SL228_POBJ

Programmation Objet

Visual C# 2015

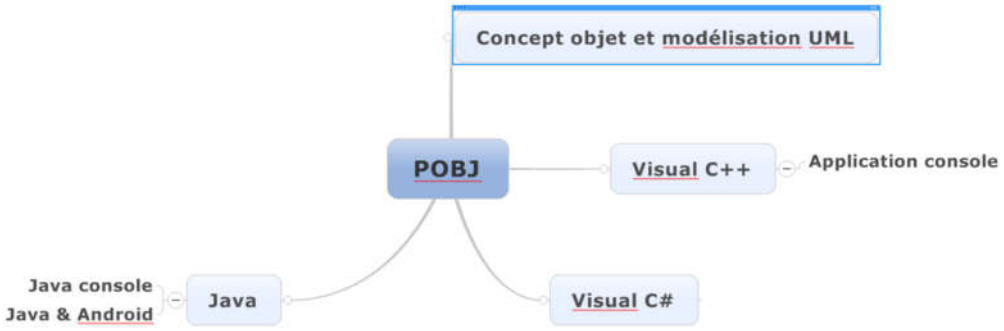
Version 1.0 2016 Ch. Huber
Version 1.1 2017 Ph. Bovey

École des Métiers Lausanne, Ecole Supérieure

CONTENU DU COURS C#

1. Introduction
2. Différences avec le C++
3. Contrôles, propriétés et événements
4. Données, opérateurs et instruction de contrôle
5. Fonctions et librairies
6. Gestion du clavier, de l'écran et de la souris
7. Gestion des fichiers
8. Gestion des communications
9. Distribution d'une application

Ensemble des modules SL228_POBJ



1. INTRODUCTION

1.1. AU REVOIR VB BONJOUR C#

Le **VB** (Visual Basic - 1991) était enseigné à L'ES depuis des années. C'est un langage qui permettait/permet de réaliser facilement des interfaces graphiques. Celui-ci avait été retenu lors de la création du cours **POBJ** (Programmation Objet).

Alors pourquoi changer ? Bien que le VB soit un langage **OO** (orienté objet), celui-ci n'en respecte pas tout le principe et de plus en plus d'applications graphiques de nos jours ce code en **C#.**
La syntaxe du C# se rapprochant plus du Java ou **C/C++** que le VB qui est un langage plus verbal.

1.2. C'EST QUOI LE C# ?

Repris du cours "*Apprenez à programmer en C# sur .Net*" de OpenClassrooms.

Il existe des centaines de langages de programmation, comme **le C**, **le C++**, **Python**, **Java**... Chacun a ses avantages et défauts.

Le **C#** (aussi écrit C Sharp et prononcé *"cé sharp"* voire *"ci sharp"* à l'anglaise) :
Alors, qu'est-ce qui caractérise le C# ?

Récent : il a été créé par Microsoft au début des années 2000, ce qui en fait un langage assez jeune (comparé au C, au C++ et à Java).

Idéal sous Windows & Windows Phone 7 : langage recommandé pour développer sous Windows (puisque développer par Microsoft), mais on s'en sert aussi pour créer des applications Windows Phone 7, pour Silverlight, ASP...

Libre : le langage est ouvert, et on peut s'en servir pour développer des applications sous Linux notamment (Mono et DotGnu).

Inspiré de Java : même s'il a ses spécificités, il ressemble globalement plus au Java qu'au C ou au C++ contrairement à ce que son nom pourrait laisser penser.

Associé au framework .NET : un langage seul comme le C# ne permet pas de faire grand-chose. On l'associe en général à une boîte à outils que l'on appelle le framework .NET (aussi écrit Dot NET et prononcé *"dotte nette"*) qui offre toutes les possibilités que l'on peut imaginer : accès réseau, création de fenêtres, appel à une base de données...

Apprécié en entreprise : si Java reste un des langages les plus utilisés en entreprise, C# se présente comme un sérieux challenger. C'est aujourd'hui une compétence recherchée en entreprise.

1.3. VISUAL STUDIO 2015

Pour de nouvelles applications que vous aimeriez développer il est préférable de chaque fois prendre les derniers versions de software. La version Visual Studio **Community** (gratuite) convient bien pour ce que l'on envisage de faire dans ce cours, mais l'école dispose des licences **Enterprise**, c'est cette version que vous trouverez sur vos poste de travail.!

1.4. ÉVOLUTION DU VISUAL STUDIO ET DU .NET FRAMEWORK

Source Wikipedia.

Versions du framework .NET

Version	CLR	Sortie	Livré avec Visual Studio	Préinstallé avec Windows		Sur-ensemble de
				Client	Serveur	
1.0	1.0	13 février 2002	.NET 2002	NC	NC	NC
1.1	1.1	24 avril 2003	.NET 2003	NC	2003	NC
2.0	2.0	7 novembre 2005	2005	NC	2003 R2	NC
3.0	2.0	6 novembre 2006	NC	Vista	2008	2.0
3.5	2.0	19 novembre 2007	2008	7	2008 R2	3.0
4.0	4	12 avril 2010	2010	NC	NC	NC
4.5	4	15 août 2012	2012	8	2012	4.0
4.5.1	4	17 octobre 2013	2013	8.1	2012 R2	4.5
4.5.2	4	5 mai 2014	NC	NC	NC	4.5.1
4.6	4	20 juillet 2015	2015	10	2016	4.5.2

1.5. BREF HISTORIQUE DU C#

Source https://fr.wikipedia.org/wiki/C_sharp#Histoire

Le projet C# a démarré au milieu des années 1990. Microsoft recherchait à obtenir un nouveau langage de programmation dérivé de C/C++. L'entreprise a tout d'abord apporté des améliorations aux langage Java, mais cette voie a été abandonnée lorsque Sun Microsystems - propriétaire du langage Java - a intenté un procès contre eux.

Microsoft se lance alors dans le développement d'une nouvelle plateforme informatique adaptée aux applications web. C'est en septembre 2000 que la plateforme .NET et C# sont présentés au public.
C# devient le langage de facto de cette plateforme, il a par ailleurs été utilisé pour implémenter une partie de la plateforme .NET.

1.5.1. EVOLUTION DES VERSIONS

Année	Version	Bibliothèque	Principal changement
2002	1.0	.NET framework 1.0 et 1.1	
2005	2.0	.NET framework 2.0	généricité ajoutée à C# et au framework
2008	3.0	.NET framework 3.5	LINQ (Language integrated queries)
2010	4.0	.NET framework 4.0	types dynamiques
2012	5.0	.NET framework 4.5	méthodes asynchrones
2015	6.0	.NET framework 4.6	version pour Linux

1.6. PRÉSENTATION DU VISUAL C#

Pour s'adapter à la nature graphique des systèmes d'exploitation Windows d'aujourd'hui, les programmes doivent être capable d'interagir graphiquement avec l'écran, le clavier et la souris.

Le Visual C# se présente comme un environnement complet, qui se compose d'un éditeur graphique (pour les dialogues), d'un éditeur de texte (pour le code C#) et d'un générateur d'application Windows (fournissant le squelette de l'application et générant les appels au fonctions Windows nécessaires à la création des fenêtres, boîte de dialogues, …).

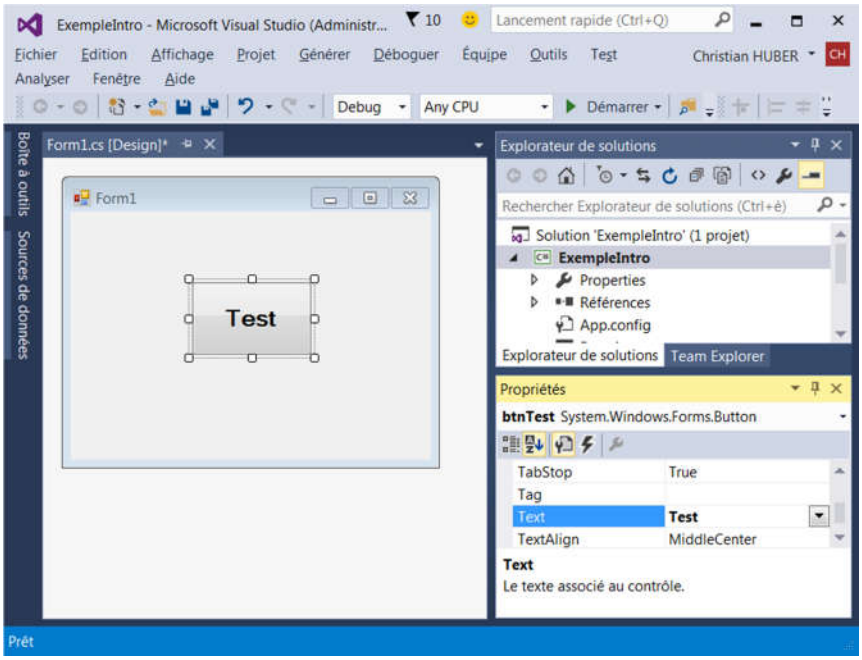
Aux éléments des dialogues comme les boutons, liste déroulantes, zone texte (appelés contrôles) sont liés des événements (click souris, focus, …). Le programmeur définis par une liste de propriétés, le comportement de chaque contrôle.

Si on prend l'exemple d'un bouton dans une boîte de dialogue: sous forme graphique le développeur va définir l'aspect graphique (dimension, disposition, etc.), et par la liste de propriétés le développeur va agir sur les aspects de détails, ajouter un texte et établir une relation entre les événements liés à ce bouton et l'action à accomplir, etc.

1.7. EXEMPLE SIMPLE - BOUTON DANS UNE FENÊTRE

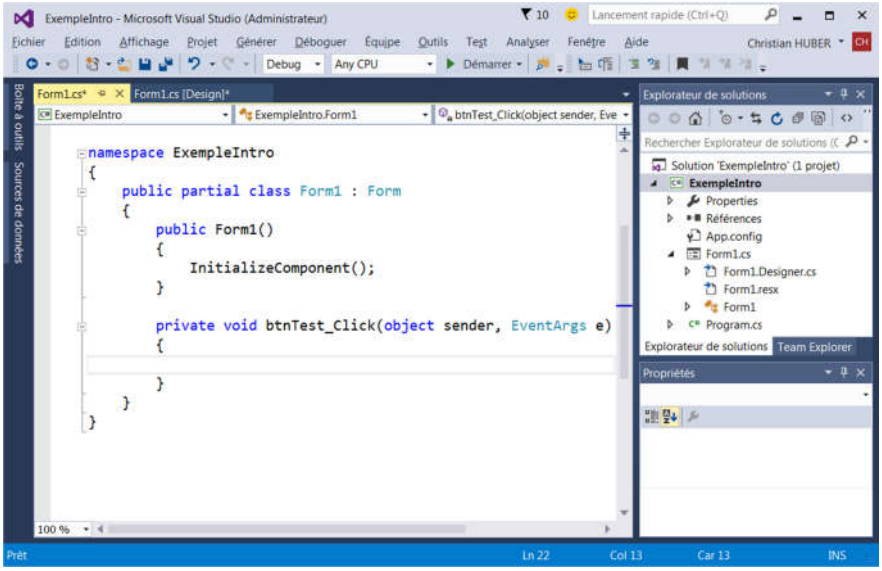
Ci-dessous, l'illustration d'un bouton qui inséré dans une feuille (FORM).

Il y a la possibilité de changer directement le nom de variable bouton **(Name -> btnTest),** ainsi que le texte à afficher (**Text -> "Test"**)



☺ Un double clic sur le bouton nous ouvre la fenêtre avec le code généré automatiquement.

Page de code de l'exemple :

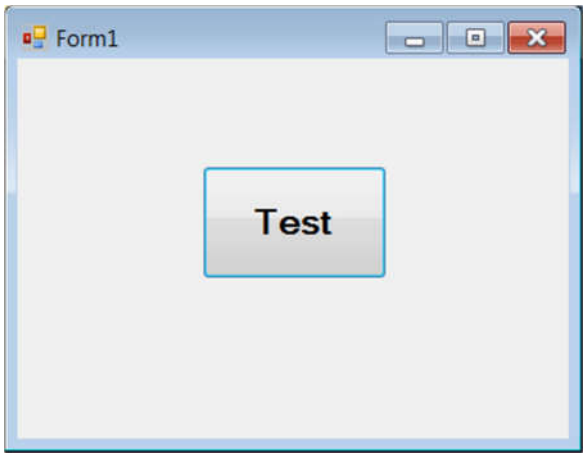


On constate que le formulaire Form1 est une classe qui dérive de Form.
L'action à accomplir doit être insérée à l'intérieur de la méthode "**btnTest_Click**".

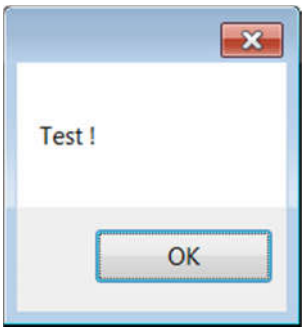
Pour afficher un message il est nécessaire d'ajouter le code suivant (en gras) :

```
private void btnTest_Click(object sender, EventArgs e)
{
      MessageBox.Show("Test !");
}
```

Lors de l'exécution on obtiendra le résultat suivant :



Avec la petite boîte correspondant au message affichée lors de l'activation du bouton **Test**.



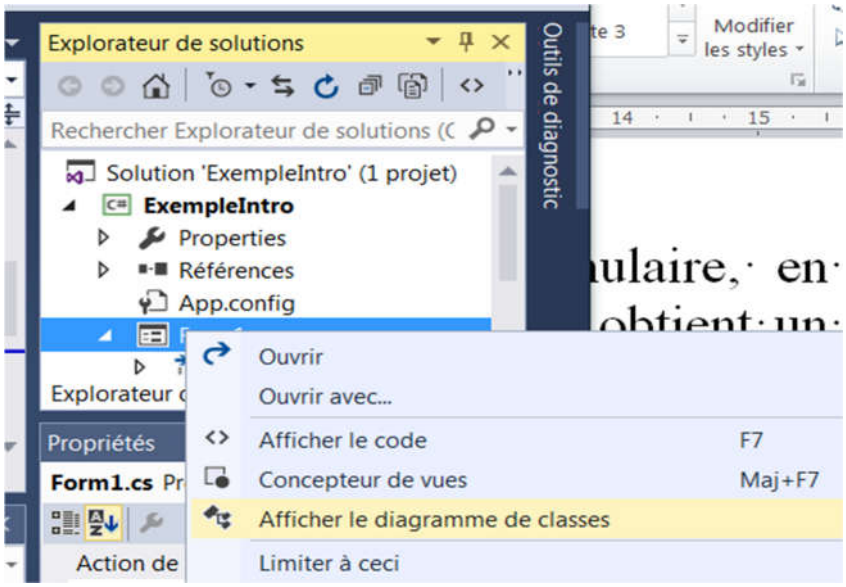
1.8. ASPECTS LANGAGE ORIENTÉ OBJET DU C#

Sans entrez trop dans le détail, tous les éléments que l'on utilise dans le C#, sont des objets. Un objet est défini par sa classe.

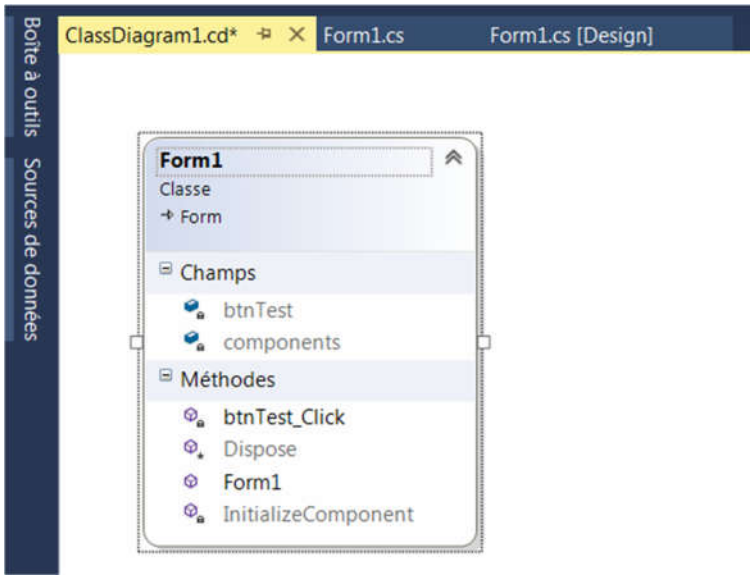
Lorsque l'on place des objets (appelé **control**, comme les boutons, TextBox, …) sur le formulaire, cela correspond à ajouter des membres à la classe correspondant au formulaire. Cet ajout est automatique et ne se voit pas au niveau du code, seul les méthodes que l'on ajoute (automatiquement par le mécanisme des événements ou manuellement) sont visible dans le code.

1.8.1. VUE "DIAGRAMME DE CLASSE"

Il est possible d'obtenir le diagramme de classe correspondant au formulaire, en sélectionnant **Form1.cs** dans l'explorateur de solution. Avec un clic-droit on obtient un menu comportant l'élément "Afficher le diagramme de classe".



On obtient :



1.8.2. QUE POSSÈDE UN OBJET C#

Les objets C# comme les contrôles possèdent :

- **Des propriétés (champs ou attributs).**
- **Des méthodes.**
- **Des évènements.**

Ce qui veut dire qu'un objet est quelque chose de complexe, encore plus complexe qu'une structure en C (une structure en C définit les propriétés de l'objet).

Pour accéder à une propriété (champ) on utilise la syntaxe:

<Nom Objet>.<Nom propriété>

Exemple pour le texte d'un bouton :

```
BtnTest.Text = "Nouveau texte";
```

Pour accéder à une méthode on utilise la syntaxe:

<Nom Objet>.<Nom méthode> (Paramètres de la méthode);

Exemple pour que le bouton devienne invisible :

```
BtnTest.Hide();
```

Pour les événements, c'est le système qui fournit une méthode qui réagit à l'événement.

Par exemple :

```
private void btnTest_Click(object sender, EventArgs e)
{

}
```

Le lien avec la source de l'événement est `object sender`, on dispose aussi d'information sur l'événement avec `EventArgs e`.

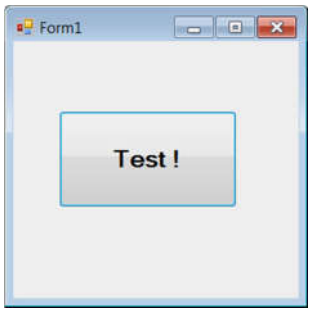
1.8.3. COMPLÉMENT DE L'EXEMPLE

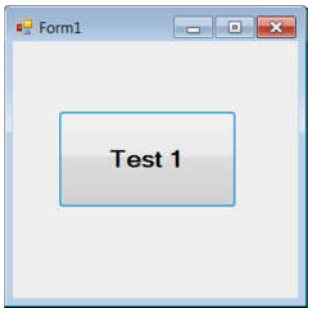
Pour illustrer les aspects pratiques, nous allons introduire une variable globale servant de compteur à chaque click sur le bouton. En plus nous afficherons sur le bouton lui-même la valeur du compteur.

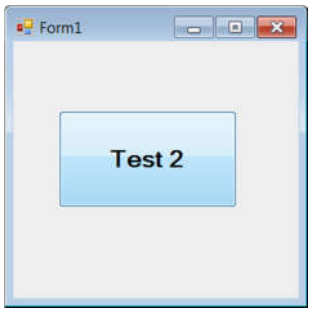
Le code devient (après suppression de l'appel au MessageBox) :

```
gram1.cd   Form1.cs*   Form1.cs [Design]*
pleIntro   ExempleIntro.Form1   btnTest_Click(object sender,
namespace ExempleIntro
{
    public partial class Form1 : Form
    {
        int Count = 0;

        public Form1()
        {
            InitializeComponent();
        }

        private void btnTest_Click(object sender, EventArgs e)
        {
            string Mess;

            Count++;
            Mess = "Test " + Count.ToString();
            btnTest.Text = Mess;
        }
    }
}
```

Lors de l'exécution, après 2 clicks on obtient :



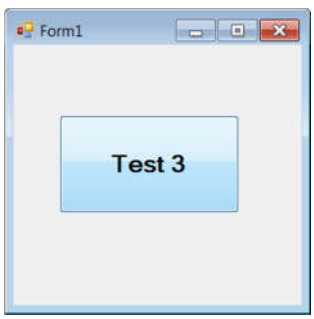


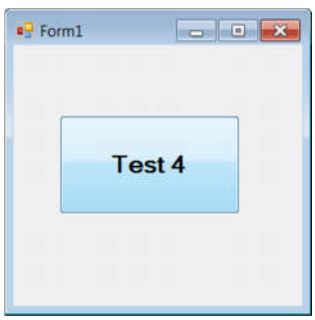


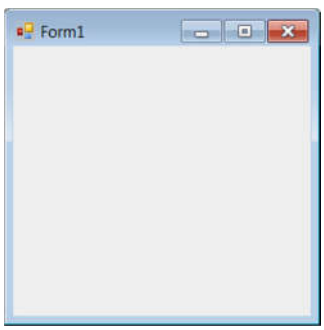
Si on ajoute encore le code suivant :

```
if ( Count == 5 ) {
     btnTest.Hide();
}
```

Lors de l'exécution, on obtient finalement :



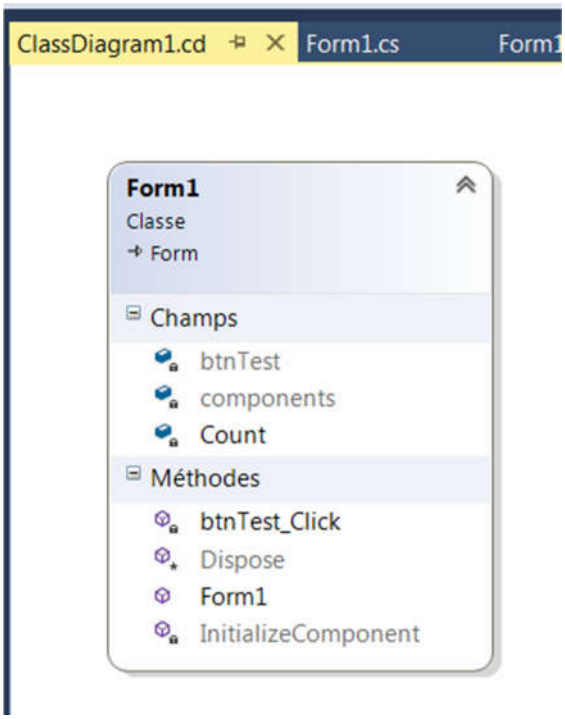




Le bouton est invisible et inactif !

1.8.4. EFFET DU COMPLÉMENT SUR LE DIAGRAMME DE CLASSE

Si on ouvre à nouveau le diagramme de classe, on constate que la variable globale **Count** apparait comme un nouveau champ dans la classe Form1.

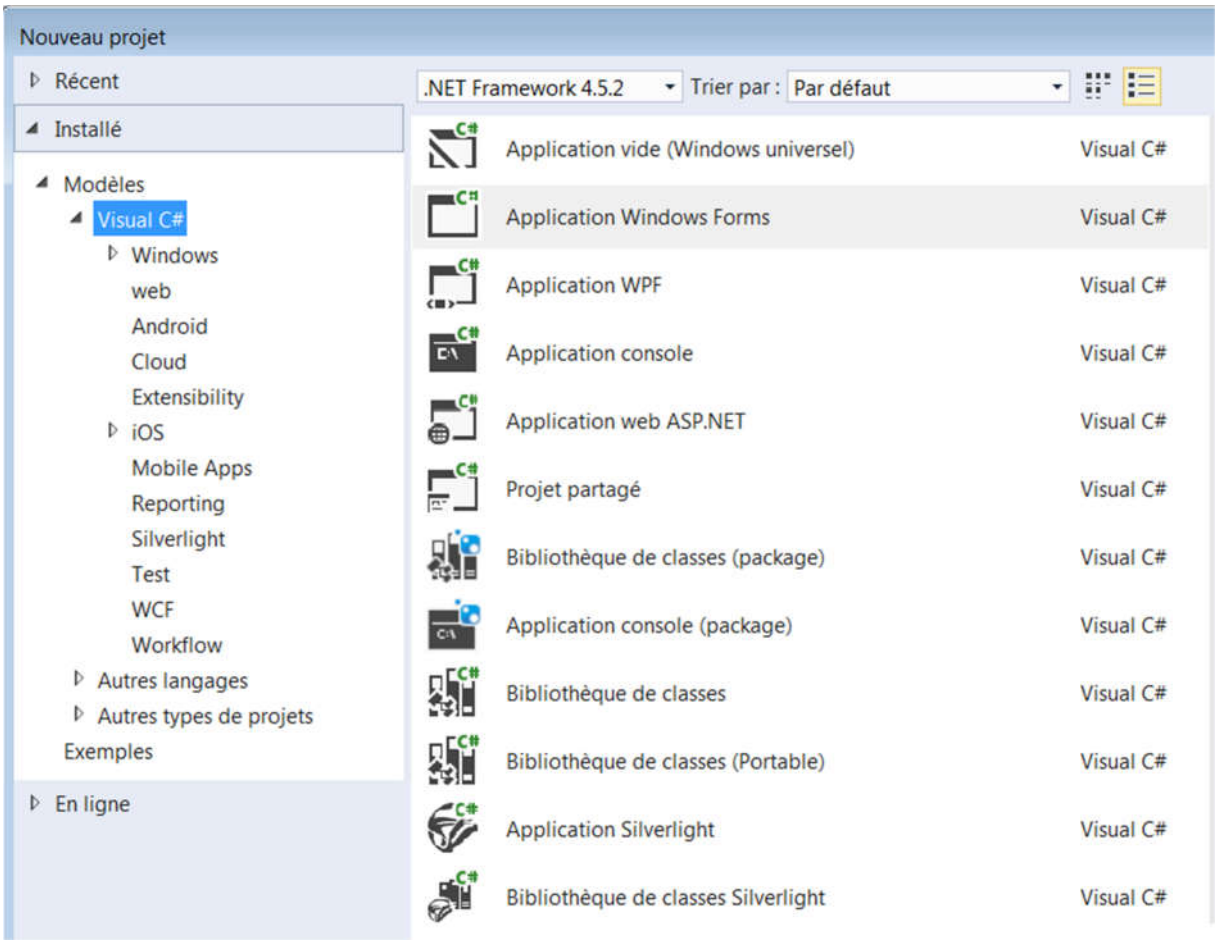


1.9. QUE PEUT-ON CRÉER AVEC C#

Voici la réponse d'Open Classrooms :

Vous savez quoi ? Avec le C# on peut créer autre chose que des applications consoles !! Dingue non ? Des applications avec des boutons et des menus, ou des sites web et même des jeux. Ces bonnes nouvelles sonnent la fin de nos applications console toutes noires et toutes tristes… mais grâce à ça, c'est le début d'une toute nouvelle aventure.

Lors de la création d'un nouveau projet la liste des application possible est relativement large.



La forme la plus ancienne est l'application **Windows Forms**, c'est celle que nous utiliserons dans le cadre du cours.

Il y a aussi les applications WPF (Application cliente Windows presentation Foundation).

Il y a aussi les applications ASP.NET et bien d'autre encore.

1.10. HISTORIQUE DES VERSIONS

1.10.1. V 1.0 FÉVRIER 2016

Création du chapitre en transformant l'ancien chapitre VB correspondant.

1.10.2. V 1.1 FÉVRIER 2017

Annonce de la nouvelle version et de la reprise par Ph. Bovey.