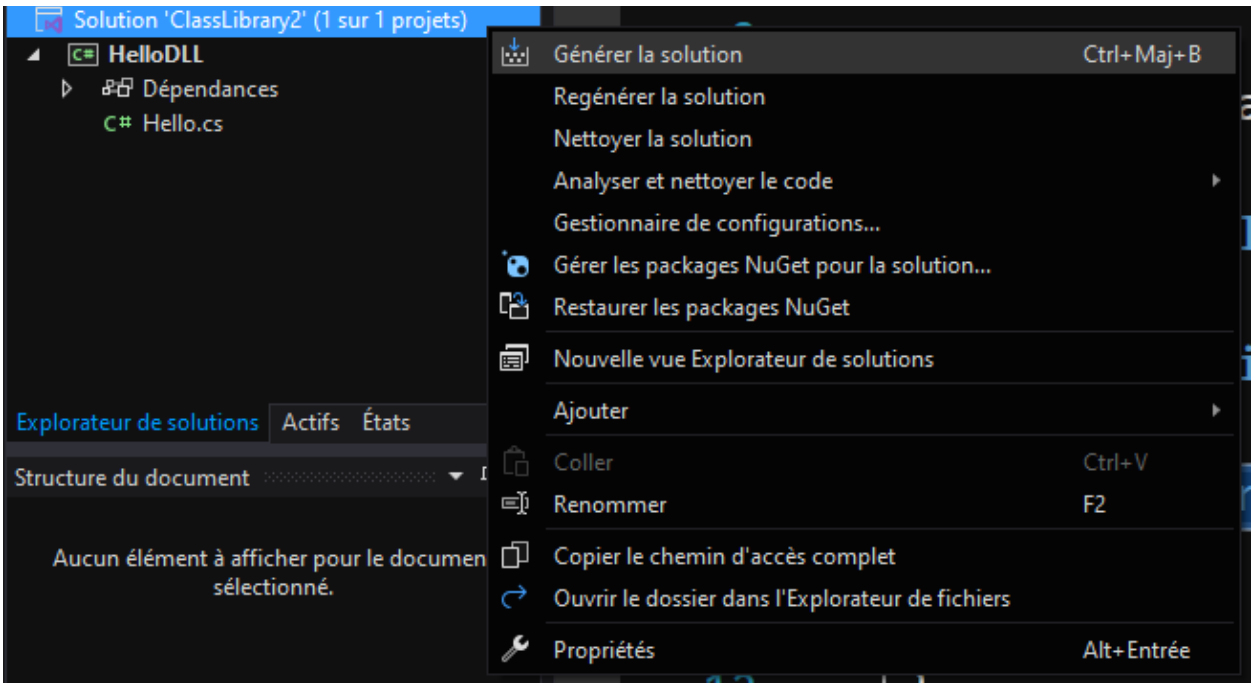


Fig7 : La génération du fichier HelloDLL.dll.

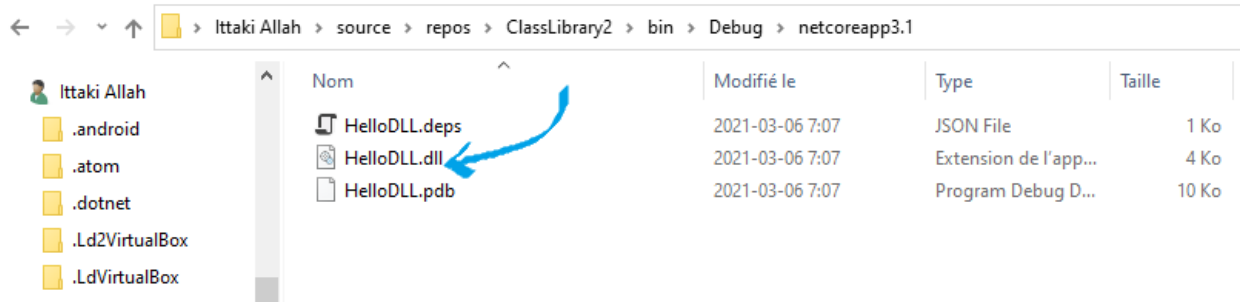


Fig8 : L'accès au fichier HelloDLL.dll.

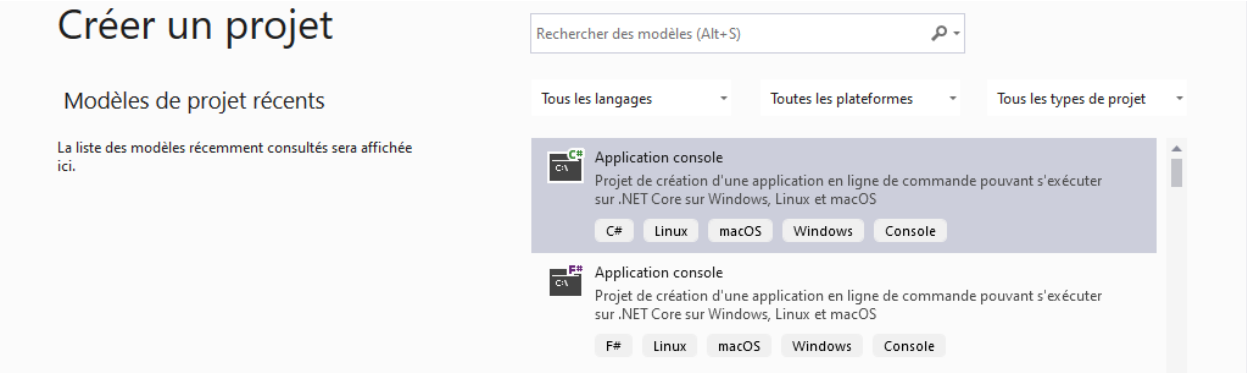


Fig9 : On crée un composant EXE en choisissant une application console.

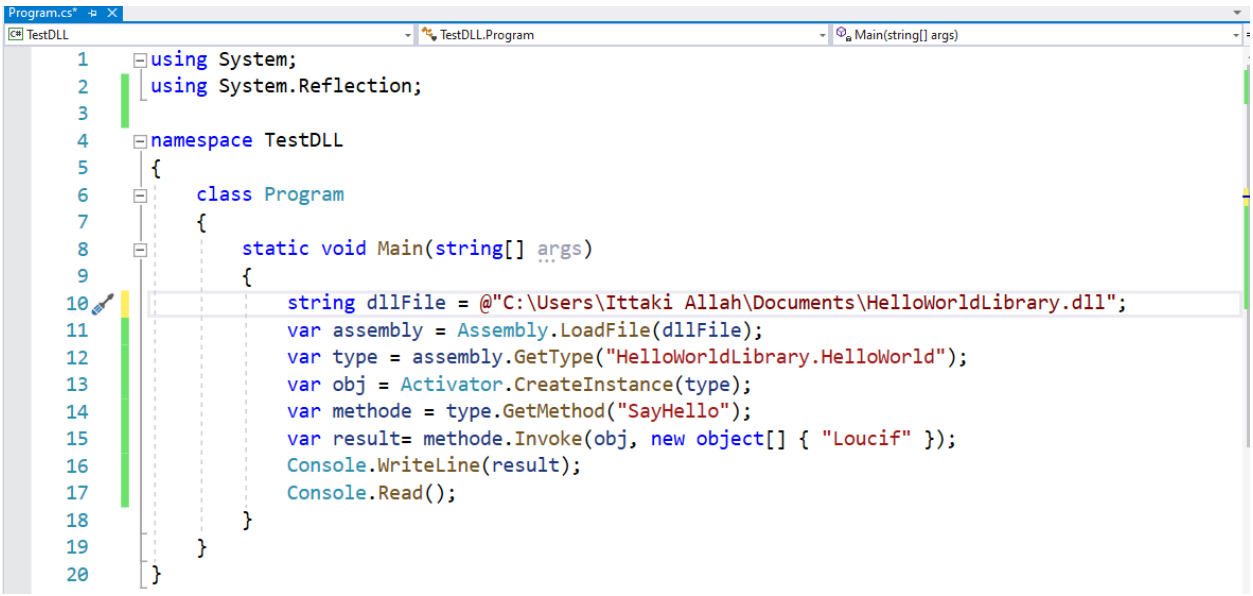


Fig10 : Le code du composant EXE en C#.