IV - Environnement de développement : les EDI/IDE

IV-A - IDE Visual Studio 2008 (Microsoft)

C'est l'**I**ntegrated **D**evelopment **E**nvironment (IDE): Environnement de développement intégré de Visual Basic Express 2008 de Microsoft. Il permet de dessiner l'interface (les fenêtres, les boutons, List, Image...) et d'écrire le code VB. Chez nous, on peut aussi dire **EDI** (Environnement de Développement Intégré).

L'IDE de Visual Basic 2008 est identique à celle de VB 2005, bien meilleur que celle de VB 2003 et l'Edition Express' (version légère par rapport à Visual Studio) est GRATUITE. Donc pas d'hésitation, chargez et utilisez VB Express 2008.

Charger sur ce lien VB Express 2008

Pour la version française, dans le cadre bleu 'Visual Basic Edition Express' dérouler la liste et choisir 'French' puis 'Download'.

Vous pouvez voir une vidéo sur l'IDE 2005 (c'est la même que pour la version 2008).

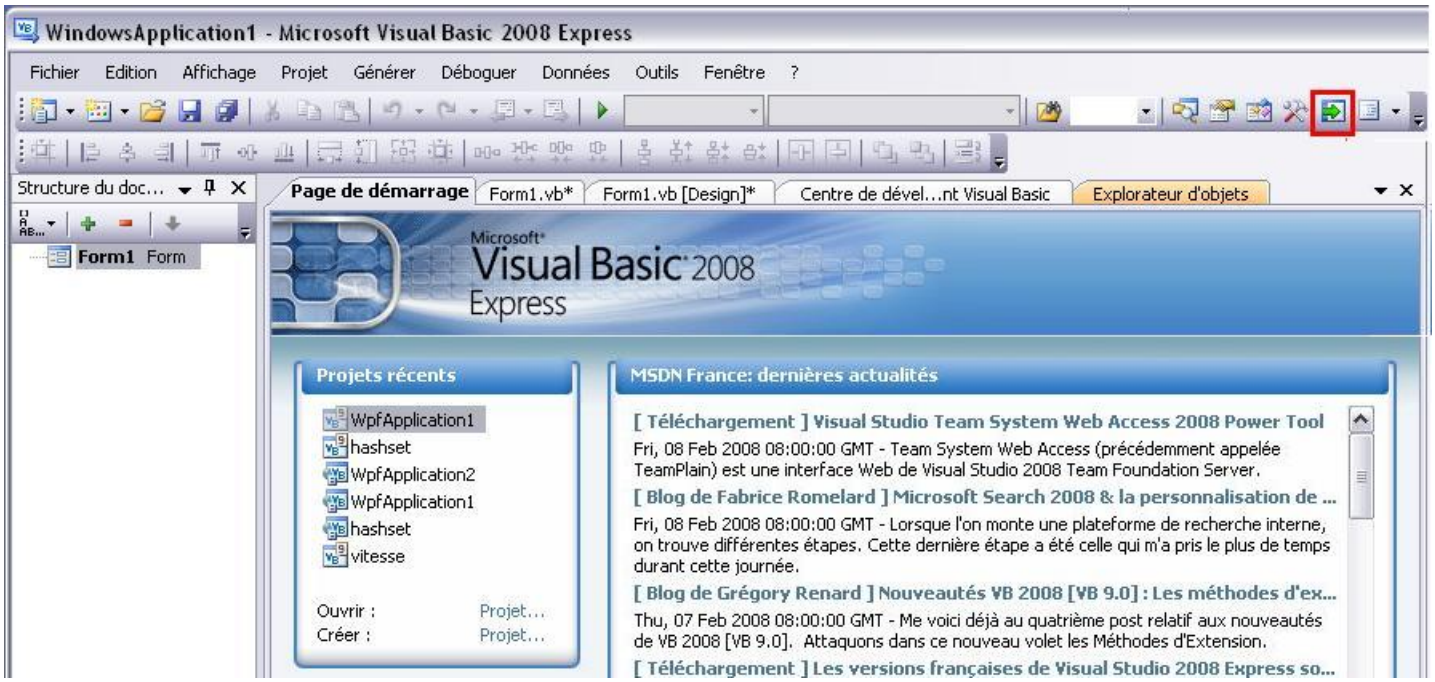

Voir la vidéo : **au format 'Flash'**> ou **au format 'Avi'** en Visual Basic 2005.

(En flash, il y a un arrêt au milieu: patientez. En Avi ne pas tenir compte des avertissements qui déclarent que le fichier n'est pas valide).

Fenêtre Projet.

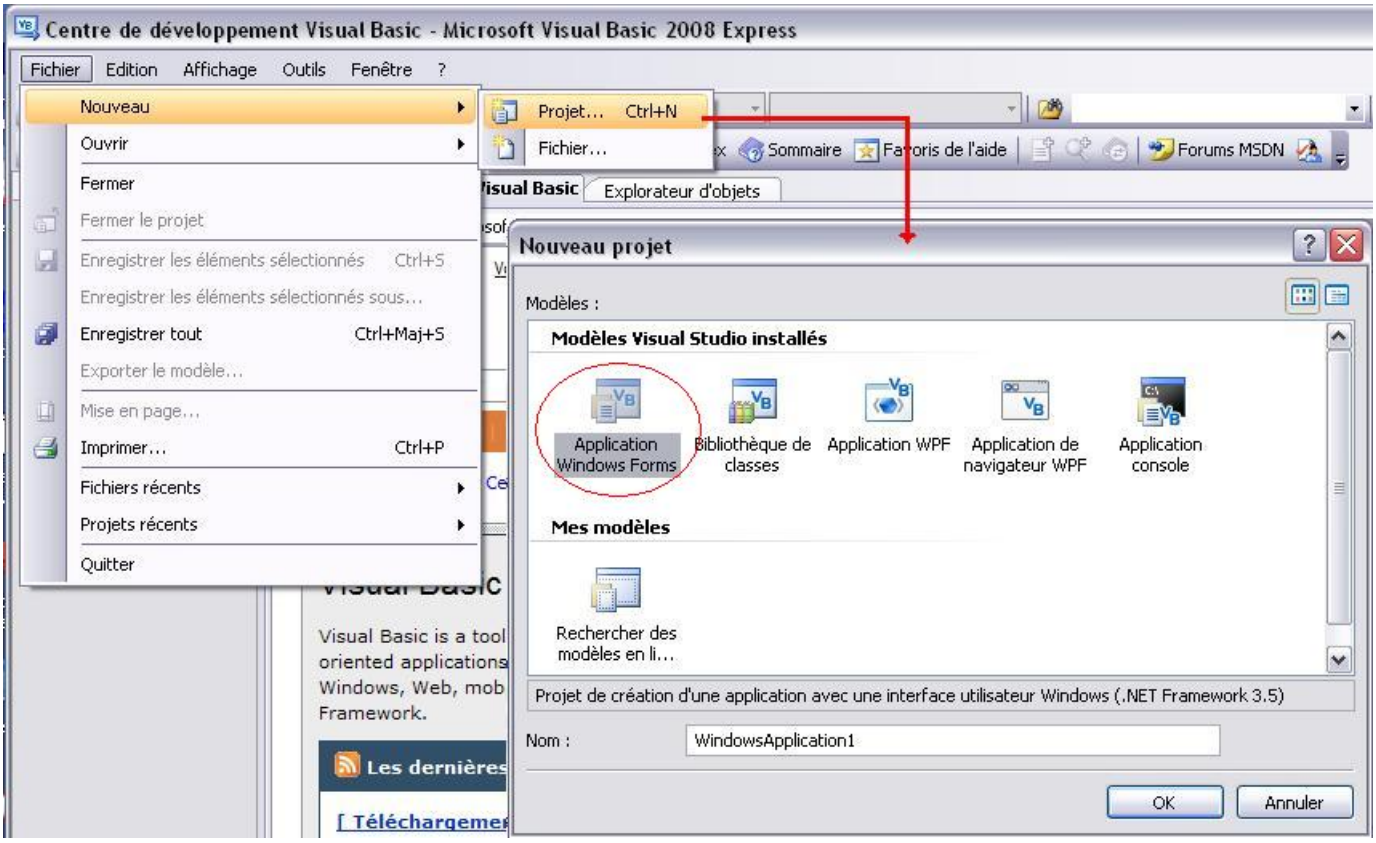
Quand on lance VB.net 2008, on ouvre l'IDE dans laquelle la fenêtre centrale charge la page du centre de développement Visual Basic de MSDN (site Microsoft); il faut être connecté à internet.

En cliquant sur le bouton 'flèche verte' en haut à droite, on affiche la Page de démarrage "Start Page" qui permet d'ouvrir un projet existant Ouvrir (Recent Projects ou Open dans la version anglaise) ou de créer un nouveau projet:Créer (Create dans la version anglaise).



On constate que les diverses fenêtres sont accessibles par des onglets. L'IDE de VB 2008 diffère peu de celui de VB 2005.

Pour créer un nouveau projet Visual Basic, il faut choisir 'Créer' à gauche ou passer par le menu 'Fichier' puis 'Nouveau' puis 'Projet' . La fenêtre suivante s'ouvre:



On a le choix à partir de VB 2008 de créer l'interface utilisateur: En Windowsforms (basé sur GDI+), interface habituelle, bien connue ou en WPF interface vectorielle élaborée n'existant pas avant VB 2008.

IV-A-1 - Interface 'Windows Forms'

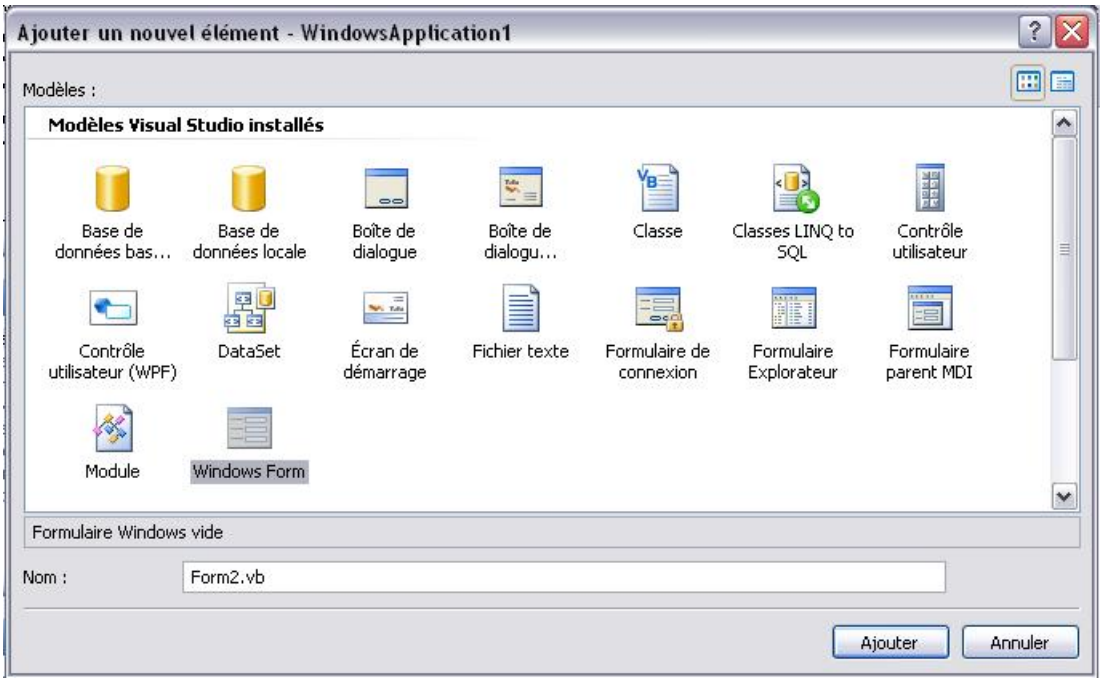
Choisir l'icône 'Application Windows forms', puis donner un nom au projet, enfin valider sur 'Ok'.

(Le chemin de l'emplacement du projet n'est pas modifiable ici, il est par défaut ' C:\Documents and Settings\Nom Utilisateur\Mes documents\Visual Studio 2008\ Projects\MonProjet')

On remarque qu'on aurait pu choisir 'Application WPF', on y reviendra.

Dans un nouveau projet, créer ou ajouter une fenêtre 'WinForm':

Pour ajouter une fenêtre (un formulaire) Menu Project, Ajouter un formulaire Windows ('Add a WindowsForms' en version anglaise):

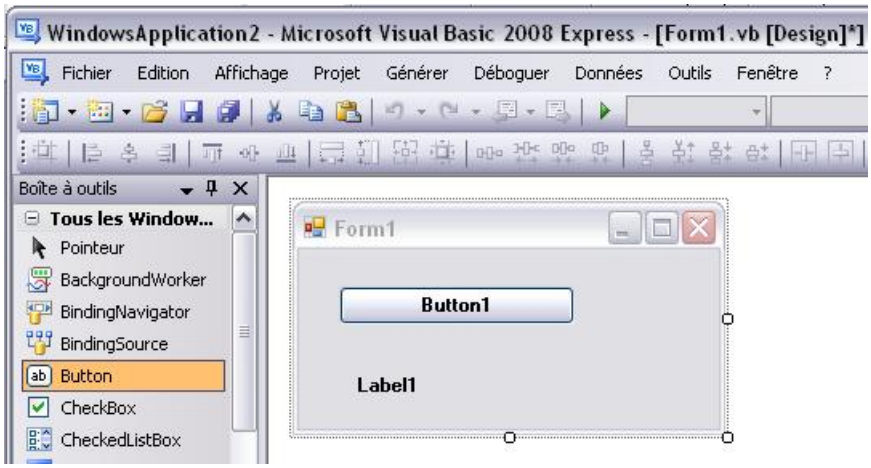


Cliquer sur Windows Form, une fenêtre (un formulaire) Form2 vide apparaît (Form1 était le nom du premier formulaire).

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

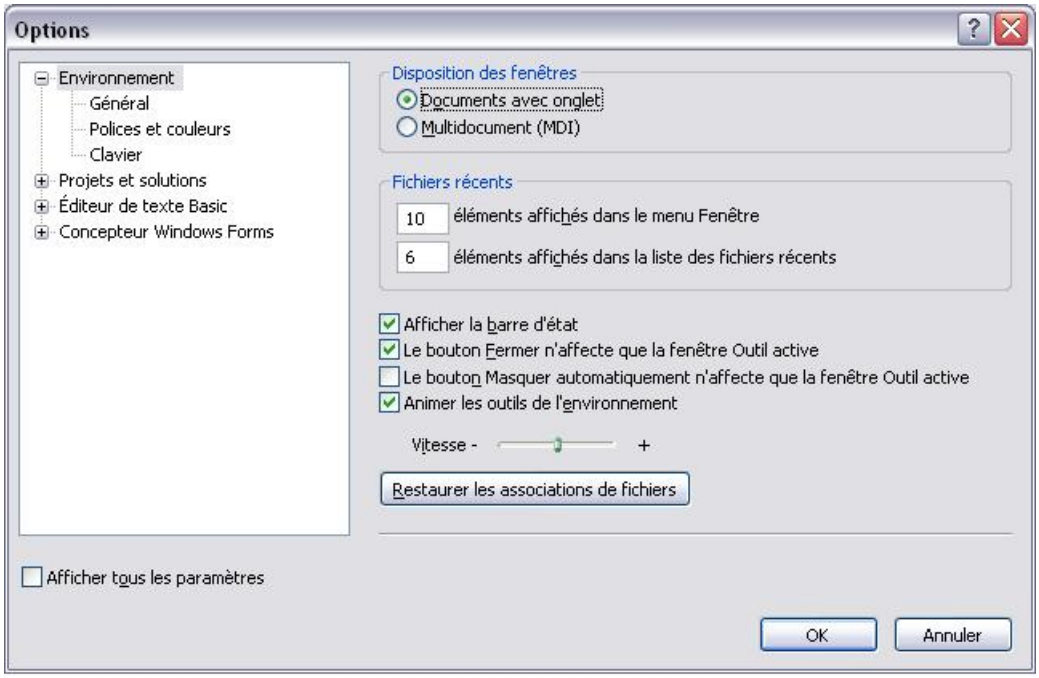
Designer:

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..

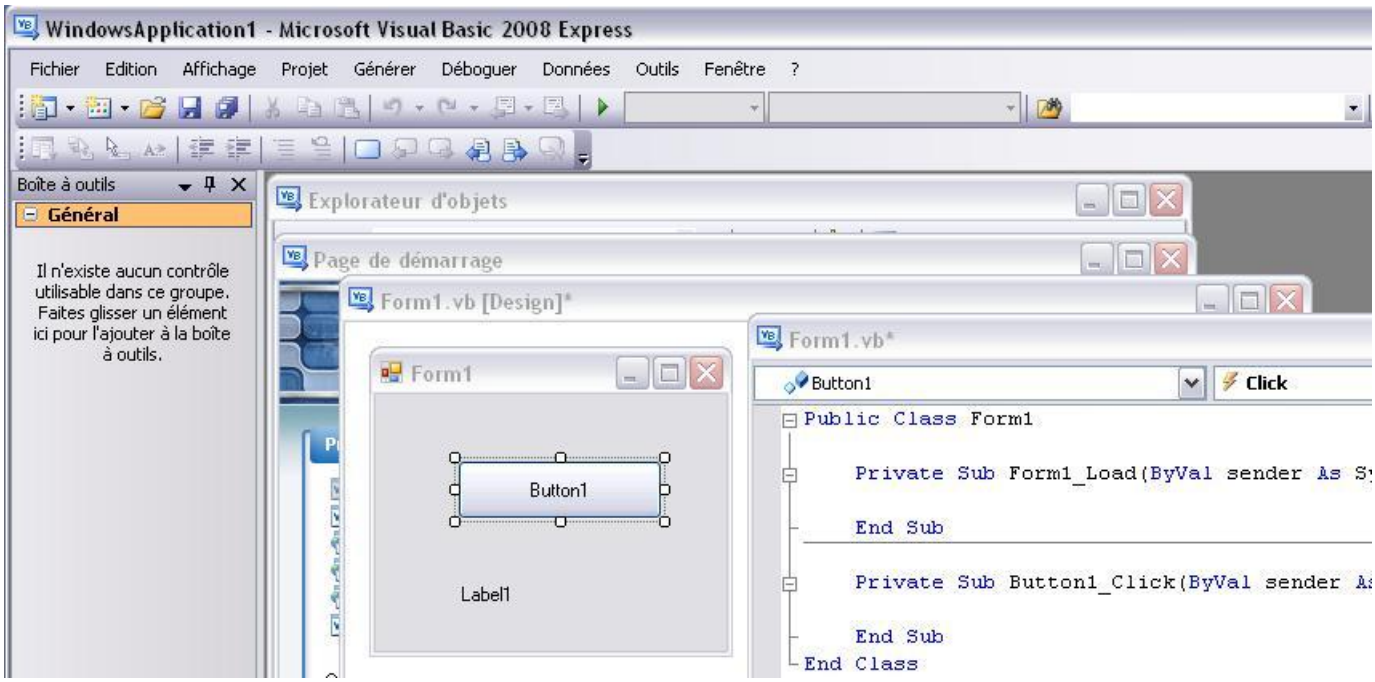


On peut passer en mode 'Multidocument Mdi' (comme en VB6) au lieu du mode 'Onglet':

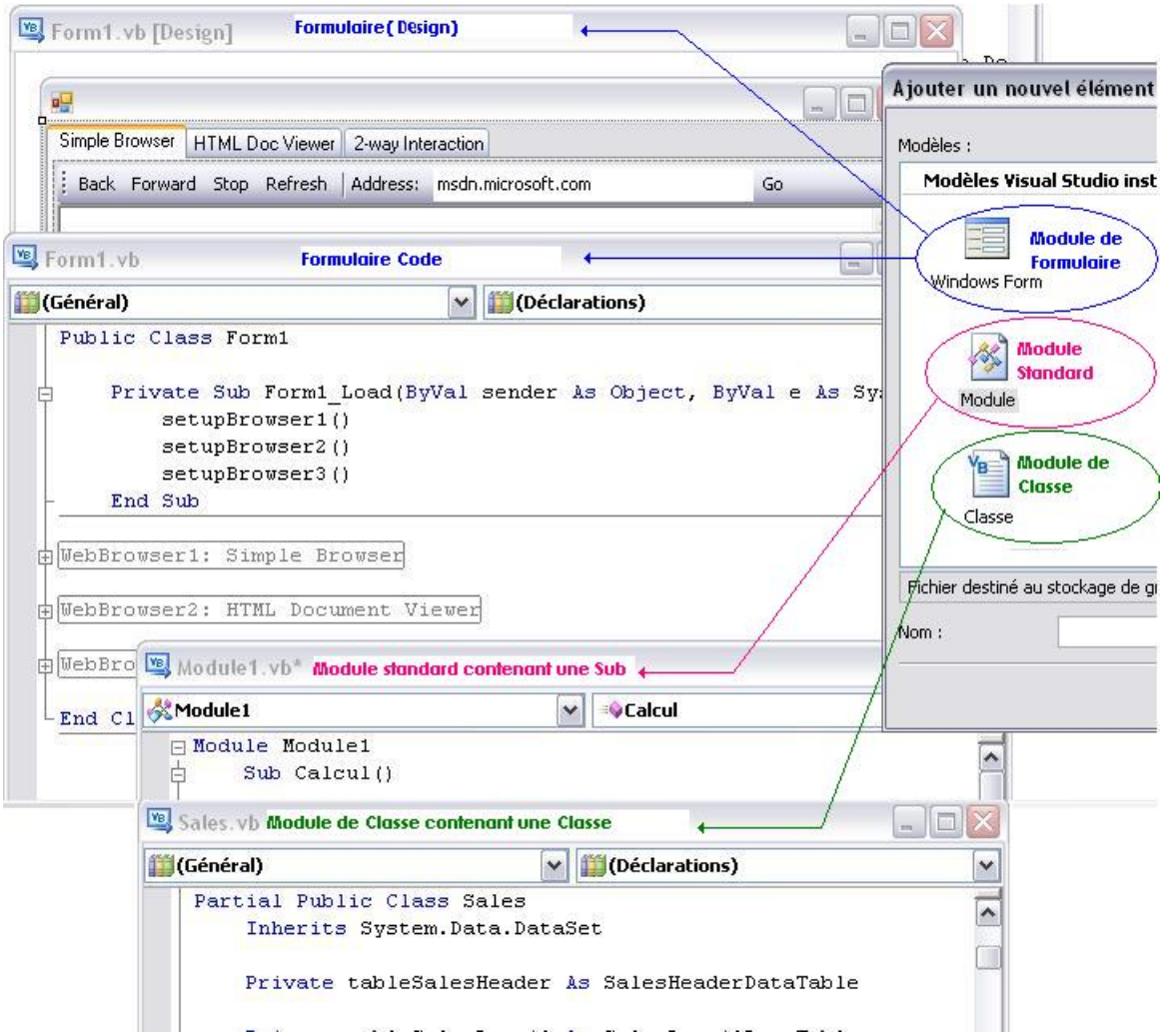
(Passer par le menu 'Outils' puis 'Options..' puis bouton 'Multidocument (Mdi)').



On obtient un mode multidocument avec plusieurs fenêtres.



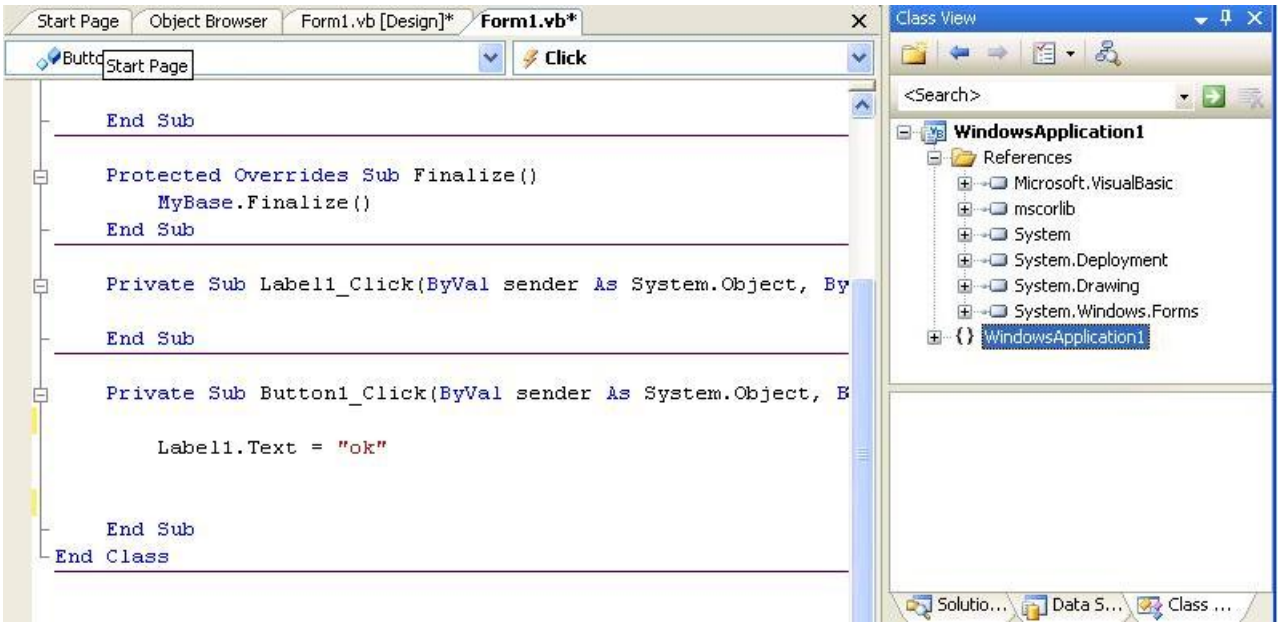
Exemple en mode Mdi montrant les 3 types de module.



A noter que si on utilise le menu 'Projet' puis 'Ajouter..' cela permet d'ajouter un formulaire, un module standard, un module de Classe.

Voir les procédures:

L'onglet Form1.vb donne accès aux procédures liées à Form1.



On peut 'tapez' du code dans les procédures.

La liste déroulante de gauche donne la liste des objets, celle de droite, les évènements correspondants à cet objet.

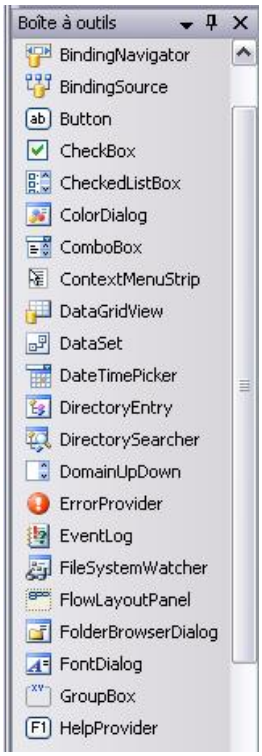
Il est possible en double-cliquant dans le formulaire ou un contrôle de se retrouver directement dans le code de la procédure correspondant à cet objet.

Ici on voit la procédure Button1_Click liée au Button1 de la fenêtre de Design.

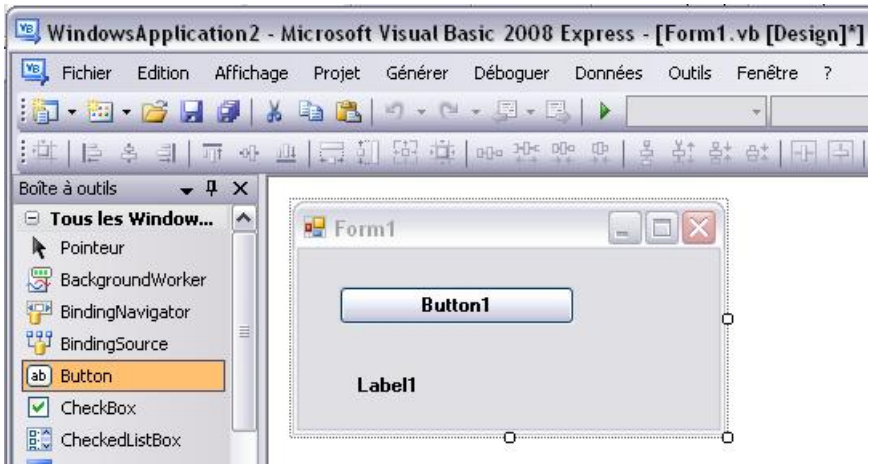
Ajouter des contrôles au formulaire 'Winform'

Ajouter un bouton par exemple:

Cliquer sur Boite à outils (Toolbox) à gauche, les contrôles apparaissent tous ou classés par ordre alphabétique.

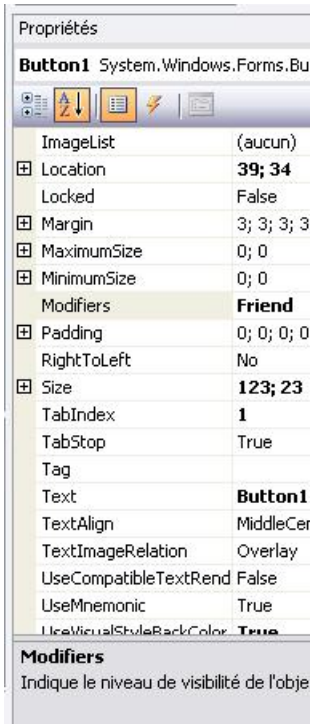


Cliquer sur 'Button' dans la boite à outils, cliquer dans la Form, déplacer le curseur sans lâcher le bouton, puis lâcher: Un bouton apparaît.



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' qui sont visibles (Text, Location..) on peut modifier directement les valeurs.



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

Exemple:

Si au niveau de la ligne 'Text' des propriétés du bouton, j'efface 'Button1' et que je tape 'Ok', dans le designer, le texte écrit sur le bouton deviendra 'OK'.

Le déplacement des contrôles ou l'accès aux principales tâches est facile:

La croix à gauche permet de déplacer le contrôle, la petite flèche à droite permet d'ouvrir un menu qui donne accès aux tâches les plus fréquentes.

L'alignement automatique des contrôles:

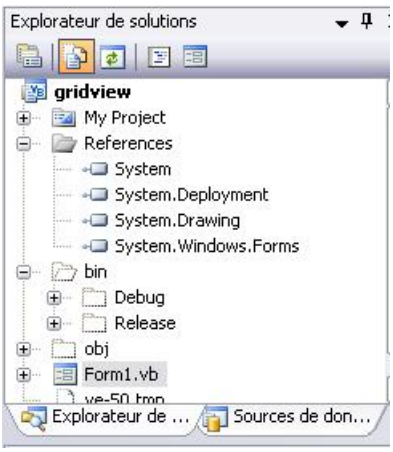
Si on modifie la taille ou l'emplacement d'un contrôle, VB signale par un trait bleu que le contrôle modifié et le contrôle voisin sont alignés:

Renommer un nom: modification automatique.

On nomme cela '**Refactoring**': Cliquer sur une variable, puis bouton droit, dans le menu cliquer sur 'Renommer'. Modifier le nom de la variable, valider. Dans toute la Classe la variable est renommée.

Voir tous les composants d'un projet:

Pour cela il faut utiliser la fenêtre **Explorateur de solutions** en haut à droite, elle permet de voir et d'avoir accès au contenu du projet (Pour voir tous les fichiers, il faut cliquer sur le deuxième bouton en haut) : gridview est le nom du programme.



MyProjet: double-cliquer dessus, vous ouvrirez la fenêtre 'propriétés du projet'.

Références qui contient les dll chargées. Pour atteindre les références, on peut aussi passer par le menu 'Projet' puis 'Propriétés' ou double cliquer sur 'MyProjet' puis choisir l'onglet 'Références.

Form1.vb est un formulaire (une fenêtre).Les formulaires, modules de classe ou standard sont tous des '.vb' Il suffit de double-cliquer dessus pour les ouvrir.

Si on ouvre la sous-liste de Form1.vb (en cliquant sur le '+'), on voit:

Form1.Designer.vb (qui montre le code qui crée le formulaire, on n'a pas à y toucher).

Form1.resx (le fichier de ressources).

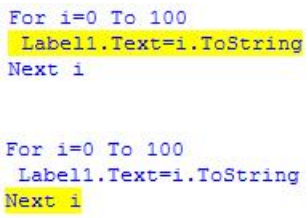
Il suffit de cliquer sur la ligne Form1 dans l'explorateur de solution pour voir apparaître la Form1 dans la fenêtre principale.

Si on clique sur un espace de noms dans la liste Références, cela montre l'arborescence des Classes.

Tester son logiciel:

On peut tester le projet 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:



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.

En mode Debug, on peut modifier une ligne et poursuivre le programme qui tiendra compte de la modification (Sauf pour les déclarations). On parle d''Edit and continue'.

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'). Le code présent dans l'IDE est le code **source**, après compilation le fichier exécutable contient du code **exécutable**.

Projet.

Dans la terminologie VB, **un projet** est une application en cours de développement.

Une '**solution**' (Team Project)regroupe un ou plusieurs projets (C'est un groupe de projets). Il n'y en a pas dans la version express.

En VB express on parle donc uniquement de projet, en fait ,VB crée aussi une solution de même nom.

Fichiers, Chemins des sources.

Si vous regardez dans ' C:\Documents and Settings\Nom Utilisateur\Mes documents\Visual Studio 2008\ Projects \MonProjet')les fichiers correspondant à un projet VB:

MonProjet.sln est le fichier solution.

MonProjet.psess est le fichier de performance (pas toujours présent).

MonProjet.suo est le fichier de User solution.

Dessous existe un répertoire nommé aussi MonProjet qui contient:

MonProjet.vbProj le fichier de projet.

Form1.vb contient un formulaire et ses procédures.

MyClasse.vb contient par exemple des classes.

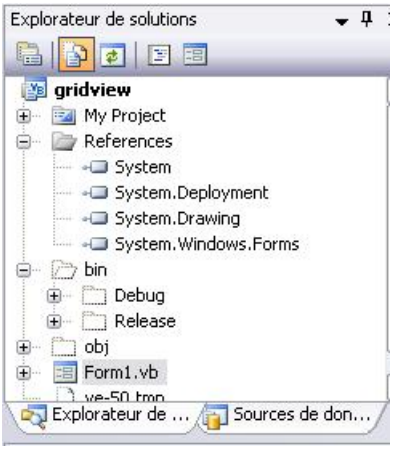
Form1.Designer.vb contient le code qui créer la fenêtre et les contrôles.

Il a encore les sous répertoires \Bin, il y a aussi un répertoire Obj et un répertoire \MyProjet

Si on compile le projet l'exécutable est dans un sous répertoire \Bin,

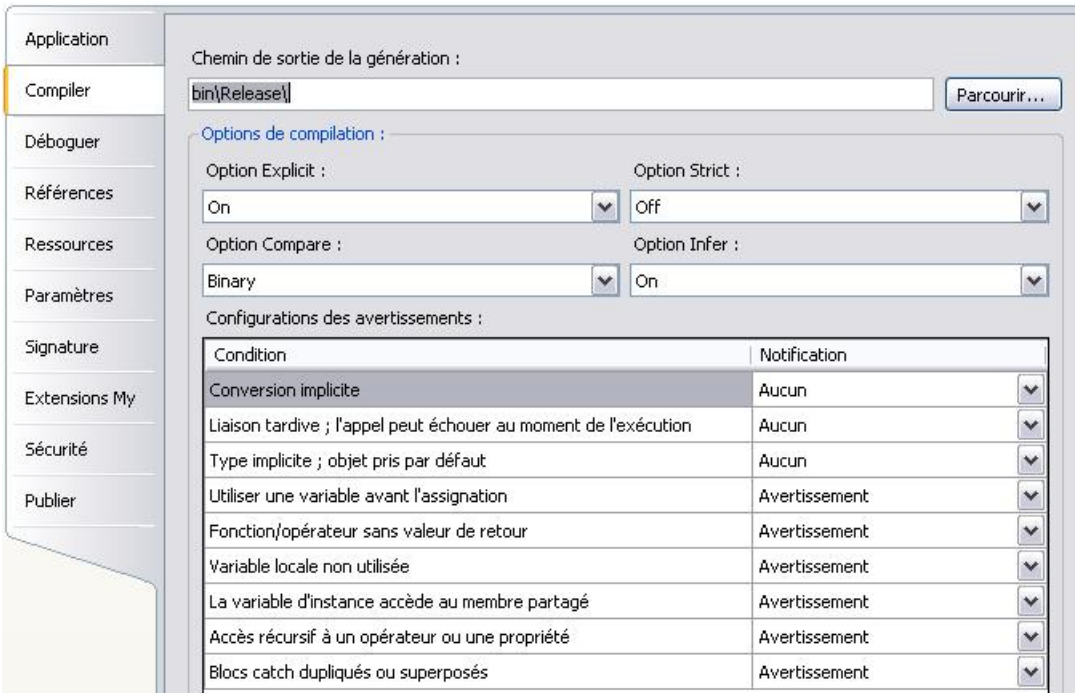
Propriétés du projet:

Toutes les propriétés de l'application peuvent être modifiées dans le 'Projet Designer' (**Propriétés du projet**), pour l'atteindre, il faut double-cliquer sur 'My Project' dans l'explorateur de solution:



Une autre manière d'ouvrir le 'Projet Designer' est de passer par les menus 'Projet' puis 'Propriétés de..'

On retrouve dans le projet designer:



Le nom de l'application, son icône, la fenêtre de démarrage, celle de fin. (Application)

Les Option Strict, Explicit compare et la nouvelle Option Infer.(Compiler).

Les références (dll liées au projet).

Les paramètres (valeurs liées à l'application).

Les ressources (texte, images, sons utilisées dans le programme).

La signature et la sécurité.

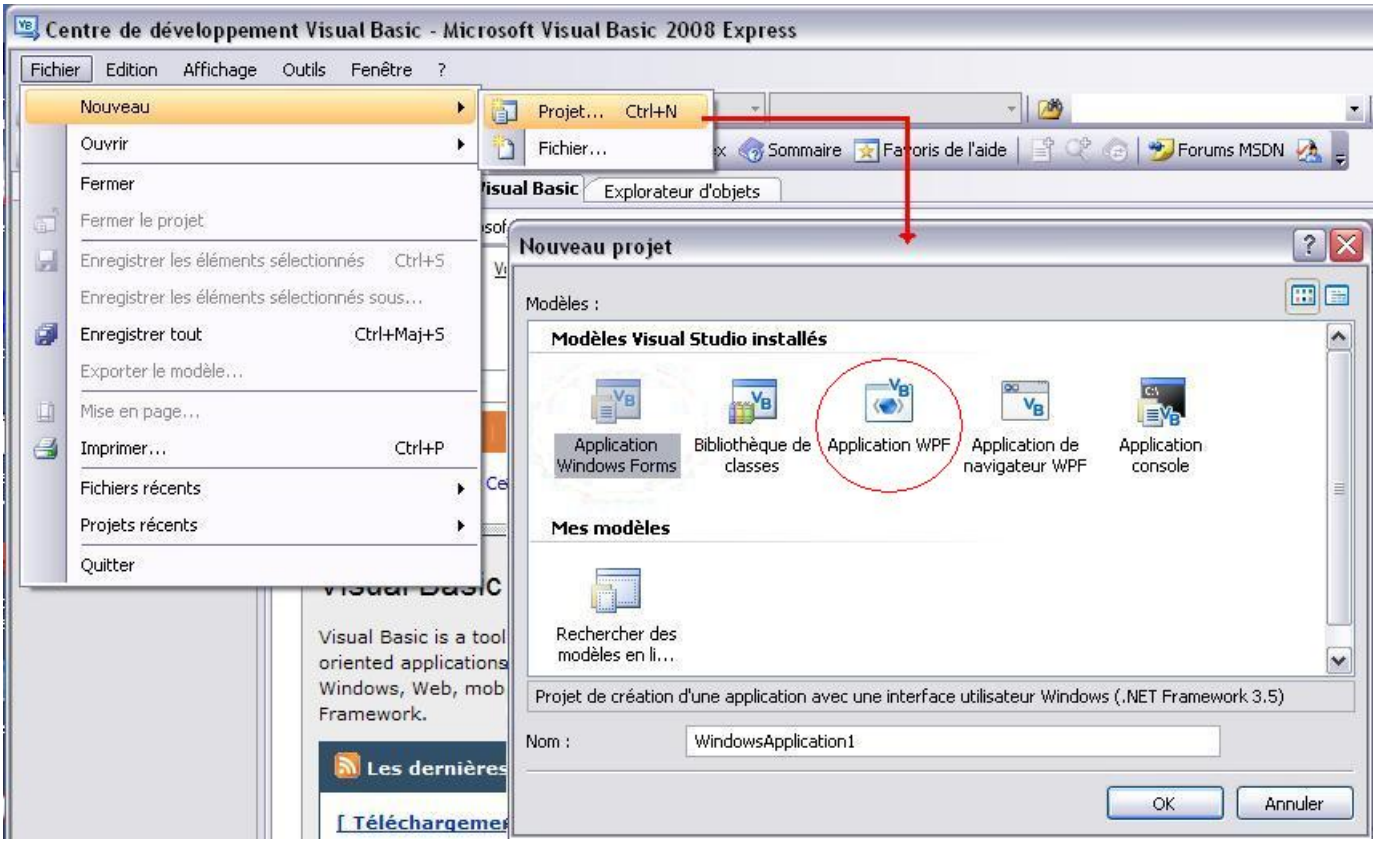
Les Extension My (nouveauté 2008).

Les paramètres relatifs à la publication (distribution et installation).

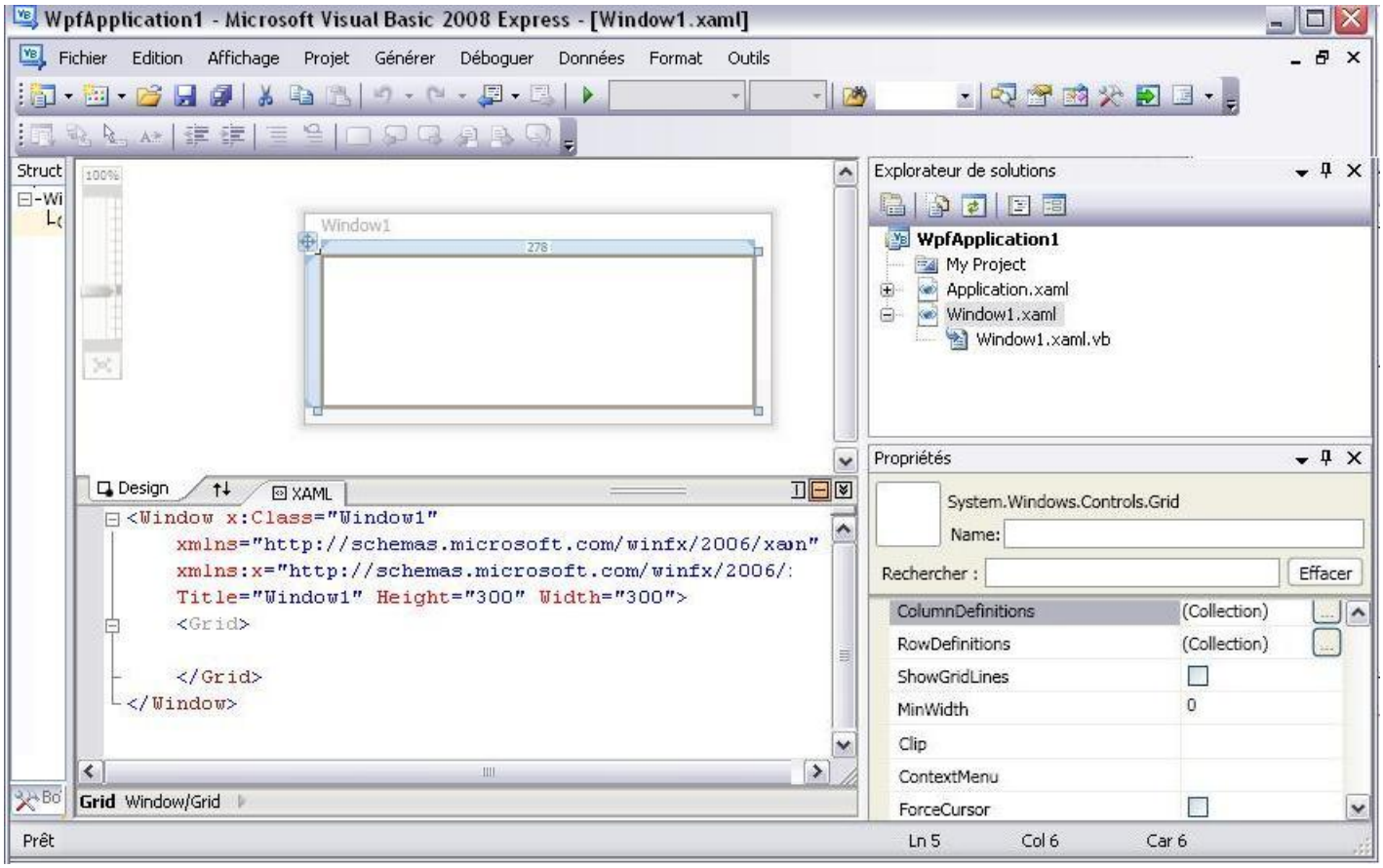
IV-A-2 - Interface WPF

Plutôt que de travailler avec les Windows Forms (formulaire habituel utilisant GDI+); en VB 2008 on peut utiliser un mode graphique vectoriel extrêmement performant pour dessiner les formulaires et contrôles: pour cela on utilise les WFP (Windows Presentation Foundation).

Pour cela: menu 'Fichier', 'Nouveau', 'Projet'.



On choisit 'Application WPF', on se retrouve dans un nouvel environnement:

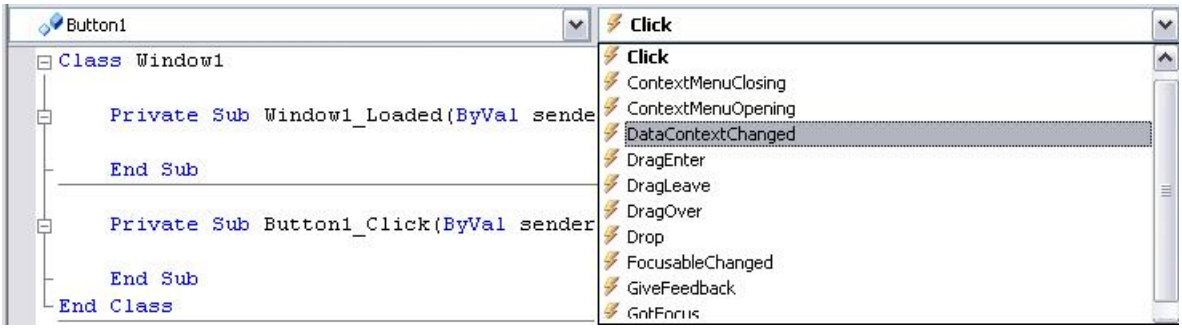


Les formulaires et contrôles sont différents de ceux des Windows Forms, ainsi que les propriétés des objets graphiques.

Il y a le 'designer' en haut qui permet de dessiner l'interface que verra l'utilisateur. Le designer génère un fichier XAML en bas qui décrit en XML l'interface.

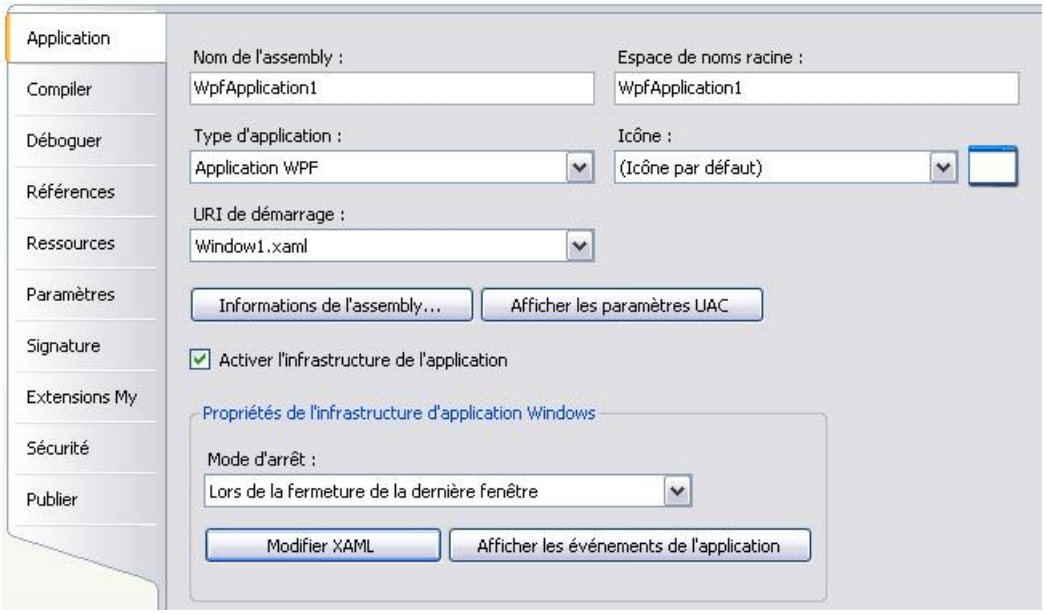
Dans la version Express, on peut dessiner des interfaces simples, les interfaces extrêmement élaborée (dégradé de couleur, animation...) peuvent être écrites en code XAML ou en utilisant un programme extérieur payant (Expression Blend). Voir le chapitre sir les WPF.

Si on double-Clique sur un bouton, par exemple, on se retrouve dans la procédure évènement correspondante:



On se rend compte que les évènements là aussi ne sont pas les mêmes que pour les WindowsForm.

Il y a aussi d'autres modifications comme dans les propriétés du projet:



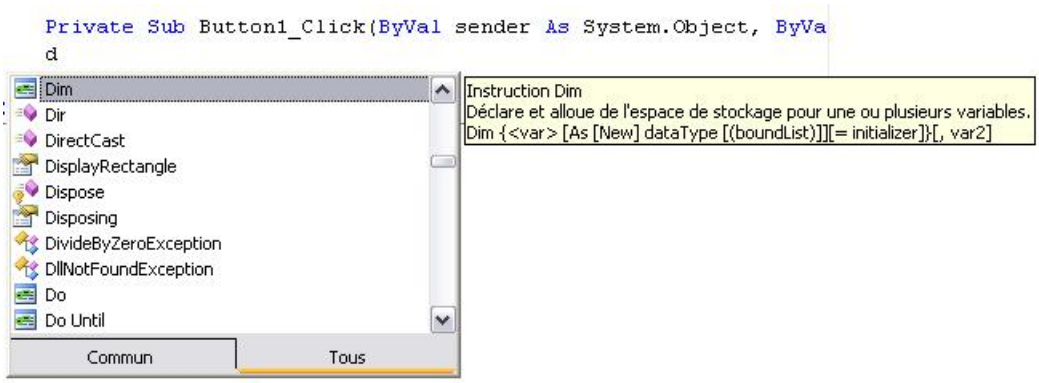
Voir le chapitre sur les WPF.

IV-A-3 - Vb propose des aides

Quand on tape du code, VB affiche, des aides:

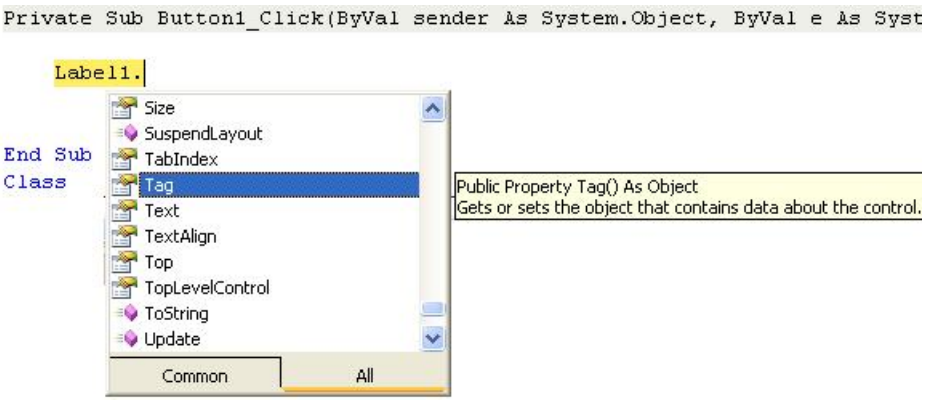
Dès que je tape une lettre VB propose dans une liste des mots.

Exemple, je tape 'd', il affiche 'Dim', 'Dir'..., de plus si je me mets sur un des mots, il ouvre une petite fenêtre d'explication sur le mot avec sa syntaxe.



VB permet de choisir dans une liste une des propriétés d'un objet.

Exemple: Je tape le nom d'un label nommé label1 puis je tape un point, cela me donne la liste des propriétés du label.



Quand je pointe dans la liste un des membres (propriété ou méthode) un carré jaune affiche la définition de la fonction avec ses paramètres et une explication.

VB aide à retrouver les paramètres d'une fonction:

Si on tape le nom d'une fonction et '(', VB affiche les paramètres possibles dans un cadre.

En plus il affiche les différentes manières d'utiliser les paramètres (les différentes signatures), on peut les faire défiler avec les petites flèches du cadre jaune.

VB aide à compléter des mots.

Si je tape App puis sur le bouton 'A->', Vb affiche la liste des mots commençant pas App

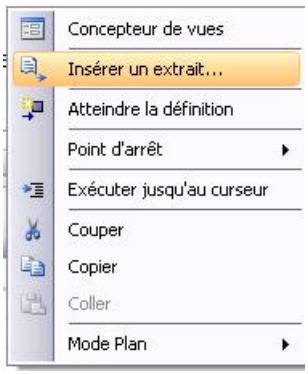
AppActivate

AppDomain

VB fournit des exemples de code.

Les **Extraits** (Snippets, bride, morceau de code) permettent d'insérer du code tout fait.

Dans le code d'une procédure, le click droit de la souris ouvre un menu.



Cliquer sur 'Insérer un extrait' (Insert Snipper). Puis par menu successif vous obtiendrez le code que vous cherchez.

Vb propose des solutions pour corriger les erreurs de code:

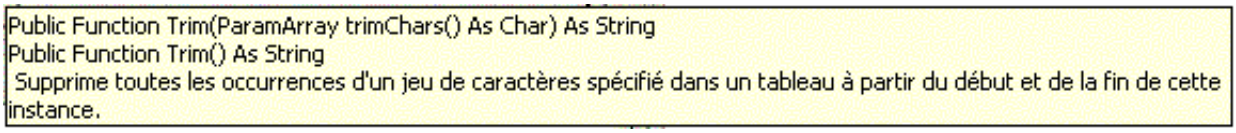
Si je veux afficher une valeur numérique (avec option Strict=On), il y a erreur, VB me propose la correction:

Il existe une abondante documentation:

Sur le Net: Msdn Framework 3.5

(http://msdn.microsoft.com/fr-fr/library/aa139616.aspx)

Dans l'IDE, VB donne accès à l'aide sur un mot Clé. Si le curseur passe sur un mot clé, un carré affiche la définition de la fonction. Si je clique sur un mot et que je tape F1 l'aide s'ouvre et un long texte donne toutes les explications. VB donne accès à l'aide sur les contrôles. Si le curseur est sur un contrôle et que je tape F1 l'aide s'ouvre pour donner accès à la description des différents membres de cet objet. Enfin il est toujours possible de rechercher des informations par le menu '?'



Erreur dans l'écriture du code.

S'il existe une erreur dans le code au cours de la conception, celle-ci est soulignée en bleu ondulé. Un carré donne la cause de l'erreur si le curseur passe sur la zone ou se trouve l'erreur.

```
Label1.Texte() = "12"
```
'Texte' n'est pas un membre de 'System.Windows.Forms.Label'.

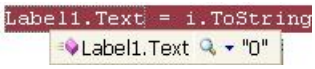
Ici la propriété 'Text' a été mal orthographiée.

Si je lance le programme en mode 'Run' et qu'il y a des erreurs, Vb me le signale et répertorie les erreurs dans la liste des taches en bas. Vb propose des solutions pour corriger les erreurs de code. (Voir plus haut)

Mode débogage (mode BREAK):

Une fois lancée l'exécution (F5), puis stoppée (par Ctrl +Alt +Pause ou sur un point d'arrêt), on peut:

Voir la valeur d'une propriété d'un objet en le pointant avec la souris:



Il s'affiche un petit cadre donnant la valeur de la propriété d'un objet.

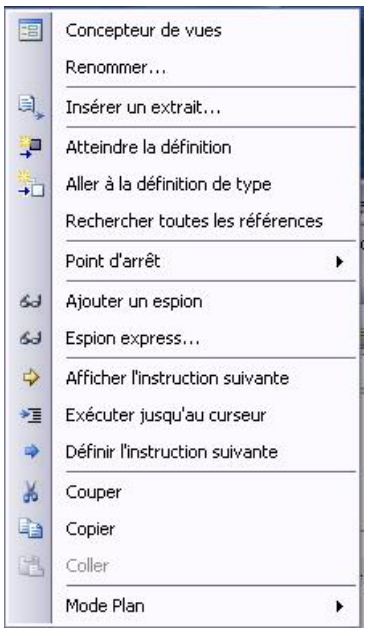
Voir la valeur d'une variable, simplement en positionnant le curseur sur cette variable.

F11 permet l'exécution pas à pas (y compris des procédures appelées).

F10 permet le pas à pas (sans détailler les procédures appelées).

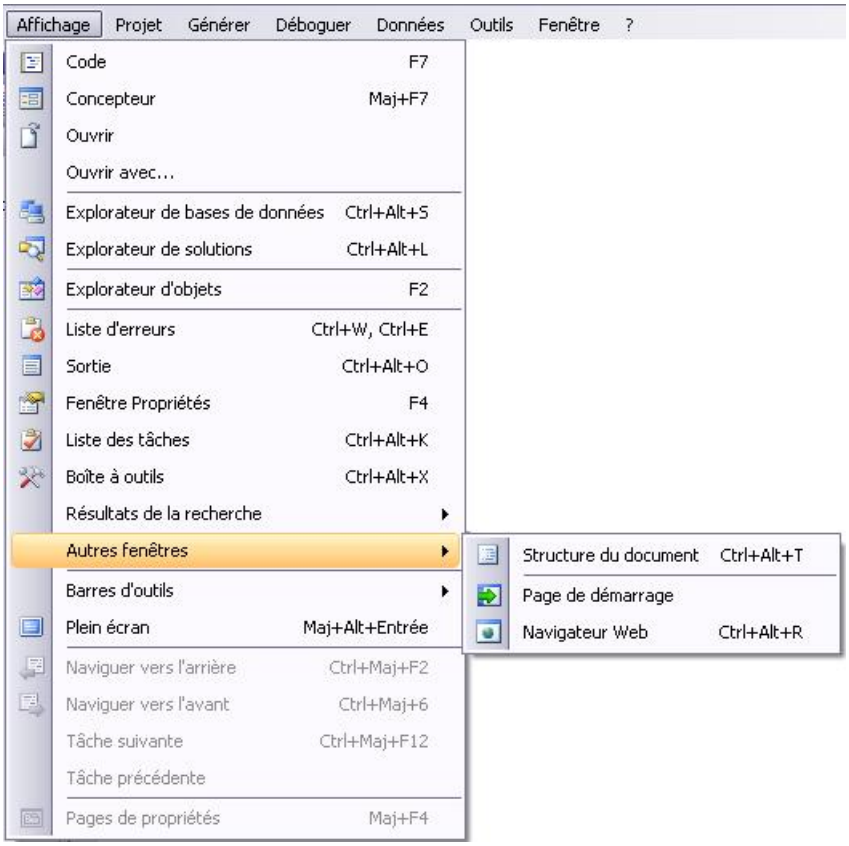
Maj+F11 exécute jusqu'à la fin de la procédure en cours.

En cliquant sur le bouton droit de la souris, on peut exécuter jusqu'au curseur (Run To Cursor), voir la définition, la déclaration de ce qui est sous le curseur (Atteinte la définition:Go To Definition)...



Il y a un chapitre sur le débogage pour apprendre à trouver les erreurs de code.

On peut grâce au menu 'Affichage' on peut avoir accès à plein de choses :



IV-B - IDE SharpDevelop (alternative gratuite) MÀJ version 2.1

C'est l'IDE (Integrated Development Environment): Environnement de développement intégré GRATUIT, en OpenSource, alternative à VisualStudio.

Depuis sa version 2 #develop est un très bon produit et n'a rien à envier à Visual Studio.

#Develop sera toujours gratuit.

C'est un logiciel libre en Open Source (GPL), fonctionne officiellement sous Windows XP et 2000 (Pas officiellement sous ME et 98)

Il parait que SharpDevelop fonctionne sous Windows 98 (non testé, si vous avez essayé, m'en faire part), Millenium (testé), NT 4, Windows 2000 (testé) , XP (testé). Alors que Visual Studio ne fonctionne pas sur un PC non NT (exit Windows 98 et Millenium).

IV-B-1 - Où le trouver ? Comment l'installer ?

Exemple de téléchargement SharpDevelop2 framework 2:

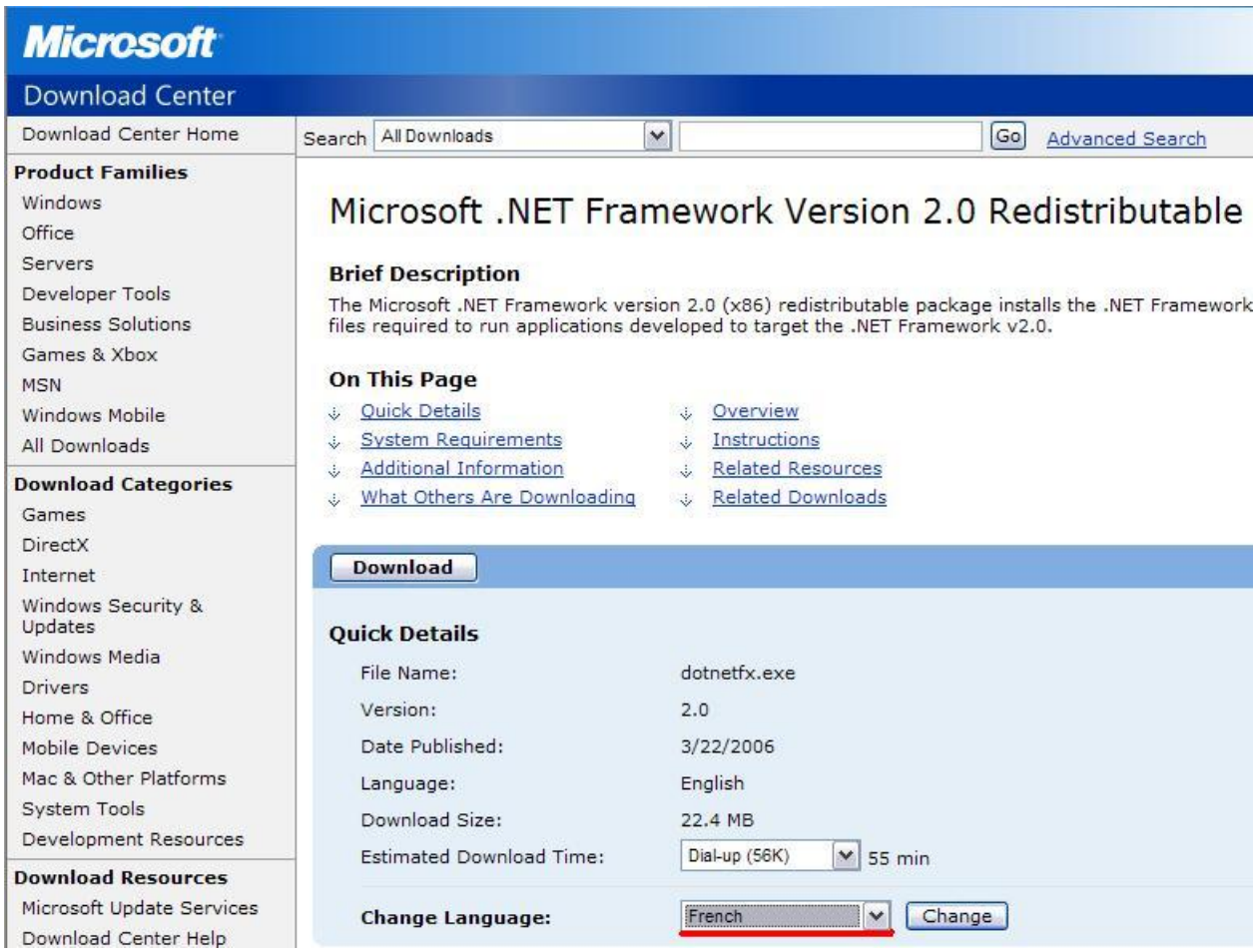
Respecter l'ordre d'installation.

1 Télécharger et installer le FrameWork. (Impérativement en premier).
Installer Microsoft .NET version 2 package.

C'est le Framework (la couche logiciel entre l'application et le système), il est téléchargeable sur le Net sur le site Microsoft.

Télécharger le FrameWork puis l'installer. (Gratuit)

⚠ *Attention, changer la langue (French) avant le téléchargement.*



2 Télécharger et installer le SDK 2.
C'est le Kit de Développement Microsoft .NET Framework: SDK du Framework 2.

En bas de la page précédente ou par le lien suivant:

Télécharger le SDK (Gratuit) Attention, changer la langue (French).

3 Télécharger et installez SharpDevelop 2.1 (beta3 le 2/2/2007)
Télécharger SharpDevelop 2.1 (Gratuit)

L'installer en exécutant le fichier 'SharpDevelop_2.1.0.2201_Setup.exe'.

Il est maintenant possible d'installer le **Framework 3.5** et d'installer **SharpDevelop 3**.

4 Configurer SharpDevelop
Lancer Sharpdevelop 2.1

Aller dans le menu 'Outils' - 'Options'

La langue de l'utilisateur est 'French', si cela n'est pas le cas, modifier la langue.

Dans 'Style visuel' : Choisir VBNET dans la liste.

'Codage' : Éditer les en-têtes standard: VB.Net

'Codage' : Modèle de code: extension '.vb

'Editeur de texte' : Surlignement, VB.net dans la liste de gauche.

Le Framework, le SDK et #develop suffisent pour faire des programmes.

Lien

Site SharpDevelop

IV-B-2 - Fenêtre Projet

Lancer SharpDevelop:

Au lancement de l'application, on peut :

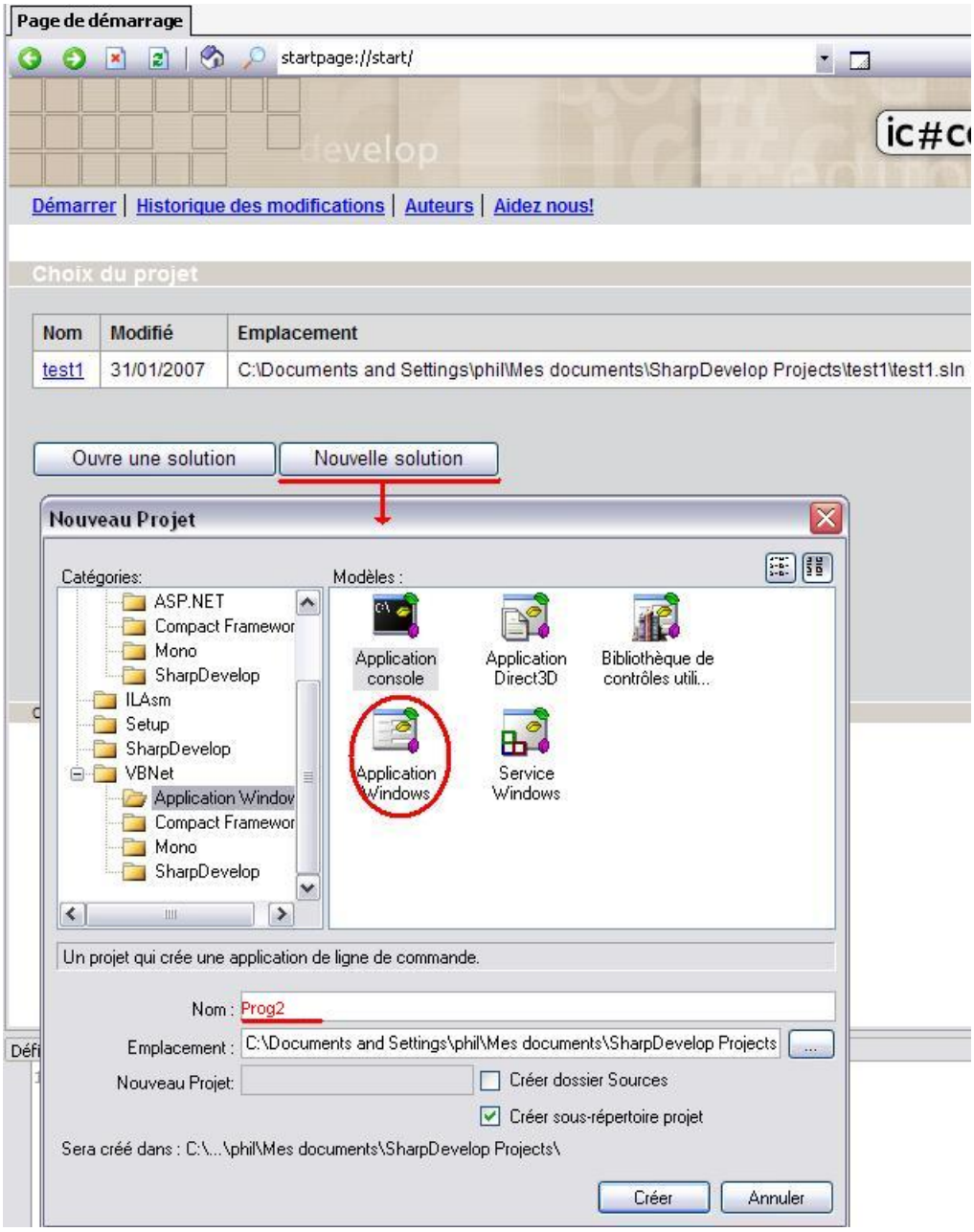
- Ouvrir une solution existante: Bouton 'Ouvrir une solution' (ou cliquer sur le nom d'un projet récent en haut)
- Créer un nouveau projet (une nouvelle solution) .

Si l'on veut rajouter des fichiers à notre projet faire :
'Fichier'-'Ouvrir'-'Fichier' et catégorie VB

Détaillons la création d'un nouveau projet.

Bouton 'Nouvelle solution' ou

Menu 'fichier'-'Nouveau'-'Solution'

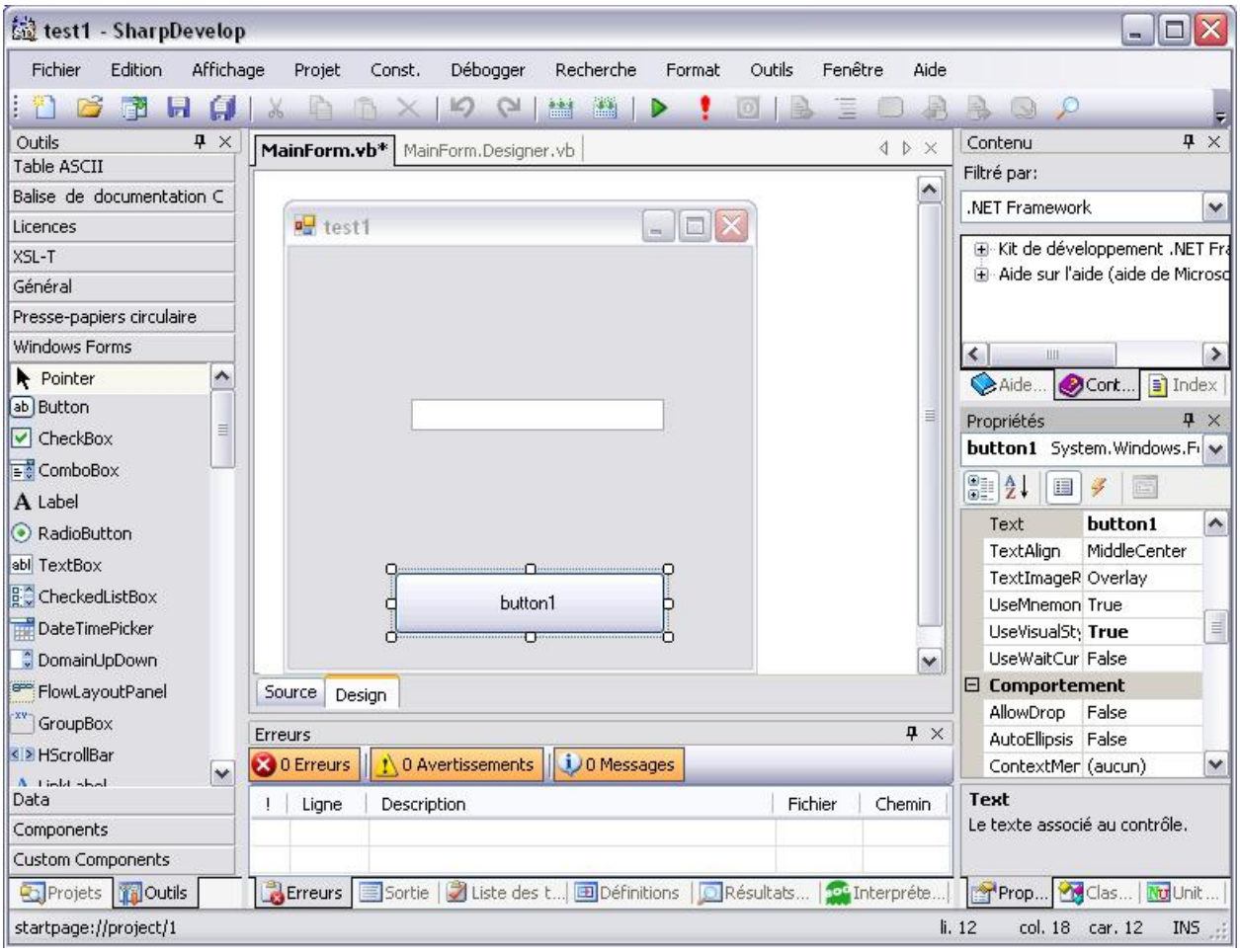


Sélectionner la catégorie 'VBNET' et choisir le type d'application à créer.(Dans le cas d'une création d'un projet Visual Basic, il faudra choisir dans les 'Modèles': Application Windows.) On remarque que #Develop permet aussi d'écrire du C#, du C++ du ILAsm un setup.

Puis il faut donner un nom au projet (il n'y a pas de nom par défaut), modifier si nécessaire le chemin de l'emplacement du projet qui est par défaut ' C:\Documents and Settings\NomUtilisateur\Mes documents \SharpDevelop Projects' (cocher si nécessaire 'Créer le répertoire source') enfin valider sur le bouton 'Créer'. Une fenêtre 'MainForm' apparaît.

Si, comme dans notre exemple, on a tapé 'Prog2', #develop crée une 'solution' nommée 'SolutionProg2'(ensemble, groupe de projets) contenant un projet (Prog2) contenant un formulaire nommé 'MainForm'

L'écran principal se présente ainsi:



Au centre, sont visible les écrans du code et des formulaires ; on peut changer d'écran grâce aux onglets du haut. Ici on voit 'MainForm'.

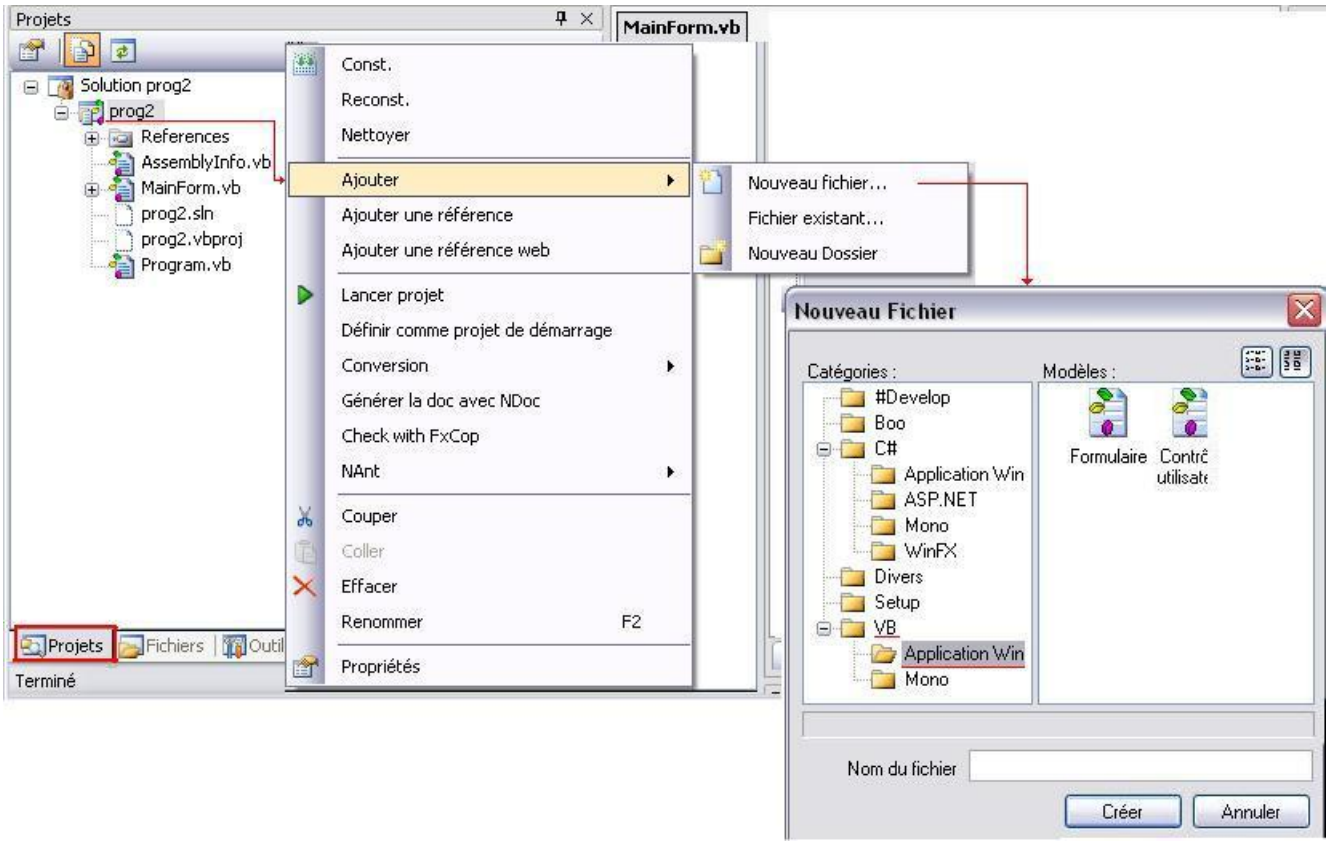
A gauche, les onglets du bas donnent accès au projet en cours (les solutions, projets, formulaires, autres fichiers: ressources, assembly..) ou aux outils : Table ascii, Presse papier et surtout (si on a un formulaire au centre et non du code) aux objets (bouton, texteBox, ListBox...)

A droite, en bas, les classes et surtout la fenêtre de Propriétés (Name, Text..) de l'objet sélectionné au centre.

En bas les fenêtres de 'sortie' (affichage de la console) liste des 'erreurs' des 'taches', définitions', 'Résultat des recherches'..

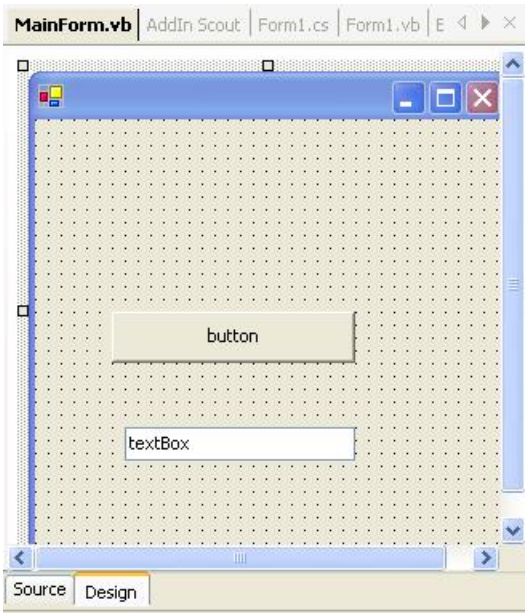
IV-B-3 - Dans un nouveau projet, créer une fenêtre

Pour ajouter une fenêtre (un formulaire) ouvrir le gestionnaire de projet et solution (Onglets en bas à gauche), il donne le nom de la solution (solutionprog2) et du projet (prog2 ici) Cliquer avec le bouton droit sur prog2 puis dans les menus sur 'Ajouter', 'Nouveau fichier'. Cela ouvre la fenêtre 'Nouveau fichier'.



Dans la fenêtre qui s'ouvre, à gauche, choisir 'VB' puis 'Application Windows' , à droite 'Formulaire' , taper un nom de formulaire (Form1 par exemple)puis 'Créer' , une fenêtre 'Form1' apparaît. La première fenêtre qui s'ouvre automatiquement quand on crée un projet se nomme 'MainForm'.

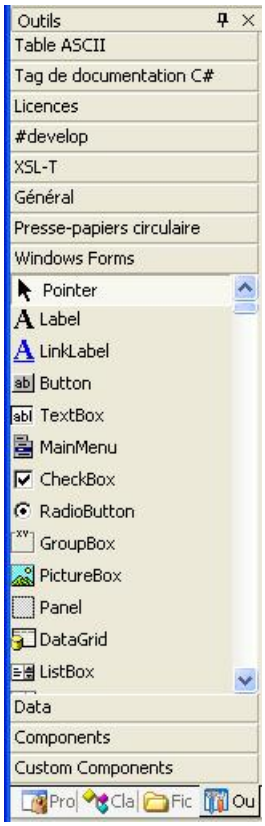
La zone de travail se trouve au centre de l'écran: On voit les onglets MainForm, Form1.vb pour chaque formulaire (fenêtre)



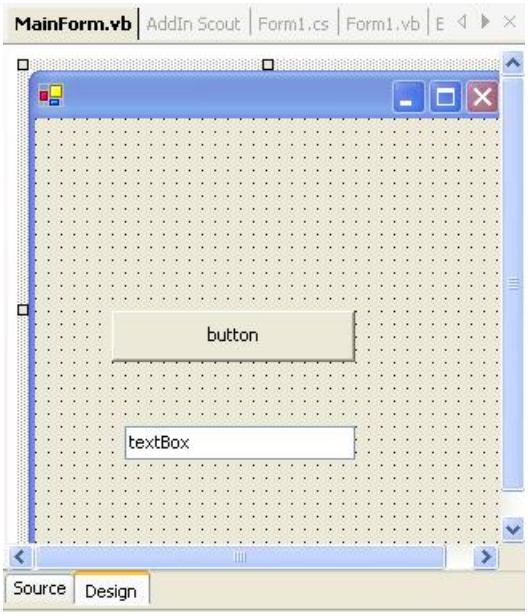
En bas les onglets 'Source' et 'Design' permettent de passer de l'affichage du code('Source') à la conception de l'interface utilisateur ('Design'): affichage de la fenêtre et de ses contrôles permettant de dessiner l'interface.

IV-B-4 - Ajouter des contrôles au formulaire

Ajoutons un bouton par exemple:



Cliquer sur l'onglet 'Outils' à gauche en bas , bouton 'Windows Forms', puis sur 'Button', cliquer dans la MainForm, déplacer le curseur sans lâcher le bouton, puis lâcher le bouton :



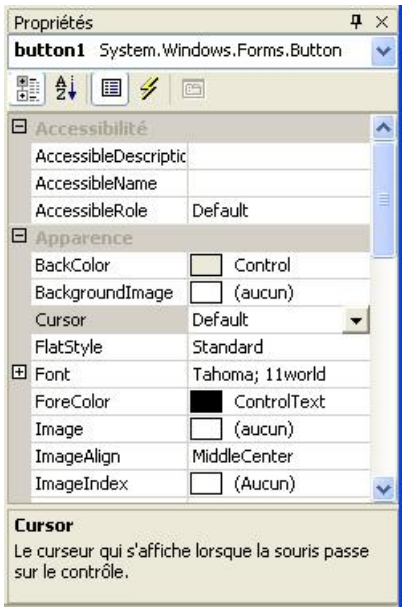
IV-B-5 - Modifier les propriétés d'un contrôle ou du formulaire

Quand une feuille ou un contrôle est sélectionné dans la fenêtre Design, ses propriétés sont accessibles dans la fenêtre de propriétés à droite en bas:(Si elles ne sont pas visibles, cliquer sur l'onglet 'Propriétés' en bas).

Ici ce sont les propriétés du contrôle 'Button1' (BackColor, Image, Texte..)

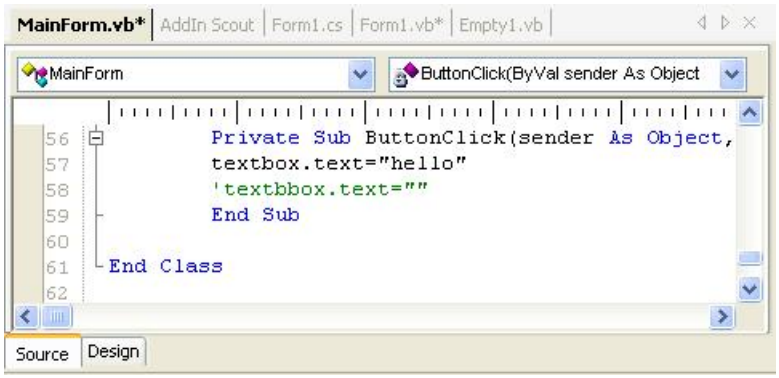
Un petit texte d'aide concernant la propriété en cours apparaît en bas.

(on peut modifier les propriétés directement.)



IV-B-6 - Voir les procédures

L'onglet 'Source' en bas donne accès aux procédures (au code) liées à Form1.



La combo déroutante de droite donne la liste des objets. Si on en choisit un, le pointeur va sur les procédures liées à cet objet.

Malheureusement, contrairement à Visual Studio, la combo de gauche ne contient que les formulaires et pas les objets. Par exemple, on aura MainForm, mais pas Label1... Du coup la recherche se fait directement dans la combo de droite et c'est forcément beaucoup moins clair dès qu'il y a beaucoup de contrôles sur un formulaire...

Il est possible en double-cliquant dans le formulaire ou sur un contrôle de se retrouver directement dans le code de la procédure correspondant à cet objet.

Si la procédure n'existe pas (ButtonClick par exemple),double-cliquez sur le bouton pour la créer.

Pour créer les autres procédures évènements, utiliser le bouton  qui est sur la fenêtre des propriétés à droite, il fait apparaître la liste des évènements, double-cliquant sur un évènement cela permet d'ouvrir la fenêtre de code et de créer les procédures.

IV-B-7 - Voir tous les composants d'un projet

Pour cela il faut utiliser La fenêtre Projet à gauche (Si elles ne sont pas visibles, cliquer sur l'onglet 'Propriétés' en bas), elles permettent de voir et d'avoir accès au contenu du projet:

le gestionnaire de projet et solution donne le nom de la solution (solutionprog2) et du projet (prog2 ici) Cliquer sur les '+' pour développer: vous verrez apparaître les formulaires, les modules.. et:

Références qui contient les espaces de nom.

Assembly: info nécessaire pour générer le projet..

IV-B-8 - Remarque relative aux fenêtres de l'IDE

Pour faire apparaître une fenêtre qui a disparu (fenêtre projet par exemple) utiliser le menu 'Affichage' puis 'projet'.

Quand la fenêtre est ancrée (accrochée aux bords), le fait de la déplacer avec sa barre de titre la 'dé ancre', et elle devient autonome.

Pour la 'ré ancrer', il faut double-cliquer dans sa barre de titre.

IV-B-9 - Tester son logiciel

On peut compiler le projet avec le premier bouton ci-dessous. Créer le projet avec le second. Lancer l'exécution avec le bouton flèche verte (débogueur actif), le point d'exclamation lance l'exécution sans débogage, le rond à droite (qui devient rouge pendant l'exécution) sert à terminer l'exécution.

La liste déroutante permet de choisir la configuration des fenêtres de l'IDE:

Défaut: c'est les fenêtres habituelles précédemment décrites.

Débogage: ouvre les fenêtres: variables locales, points d'arrêt, modules chargés..

Texte simple: uniquement les fenêtres centrales.

Editer: ouvre la fenêtre Edit Layout?

La sauvegarde du projet se fait comme dans tous les logiciels en cliquant sur l'icône du paquet de disquettes.

IV-B-10 - Fichiers, Chemins des sources

Avant, en #develop 1:

.prjx est le fichier de projet.

.cmbw est le fichier solution.

Avec Sharpdevelop 2 c'est comme en VB: les solutions sont maintenant des fichiers .sln

.vb sont tous les fichiers Visual Basic (Feuille module...)

Les sources sont par défaut dans ' C:\Documents and Settings\NomUtilisateur\Mes documents\SharpDevelop Projects'

Si on compile le projet l'exécutable est dans un sous répertoire \Bin\Debug ou \Bin\Realese.

Si vous avez plusieurs versions du framework sur votre machine (version 1.0, version 1.1 voire version 2.0 Bêta), il vous est possible de choisir le compilateur dans les options du projet.

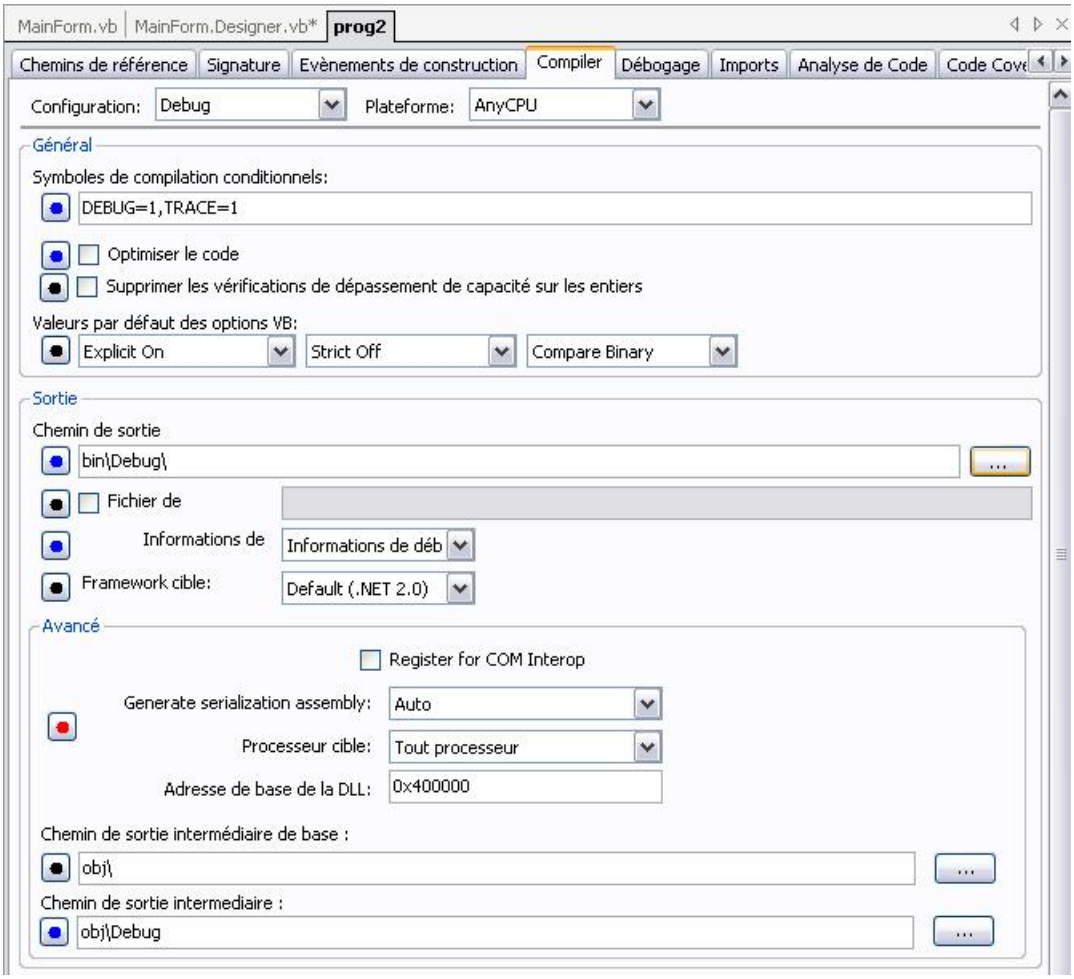
Visual Studio 2003 à version 1.1 du framework.

Visual Studio 2005 à version 2.0 du framework.

IV-B-11 - Propriétés du projet

Menu 'Projet', 'Option du projet' permet l'accès aux propriétés du projet en cours.

Le quatrième onglet (Compiler) est le plus intéressant:



On peut:

Compiler le programme en mode 'Debug' ou 'Release'.

Forcer le programmeur à travailler en Option Strict= On (empêcher les conversions automatiques).

Option Explicit=On (Forcer la déclaration des variables).

Choisir le Framework avec lequel on travaille (1 ou 2, pas le trois encore).

...

Dans l'onglet Import, on peut importer des espaces de noms.

IV-B-12 - #Develop propose des AIDES

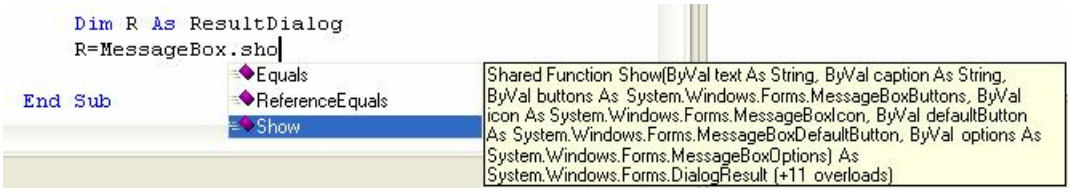
La fenêtre d'aide à droite donne accès à des aides:

De #develop en anglais, non à jour!!

Du Framework

De zipLib

Si vous avez installé le SDK (SDK Framework .Net et/ou SDK Direct X) , vous avez accès à l'aide (partie en haut à droite de l'écran) , et donc également à l'intellisense, qui affiche les propriétés, les méthodes des objets, les paramètres des fonctions, des types, □ , des différents objets.



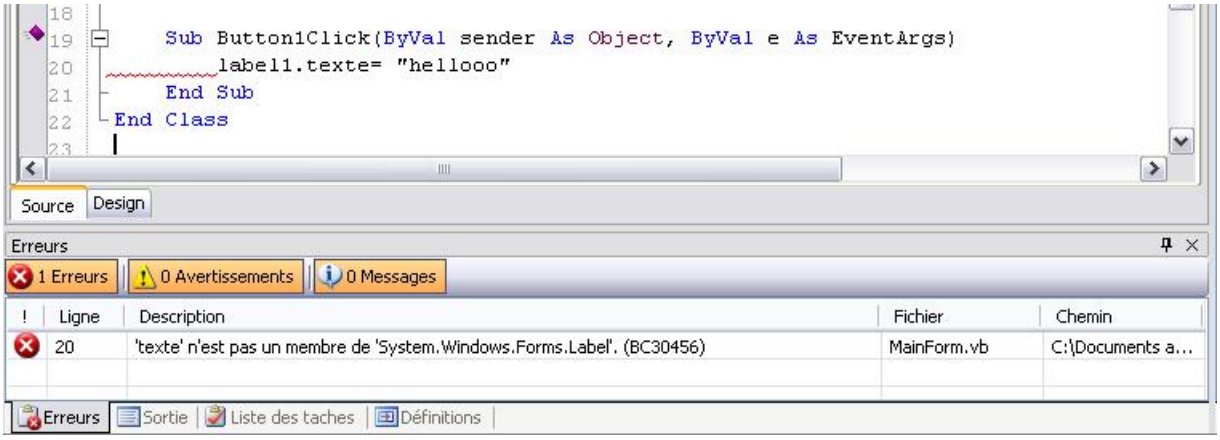
Ici par exemple on a tapé MessageBox. , la liste des membres (Equals, Show...) est affichée.

IV-B-13 - Erreur de compilation

Si on fait une faute dans le code, elle est détectée lorsque l'on lance l'exécution.

Ici on a tapé 'Texte' à la place de 'Text'.

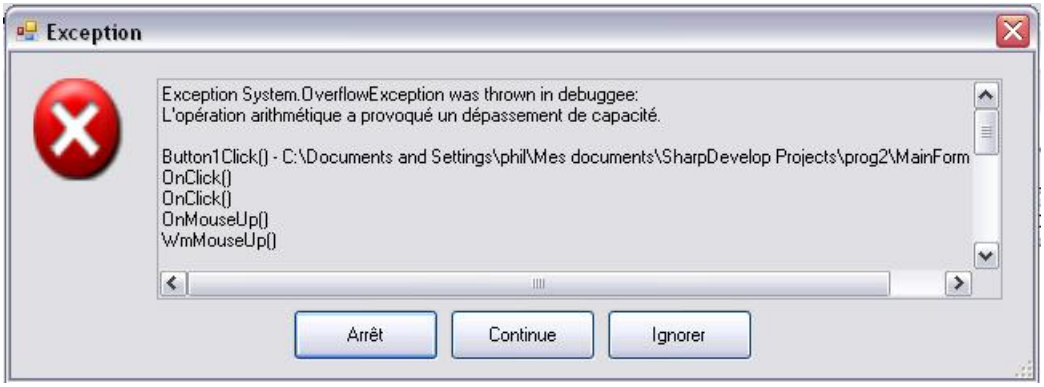
La ligne en cause est soulignée en rouge et la fenêtre des erreurs située en bas s'ouvre, elle indique et décrit l'erreur:.



L'aide dynamique à droite propose des liens en rapport avec le contexte.

IV-B-14 - Erreur d'exécution : Exceptions

Si il y a une erreur d'exécution (division par zéro par exemple), l'exécution s'arrête et la fenêtre d'exception s'ouvre:



On peut choisir d'arrêter le programme, de continuer, d'ignorer.

IV-B-15 - Débogage

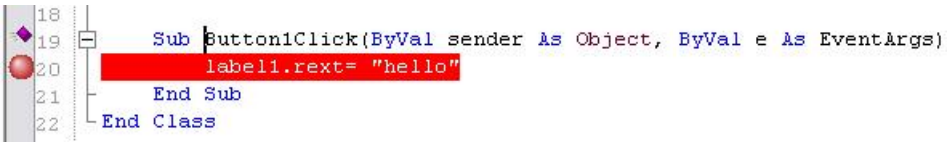
Le debogueur est maintenant intégré dans la version 2.

Une fois l'exécution lancée, on peut:

Suspendre l'exécution par ALT+CTRL+B , reprendre par F6

Ajouter des points d'arrêt.

grâce à des points d'arrêt (pour définir un point d'arrêt en mode de conception, cliquez en face d'une ligne dans la marge grise, cela fait apparaître un rond et une ligne rouge. Quand le code est exécuté, il s'arrête sur cette ligne).



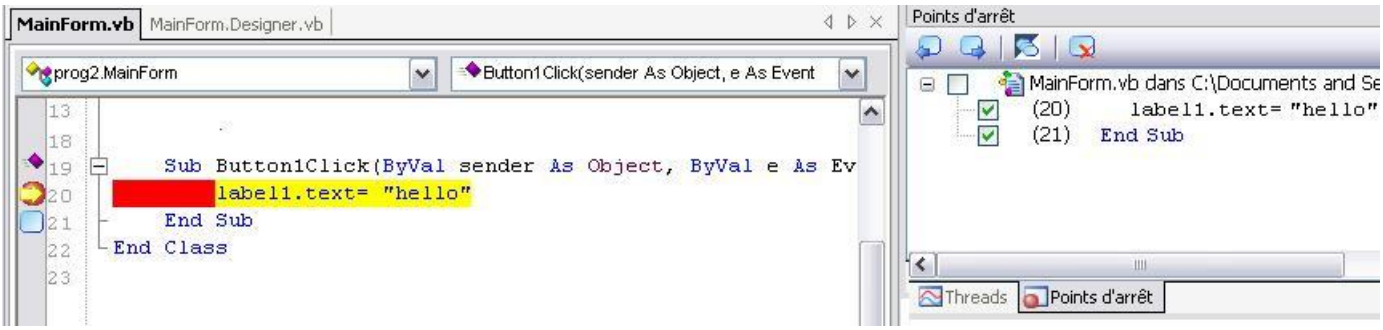
(Recliquer sur le rond pour l'enlever).

Ajouter des marques pages.

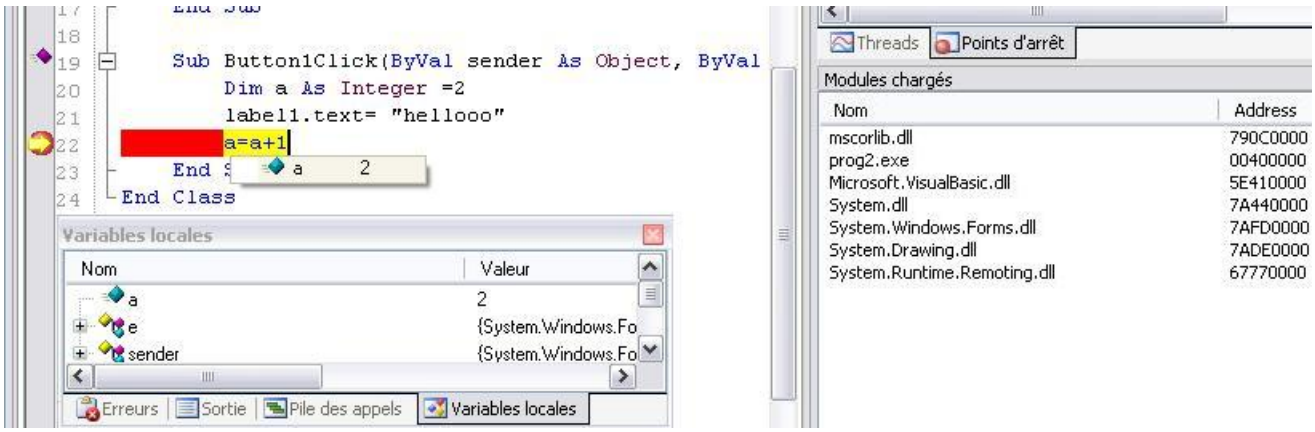
On peut ajouter des marques pages, en cliquant (quand on est sur la ligne à marquer) sur le petit carré bleu de la barre d'outils:

 Ensuite, on peut se déplacer de marque en marque avec les 2 boutons qui suivent.

En mode 'Run', si on clique sur l'onglet 'Points d'arrêt' à droite, on voit la liste des points d'arrêt et des marques; on peut rendre inactif tous les points d'arrêt (3eme bouton) ou un seul en le décochant dans la liste.



Voir la valeur d'une variable, simplement en positionnant le curseur sur cette variable.



En plus en mode Run , la fenêtre 'Variables locales' située en bas affiche la valeur de toutes les variables de la procédure. (Y compris 'e' et 'sender' qui sont les paramètres de la Sub)

Enfin à droite on peut voir les modules chargés et les threads.

Exécution pas à pas:

F11 permet l'exécution pas à pas (y compris des procédures appelées).

F10 permet le pas à pas (sans détailler les procédures appelées).

Maj+F11 exécute jusqu'à la fin de la procédure en cours.

On peut aussi utiliser les boutons : 

Attention, il n'est pas possible de modifier les fichiers sources à partir du moment où vous avez démarré le débogage.

Fonctions présentes dans #develop 1 mais pour l'instant absente dans #develop 2: C++ NProf Wix NAnt, générateur de MessageBox.

Créer un installateur (en anglais).

IV-B-16 - Conclusion

Programme permettant de faire du VB.net gratuitement (rapport qualité/prix infiniment élevé).

CONCLUSION D'UN UTILISATEUR:

SharpDevelop est un IDE agréable à utiliser, pour le développement des programmes .NET, en mode WYSIWYG.

Il est possible d'atteindre un niveau de qualité équivalent à Visual Studio ou à Borland C# Builder en faisant une installation complète. Très ouvert, on peut lui rajouter des plugins. Certains programmes externes peuvent être utilisés également avec Visual Studio ou Borland C# Builder.

SharpDevelop est en perpétuelle évolution.

Un forum permet de déposer le descriptif des erreurs rencontrées mais également de vos demandes de modifications, si vous pensez à une évolution qu'il serait bien que SharpDevelop possède. En plus vous pouvez récupérer le code source et pouvez donc modifier à loisir l'IDE.

Bien sur, pour les débutants, il manque les assistants de Visual Studio (Crystal report, ADO .NET, □). Le problème avec les assistants est qu'une fois qu'on pratique un peu, ils deviennent vite un gêne, et souvent, il faut repasser derrière eux, pour enlever le superflu de code qu'ils ont écrit (souvent ils n'optimisent pas le code).

Il manque également la partie UML de Visual Studio Architecte, mais là on attaque le haut du panier des développeurs.

Par contre, SharpDevelop apporte en plus :

- Aide à la génération automatique des MessageBox
- Aide à la conversion C# vers VB.NET et de VB.NET vers C#
- Aide à la génération d'expression régulière.

Il fournit les logiciels :

- NDoc : permet de faire des fichiers d'aide compilée au format MSDN, à partir de lignes commentées dans le code.
- NUnits : permet de faire des tests unitaires (!).
- SharpQuery : Permet de se connecter aux bases de données .

IV-B-17 - J'ai besoin d'aide

Comment créer facilement un installateur (SetUp) avec #develop? Certains utilisateurs utilisent **Inno Setup**.

Comment utiliser NDoc NUnits?

Comment utiliser simplement des ressources?

Comment utiliser des bases de données?

Qui utilise le menu 'Outils' et ses différentes options?

Merci à Fabrice SAGE pour son aide.

Merci à Hubert WENNEKES, CNRS Institut de Biologie de Lille pour son aide.

Remarque pour les forts:

On peut s'étonner qu'il n'y aie pas Handles Button1.Click à la fin de la ligne suivante (comme dans VB 2005)

```
Sub Button1Click(ByVal sender As Object, ByVal e As EventArgs)

End Sub
```

En fait si on va voir dans InitializeComponent, il y a un AddHandler après la description du bouton.

```
Private Sub InitializeComponent()

...

AddHandler Me.button1.Click, AddressOf Me.Button1Click
```

V - Langage Visual Basic

V-A - Introduction

Nous allons étudier :

Le langage Visual Basic.Net qui est utilisé dans les procédures.

Comme nous l'avons vu, le langage Visual Basic sert à

- Agir sur l'interface (Afficher un texte, ouvrir une fenêtre, remplir une liste, un tableau, poser une question).
  Exemple afficher 'Bonjour" dans un label:

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

- Effectuer des calculs, des affectations en utilisant des variables.
  Exemple: Mettre dans la variable B la valeur de A+1

```
B=A+1
```

- Faire des tests avec des structures de décision: évaluer des conditions des comparaisons et prendre des décisions.
  Exemple: SI A=1 ..

```
If A=1 Then...End If
```

- Travailler en boucle pour effectuer une tache répétitive.
  Exemple faire 100 fois..

```
For I=0 To 100... Next I
```

Tout le travail du programmeur est là.

Dans VB.Net nous avons à notre disposition 2 sortes de choses:

V-A-1 - Les Classes du framework

Le Framework (un framework est un ensemble de Classes) en est à sa version 3.5. VB les utilise.

Les classes du Framework permettront de créer des objets de toutes sortes: objet chaîne de caractères, objet image, objet fichier.... On travaille sur ses objets en utilisant leurs propriétés, leurs méthodes.

Il existe des milliers de 'Classes': les plus utilisées sont les classes String (permettant de travailler sur des chaînes de caractères), Math (permettant d'utiliser des fonctions mathématiques), Forms (permettant l'usage de formulaire, de fenêtre et donnant accès aux contrôles: bouton, case à cocher, liste..)

Elles sont communes à tous les langages utilisant le FrameWork (VB, C#, C....)

Ces Classes ont de multiples méthodes (rappel de la syntaxe: Classe.Methode).

Exemple d'utilisation de la Class TextBox (contrôle contenant du texte) et de sa méthode Text:

```
TextBox1.Text="Hello"
```

'Affiche "Hello" dans le Textbox.

Parfois la Classe n'est pas chargée par défaut au démarrage de VB, il faut dans ce cas 'l'importer' en haut du module. Si par exemple, je veux utiliser les propriétés de la classe Math, il faut écrire en haut du module:

```
Imports System.Math
```

V-A-2 - Les instructions de Microsoft.VisualBasic

Vb permet d'utiliser **des instructions Visual Basic**; seul VB peut les utiliser et de lui seul (pas C#).

Il s'agit d'instructions, de mots clé qui ont une syntaxe similaire au basic mais qui sont du VB.Net.

Exemple:

```
A = Mid(MaString, 1, 3)
```

'Mid retourne une partie de la chaîne de caractères.

Il y a aussi les **Classes de compatibilité VB6**. Elles ne dépayseront pas ceux qui viennent des versions antérieures de VB car elles reprennent la syntaxe utilisée dans VB6 et émulent les fonctions VB6 qui ont disparues de VB.Net. Ce sont des fonctions VB6 qu'on ajoute à VB.net par soucis de compatibilité, mais ce n'est pas du VB.Net. Il faut les oublier!!

L'outils d'import automatique de VB6 vers VB.Net en met beaucoup dans le code. Il faut à mon avis éviter de les utiliser car ce n'est pas vraiment du VB. Ce cours 'pur' VB.Net n'en contient pas.

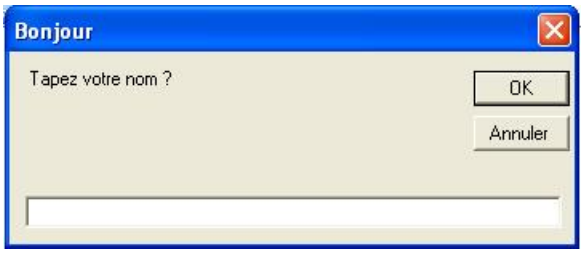
Pour le moment cela peut paraître un peu compliqué, mais ne vous inquiétez pas, cela va devenir clair.

V-A-3 - Saisir, Afficher

Dans l'étude du langage VB, on s'occupe du code, on ne s'occupe pas de l'interface (les fenêtres, les boutons, l'affichage du texte...), mais parfois, on a besoin, pour faire fonctionner des exemples, de saisir des valeurs, de les afficher:

- **Saisir une valeur**, pour cela on utilise une InputBox, c'est une boite qui s'ouvre, l'utilisateur y tape un texte puis il clique sur 'ok'; on retrouve ce qu'il a tapé dans la variable Réponse.

```
Réponse= InputBox()
```



- **Afficher des résultats**, pour le moment on affichera du texte de la manière suivante:

dans une fenêtre, dans des TextBox:

```
TextBox1.Text="TextBox1"
```

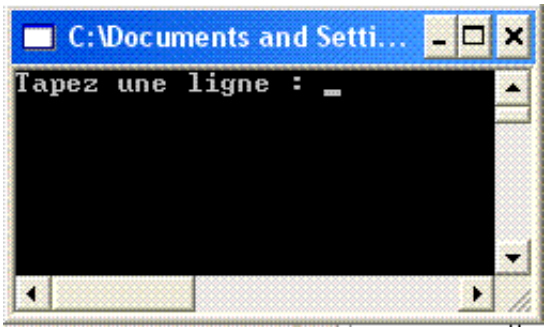
ou un label:

```
Label1.Text="Résultat"
```

ou

sur la console:

```
Consol.Out.WriteLine ("Résultat")
```



ou

dans une Boite de message:

```
MsgBox("Bonjour")
```

V-B - Les 'Algorithmes'

Ici nous allons étudiez les principes généraux de la programmation, ils sont valables pour tous les langages de programmation. Ici il faut simplement comprendre le principe de ce qui est expliqué.

V-B-1 - Pour écrire un programme

Pour écrire un programme, aller du problème à résoudre à un programme exécutable, il faut passer par les phases suivantes:

- Analyse du cahier des charges.
  Il doit être clair, exhaustif, structuré.
- Analyse générale du problème.
  Il existe des méthodes pour professionnels (MERISE, JACKSON..), nous utiliserons plutôt l'analyse procédurale: Le problème global est découpé en sous problèmes nommés fonctions. Chaque fonction ne contient plus qu'une partie du problème. Si une fonction est encore trop complexe, on itère le processus par de nouvelles fonctions à un niveau plus bas.

  Cela s'appelle la 'Conception structurée descendante'. La 'Conception ascendante' existe aussi: en assemblant des fonctions préexistantes, on résout le problème: attention, il faut que les fonctions préexistantes soient cohérentes. (Pour le moment on ne fait pas de programmation objet)
- Analyse détaillée.
  Chaque fonction est mise en forme, la logique de la fonction est écrite dans un pseudo langage (ou pseudo code) détaillant le fonctionnement de la fonction. Ce pseudo code est universel, il comporte des mots du langage courant ainsi que des mots relatifs aux structures de contrôle retrouvées dans tous les langages de programmation.
- Codage.
  Traduction du pseudo code dans le langage que vous utilisez.
- Test
  Car il faut que le programme soit valide.

Exemple simpliste:

- Analyse du cahier des charges.
  Création d'un programme affichant les tables de multiplication, d'addition, de soustraction.
- Analyse générale du problème.
  Découpons le programme en diverses fonctions:

  Il faut créer une fonction 'Choix de l'opération', une fonction 'Choix de la table', une fonction 'TabledeMultiplication', une fonction 'TabledAddition', une fonction 'Affiche'...
- Analyse détaillée.
  Détaillons la fonction 'TabledeMultiplication'

  Elle devra traiter successivement (pour la table des 7 par exemple)

  1X7
  2X7
  3X7..

  Voici l'algorithme en pseudo code.

```
Début
```

```
        Pour i allant de 1 à 10

            Ecrire (i*7)

        Fin Pour

    Fin
```

- Codage.
  Traduction du pseudo code en Visual Basic, en respectant la syntaxe du VB.

```
    Sub MultiplicationPar7

    Dim i As Integer

    For i=1 to 10

        Call Affiche(i*7)

    next i.

    End Sub
```

- Test
  Ici il suffit de lancer le programme pour voir s'il marche bien..

Pour des programmes complexes, il existe d'autres méthodes.

V-B-2 - Définition de l'algorithme

Un problème est traitable par informatique si :

- on peut parfaitement définir les données (entrées) et les résultats (sorties),
- on peut décomposer le passage de ces données vers ces résultats en une suite d'opérations élémentaires exécutables.

L'algorithme détaille, en pseudo code, le fonctionnement de ce passage et en décrit la logique.

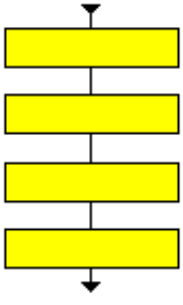
L'algorithme est une succession de tests, décisions et actions dans le but de décrire le comportement d'une entité (objet, programme, personne). Définition du Dicodunet.

On écrit bien 'algorithme' et non 'algorythme'.

Etudions cette logique valable pour tous les langages de programmation (ceux qui sont des langages impératifs):

Pour représenter n'importe quel algorithme, il faut disposer des trois possibilités suivantes:

- la **séquence** qui indique que les opérations doivent être exécutées les unes après les autres.



- le **choix** qui indique quelles instructions doivent être exécutées en fonction de circonstances.