

Prise en main de l'IDE Visual C# 2010

I - Introduction

Présentation

Un Environnement de Développement Intégré (EDI) ou IDE en anglais (Integrated Development Environment) est un logiciel regroupant au minimum les fonctionnalités suivantes : un éditeur de texte, un compilateur, des outils d'aide à la programmation et un débogueur. Un EDI fournit aussi des outils et des bibliothèques propres permettant de créer des interfaces graphiques IHM (Interface Homme-Machine en français) ou GUI (Graphical User Interface en anglais). Nous allons utiliser l'IDE Visual C++ 2010 en vous présentant successivement le menu, les différentes fenêtres de travail, la fenêtre d'édition, les barres d'outils et l'interface de débogage.

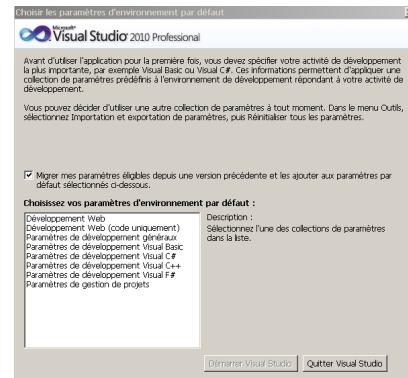
Historique

Depuis Visual 6.0 datant de 1998, Microsoft avait du mal à implanter un nouvel IDE connaissant un vif succès. Les versions qui lui succédèrent : Visual 2002 (7.0) et Visual 2003 dot net (7.1) ne générèrent pas un grand engouement. En effet, les environnements de développements des principaux concurrents (Borland, Eclipse, Linux,...) étaient depuis fort longtemps plus avancés et plus faciles à utiliser que les produits Microsoft. Ainsi, Visual C++ 6.0 malgré son côté spartiate resta un standard pendant de nombreuses années. Fin octobre 2005, Visual Studio .Net 2005 (8.0) arrivait et ce fut enfin un vif succès. Les développeurs le décrivent comme un environnement orienté vers la « productivité » permettant d'accomplir de nombreuses tâches avec facilité et rapidité. Ensuite, lui succéda, Visual 2008 (9.0) sans important changement de l'IDE et la version actuelle Visual 2010 (10.0) qui abandonna la traditionnelle couleur d'interface -gris souris- pour un mélange entre bleu roi et cyan, on n'arrête pas le progrès...

Une version allégée « Visual Express » est éditée gratuitement. Cette version contient les principaux composants utiles. Cependant, nous vous mettons en garde car les projets créés sous la version de l'école seront incompatibles avec la version Express. Les autres éditions existantes sont « Professionnal » à destination des développeurs, « Premium » permettant la gestion du travail en équipe et offrant des fonctionnalités de tests avancées puis pour terminer la version « Ultimate » ajoutant quelques fonctionnalités de débogage et de tests.

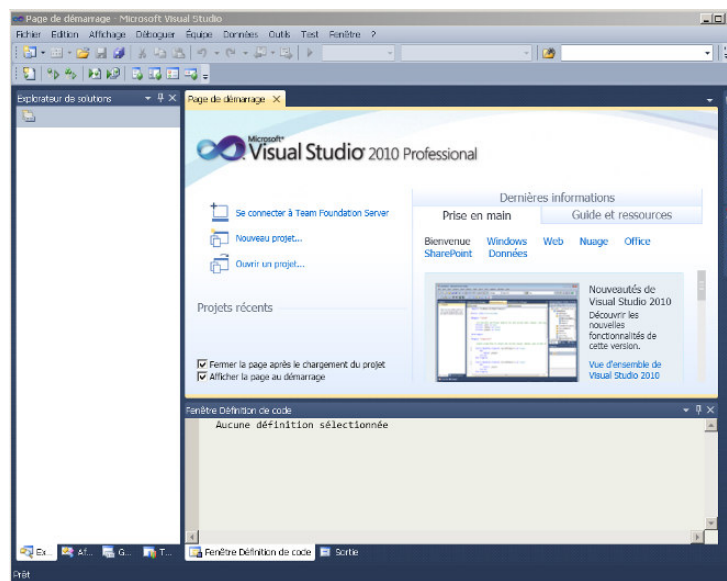
Lancez Visual C#

Le programme, si vous ne l'avez pas encore mis dans vos raccourcis du bureau peut se lancer à partir de Démarrer > Programmes > Microsoft Visual Studio 2010 > Microsoft Visual Studio 2010. La fenêtre de lancement s'ouvre. Si vous ne vous êtes jamais connecté sur la station actuelle, le démarrage peut prendre du temps car il faut créer votre profil ou initialisé le premier lancement du logiciel. Après cela, vous arrivez à une fenêtre de sélection où vous choisissez de travailler en C#.

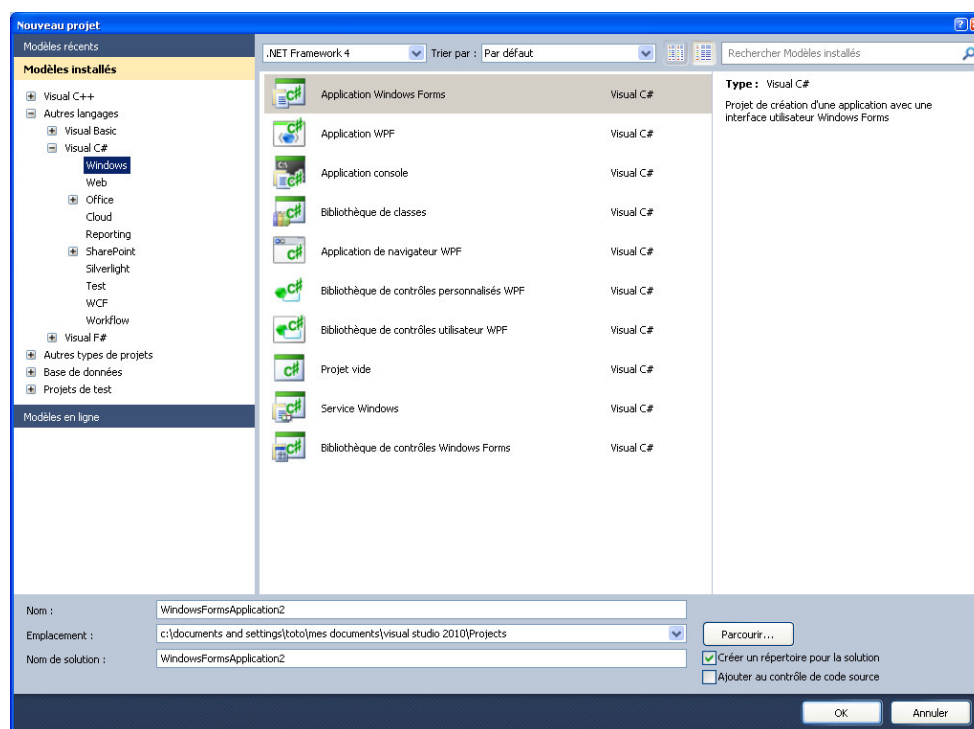


Créer son premier projet

Vous vous retrouvez face à une interface incluant diverses fenêtres, interface un peu déroutante au début :



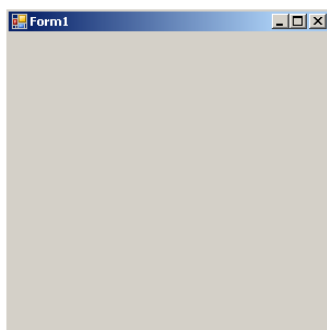
Nous allons créer un premier projet succinct nous permettant de découvrir l'environnement. Pour cela, allez dans le menu en haut à gauche et choisissez : Fichier > Nouveau > Projet.



Il existe deux grands types d'application dans le monde informatique : les applications dites « **Fenêtrées** » constituées d'une interface avec des boutons, des cases à cocher, des listes déroulantes... et des applications dites « **Console** » lancées dans une interface texte et n'utilisant pas la souris ni les graphismes. Les applications « fenêtrées » correspondent à ce que l'on rencontre tous les jours. Ces applications ne peuvent pas faire de sortie texte à partir d'une simple commande. Elles ne contiennent pas de fonction principale `main()` correspondant au démarrage du programme. Les points d'entrée de votre programme sont multiples et correspondent à des fonctions lancées suite à un évènement utilisateur du type : click sur un bouton, déplacement de la fenêtre, déplacement de la souris. A l'inverse, les applications « console » correspondent aux applications en mode texte (pensez au DOS, au terminal Linux). Elles n'ont pas de fenêtres, pas de boutons ou d'autres interfaces visuelles, elles n'utilisent pas la souris et ne peuvent pas gérer de graphisme. La programme a un unique point d'entrée représenté par la fonction `main()`. Le fonctionnement et l'architecture d'un programme « Fenêtrée » et d'un programme « Console » étant complètement différents, il vous sera impossible de migrer un projet vers un autre mode sans recréer un nouveau projet de zéro. Le choix est donc définitif.

Sous Visual, on retrouve ces deux choix. Pour les applications fenêtrées, nous utiliserons les « **Application Windows Forms** » (Fenêtre = Form) et pour les applications « Console » les « **Application Console** ». Pour continuer, donnez un nom original à votre projet (test, toto, TP...) et vérifiez que le répertoire de travail correspond à un emplacement où vous avez les droits d'écriture (C:\temp pour l'ESIEE).

Une fois ces paramètres rentrés, cliquez sur OK. Visual va alors créer plusieurs fichiers servant de squelette à votre projet. Le disque dur mouline, nous attendons quelques secondes et miracle, Visual répond à nouveau. La maquette de votre projet a été finalisée. Pour tester si tout s'est bien passé, appuyez sur **F5**, ce qui provoque la compilation et l'exécution du programme. Vous devriez voir apparaître la fenêtre suivante :



Un bug ? Un trou noir dans le système d'exploitation ? Non, une simple fenêtre vide qui attend vos modifications.

Comment gérer votre espace de sauvegarde et vos archives

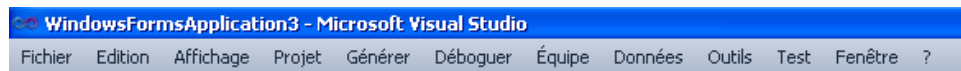
Evitez de choisir votre répertoire réseau (U:\moncompte), car les accès disque sont ralentis surtout pendant l'étape de création du programme (compilation) qui peut alors prendre plusieurs secondes à plusieurs minutes. N'oubliez pas que tout ce que vous sauvegardez sur le bureau Windows est remis à zéro quand vous vous déloguez. Voici comment procéder avec sagesse :

- En début de séance, allez chercher votre projet zippé sur votre compte réseau, copiez et collez le sur le bureau Windows ou dans C:\temp. Dézippez votre fichier, rentrez dans le répertoire et double cliquez sur le fichier portant l'extension SLN (=solution).
- Effectuez votre séance de TP
- En fin de séance, quittez Visual. Allez dans le répertoire du projet. Supprimez les sous-répertoires BIN et OBJ stockant les fichiers de compilation intermédiaires de plusieurs centaines de Mo.
- Copiez votre projet, éventuellement zippé, sur votre disque réseau ou sur votre clef USB. Changez de nom de temps en temps pour garder des archives.

C'est une manière simple et efficace de travailler. Ainsi vous conservez des versions antérieures de votre programme. Lors des dates limites où vous devez rendre vos projets par mail, vous allez effectuer la même manipulation et rendre un fichier zippé sans les répertoires BIN et OBJ. **L'archivage des données est sous votre responsabilité.** Les projets non rendus sous prétexte de disparation involontaire du répertoire de travail sont sanctionnés par une note de TP nulle.

II Le Menu

La seule chose qui ne se déplace pas dans l'interface de Visual, c'est le menu !!! Positionné en haut, juste sous le bandeau de l'application, il vous permet d'accéder à divers sous-menus :



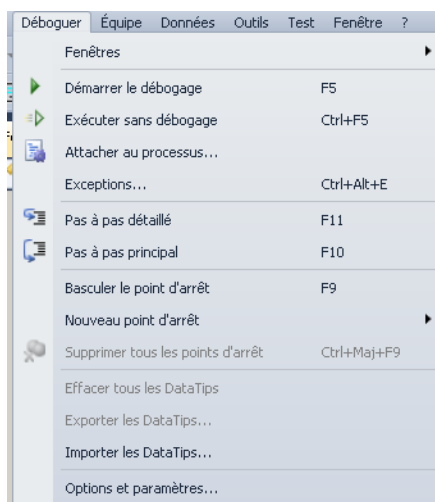
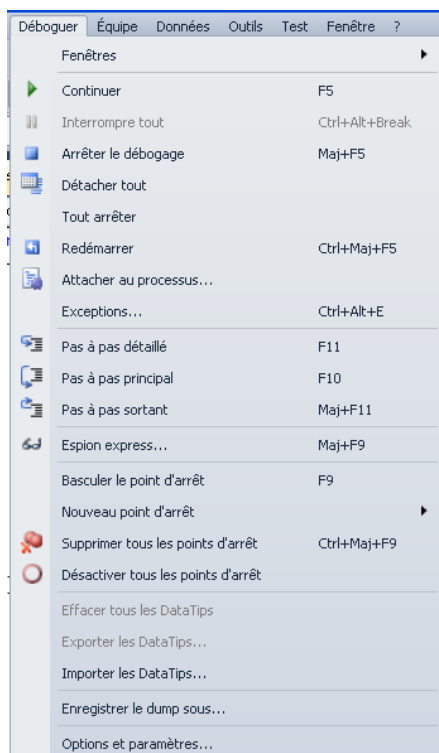
Nous rappelons que vous pouvez utiliser les raccourcis clavier pour sélectionner un sous-menu particulier. Pour cela, maintenez la touche ALT enfoncée et vous allez voir certaines lettres qui se soulignent dans les titres des sous-menus. L'appui sur la touche adéquate permet alors d'ouvrir le sous-menu correspondant. Vous pouvez aussi appuyer et relâcher la touche ALT, un encadrement bleu apparaît sur un des titres du menu. En le déplaçant grâce aux flèches du clavier et en tapant sur ENTREE vous pouvez ainsi ouvrir le sous-menu correspondant.



Le menu de Visual n'est pas si statique que l'on pourrait penser. En effet, il peut se reconfigurer suivant l'activité en cours. Plus tard dans cette fiche lorsque vous serez dans la fenêtre d'édition, un nouvel item « Refactoriser » apparaît, voir ci-dessous. Il permet de faire des recherches & remplacements de texte dans l'ensemble de votre programme.



De même, lorsque vous serez en mode débogage, le contenu de la section « Déboguer » va changer et inclure beaucoup plus d'options. Attention à ces petites farces qui parfois amènent à chercher une fonctionnalité pendant un long moment.

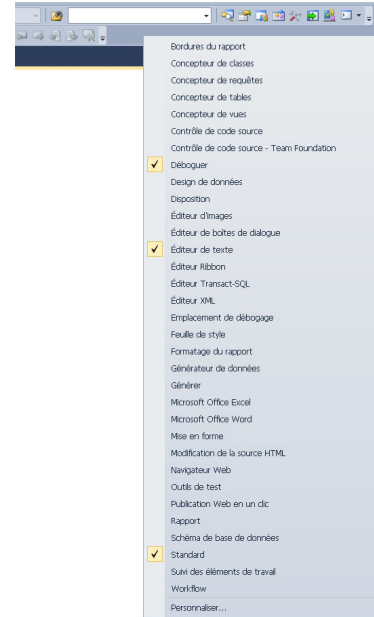


Le contenu du menu Déboguer en mode « Débogage » et en mode « Normal »

III Les barres d'outils

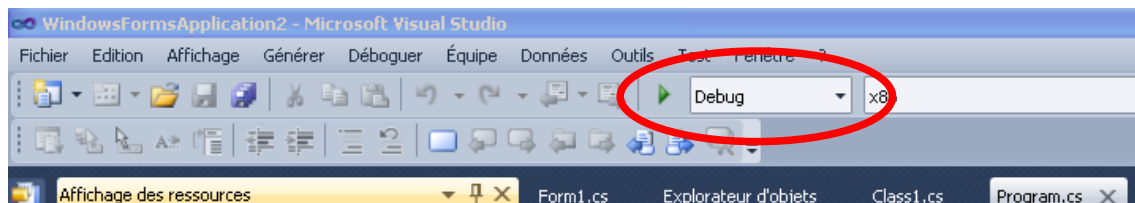
En dessous du Menu se trouve la zone des barres d'outils. Une barre d'outils regroupe une série d'icônes spécifiques à un thème (compilation, éditeur de fenêtre...). Ces différentes barres peuvent être affichées en passant par Menu > Affichage > Barre d'outils. Vous obtenez la liste complète des barres affichables en faisant un click droit n'importe où sur la zone des barres d'outils, voir figure à droite.

Peu nous seront finalement utiles. Nous utiliserons principalement la barre « Standard » et éventuellement les barres « Debug » et « Générer ». Il faut préciser que toutes les fonctionnalités accessibles par les barres d'outils sont également disponibles à l'intérieur des menus. Il s'agit juste d'une manière plus esthétique et plus pratique d'accéder aux différentes opérations.



Vous pouvez voir ici la barre Générer. A gauche se trouvent une série de points verticaux permettant de déplacer cette barre en cliquant/glissant là où il y a de la place. Vous pouvez la faire bouger dans la zone des barres d'outils, s'en suit alors un véritable jeu de domino où les barres se poussent les unes contre les autres. La flèche à droite de chaque barre correspond aux options qui permettent d'insérer de nouveaux icônes. En effet par défaut, une barre ne présente que les icônes des opérations les plus souvent utilisées. Vous pouvez ainsi personnaliser chaque barre.

Les options de compilation (Affichage > Barre d'outils > Standard)



Il existe deux types de compilation dans les environnements de développement : « Debug » et « Release ». Le mode Debug permet d'accéder aux fonctionnalités de débogage. En effet, lorsque le programme s'arrête sur un breakpoint, cela est rendu possible car des informations supplémentaires ont été

ajoutées dans le programme exécutable. Comptez une perte de 30 à 50% d'efficacité en mode Debug.

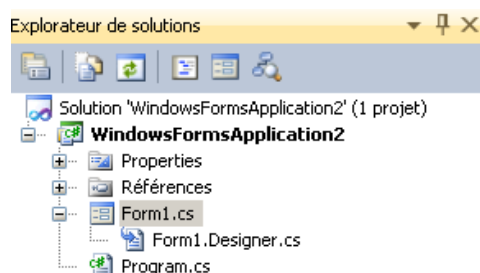
Le mode Release correspond au mode utilisé pour produire le programme distribué aux utilisateurs. Un maximum d'optimisation sont activées et le débogage est impossible, les breakpoints sont tout simplement inactifs dans ce mode. L'inconvénient majeur, hormis l'absence de débogage, est la fragilité de l'environnement face aux bugs. Ainsi, en mode Debug, lorsqu'un bug majeur stoppe votre programme, vous aurez la possibilité de reprendre la main et de diagnostiquer les variables produisant l'erreur. Mais, en mode Release, le programme s'arrêtera brutalement et l'environnement de développement pourra aussi dans la foulée subir quelques dégâts (comportement anormal, blocage des menus, impossibilité de relancer le programme...). Les deux modes coexistent et vous les choisirez suivant ce que vous voulez faire. Majoritairement, vous travaillerez 99% du temps en mode Debug et vous ne toucherez pas aux paramètres de compilation.

IV – Les fenêtres de projet

Il existe une multitude de fenêtres sous l'environnement Visual, il n'est pas rare de les perdre et de ne plus savoir comment les retrouver. Les fenêtres s'organisent en quatre positionnements : au dessus, en dessous, à droite et à gauche. Vous comprendrez rapidement que la position n'est pas un critère d'importance car leurs localisations est facilement personnalisables. Généralement ces fenêtres s'ouvrent grâce au Menu Affichage. Cependant, Visual aura pu les déplacer dans Menu > Affichage > Autres fenêtres.

L'explorateur de solutions (Menu : Affichage > Explorateur de Solutions)

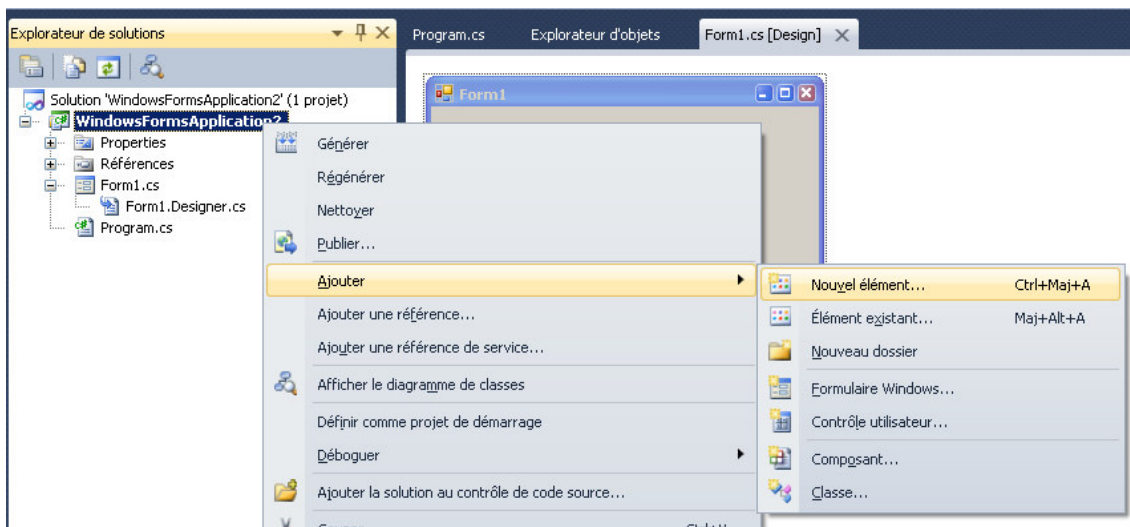
Dans le monde informatique moderne, on développe des « solutions » pour répondre aux problèmes des clients survenus suite à l'utilisation des solutions dans lesquelles ils avaient investi précédemment. Il s'agit ici d'un choix commercial visant à n'utiliser que des termes positifs face à un client pour favoriser la vente. Mais, pour nous simples développeurs, une **solution** correspond en fait à un groupement de plusieurs **projets** à l'intérieur de l'environnement de développement.



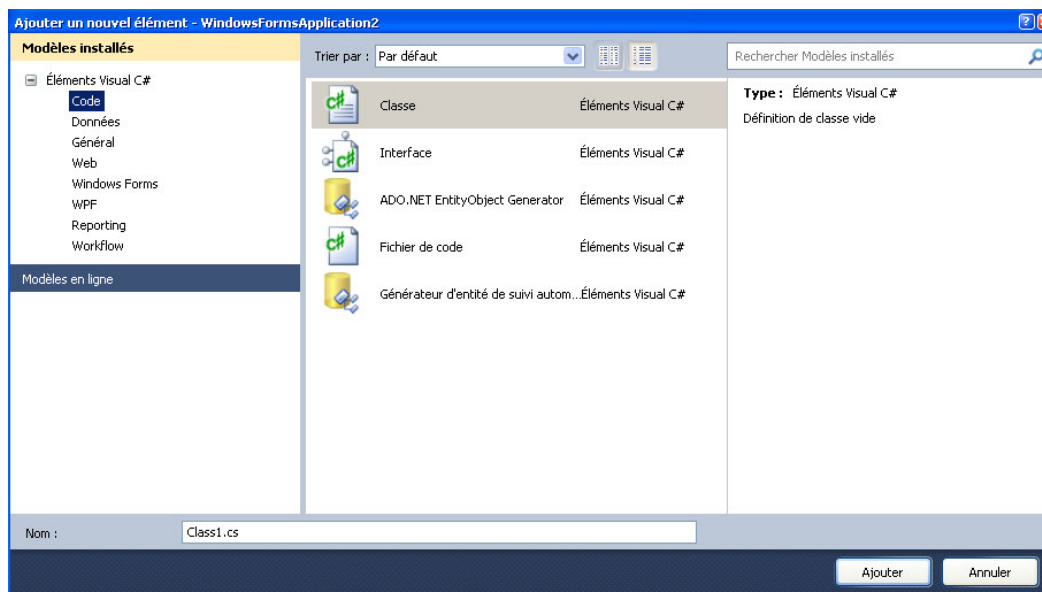
Cette fenêtre correspond à un mini explorateur de fichiers qui permet d'accéder rapidement et simplement à l'ensemble des fichiers et des ressources de votre projet. Un simple double click sur « Program.cs » a pour effet d'ouvrir directement le fichier dans la fenêtre d'édition. A vous de tester.

Important : pour ajouter des fichiers existants à votre projet vous devez utiliser cette interface. En effet, tout fichier non listé dans l'explorateur de solutions, même s'il est présent dans le répertoire du projet, sera tout simplement ignoré et bon nombre d'erreurs vont alors apparaître.

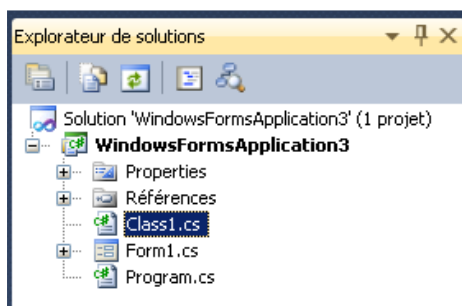
Pour créer un nouveau fichier et l'ajouter à votre projet, faites un click droit dans l'explorateur sur le nom du projet puis faites Ajouter > Nouvel élément.



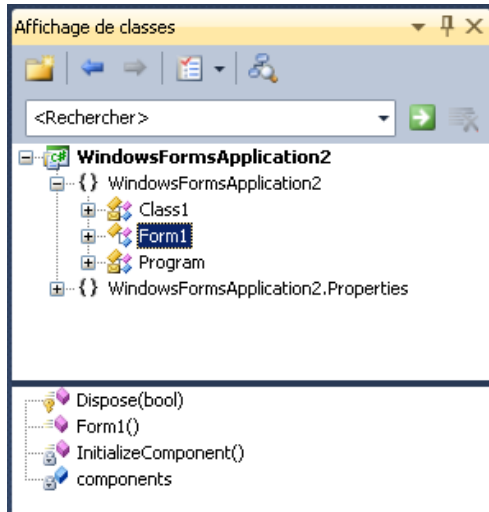
Dans la fenêtre, choisissez alors « code », puis « classe » et donnez un nom au fichier.



Votre nouveau fichier apparaît dans l'explorateur.



Si vous voulez utiliser des fichiers provenant de l'extérieur, copiez les dans le répertoire de travail et faites Ajouter > élément existant. Vous pouvez aussi ajouter plusieurs fichiers en faisant une sélection multiple dans l'interface (pour cela appuyez sur shift ou ctrl tout en cliquant sur les fichiers qui vous intéressent).

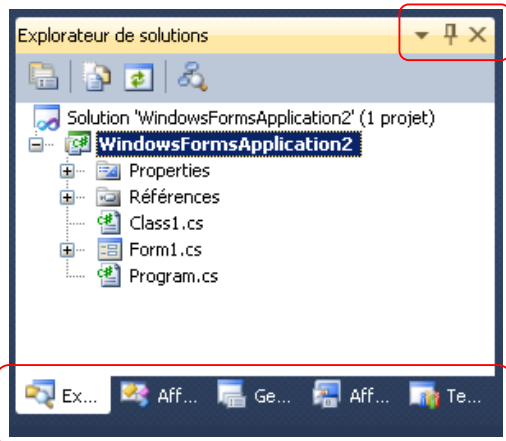


Affichage de classes (Menu : Affichage > Affichage de classes)

Cette interface ne tient plus compte de l'organisation des fichiers. Elle liste dans la fenêtre supérieure l'ensemble des classes développées dans votre projet comme un annuaire. Un click sur le nom d'une classe fournit l'affichage de ses méthodes et de ses paramètres dans la fenêtre inférieure. Un double-click sur leurs noms vous fera accéder dans la fenêtre d'édition au code correspondant. Pratique non ?

V – Disposition des fenêtres de l'environnement

Naviguer d'une fenêtre à l'autre



En ayant ouvert ces fenêtres, des onglets sont apparues en bas à gauche. Le simple click sur l'un des onglets fait basculer d'une fenêtre à l'autre.



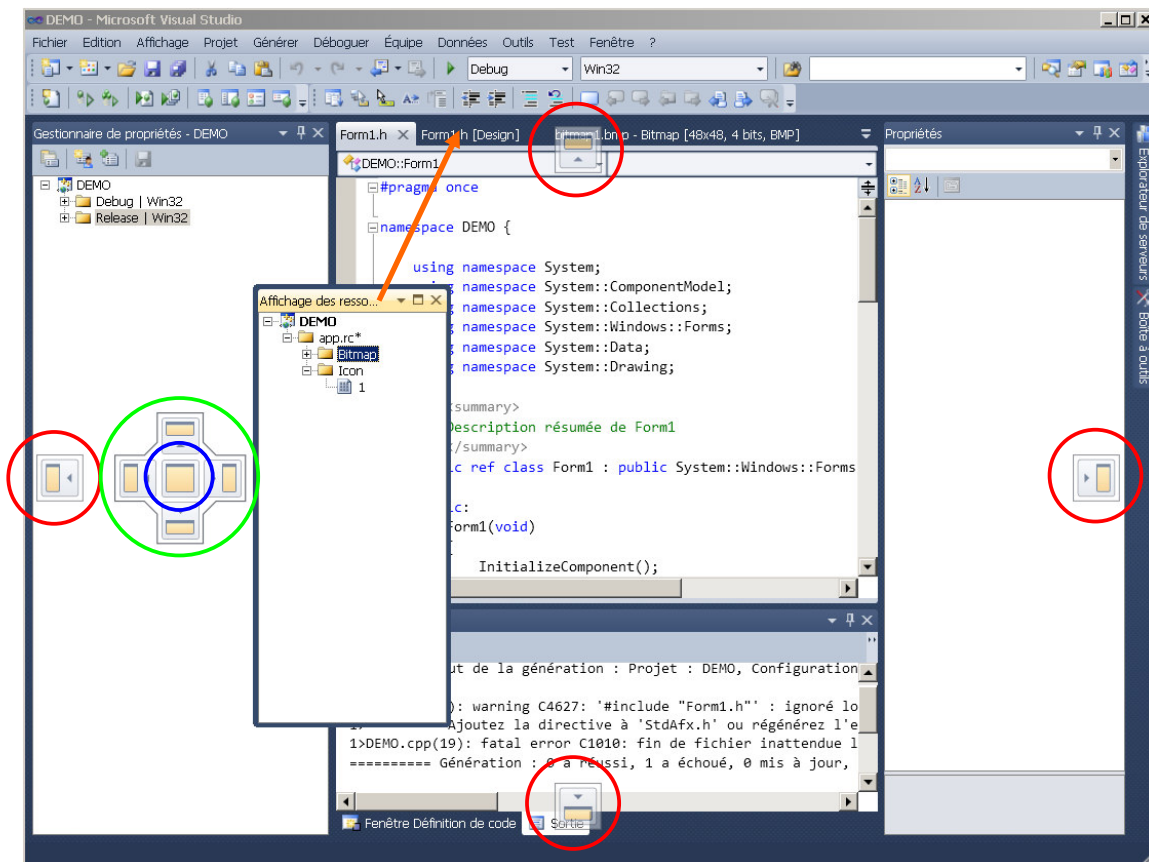
En haut à droite, vous avez trois icônes spéciaux. La croix fait disparaître l'onglet en cours (attention si vous y tenezz !!!), il faudra alors le rouvrir en passant par Menu > Affichage. La punaise permet de fixer les onglets, c'est-à-dire qu'ils sont visibles en permanence à l'écran. Si vous décliquez la punaise, les onglets vont se masquer sur le bord gauche de l'écran, voir illustration à droite.

Dans ce cas là, il suffit de passer le curseur de la souris sur le titre de l'onglet pour le faire se rouvrir, en cliquant sur la punaise vous fixez le groupe d'onglets à nouveau. Le petit triangle tête vers le bas fournit l'ensemble de ces fonctions mais sous la forme d'un menu : « masquer » veut dire fermer définitivement (!), « masquer automatiquement » est équivalent à la punaise, « Ancrer » correspond à la punaise activée et « flottante » permet de décrocher l'onglet et de le transformer en une fenêtre indépendante qui flotte au dessus de l'environnement.

Un peu de fun

Lorsqu'une fenêtre est flottante, on peut faire un click droit sur son bandeau de titre et demander à ce qu'elle devienne ancrable. La fenêtre va se repositionner à sa position précédente.

Remettez la fenêtre en mode flottant. Cliquez maintenant sur le bouton gauche de votre souris sur le bandeau de titre de la fenêtre. Ceci fait apparaître à l'écran une interface assez surprenante.



Vous trouvez 4 ancrs ici en rouge, permettant de positionner la fenêtre flottante entre le bord de l'écran et la fenêtre désignée par l'ancre. Faites glisser cette fenêtre sur une des quatre ancrs (en haut, en bas, à gauche, à droite). Lorsque vous approchez d'une ancre, l'endroit où la fenêtre va être localisée est

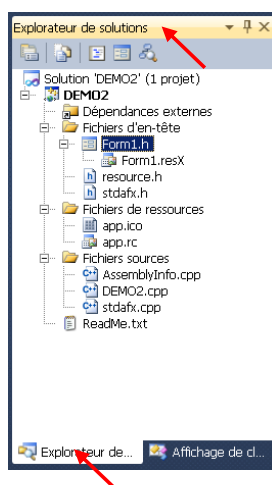
indiqué par une ombre bleue à l'écran. Si cela vous convient, vous pouvez alors relâcher le bouton de la souris.

En vert se trouve une interface de positionnement active seulement dans la fenêtre où se trouve l'icône de la souris. Elle permet de choisir comment vous voulez positionner la fenêtre flottante par rapport à cette fenêtre (en dessous, au dessus, à droite, à gauche).

Pour grouper deux fenêtres dans une seule avec un système d'onglet, il suffit de choisir le carré au centre de la fenêtre de positionnement (en bleu) ou de positionner le bandeau de la fenêtre sur le bandeau des onglets (flèche en orange).

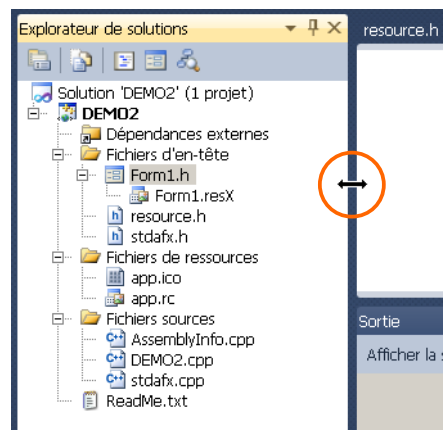
Passez un certain temps à tester les multiples possibilités de cette interface

Ps : parfois l'explorateur de solutions bogue et ouvre les fichiers non pas dans la fenêtre d'édition au centre mais dans sa propre fenêtre (petit farceur). Ce problème peut être corrigé en mettant l'explorateur en fenêtre flottante et en l'encrant tout à gauche.



Si vous voulez déplacer un groupe de fenêtres (plusieurs fenêtres regroupées grâce à des onglets), vous pouvez simplement cliquer sur le bandeau du groupe et faire glisser votre souris, le groupe devient alors flottant. Si vous voulez « décrocher » seulement une fenêtre à l'intérieur du groupe, faites la même manipulation en cliquant/glissant sur l'onglet de la fenêtre concernée.

Pour régler la position de la séparation entre deux groupes de fenêtres, il vous suffit de positionner votre curseur de souris à leur frontière, il changera alors de forme et vous pourrez par un simple cliquer/glisser faire changer la position du bord.



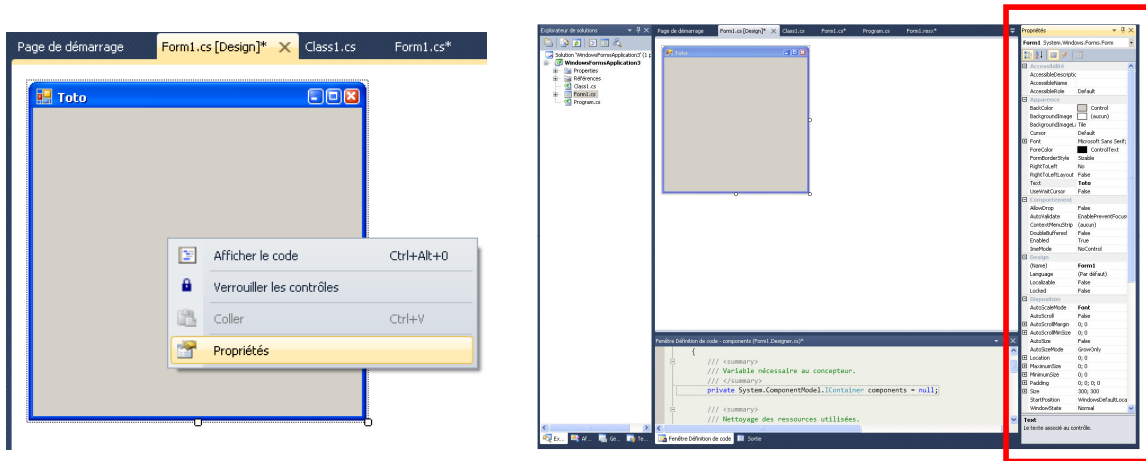
Réinitialisation de l'espace de travail

Si vous fermez une fenêtre et que vous la réactivez par le menu Affichage, elle réapparaîtra à sa dernière position. N'espérez pas récupérer les positions d'origine des fenêtres de cette manière. Pour réinitialiser les fenêtres de

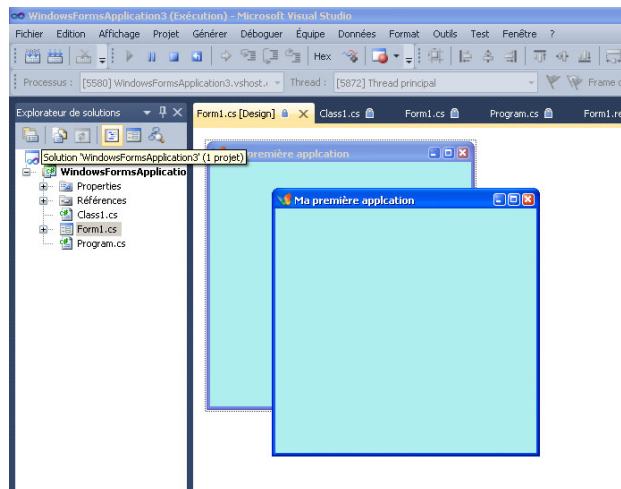
l'environnement comme elles étaient à la première ouverture de Visual, effectuez dans le Menu : Fenêtre > Rétablir la disposition de fenêtre.

VI – L'éditeur d'IHM

Vous allez réaliser votre premier programme. Dans l'explorateur de projet, double-cliquez sur « Form1.cs » pour ouvrir l'assistant d'édition de votre fenêtre. Nous allons d'abord modifier quelques propriétés basiques. Pour cela, faites un click droit sur la fenêtre et choisissez propriétés :

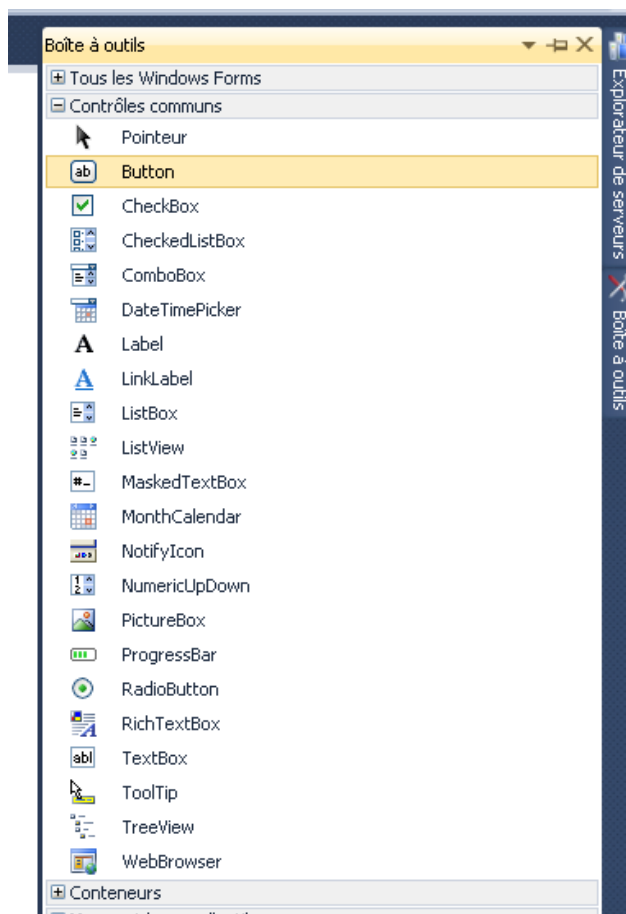


Cela fait apparaître la fenêtre des propriétés sur la droite, voir illustration. Vous pouvez changer le titre de cette fenêtre en modifiant le champ « Apparence > Text ». Vous pouvez changer la couleur de fond de la fenêtre dans le champ « Apparence > BackColor ». Vous pouvez changer l'icône en modifiant « Style de la fenêtre > Icon », une recherche sur l'ordinateur vous donnera d'autres fichiers « .ico ». En appuyant sur F5, vous devriez voir la nouvelle fenêtre d'application relookée, voir exemple ci-dessous :



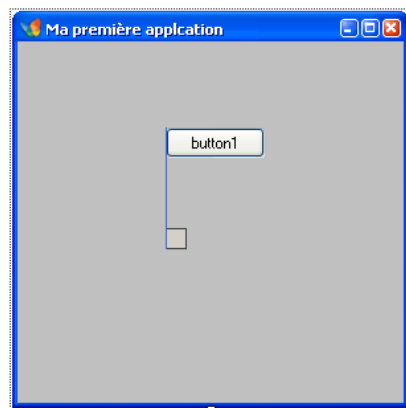
Une **IHM** (interface homme machine) est un terme générique servant à décrire toute interface visuelle interactive permettant de déclencher des évènements suite à des clics et de choisir différentes options par des listes, menus, cases à cocher.

Boite à outils : Affichage > Boites à outils – CTRL ALT X



En activant cette fenêtre, vous allez disposer d'un ensemble de composants (boutons, labels, barre de progression...) que vous allez glisser déposer dans votre maquette. Choisissez la section « Contrôles Communs », cliquez sur « Button », laissez appuyer, glissez vers votre maquette, la boite à outils peut alors disparaître, ça n'a pas d'importance, puis relâchez le bouton de la souris au milieu de votre maquette.

Recommencez cette manipulation en posant un Label sur votre maquette. Vous pouvez remarquer que l'assistant vous aide à positionner vos différents objets. En effet, si vous cherchez à vous positionner au même niveau que votre bouton juste en dessous, l'assistant va dessiner des traits d'alignement pour vous montrer si vous êtes au bon endroit. Vous pouvez ensuite cliquer sur le bouton et changez sa taille en utilisant les poignées. En ouvrant la fenêtre de propriétés, vous pouvez modifier les attributs de ce bouton. Par exemple, changez son contenu dans le champ « Apparence > Title », la police du texte dans « Apparence > Font ». En appuyant sur F5, vous devriez voir apparaître votre

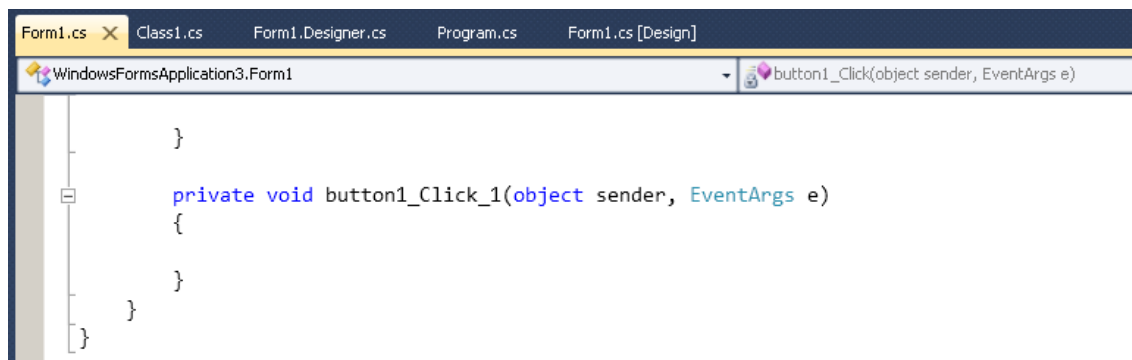


fenêtre avec ces nouveaux éléments :



En double cliquant dans la fenêtre sur une ligne où est citée une erreur, vous faites apparaître la ligne en question dans la fenêtre d'édition. De plus, le curseur est positionné sur l'erreur détectée.

Nous allons un peu programmer. Je vous rassure, rien de difficile. Une facilité fournie par les environnements de développement moderne est l'assistance à la création de code. Par exemple en double cliquant sur le bouton « Clique-moi » sur la maquette de la fenêtre, Visual crée et déclare automatiquement la fonction qui sera appelée suite à cet événement. Ensuite, il ouvre le fichier concerné dans l'éditeur et positionne le curseur à l'endroit précis où vous devez taper le code de cette fonction. Le gain en productivité et la facilité sont au rendez-vous. Vous devriez obtenir la vue suivante :



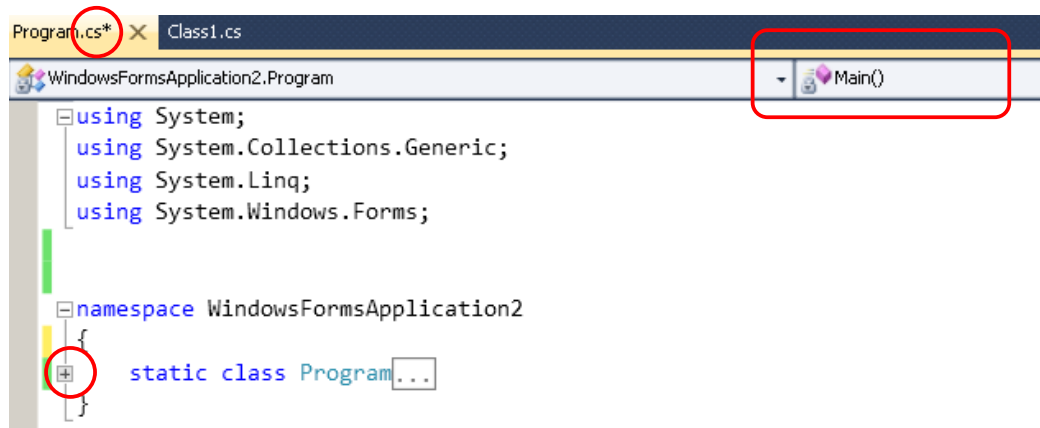
Vous allez taper le code suivant :

```
private void button1_Click_1(object sender, EventArgs e)
{
    label1.Text = "Aïe";
}
```

Le terme « label1 » désigne le nom du Label positionné sur la fenêtre dans l'assistant de conception. Si vous avez changé le nom de ce composant pour un autre, veuillez adapter le code en fonction. « .Text » permet d'accéder au champ « Apparence > Text » du bouton comme vous le faisiez précédemment dans la fenêtre de propriétés. Appuyez sur F5 et cliquez sur le bouton pour voir le résultat.

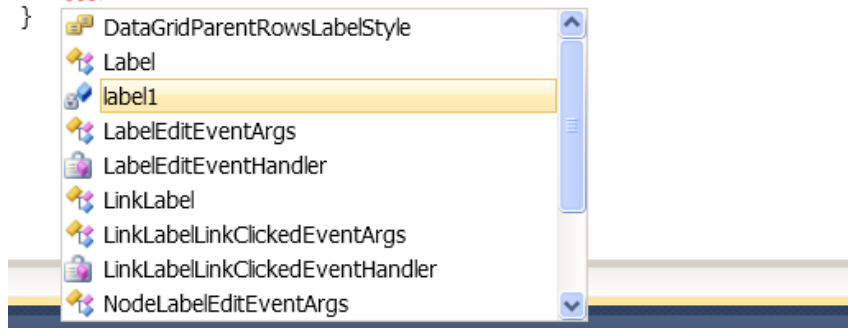
VII Fenêtre centrale d'édition (Menu : Affichage > Code)

Nous allons nous pencher sur la fenêtre d'édition de code généralement positionnée au milieu de l'écran. Pour ouvrir d'autres fichiers, cliquez sur « Program.cs » et « class1.cs » dans l'explorateur de solutions. Vous pouvez passer d'un fichier à l'autre en cliquant sur chaque onglet.



L'étoile signale que des modifications ont été effectuées dans le fichier depuis sa dernière sauvegarde. Un simple CTRL-S permet de sauvegarder le document en cours d'édition et l'étoile disparaît alors. Le +/- permet de réduire le code de la fonction sur une seule ligne. Ainsi on peut compacter simplement l'affichage du code source pour augmenter la lisibilité. En haut à droite, vous trouvez une liste contenant les fonctions du fichier, elle permet d'y accéder rapidement. Revenez dans la fonction de tout à l'heure et tapez juste en dessous de votre première ligne « lab », une des fonctionnalités d'assistance au développement va alors s'activer et vous devriez voir ceci :

```
private void button1_Click_1(object sender, EventArgs e)
{
    label1.Text = "Aïe";
    lab
}
```

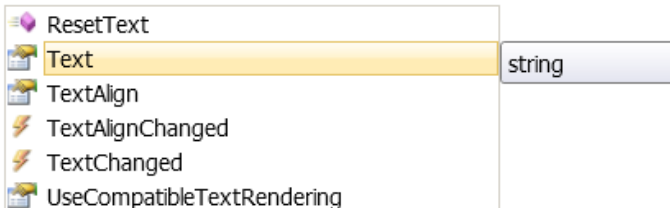


Visual vous indique tous les termes dans le projet (classes, fonctions...) commençant par lab et bien sur « label1 » y figure en bonne place, à ce niveau un simple appui sur entrée va compléter automatiquement votre texte en « label1 ». Vous devriez obtenir ceci :

```
private void button1_Click_1(object sender, EventArgs e)
{
    label1.Text = "Aïe";
    label1
}
```

Appuyez ensuite sur « . » et vous allez voir apparaître l'ensemble des propriétés et méthodes associées à votre objet. Un simple appuie sur « te » nous amène au bon champ :

```
private void button1_Click_1(object sender, EventArgs e)
{
    label1.Text = "Aïe";
    label1.te
}
```



Vous pouvez remarquer qu'il fallait utiliser une majuscule pour accéder à la propriété « Text », l'outil précédent sert aussi de correction automatique pour vous aider dans la syntaxe. Remarquer en droite le terme « String » qui apparaît 2/3 secondes après. Cela signifie que le champ « Text » est une variable de type « string » (=chaîne de caractère) et qu'elle doit donc recevoir un contenu de ce type là sinon elle va lever une erreur d'exécution. Finissez votre ligne pour obtenir cela :

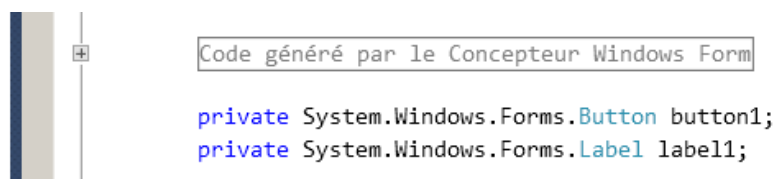
```
private void button1_Click_1(object sender, EventArgs e)
{
    label1.Text = "Aïe";
    label1.Text = "Ouille";
}
```

Le module gérant ce type de fonctionnalités s'appelle << Intellisense >> sous Visual 2010. Il facilite la vie du programmeur et permet d'augmenter la productivité. D'autres fonctions sont disponibles et vous les découvrirez plus tard. Par exemple, en tapant le nom d'une fonction, l'IntelliSense va vous rappeler les paramètres qu'elle prend, ce qui est pratique !

Remarque : Malheureusement dans la dernière version de Visual 2010, les fonctions INTELLISENSE ne sont plus actives pour les projets C++ pour les applications fenêtrées alors qu'elles fonctionnaient très bien dans les versions 2005 & 2008. La cause officielle évoquée par Microsoft est un manque de temps pour mettre à jour dans l'IntelliSense les nouvelles fonctionnalités du C++ ! Mensonge ? Réalité ? Quoiqu'il en soit, cela fournit un argument aux développeurs pour migrer vers le C#...

Nous allons étudier les fonctions permettant de « refactoriser » le code. Imaginez que vous vouliez changer le nom du label de votre application et remplacer « label1 » par « MessagePrincipal ». Cette opération doit s'effectuer à tous les endroits dans le code où « label1 » est cité et cela peut requérir quelques minutes.

Ouvrez « Form1.Designer.cs » pour accéder au code de Form1. Double-cliquez sur « label1 » en bas de la page pour sélectionner le texte :



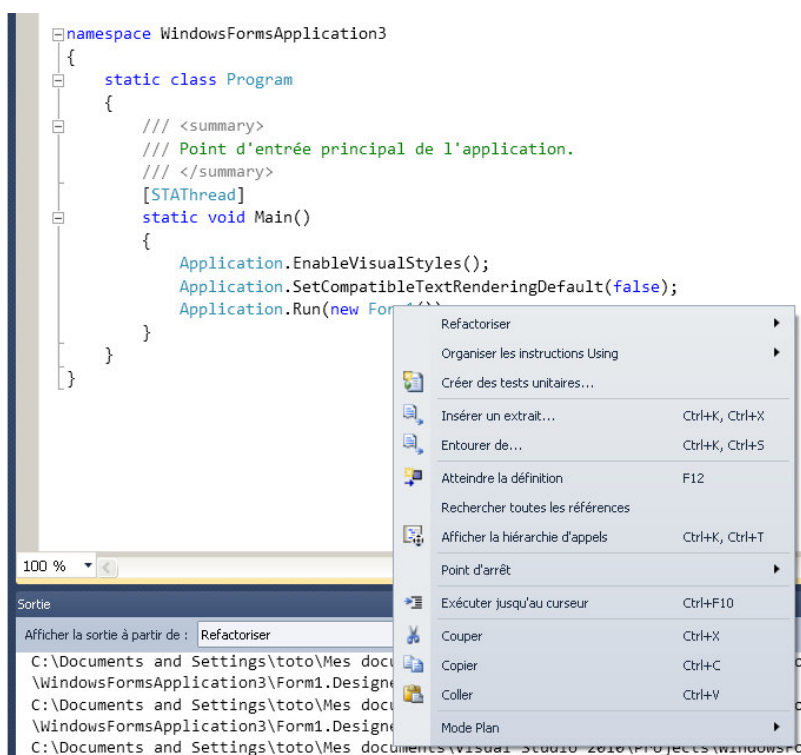
```
Code généré par le Concepteur Windows Form

private System.Windows.Forms.Button button1;
private System.Windows.Forms.Label label1;
```

Allez dans Menu > Refactoriser > Renommer, entrez « MessagePrincipal » et appuyez sur ok. Visual présente les zones du programme où « label1 » est utilisé. Un click sur OK lance le remplacement automatique. Allez dans Form1.cs et regardez votre fonction, vous devriez voir ceci :

```
private void button1_Click_1(object sender, EventArgs e)
{
    MessagePrincipal.Text = "Aïe";
    MessagePrincipal.Text = "Ouille";
}
```

Ouvrez Form1.cs et faites un click droit sur « Form1() », vous avez alors accès à diverses opérations :

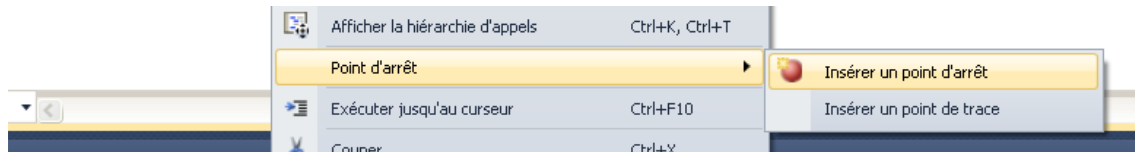


```
namespace WindowsFormsApplication3
{
    static class Program
    {
        /// <summary>
        /// Point d'entrée principal de l'application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

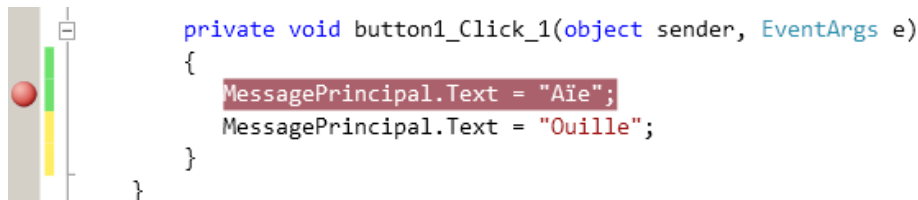
Par exemple, vous pouvez choisir « Atteindre la définition ». Sur cette ligne, la fonction Form1() est appelée, mais vous ne connaissez pas son code. En choisissant « atteindre la définition », Visual ouvre directe le fichier et vous positionne sur la zone où le code de Form1() est visible. Essayez cette fonctionnalité.

VIII Débogage

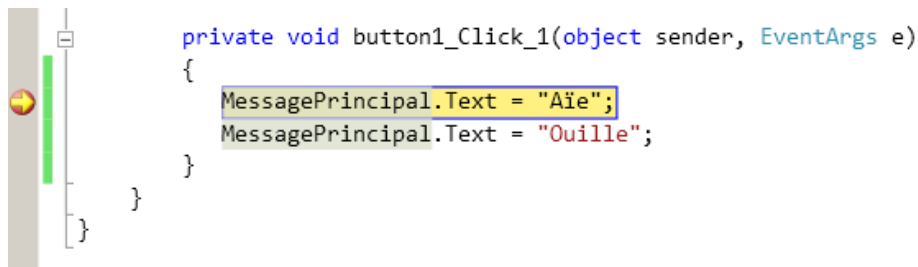
Revenez dans « form1.cs » et faites un click droit sur « MessagePrincipal.
Text = "Aïe"; » et mettez un point d'arrêt :



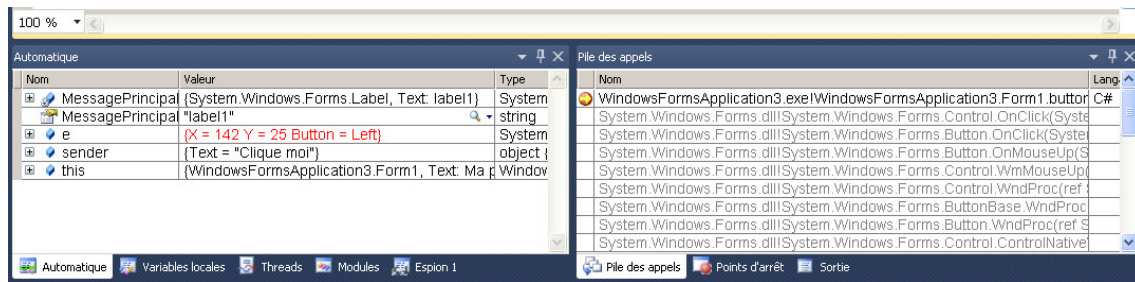
Vous devriez voir apparaître un rond rouge en marge de cette ligne :



Cliquez sur ce rond rouge, il devrait disparaître. Recliquez à nouveau il devrait réapparaître. Par la suite, vous cliquerez directement dans la marge pour faire apparaître un breakpoint. Le raccourci clavier **F9** est aussi très souvent utilisé pour placer un breakpoint sur la ligne courante. Testez-le. Lorsque le programme s'exécute en mode Debug et qu'une ligne où est positionnée un breakpoint est activée, le programme se gèle momentanément et vous pouvez alors « ausculter » vos variables pour rechercher l'origine d'une erreur. Lancez maintenant votre programme en appuyant sur **F5**. A ce niveau là, le programme est actif et il s'exécute normalement. Cliquez sur le bouton de votre fenêtre. Le breakpoint va être rencontré et le programme est stoppé momentanément. La prochaine ligne à exécuter s'affiche alors en jaune dans la fenêtre d'édition. Cette ligne n'a PAS ENCORE ETE EXECUTEE, le programme s'est arrêté juste avant.



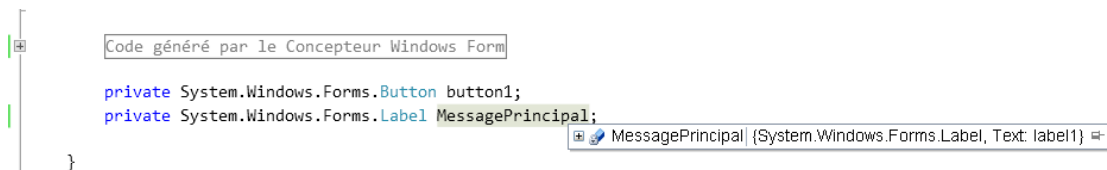
Remarquez que la fenêtre en dessous de votre fenêtre d'édition a légèrement changé :



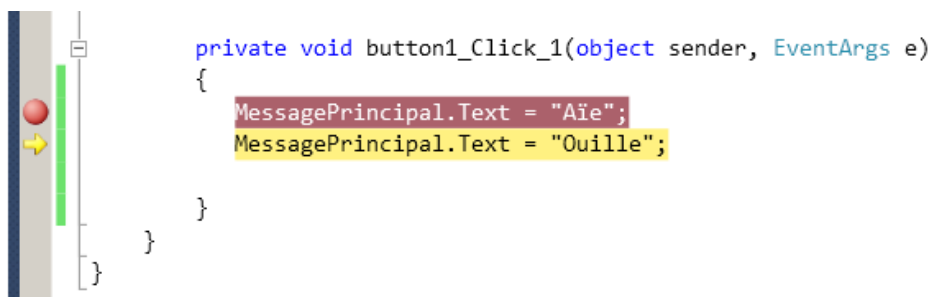
Des fenêtres de diagnostic sont apparues. Pour connaître les différentes fonctionnalités du débogage, vous aurez un TP. Nous donnons succinctement le rôle des fenêtres apparaissant :

- Variables locales / automatiques : permet de visualiser les valeurs de toutes les variables de la fonction courante
- Espion : donnez des expressions, elles seront alors évaluées en permanence. Par exemple : « sin(t) » ou tout simplement « a » si vous voulez accéder à la valeur de cette variable. Vous avez par défaut quatre fenêtres espion disponibles, appelées Espion 1 à 4.
- Pile des appels : permet de connaître l'historique des fonctions appelantes. Fonctionnalité très pratique : vous pouvez double-cliquer sur une ligne particulière dans cette fenêtre et vous accédez au code des différentes fonctions appelantes ainsi qu'aux valeurs de leurs variables. Attention en remontant trop haut, vous pouvez tomber sur des bibliothèques Windows qui n'appartiennent pas à votre programme, inutile d'aller si loin.
- Point d'arrêt : liste l'ensemble des points d'arrêts positionnés dans les différents fichiers. Permet de les activer / désactiver par un simple click en cochant ou décochant la ligne concernée. Le rond rouge prend alors un aspect creux pour signifier que le breakpoint est désactivé.

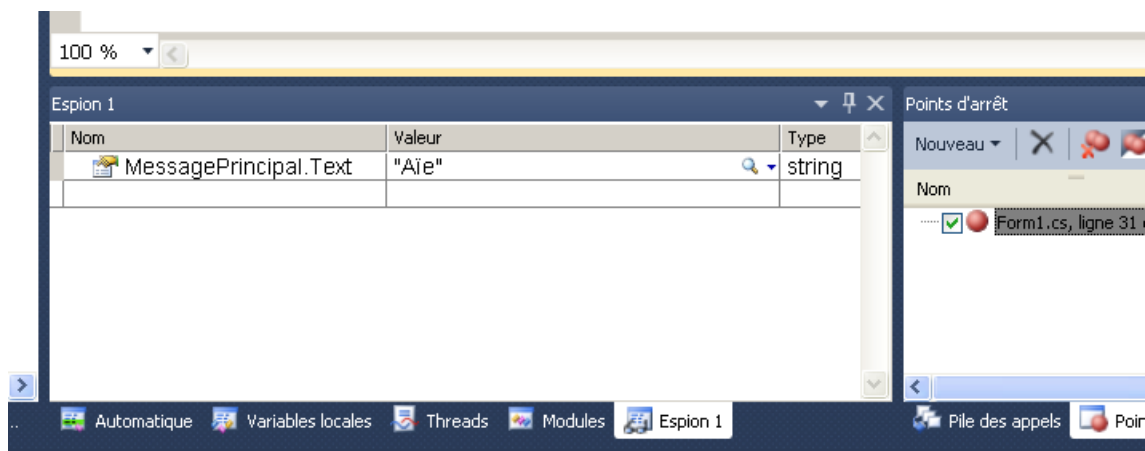
Allez dans « Form1.Designer.cs » et positionnez votre curseur au dessus de « MessagePrincipal », vous devriez voir les attributs de votre label :



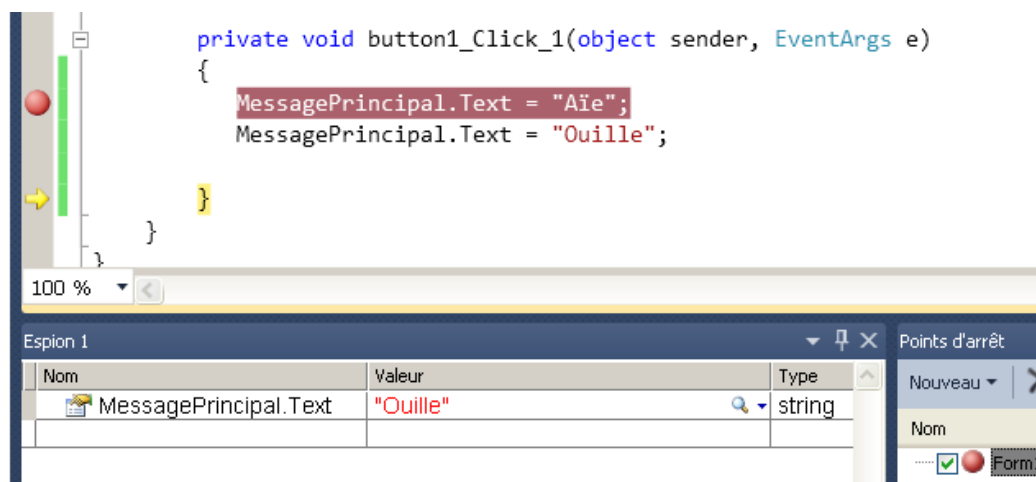
Vous remarquez que la propriété Text est alors positionnée à « label1 », ce qui correspondait à ce que l'on voyait à l'écran. Appuyez sur maintenant sur **F10**. Cette touche à pour effet de lancer l'exécution de la ligne en cours et de stopper le programme immédiatement juste après. Vous devriez voir la fenêtre suivante :



La première ligne est rouge car un breakpoint actif est positionné à cet endroit. La deuxième est jaune car elle correspond à la prochaine ligne à exécuter. Ouvrez la fenêtre espion en bas et tapez « MessagePrincipal.Text » dans le champ « Nom ». Vous devriez voir ceci au final :



L'espion vous permet de suivre le contenu d'une variable sans avoir à positionner le curseur souris sur nom comme nous l'avions fait précédemment. Les deux manières sont équivalentes ; choisissez la plus adéquate. Nous constatons que le label de votre fenêtre a bien été modifié et qu'il faut « Aïe » actuellement. Si vous cherchez à voir la fenêtre de votre programme, cela n'est pas. En effet, comme le programme est gelé à cet instant, les procédures d'affichage habituelles sont inactivées et la fenêtre est comme figée. Ainsi, l'opération « MessagePrincipal.Text = "Aïe"; » est effectuée correctement, cependant son affichage dans la fenêtre ne sera pas rafraîchi. Appuyez à nouveau sur F10. La ligne est exécutée, dans l'espion le champ a changé :



La prochaine ligne à exécuter se trouve sur l'accolade fermante, étrange non ? En effet, une accolade ne semble pas être une instruction ni une opération. En fait, en arrivant à la fin d'une fonction, beaucoup d'opérations sont effectuées implicitement : destruction de certaines variables, retour à la fonction appelante... Ces opérations sont effectuées sur la ligne portant l'accolade fermante. Appuyez

maintenant sur F5, cela relance le programme à pleine vitesse ! La fenêtre réapparaît et le contenu du label a été mis à jour.

Remarque : un click droit sur le rond rouge du breakpoint permet de lui associer une condition d'arrêt plus poussée. Cette condition doit être validée pour que le breakpoint devienne actif.

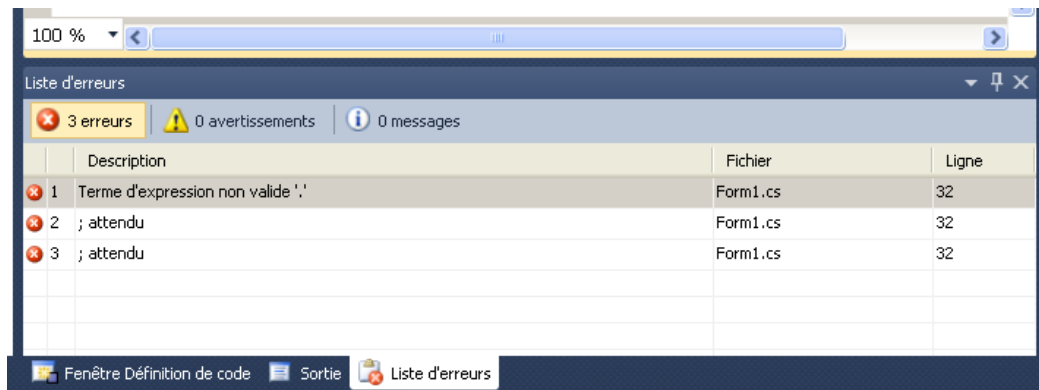
IX Compilation

Fenêtre des erreurs (Menu Affichage > Liste d'erreurs ou ALT A T)

Allez dans Form1.cs et tapez n'importe quoi dans la fonction button1_click

```
private void button1_Click_1(object sender, EventArgs e)
{
    zerzerzer
    MessagePrincipal.Text = "Aïe";
}
```

Ouvrez la fenêtre des erreurs et vous devriez voir apparaître ceci :



Lorsque des erreurs sont signalées, notamment après la compilation du programme en tapant sur la touche F5, cette fenêtre vous permet d'accéder à la liste des points posant problème. Examinez toujours la première ligne, une erreur amène plusieurs messages d'erreurs car le compilateur se perd. Inutile d'essayer de d'étudier les erreurs suivantes tant que la première n'est pas corrigée, à la correction de la première, plusieurs d'entre elles vont disparaître dans la foulée. En double cliquant sur la première ligne de cette fenêtre, le fichier correspondant est ouvert et le curseur se positionne automatiquement à la bonne ligne, très pratique à l'usage.

CONCLUSION et CONSEILS FINALS

J'espère que vous avez apprécié cette introduction assez complète mais qui présente l'essentiel des fonctionnalités d'un environnement efficace et pratique pour le développement.

Il se peut que par moment, vous ayez l'impression que Visual déraile et se met à raconter n'importe quoi. Cela arrive, erreur signalée à une ligne où il n'y a rien d'écrit, fonction retournant une valeur correspondant à l'ancienne version de cette fonction, refus de compilation alors qu'un fichier a été modifié... Il y a énormément de fonctionnalités déployées pour faciliter la vie du programmeur : auto-complétion, mise à jour en dynamique des informations dans les fenêtres, Intellisense... Lorsque ces fonctionnalités se synchronisent mal ou que l'une d'entre elles se bloque, cela peut entraîner des comportements étranges.

En cas de problème, je vous conseille alors de suivre les opérations suivantes jusqu'à résolution du problème :

- Sauvegardez tous les fichiers ouverts dans la fenêtre d'édition, c'est-à-dire Menu > Fichier > Enregistrez tout. Générez la solution (F7) ou Menu > Générer > Générez la solution. Vérifiez
- Lancez une reconstruction complète du projet. En général, cela est suffisant : Menu > Générer > Régénérez la solution. Vérifiez.
- Toujours pas... bon... Demandez un nettoyage des fichiers temporaires du projet : Menu > Générer > Nettoyer la solution. Puis Régénérez la solution. Vérifiez.
- Vous avez peut-être deux fois le même fichier ouvert dans la fenêtre d'édition. L'un contenant le code à jour et l'autre la version d'il y a une heure. Vérifiez.
- Cela résiste ? Peut-être avez-vous un fichier ouvert qui n'est pas celui du projet. Comment est ce possible ? Vous avez ouvert votre projet dans c:\temp et par la suite vous avez ouvert un fichier class1.cs sur le bureau Windows en double cliquant dessus. Ce fichier est ouvert dans votre fenêtre d'édition de Visual, vous allez entrer les modifications voulues et les sauvegardez. Cependant, à la compilation, ce sera le fichier class1.cs présent dans le répertoire du projet actuel c:\temp\class1.cs qui sera utilisé. Fermez vos fichiers dans l'éditeur et rouvrez-les à partir de l'explorateur de solutions. Vérifiez
- Toujours pas ? Un marabout s'impose. Il faut employer les grands moyens. Allez dans le répertoire du projet, effacez à la main les répertoires Release et Debug. Eteignez la machine, secouez la souris et le clavier. Relancez
- A ce niveau, je n'ai plus de conseils en stock. C'est la fin. Soit votre binôme attire vraiment la guigne, soit par chance vous avez une archive... unique bouée de secours dans cette situation.