

Premiers pas : l'environnement de développement Visual Studio.NET



BIENVENUE



**Vous allez entrer dans un monde étrange et merveilleux
... celui de la POO ...**

La quoi???

POO...c'est-à-dire Programmation Orientée Objet...

**Comprendo Non, ne pars pas en courant ce n'est pas
un gros mot ni une menace ... mais tu as 2 choix (coche
la bonne case) :**

☐ Je suis partant pour l'aventure ... oupi ! (je continue mon cours)

☐ Ca va pas non, plutôt fuir ... j'éteins mon PC et je quitte la salle en courant (non c'est une blague !!)



Les applications Windows sont des **programmes directement exécutables** qui **utilisent des fenêtres Windows**: des programmes de traitement de texte, d'image, de musique, des jeux, de petits utilitaires, des logiciels métiers (médicaux)...



Bon jusque là ça va !!! Mais un Objet, c'est quoi ? parce que un programme je sais ce que c'est mais orienté objet !!!????

Les objets

A- Dans la vie courante:

Mon gâteau est un **objet**, cet objet existe.



Pour créer un gâteau, j'utilise un 'moule à gâteau', ce moule a les caractéristiques du gâteau (forme, hauteur..), mais je ne peux pas manger le moule!! Le moule se nomme la **Classe**.

Avec le moule, je crée un ou plusieurs gâteaux.

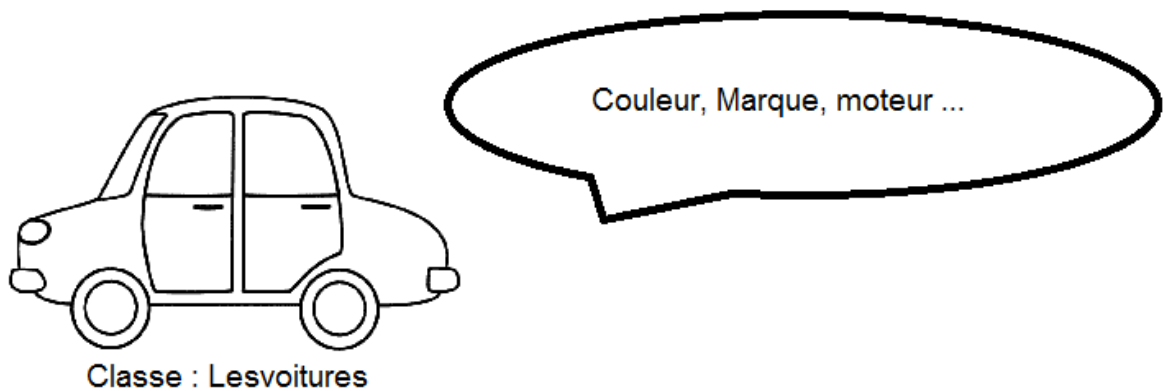


Autre exemple:

Ma voiture est un **objet**, cet objet existe, on peut l'utiliser...en tout cas moi je peux !



Ma voiture fait partie de "Les voitures", du type, du genre "Les voitures". "Les voitures" c'est une classe (**Class**) qui a ses caractéristiques : "Les voitures" ont une couleur, un moteur..., elles roulent en transportant des passagers...

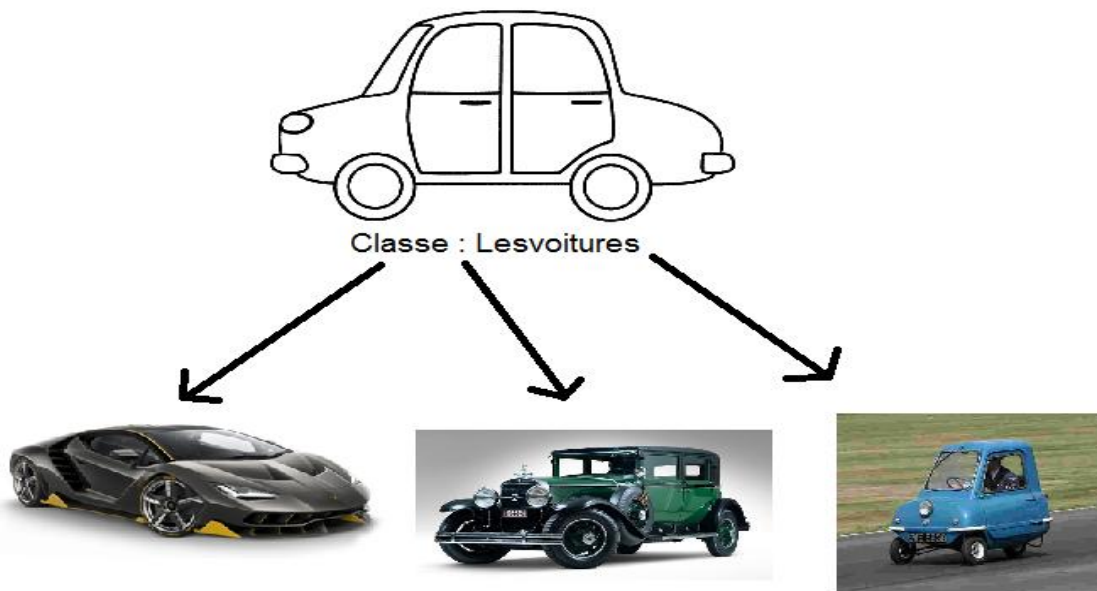


...mais je ne peux pas utiliser "Les voitures", la Classe;

Pour me déplacer, il faut avoir un objet "voiture".



Avec la Classe je vais créer des Objets :



A vous de choisir laquelle !!!



De manière générale, une **classe est une représentation abstraite** de quelque chose, tandis qu'un **objet est un exemple utilisable** de ce que représente la classe.

Pour fabriquer ma voiture, je prends les caractéristiques de la **class** "Les voitures" (c'est comme un moule) et je fabrique une voiture, je la nomme 'MaVoiture'.

Dim MaVoiture As New LesVoitures

MaVoiture est maintenant un nouvel objet de **type** 'LesVoitures'.

Un **Objet** est créé selon un 'modèle' qu'on appelle une **Classe**.

On dit aussi qu'il faut **instancier** un objet à partir de la Classe.

'Mavoiture' est une **instance** de la classe 'LesVoitures'. (On dit aussi une '**occurrence**' de la classe)

Remarque: Avec la même classe on peut instancier plusieurs Objets : ici avec la classe LesVoitures, on peut instancier 3 objets : VoitureSport, VoitureAncienne, VoitureEtudiants...

Bon pour pas vous obliger à choisir je vais choisir MaVoiture comme cela c'est neutre et vous lui mettrez les propriétés (puissance, marque...) que vous voudrez

Les propriétés ????



Propriétés (ou Attributs) :

Prenons MaVoiture.

Elle a des **propriétés** : une marque, une couleur, une puissance...

Pour indiquer la couleur de ma voiture on utilise la notation :

MaVoiture.couleur



Syntaxe : **Objet.Propriété** (Il y a **un point** entre les 2 mots)

Pour modifier la couleur et avoir une voiture verte on écrit :

`MaVoiture.couleur= "Vert"`

Et la voiture devient verte !!



Génial !! mais une voiture ça roule ? non ?

Enfin c'est sensé rouler !!!

Mais "Rouler" c'est une action... "le fait de rouler".

En POO, on appelle ça une méthode.



Pas de manique !!



Méthodes :

`MaVoiture` fait des choses : elle roule par exemple.

Pour faire rouler la voiture j'appelle la **méthode** 'Roule'

`MaVoiture.Roule()`

Syntaxe : `Objet.Méthode` (Il y a **un point** entre les 2 mots)

Si c'est possible pour cette méthode je peux indiquer à quelle vitesse la voiture doit rouler :

`MaVoiture.Roule(100)` j'ai ajouté **un paramètre**.

Le paramètre est un renseignement envoyé avec la méthode.

Il est possible parfois d'indiquer en plus si la voiture doit rouler en marche avant ou en marche arrière.

`MaVoiture.Roule(10, Arriere)`

Evènement :

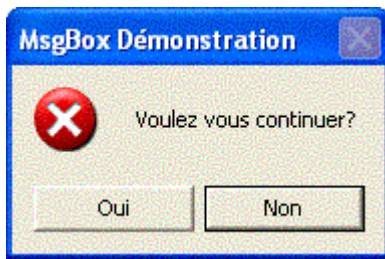
Des évènements peuvent survenir sur un objet.

`MaVoiture_démarre` est un évènement, quand la voiture se met en route (si par exemple j'ai fait `MaVoiture.Roule(10, Arriere)`), cet évènement `démarre` se déclenche automatiquement.

B- Et dans Visual Basic.net:

Une application Windows se compose de **fenêtres** (nommées aussi **formulaires**) dans lesquelles se trouvent des **contrôles** (bouton, liste, texte...)

Exemple de fenêtre avec 2 boutons, une zone de texte (un label) et une icône:



Dans une application Windows, il y a aussi des lignes de code utilisant des variables pour faire des calculs.

En Visual Basic.Net tout est objet :

- les fenêtres (on dit les formulaires),
- les variables,
- les contrôles (les boutons, liste, image, case à cocher..)

...

Il faut un **moule** pour faire un objet. Le moule c'est une **classe**.

Le moule va servir à créer un objet, on dit **une instance**.

On peut créer, **instancier** une multitude d'objets avec le même moule.

Pour créer, démouler un objet, on utilise les mots clé `Dim` et `As New`.

`Dim objet As New Classe`

`New` est un **constructeur**.

Exemple : créer une fenêtre (un formulaire) :

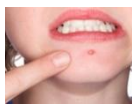
Je dessine une fenêtre `FormDémarrage` (c'est la Classe, le moule)

puis

`Dim UneFenêtre As New FormDémarrage`

Crée une fenêtre qui se nomme ' UneFenêtre ' à partir du moule, du modèle (`FormDémarrage`) que j'ai dessiné.

Autre exemple le bouton :



...mais non pas celui là !! celui-ci :



`Dim Monbutton as New Button`

Créer un bouton nommé ' Monbutton ' avec les attributs habituels des boutons (**Class Button**)

Tout objet a des propriétés.

On utilise la syntaxe : `Objet.Propriété` (Il y a **un point** entre les 2 mots)

Si `UneFenêtre` est une fenêtre, `UneFenêtre.BackColor` indique la couleur de fond de la fenêtre.

S'il y a un bouton, la couleur de fond du bouton sera :

`Bouton.BackColor`

Ou

`UneFenêtre.Bouton.BackColor`

Noter la syntaxe : La couleur du bouton qui est dans la fenêtre `UneFenêtre`

En fait une propriété c'est une sorte de variable.

Comment modifier cette propriété?

`Bouton.BackColor=Color.Red` 'modifie la couleur de fond du bouton



Autre exemple:

La propriété **Visible**: si elle a la valeur `True` (Vraie) l'objet est visible; si elle est à `False` l'objet n'est pas visible.

`Bouton.Visible=False` 'fait disparaître le bouton



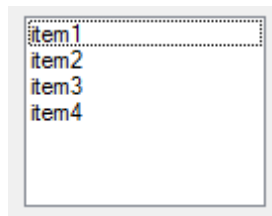
<=Ici il y a un bouton invisible!! oui, oui!! Ben tu ne vois pas qu'il est invisible ?

Les objets ont des méthodes parfois.

Une méthode agit sur l'objet ou fait agir l'objet.

Prenons un exemple simplifié.

Les **Listes** (liste déroulante) ont des lignes (Items) et une méthode **Clear** qui permet de les vider.



Si je veux vider toutes les lignes d'une liste nommé Liste1, je fais:

`Liste1.Items.Clear()`



Les propriétés et méthodes se nomment **les membres** d'un objet.

IMPORTANT !!! ... Certains objets ont des évènements:

Reprenons notre bouton. Quand l'utilisateur click dessus, l'évènement [Bouton_Click](#) survient.

Ce sont les objets **contrôles** (bouton, case à cocher..) et les formulaires qui ont des évènements.

Visibilité:

Quand un objet est créé, il est **visible et utilisable**, uniquement dans la partie du programme où il a été défini.

Par exemple habituellement, je peux voir et modifier la couleur d'un bouton uniquement dans le code de la fenêtre où il est situé.

Pour les variables on parle de **portée**: la variable peut être locale (**Private**) ou de portée générale (**Public**) visible partout.

En résumé :

En Visual Basic.net **tout est objet**.

Les **Classes** sont des types d'objet.

Pour créer (instancier) un objet à partir d'une Classe, il faut utiliser les mots clé [Dim ..As New](#):

[Dim Objet As New Class](#)

Un objet a :

- Des **propriétés**.
- Des **méthodes**.



Attention, par abus de langage, on emploie parfois indifféremment les mots 'Classe' et 'Objet', mais il est préférable de ne pas confondre le modèle et l'objet lui même.

Premier programme: événements

Principes de la programmation VB

Le programmeur va dessiner l'interface utilisateur (fenêtre, bouton, liste..), il va ensuite **uniquement** écrire les actions à effectuer quand certains événements se produisent sur cette interface.

C'est **Visual Basic** qui va entièrement s'occuper de la gestion des événements.

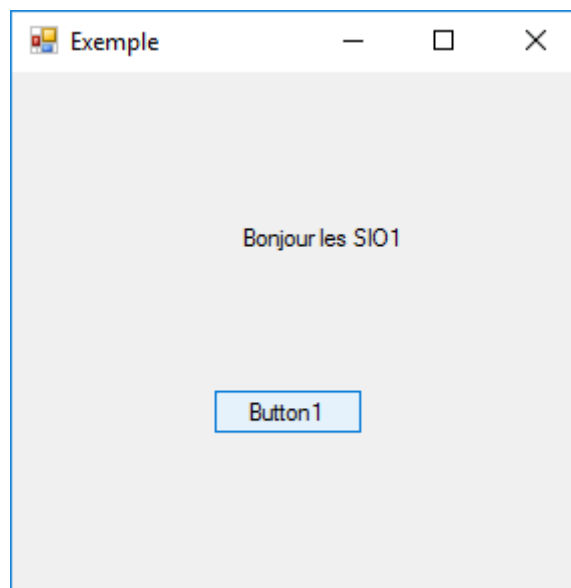
Exemple :le premier programme:

Il affiche 'Bonjour' quand on clique sur un bouton.

C'est pas original: le premier programme, dans tous les cours d'informatique, permet d'afficher 'Bonjour' (ou 'Hello Word').

- **Que voit l'utilisateur du programme?**

L'utilisateur final, celui qui utilise le logiciel, voit une **fenêtre** avec un **bouton**, S'il appuie sur ce bouton il voit s'afficher « **Bonjour** ».



- **Que se passe t-il dans le programme?**

Quand l'utilisateur clique sur le bouton cela déclenche automatiquement **un événement** (**Button1_Click**). Cet événement contient du code qui affiche « Bonjour ».

- **Que doit faire le programmeur pour arriver à ce résultat?**

Pour atteindre ce résultat, **le programmeur va dessiner la fenêtre, le bouton, la zone d'affichage du texte (un label)** puis il va simplement indiquer dans l'évènement Button_Click d' **afficher « Bonjour »**.



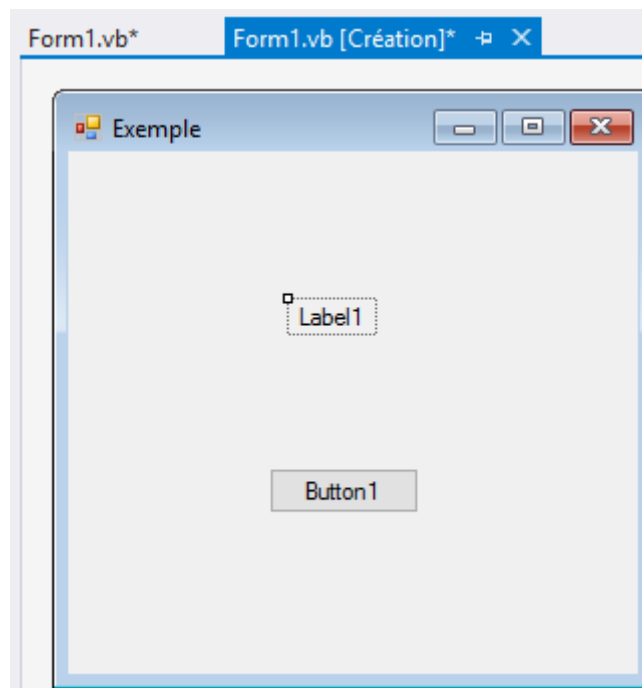
Le fait de déterminer la procédure à appeler ou de réaliser l'appel est entièrement pris en charge par VB.

En pratique, que fait le programmeur ??? : voyons !



A- Il dessine l'interface utilisateur

(Ce que verra l'utilisateur final, c'est l'interface utilisateur : une fenêtre avec des boutons, des listes, du texte..) :



Il ouvre un projet :

Comment ?? ... comme cela :

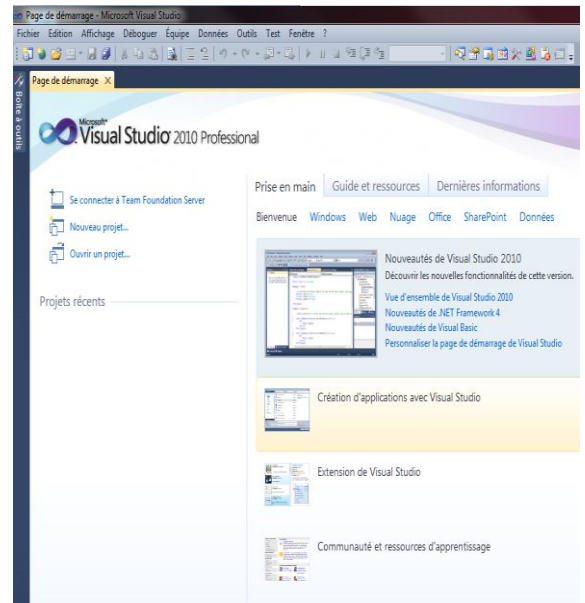
Fenêtre Projet.

Quand on lance VB.net, le logiciel présente une **Page de démarrage** "Start Page" qui permet:

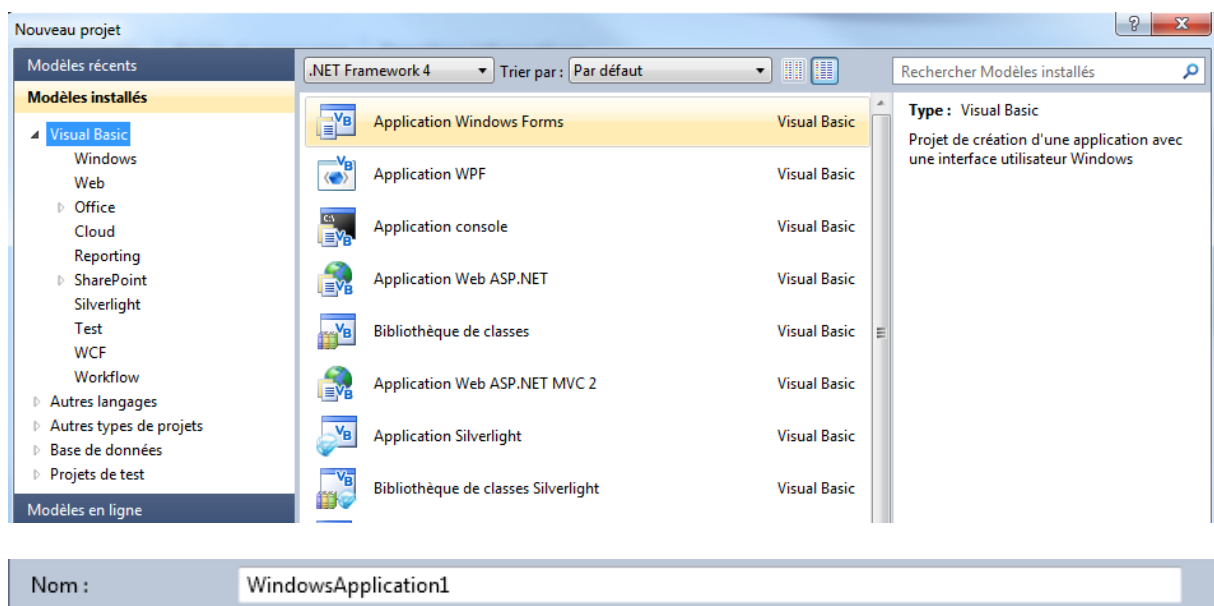
d'ouvrir un projet existant

ou

de créer un nouveau projet



Pour créer un projet Visual Basic normal (WinForms), il faudra choisir 'Nouveau Projet'. La fenêtre suivante s'ouvre:

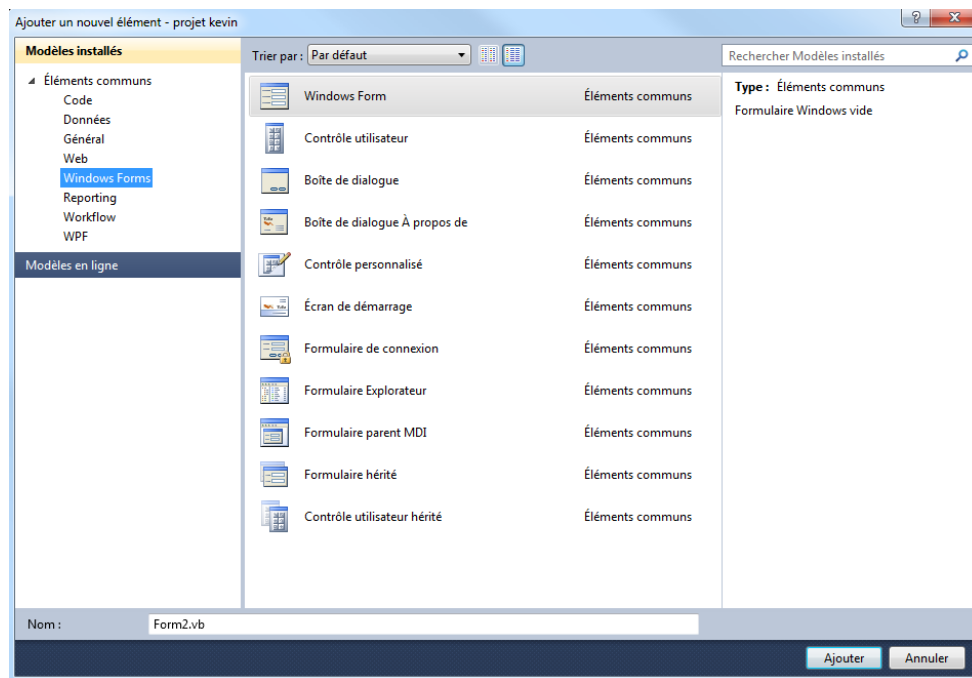


Choisir l'icône '**Application Windows**', puis donner un **nom** au projet(**IMPORTANT**) au lieu de « WindowsApplication1 », enfin valider sur 'Ok'.

(Le **chemin de l'emplacement du projet** n'est pas modifiable ici, il est par défaut 'C:\Users\nom de l'utilisateur\Documents\Visual Studio...\ Projects\MonProjet')

Il crée une fenêtre 'Form1'.

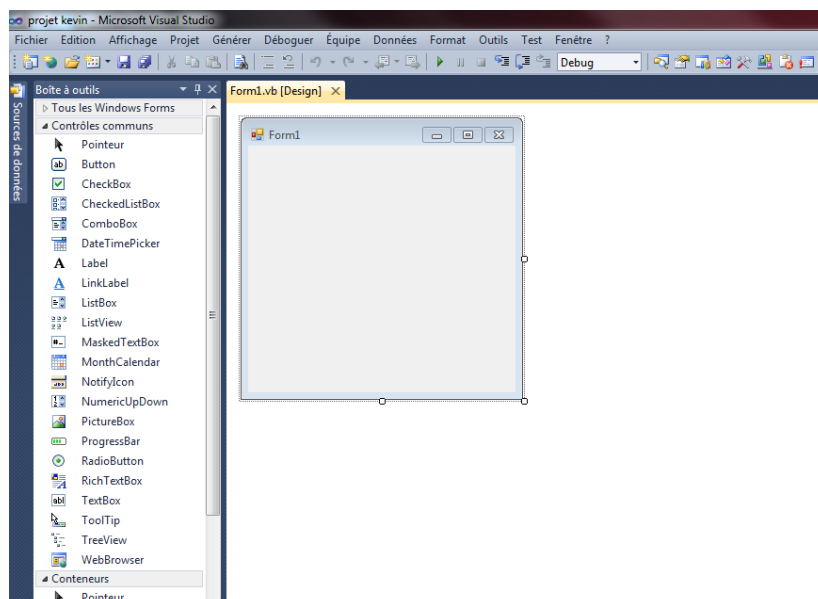
Pour ajouter un fenêtre (un formulaire) Menu **Project**, **Ajouter un formulaire Windows**:



cliquer sur **Windows Form**, une fenêtre (un formulaire) 'Form2' **vide** apparaît ('Form1' était le nom du premier formulaire).

Il y a des fenêtres toutes faites pour accélérer le travail (les templates) comme les 'Ecran de démarrage' les 'Formulaire Explorateur'...

La zone de travail se trouve au centre de l'écran: C'est l'onglet **Form1.vb[Design]** ci-dessous qui donne donc accès au dessin de la feuille (du formulaire); on peut ajouter des contrôles, modifier la taille de ces contrôles..



Il ajoute un bouton :

Pour cela il utilise la Boîte à outils:

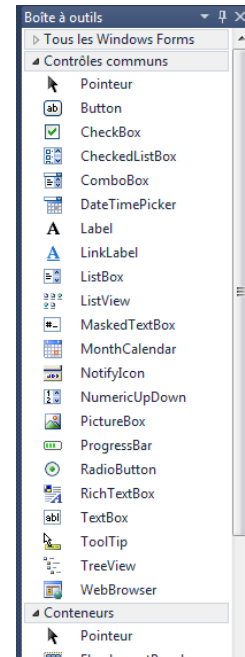
Il clique sur 'Boîte à Outils' à gauche, bouton Windows Forms, puis bouton '**Button**', il clique dans Form1: le dessin d'un bouton apparaît.

🔴* **Attention : dès que vous créez un objet (bouton, label ..) vous devez donner un nom (dans propriété : NAME !)**

Pour l'exemple, Il **ajoute un label**.

Un label est un contrôle qui permet d'afficher un texte.

Comme pour le bouton il clique sur 'Boîte à Outils' à gauche, bouton Windows Forms, bouton 'Label' et met un contrôle label sur la fenêtre.



B- Il va écrire le code correspondant aux événements :

Il double-clique sur le bouton qu'il a dessiné :

Une fenêtre de conception de code s'ouvre et il apparaît :

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
```

```
End Sub
```

Cela correspond à la **procédure événement** en rapport avec l'évènement : 'On a cliqué sur le bouton1'.

Quand le programme fonctionne, quand l'utilisateur du logiciel clique sur le bouton1, le code situé entre Private Sub Button1Click et End Sub est effectué.

Une procédure est un ensemble de lignes de code qui commence par Sub et se termine par End Sub (ou Function..End Function).

Comment indiquer dans cette procédure d'afficher "Bonjour"?

Le label possède une propriété nommée '.text' qui contient le texte à afficher.

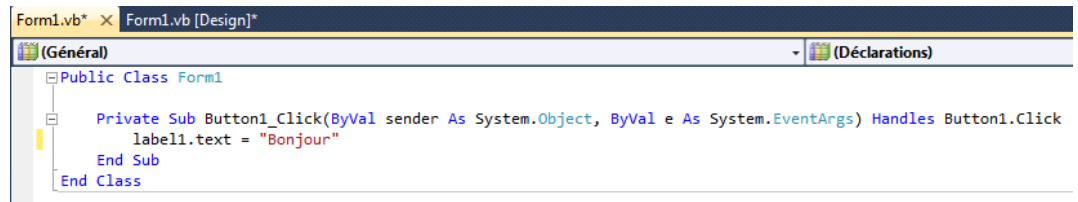
Il faut taper le code qui modifie cette propriété '.text', qui y met la chaîne de caractère "Bonjour":

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
```

```
Label1.Text = "Bonjour"
```

```
End Sub
```

Cela donne:




Voila votre premier programme est écrit.

Comment exécuter ce programme?

Il est possible de tester immédiatement le programme en mode débogage, sans quitter l'environnement de développement:

Utiliser le menu 'Déboguer' puis 'Démarrer' qui lance l'exécution du programme.

On peut aussi taper sur **F5** pour lancer le programme.

Ou grâce à  : **lancer l'exécution avec le premier bouton** (mode 'Run', le second servant à arrêter temporairement l'exécution (mode 'Debug'), le troisième à terminer l'exécution (Retour au mode 'Design' ou 'Conception').

Quand on est en arrêt temporaire en mode 'Debug', la ligne courante, celle qui va être effectuée, est en jaune:

```
For i=0 To 100
Label1.Text=i.ToString
Next i
```

Si on tape la touche **F10** (exécution pas à pas), la ligne 'Label1.Text=i.ToString' est traitée et la position courante passe à la ligne en dessous.

```
For i=0 To 100
Label1.Text=i.ToString
Next i
```


Il y a maintenant le 'Edit and continue': en mode Debug, on peut modifier une ligne et poursuivre le programme qui tiendra compte de la modification (Sauf pour les déclarations). La **sauvegarde** du projet se fait comme dans tous les logiciels en cliquant sur l'icône du paquet de disquettes.

On peut compiler le programme pour créer un exécutable par le menu **Générer** ('Build')

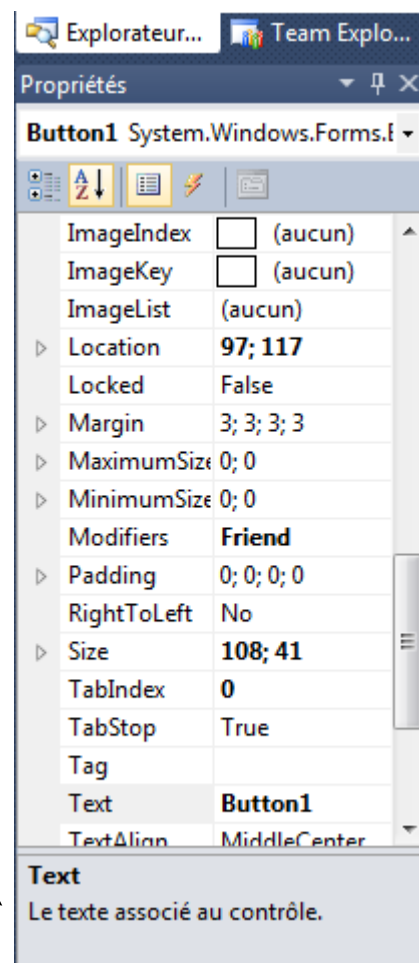
Modifier les propriétés d'un contrôle ou du formulaire.

Quand un formulaire ou un contrôle est sélectionné dans la fenêtre Design, **ses propriétés sont accessibles dans la fenêtre de 'Propriétés' (Properties) à droite en bas:**

Ici ce sont les propriétés du contrôle 'Button1' (Text, Location..)

(on peut modifier directement les valeurs.)

Dessous, il y a une explication succincte de la propriété sélectionnée (Si elle n'apparaît pas, faire sur la propriété: click droit puis dans le menu 'Description').



Ca marche ???

