

Partie I

# Introduction à Visual C# et Visual Studio 2008

Dans cette partie :

<b>Chapitre 1 : Bienvenue dans l'univers de C# . . . . .</b>	<b>3</b>
<b>Chapitre 2 : Variables, opérateurs et expressions. . . . .</b>	<b>29</b>
<b>Chapitre 3 : Écrire des méthodes et définir leur portée. . . . .</b>	<b>49</b>
<b>Chapitre 4 : Commandes de prise de décision . . . . .</b>	<b>67</b>
<b>Chapitre 5 : Assignment composée et instruction d'itération . . . . .</b>	<b>85</b>
<b>Chapitre 6 : Erreurs et exceptions . . . . .</b>	<b>103</b>



## Chapitre 1

# Bienvenue dans l'univers de C#

### Au terme de cette leçon, vous saurez :

- Utiliser l'environnement de programmation Microsoft Visual Studio 2008.
- Créer une application console C#.
- Expliquer le but des espaces de noms.
- Créer une application graphique simple C#.

Microsoft Visual C# est un puissant langage orienté composant créé par Microsoft. C# joue un rôle essentiel dans l'architecture de Microsoft .NET Framework, et certaines personnes ont comparé son rôle à celui joué par C dans le développement d'UNIX. Si vous connaissez déjà un langage comme C, C++ ou Java, vous trouverez que la syntaxe de C# en est très proche. Si vous êtes habitué à programmer dans d'autres langages, vous devriez rapidement vous familiariser avec la syntaxe de C# et vous n'aurez qu'à apprendre à placer les accolades et les points-virgules aux bons endroits.

La Partie I présente les bases de C#. Vous découvrirez comment déclarer des variables et comment utiliser des opérateurs arithmétiques, comme le signe plus (+) ou le signe moins (-) pour manipuler des valeurs dans les variables. Vous étudierez également la manière d'écrire des méthodes et de transformer des arguments en méthodes. Vous apprendrez aussi à vous servir des instructions de prise de décision, telles que *if*, et des instructions d'itération, comme *while*. Enfin, vous analyserez la manière dont C# utilise des exceptions pour gérer avec souplesse les erreurs. Quand vous maîtriserez toutes ces bases de C#, vous pourrez évoluer vers des fonctions plus avancées qui sont abordées dans les Parties II à VI.

## Débuter dans l'environnement Visual Studio 2008

Visual Studio 2008 est un environnement de programmation riche en outils comportant toutes les fonctionnalités nécessaires pour créer des projets C# de toute taille. Vous pouvez même créer des projets qui combinent harmonieusement des modules compilés utilisant des langages de programmation différents. Dans le premier exercice, vous lancerez l'environnement de programmation Visual Studio 2008 et vous apprendrez à concevoir une application console.



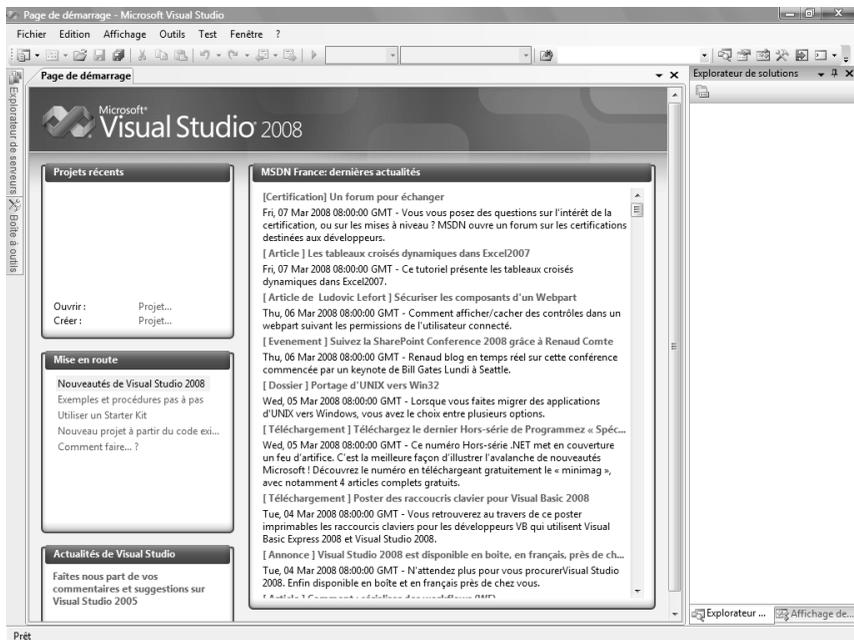
**Note** Une application console est une application qui s'exécute dans une fenêtre d'invite de commandes, et non pas dans une interface utilisateur graphique.

## Création d'une application console dans Visual Studio 2008

■ Si vous utilisez Visual Studio 2008 Standard Edition ou Visual Studio 2008 Professional Edition, effectuez les opérations suivantes pour démarrer Visual Studio 2008 :

1. Dans la barre de tâches de Windows, cliquez sur le bouton *Démarrer*, pointez *Tous les programmes*, puis pointez le groupe de programmes *Microsoft Visual Studio 2008*.
2. Dans le groupe de programmes Microsoft Visual Studio 2008, cliquez sur *Microsoft Visual Studio 2008*.

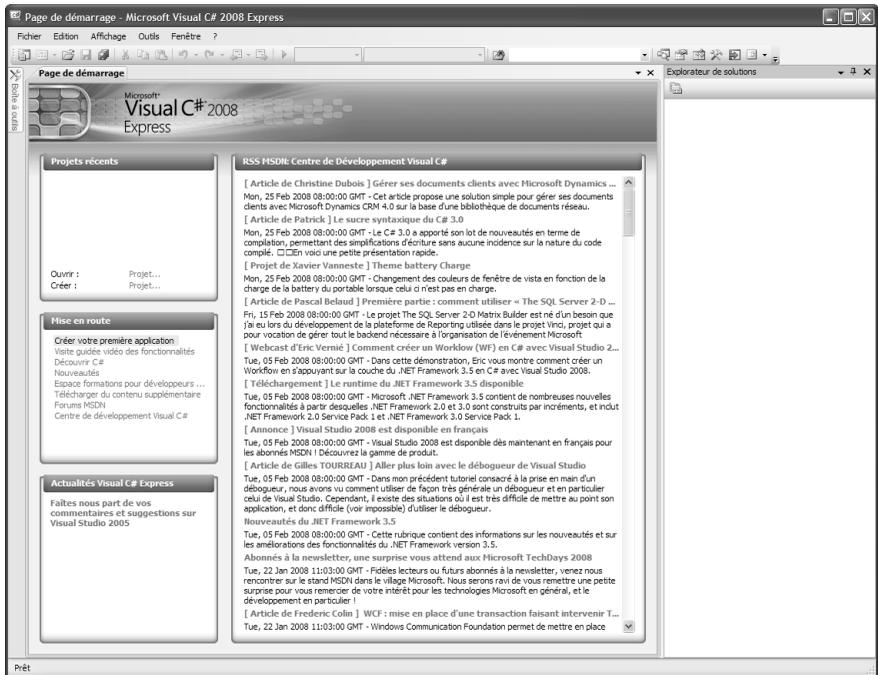
Visual Studio 2008 affiche au démarrage la fenêtre suivante :



**Note** Si c'est la première fois que vous exécutez Visual Studio 2008, vous verrez certainement une boîte de dialogue vous demandant de choisir les paramètres par défaut de votre environnement de développement. Visual Studio 2008 s'adapte à votre langage de développement favori : ainsi, les diverses boîtes de dialogue et les différents outils de l'environnement de développement intégré (ou IDE pour *integrated development environment*) ont leurs paramètres par défaut définis en fonction du langage choisi. Sélectionnez *Paramètres de développement Visual C#* dans la liste, puis cliquez sur le bouton *Démarrer Visual Studio*. Après un bref instant, l'IDE de Visual Studio 2008 apparaît.

■ Si vous utilisez Visual C# 2008 Express Edition, cliquez sur le bouton *Démarrer* dans la barre de tâches Microsoft Windows, pointez *Tous les programmes*, puis cliquez sur *Microsoft Visual C# 2008 Express Edition*.

Visual C# 2008 Express Edition affiche au démarrage la fenêtre suivante :



**Note** Afin d'éviter la répétition, tout au long du livre, j'indiquerai simplement, « Démarrer Visual Studio » lorsque vous aurez besoin d'ouvrir Visual Studio 2008 Standard Edition, Visual Studio 2008 Professional Edition, ou Visual C# 2008 Express Edition. En outre, à moins d'être clairement spécifié, toutes les références à Visual Studio 2008 s'appliquent indifféremment à Visual Studio 2008 Standard Edition, Visual Studio 2008 Professional Edition, ou Visual C# 2008 Express Edition.

- Si vous utilisez Visual Studio 2008 Standard Edition ou Visual Studio 2008 Professional Edition, effectuez les tâches suivantes pour créer une nouvelle application console.

1. Dans le menu *Fichier*, pointez *Nouveau*, puis cliquez sur *Projet*.

La boîte de dialogue *Nouveau projet* s'ouvre. Cette boîte de dialogue liste les modèles que vous pouvez utiliser comme point de départ pour construire une application. Les modèles sont présentés en fonction du langage de programmation et du type d'application que vous utilisez.

2. Dans le volet *Types de projets*, cliquez sur *Visual C#*. Dans le volet *Modèles*, cliquez sur l'icône *Application console*.

3. Dans le champ *Emplacement*, si vous utilisez le système d'exploitation Windows Vista, saisissez **C:\Utilisateurs\VotreNom\Documents\Visual C Sharp Étape par étape\Chapitre 1**. Si vous utilisez Microsoft Windows XP ou Windows Server 2003, saisissez **C:\Documents and Settings\VotreNom\Mes Documents\Visual C Sharp Étape par étape\Chapitre 1**.

Remplacez le texte *VotreNom* dans ces chemins par votre nom d'utilisateur Windows.



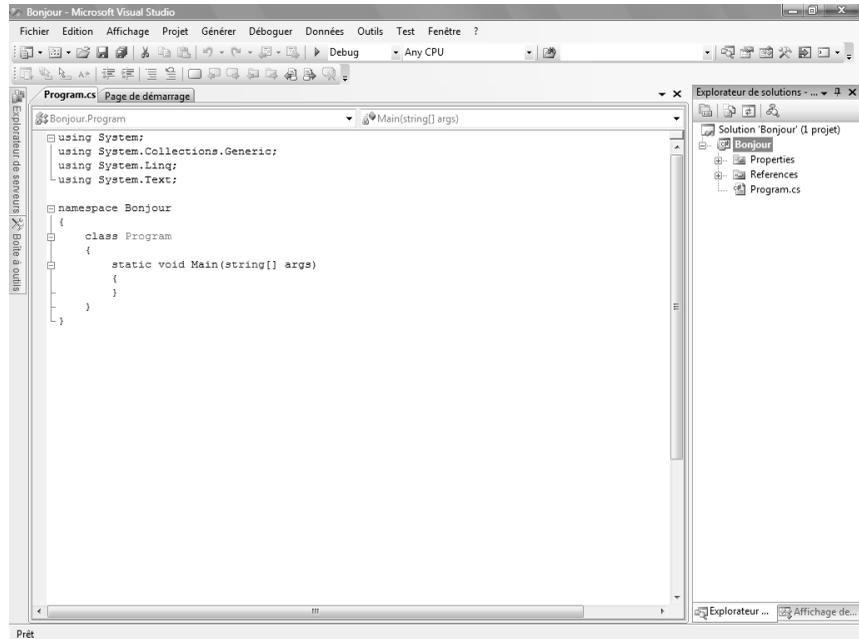
**Note** Par souci de concision, nous désignerons désormais le chemin "C:\Utilisateurs\VotreNom\Documents" ou "C:\Documents and Settings\VotreNom\Mes Documents" sous la dénomination votre dossier Documents.



**Astuce** Si le dossier que vous avez spécifié n'existe pas, Visual Studio 2008 le crée pour vous.

4. Dans le champ *Nom*, saisissez **Bonjour**.
  5. Assurez-vous que la case *Créer le répertoire pour la solution* est cochée, puis cliquez sur OK.
- Si vous utilisez Visual C# 2008 Express Edition, la boîte de dialogue *Nouveau projet* ne vous autorise pas à spécifier l'emplacement de vos fichiers de projets; il les place par défaut dans le dossier C:\Utilisateurs\VotreNom\AppData\Local\Temporary Projects. Modifiez cela en effectuant la procédure suivante :
1. Dans le menu *Outils*, cliquez sur *Options*.
  2. Dans la boîte de dialogue *Options*, cochez la case *Afficher tous les paramètres*, puis cliquez sur *Projets et solutions* dans l'arborescence du volet gauche.
  3. Dans le volet droit, dans la zone de texte *Emplacements des projets Visual Studio*, spécifiez le dossier *Visual C Sharp Étape par étape\Chapitre 1* sous votre dossier Documents.
  4. Cliquez sur *OK*.
- Si vous utilisez Visual C# 2008 Express Edition, effectuez les tâches suivantes pour créer une nouvelle application console.
1. Dans le menu *Fichier*, cliquez sur *Nouveau projet*.
  2. Dans la boîte de dialogue *Nouveau projet*, cliquez sur l'icône *Application console*.
  3. Dans le champ *Nom*, saisissez **Bonjour**.
  4. Cliquez sur *OK*.

Visual Studio crée le projet en utilisant le modèle Application console et affiche le code de démarrage suivant pour le projet :



La *barre de menus* en haut de l'écran vous permet d'accéder aux fonctions que vous utiliserez dans l'IDE. Vous pouvez utiliser le clavier ou la souris pour accéder aux menus et aux commandes. La *barre d'outils* se situe sous la barre de menus et propose des raccourcis pour exécuter les commandes les plus utilisées. La fenêtre *Code*, qui occupe la majeure partie de l'IDE, affiche le contenu des fichiers source. Dans un projet multifichier, si vous modifiez plusieurs fichiers, chaque fichier source possède son propre *onglet* avec son nom. Vous pouvez cliquer sur l'onglet pour faire apparaître au premier plan, dans la fenêtre *Code*, le fichier source. L'*Explorateur de solutions* affiche, entre autres, le nom des fichiers associés au projet. Vous pouvez faire un double-clic sur le nom d'un fichier dans l'*Explorateur de solutions* pour faire apparaître ce fichier source au premier plan dans la fenêtre *Code*.

Avant d'écrire le code, examinez les fichiers se trouvant dans la liste de l'*Explorateur de solutions*, que Visual Studio 2008 a créés comme éléments de votre projet :

- **Solution 'Bonjour'** : c'est le fichier solution qui est au sommet de la liste ; il en existe un par application. Si vous vous servez de l'Explorateur Windows pour rechercher votre dossier Documents\Visual C Sharp Étape par étape\Chapitre 1\Bonjour, vous verrez que le nom de ce fichier est en fait Bonjour.sln. Chaque fichier solution contient des références à un ou plusieurs fichiers projet.

- **Bonjour** : c'est le fichier projet C#. Chaque fichier projet répertorie un ou plusieurs fichiers contenant le code source et d'autres éléments relatifs à ce projet. L'ensemble du code source d'un projet doit être écrit dans le même langage de programmation. Dans l'Explorateur Windows, ce fichier qui s'appelle `Bonjour.csproj` est stocké dans votre dossier `\Mes Documents\Visual C Sharp Étape par étape\Chapitre1\Bonjour\Bonjour`.
- **Properties** : c'est un dossier du projet Bonjour. Si vous le déployez, vous verrez qu'il contient un fichier nommé `AssemblyInfo.cs`. Ce dernier est un fichier spécial que vous pouvez utiliser pour ajouter des *attributs* à un programme, comme le nom du créateur, la date d'écriture du programme, etc. Vous pouvez spécifier d'autres attributs pour modifier la manière dont s'exécute le programme. Apprendre à utiliser ces attributs dépasse le cadre de ce livre.
- **References** : c'est un dossier qui comporte des références au code compilé que votre application peut utiliser. Quand un code est compilé, il est converti en un *assembly* et se voit attribuer un nom unique. Les développeurs se servent d'assemblies pour regrouper des routines de code utiles qu'ils ont écrites afin de les distribuer à d'autres développeurs pour qu'ils puissent les utiliser dans leurs applications. De nombreuses fonctions, que vous utiliserez quand vous écrirez des applications avec ce manuel, se serviront d'assemblies fournis par Microsoft avec Visual Studio 2008.
- **Program.cs** : c'est un fichier source C#. C'est celui qui s'affiche dans la fenêtre Code quand le projet est créé. Vous écrirez le code de l'application console dans ce fichier qui contient du code que Visual Studio 2008 génère automatiquement.

## Écrire votre premier programme

Le fichier `Program.cs` définit une classe intitulée `Program` qui comporte une méthode appelée *Main*. Toutes les méthodes doivent être définies dans une classe. Vous en apprendrez plus sur les classes dans le chapitre 7, « Classes et objets ». La méthode *Main* est spéciale car elle correspond au point de lancement du programme. Ce doit être une méthode statique (les méthodes sont abordées en détail dans le chapitre 3, « Écrire des méthodes et définir leur portée », et les méthodes statiques sont traitées dans le chapitre 7).



**Important** C# est un langage sensible à la casse. Vous devez écrire *Main* avec un *M* majuscule.

Dans les exercices suivants, vous allez écrire du code pour afficher le message *Bonjour* sur la console, puis vous générerez et exécuterez l'application console Bonjour ; pour finir, vous étudierez la manière dont les espaces de noms sont utilisés pour séparer les éléments du code.

## Écriture du code à l'aide de la technologie IntelliSense

1. Dans la fenêtre *Code* qui affiche le fichier *Program.cs*, placez le curseur dans la méthode *Main* directement après l'accolade ouvrante, {, puis pressez Entrée pour créer une nouvelle ligne. Dans la nouvelle ligne, saisissez le mot **Console**, qui est le nom d'une classe intégrée. Quand vous saisissez la lettre C au début du mot *Console*, une liste IntelliSense apparaît. Cette liste comporte tous les mots-clés C# et les types de données valides dans ce contexte. Vous pouvez continuer votre saisie ou faire défiler la liste et effectuer un double-clic sur l'élément *Console*. Sinon, après avoir écrit *Con*, la liste IntelliSense se positionnera automatiquement sur l'élément *Console* et vous n'aurez qu'à appuyer sur la touche de tabulation ou la touche Entrée pour le sélectionner.

*Main* devrait alors ressembler à ceci :

```
static void Main(string[] args)
{
    Console
}
```



**Note** *Console* est une classe intégrée qui contient les méthodes permettant d'afficher des messages à l'écran et de récupérer les entrées depuis le clavier.

2. Mettez un point directement après *Console*. Une autre liste IntelliSense apparaît, présentant les méthodes, les propriétés et les champs de la classe *Console*.
3. Faites dérouler la liste, sélectionnez *WriteLine*, puis appuyez sur Entrée. Vous pouvez également continuer à saisir les caractères *W, r, i, t, e, L* jusqu'à ce que *WriteLine* soit sélectionné, puis appuyer sur Entrée.

La liste IntelliSense se ferme et le mot *WriteLine* est ajouté au fichier source. *Main* devrait maintenant ressembler à ceci :

```
static void Main(string[] args)
{
    Console.WriteLine
}
```

4. Saisissez une parenthèse ouvrante, (. Une autre astuce IntelliSense s'affiche.

Celle-ci présente les paramètres de la méthode *WriteLine*. En fait, *WriteLine* est une *méthode surchargée*, ce qui signifie que la classe *Console* contient plus d'une méthode appelée *WriteLine* (elle fournit en fait 19 versions différentes de cette méthode). Chaque version de la méthode *WriteLine* peut être utilisée pour transmettre en sortie différents types de données (les méthodes surchargées sont abordées au chapitre 3). *Main* devrait maintenant ressembler à ceci :

```
static void Main(string[] args)
{
    Console.WriteLine(
}
```



**Astuce** Vous pouvez cliquer sur les flèches de l'astuce pour faire défiler les différentes versions de *WriteLine*.

5. Saisissez une parenthèse fermante, `)`, suivie d'un point-virgule, `;`.

*Main* devrait maintenant ressembler à ceci :

```
static void Main(string[] args)
{
    Console.WriteLine();
}
```

6. Déplacez le curseur, et saisissez la chaîne **"Bonjour"**, sans oublier les guillemets, entre les parenthèses suivant la méthode *WriteLine*.

*Main* devrait maintenant ressembler à ceci :

```
static void Main(string[] args)
{
    Console.WriteLine("Bonjour");
}
```



**Astuce** Habituez-vous à saisir des paires de caractères correspondantes, comme ( et ) ou { et }, *avant* d'insérer leur contenu. Il est fréquent d'oublier le caractère fermant si vous saisissez d'abord le contenu.

## Icônes IntelliSense

Lorsque vous saisissez un point après le nom d'une classe, IntelliSense affiche le nom de chaque membre de cette classe. À gauche de chaque nom de membre se trouve une icône qui indique le type de celui-ci. Voici les principales icônes et leurs types :

Icône	Signification
	méthode (abordée dans le chapitre 3)
	propriété (abordée dans le chapitre 15)
	classe (abordée dans le chapitre 7)
	structure (abordée dans le chapitre 9)
	énumération (abordée dans le chapitre 9)
	interface (abordée dans le chapitre 13)

Icône	Signification
	délégué (abordé au chapitre 17)
	méthode d'extension (abordée dans le chapitre 12)

Vous verrez également d'autres icônes IntelliSense apparaître lorsque vous saisissez du code dans différents contextes.



**Note** Vous verrez fréquemment des lignes de code contenant deux caractères slash consécutifs suivis de texte ordinaire. Ce sont des commentaires. Ils sont ignorés par le compilateur, mais sont très utiles pour les développeurs, puisqu'ils permettent de documenter les actions d'un programme. Par exemple :

```
Console.ReadLine(); // Attendre que l'utilisateur appuie sur la touche Entrée
```

Tout le texte à partir des caractères slash jusqu'à la fin de la ligne sera ignoré par le compilateur. Vous pouvez ajouter des commentaires sur plusieurs lignes en commençant par un slash suivi d'un astérisque (/\*). Le compilateur ignorera tout ce qui se trouve jusqu'à la prochaine séquence (\*/), qui peut se trouver plusieurs lignes plus bas. Vous êtes vivement encouragé à documenter votre code en y insérant autant de commentaires explicatifs que nécessaire.

## Génération et exécution de l'application console

1. Dans le menu *Générer*, cliquez sur *Générer la solution*.

Cette action compile le code C#, créant un programme que vous pouvez exécuter. La fenêtre *Sortie* apparaît sous la fenêtre *Code*.



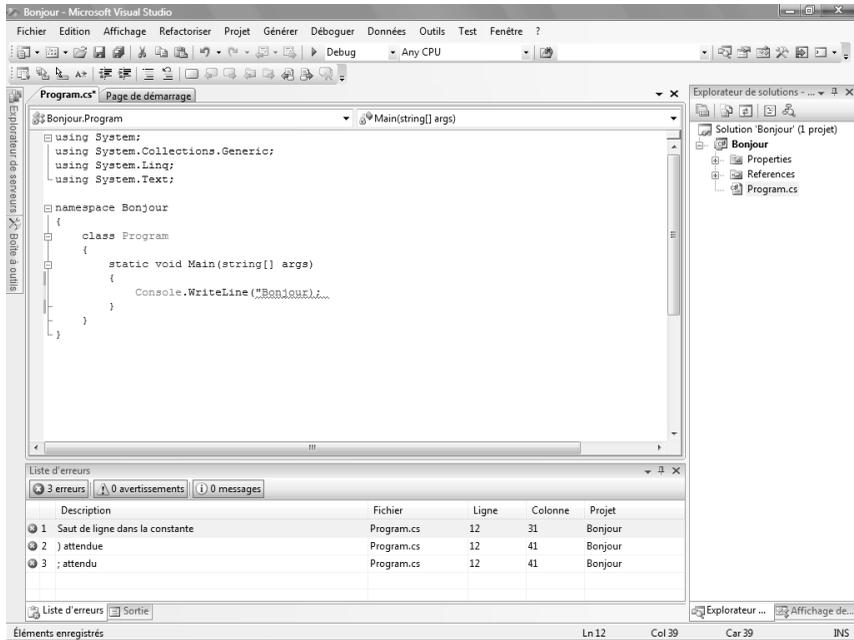
**Astuce** Si la fenêtre *Sortie* n'apparaît pas, dans le menu *Affichage*, cliquez sur *Sortie* pour la faire apparaître.

Dans la fenêtre *Sortie*, vous devez voir des messages similaires à celui reproduit ci-dessous qui indique comment le programme a été compilé :

```
----- Début de la génération : Projet : Bonjour, Configuration : Debug Any CPU -----
C:\Windows\Microsoft.NET\Framework\v3.5\Csc.exe /noconfig /nowarn:1701,1702 ...
Compilation terminée -- 0 erreurs, 0 avertissements
Bonjour -> C:\Users\Dominique\Documents\Visual C Sharp Étape par étape\...
===== Génération : 1 a réussi ou est à jour, 0 a échoué, 0 a été ignoré =====
```

Si vous avez fait des erreurs, elles apparaîtront dans la fenêtre *Liste d'erreurs*. La figure suivante montre ce qui se passe si vous oubliez de saisir le guillemet fermant

après le texte *Bonjour* dans la commande *WriteLine*. Notez bien qu’une seule erreur peut parfois provoquer plusieurs erreurs de compilation.



**Astuce** Vous pouvez faire un double-clic sur un élément dans la fenêtre *Liste d'erreurs*, et le curseur sera alors placé sur la ligne qui provoque l'erreur. Vous remarquerez aussi que, lors de la saisie, Visual Studio affiche un trait ondulé rouge sous les lignes de code qu'il ne compile pas.

Si vous avez soigneusement suivi les instructions précédentes, il ne devrait pas y avoir d'erreurs ou d'alertes, et le programme devrait se générer avec succès.

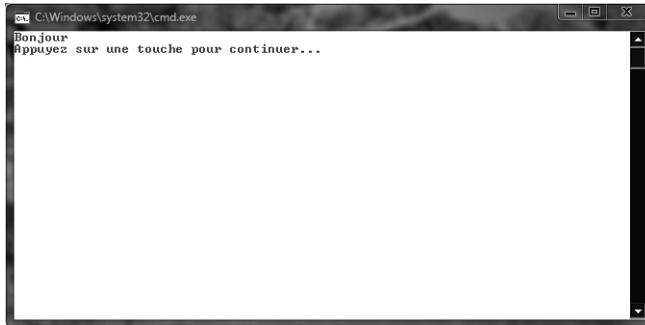


**Astuce** Vous n'avez pas besoin d'enregistrer le fichier de façon explicite avant la génération car la commande *Générer la solution* enregistre automatiquement le fichier. Si vous utilisez Visual Studio 2008 Standard Edition ou Visual Studio 2008 Professional Edition, le projet est enregistré dans l'emplacement spécifié au moment de sa création. Si vous utilisez Visual C# 2008 Express Edition, le projet est enregistré dans un emplacement temporaire puis est copié vers le dossier que vous avez spécifié dans la boîte de dialogue *Options*, uniquement si vous enregistrez de façon explicite le projet en utilisant la commande *Enregistrer tout* du menu *Fichier* ou si vous fermez Visual C# 2008 Express Edition.

Un astérisque après le nom du fichier dans l'onglet au-dessus de la fenêtre *Code* indique que le fichier a été modifié depuis sa dernière sauvegarde.

2. Dans le menu *Débuguer*, cliquez sur Exécuter sans débogage.

Une fenêtre de Commande s'ouvre et le programme s'exécute. Le message Bonjour apparaît, puis le programme attend que l'utilisateur appuie sur n'importe quelle touche, comme l'illustre la figure suivante :



**Note** L'invite "Appuyez sur une touche pour continuer. . ." est générée par Visual Studio ; vous n'avez pas écrit de code pour faire cela. Si vous exécutez le programme au moyen de la commande *Démarrer le débogage* dans le menu *Déboguer*, l'application s'exécute, mais la fenêtre de commande se ferme immédiatement sans attendre que vous appuyiez sur une touche.

3. Assurez-vous que la fenêtre de commande affichant la sortie du programme est active, puis appuyez sur Entrée.

La fenêtre de commande se ferme et vous êtes à nouveau dans l'environnement de programmation de Visual Studio 2008.

4. Dans l'*Explorateur de solutions*, cliquez sur le projet Bonjour (pas la solution), puis cliquez sur le bouton *Afficher tous les fichiers* dans la barre d'outils *Explorateur de solutions* (il s'agit du deuxième bouton à partir de la gauche dans la barre d'outils de la fenêtre Explorateur de solutions).

Les entrées nommées *bin* et *obj* apparaissent au-dessus du fichier Program.cs. Ces entrées correspondent directement aux dossiers intitulés *bin* et *obj* dans le dossier du projet (Visual C Sharp Étape par étape\Chapitre 1\Bonjour\Bonjour). Visual Studio crée ces dossiers lorsque vous générez votre application et ils contiennent la version exécutable du programme et d'autres fichiers utilisés pour générer et déboguer l'application.

Dans l'*Explorateur de solutions*, cliquez sur le signe plus (+) à gauche de l'entrée *bin*. Un autre dossier nommé *Debug* apparaît.

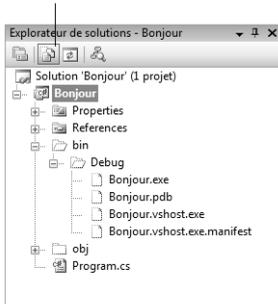


**Note** Il est possible que vous voyiez aussi un dossier nommé *Release*.

5. Dans l'*Explorateur de solutions*, cliquez sur le signe plus (+) à gauche du dossier *Debug*.

Quatre autres éléments nommés Bonjour.exe, Bonjour.pdb, Bonjour.vshost.exe, et Bonjour.vshost.exe.manifest apparaissent de la manière suivante :

Afficher tous les fichiers



**Note** Si vous utilisez Visual C# 2008 Express Edition, il est possible que vous ne puissiez pas voir tous ces fichiers.

Le fichier Bonjour.exe correspond au programme compilé, et il s'agit du fichier qui s'exécute lorsque vous cliquez sur *Exécuter sans débogage* dans le menu *Déboguer*. Les autres fichiers contiennent des informations utilisées par Visual Studio 2008 si vous exécutez votre programme en mode *Débogage* (lorsque vous cliquez sur *Démarrer le débogage* dans le menu *Déboguer*).

## Utiliser des espaces de noms

L'exemple que vous venez d'étudier correspond à un tout petit programme. Toutefois, les petits programmes peuvent rapidement devenir plus importants. Au fur et à mesure de leur développement, deux problèmes surgissent. Tout d'abord, il est bien plus difficile de comprendre et de maintenir des programmes importants que des programmes plus petits. Deuxièmement, l'augmentation de la taille du code implique en général plus de noms, plus de méthodes, et plus de classes. Plus le nombre de noms augmente, plus il y a de risques d'échec lors de la génération du projet, puisque plusieurs noms peuvent entrer en conflit (notamment quand le programme utilise des bibliothèques tierces écrites par des développeurs qui ont également utilisé une grande variété de noms).

Dans le passé, les programmeurs ont tenté de résoudre ce problème de conflit de noms en préfixant les noms avec une sorte de qualificatif (ou des ensembles de qualificatifs). Cette solution n'est pas appropriée puisqu'elle n'est pas évolutive ; les noms s'allongent et vous passez moins de temps à écrire du code et plus de temps à saisir (ce n'est pas la même chose), à lire et à relire des noms incompréhensiblement longs.

Les espaces de noms vous aident à résoudre ce problème en créant un conteneur nommé pour les autres identificateurs comme les classes. Deux classes portant le même nom ne

pourront plus être confondues si elles se trouvent dans des espaces de noms différents. Vous pouvez créer une classe appelée *Salut* dans l'espace de noms intitulé *Bonjour*, comme ceci :

```
namespace Bonjour
{
    class Salut
    {
        ...
    }
}
```

Vous pouvez ensuite faire référence à la classe *Salut* sous la forme *Bonjour.Salut* dans vos programmes. Si un autre développeur crée également une classe *Salut* dans un autre espace de noms, comme *NouvelEspaceDeNoms* et l'installe sur votre ordinateur, vos programmes fonctionneront quand même normalement, puisqu'ils utilisent la classe *Bonjour.Salut*. Si vous voulez faire référence à la classe *Salut* de l'autre développeur, vous devez la spécifier sous la forme *NouvelEspaceDeNoms.Salut*.

Il est judicieux de définir toutes vos classes dans des espaces de noms et l'environnement Visual Studio 2008 suit ces recommandations en utilisant le nom de votre projet comme espace de noms de niveau supérieur. Le SDK du .NET Framework adhère également à cette recommandation ; chaque classe du .NET Framework se trouve dans un espace de noms. Par exemple, la classe *Console* se situe dans l'espace de noms *System*. Cela signifie que son nom complet correspond à *System.Console*.

Bien entendu, si vous deviez écrire à chaque fois le nom complet d'une classe que vous avez utilisée, cela ne présenterait pas plus d'intérêt que de préfixer avec des qualificatifs ou bien que de nommer la classe avec un nom unique comme *SystemConsole* sans s'encombrer d'un espace de noms. Heureusement, vous pouvez résoudre ce problème grâce à une directive *using* dans vos programmes. Si vous retournez dans le programme *Bonjour* dans Visual Studio 2008 et examinez le fichier *Program.cs* dans la fenêtre *Code*, vous remarquerez les instructions suivantes au début du fichier :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

L'instruction *using* inclut un espace de noms dans la portée. Dans le code qui suit dans le même fichier, vous n'êtes plus obligé de qualifier explicitement des objets avec l'espace de noms auquel ils appartiennent. Les quatre espaces de noms indiqués contiennent des classes qui sont utilisées si souvent que Visual Studio 2008 ajoute automatiquement ces instructions *using* à chaque fois que vous créez un nouveau projet. Vous pouvez ajouter d'autres directives *using* au début d'un fichier source.

L'exercice suivant approfondit le concept d'espace de noms.

## Test des noms pleinement qualifiés

1. Dans la fenêtre *Code* affichant le fichier *Program.cs*, mettez en commentaire la première directive *using* au début du fichier de la manière suivante :

```
//using System;
```

2. Dans le menu *Générer*, cliquez sur *Générer la solution*.

La génération échoue, et la fenêtre *Liste d'erreurs* affiche le message d'erreur suivant :

```
Le nom 'Console' n'existe pas dans le contexte actuel.
```

3. Dans la fenêtre *Liste d'erreurs*, faites un double-clic sur le message d'erreur.

L'identificateur qui provoque l'erreur est sélectionné dans le fichier source *Program.cs*.

4. Dans la fenêtre *Code*, modifiez la méthode *Main* pour utiliser le nom complet qualifié *System.Console*.

*Main* devrait ressembler à ceci :

```
static void Main(string[] args)
{
    System.Console.WriteLine("Bonjour");
}
```

Note Lorsque vous saisissez *System.*, les noms de tous les éléments dans l'espace de noms *System* sont affichés par IntelliSense.

5. Dans le menu *Générer*, cliquez sur *Générer la solution*.

La génération devrait être réalisée avec succès cette fois-ci. Si ce n'est pas le cas, assurez-vous que *Main* apparaît exactement comme dans le code précédent, puis essayez à nouveau de générer.

6. Exécutez l'application pour vous assurer qu'elle fonctionne encore en cliquant sur *Exécutez sans débogage* dans le menu *Déboguer*.

### Espaces de noms et assemblies

Une instruction *using* place simplement les éléments d'un espace de noms dans la portée si bien que vous n'avez plus besoin de qualifier pleinement les noms des classes dans votre code. Les classes sont compilées dans des *assemblies*. Un assembly est un fichier qui a habituellement l'extension *.dll*, bien les programmes exécutables ayant l'extension *.exe* soient aussi des assemblies.

Un assembly peut contenir de nombreuses classes. Les classes de la bibliothèque de classes du .NET Framework, comme *System.Console*, sont fournies dans les assemblies qui sont installés sur votre ordinateur avec Visual Studio. Vous constaterez que la bibliothèque de classes du .NET Framework contient plusieurs milliers de classes. Si elles se trouvaient toutes dans le même assembly, l'assembly

serait énorme et difficile à maintenir. De plus, si Microsoft mettait à jour une méthode dans une classe, il serait dans l'obligation de distribuer l'ensemble de la bibliothèque de classes à tous les développeurs !

Pour cette raison, la bibliothèque de classes du .NET Framework se décompose en un certain nombre d'assemblies où les classes sont regroupées de manière thématique. Par exemple, il y a un assembly "core" qui contient toutes les classes courantes, telles que *System.Console*, et il y a d'autres assemblies qui contiennent des classes pour manipuler des bases de données, pour accéder aux services Web, pour générer des interfaces utilisateurs graphiques, etc. Si vous souhaitez utiliser une classe d'un assembly, vous devez ajouter à votre projet une référence à cet assembly. Vous pouvez ensuite ajouter des instructions *using* à votre code qui place les éléments des espaces de noms de cet assembly dans la portée.

Vous noterez qu'il n'y a pas nécessairement une stricte équivalence entre un assembly et un espace de noms ; un assembly peut contenir des classes de plusieurs espaces de noms, et un espace de noms peut couvrir plusieurs assemblies. De prime abord, cela peut sembler déconcertant, mais vous allez vite vous y habituer.

Lorsque vous utilisez Visual Studio pour créer une application, le modèle que vous sélectionnez comporte automatiquement des références vers les assemblies qui conviennent. Par exemple, dans l'*Explorateur de solutions* du projet Bonjour, cliquez sur le signe plus (+) à gauche du dossier *References*. Vous verrez alors qu'une application console inclut automatiquement des références à des assemblies nommés *System*, *System.Core*, *System.Data*, et *System.Xml*. Vous pouvez ajouter à un projet des références à d'autres assemblies en faisant un clic droit sur le dossier *References* et en cliquant sur *Ajouter une référence*.

## Créer une application graphique

Jusqu'à présent, vous avez utilisé Visual Studio 2008 pour créer et exécuter une application console minimale. L'environnement de programmation Visual Studio 2008 contient également tout ce dont vous avez besoin pour créer des applications graphiques Windows. Vous pouvez concevoir l'interface utilisateur d'une application Windows de manière interactive en créant un formulaire. Visual Studio 2008 génère alors les instructions du programme pour implémenter l'interface utilisateur que vous avez conçue.

Visual Studio 2008 offre deux modes d'affichage pour une application graphique : *Concepteur* et *Code*. On utilise la fenêtre *Code* pour modifier et maintenir le code et la logique de l'application, et on se sert de la fenêtre *Concepteur* pour concevoir l'interface utilisateur. Vous pouvez basculer entre les deux affichages quand vous le souhaitez.

Dans les exercices suivants, vous allez apprendre à créer une application graphique en utilisant Visual Studio 2008. Celui-ci affichera un formulaire simple comportant une zone

de texte où vous pouvez saisir votre nom et un bouton qui, quand vous cliquez dessus, affiche un message d'accueil personnalisé dans une boîte de dialogue.



**Note** Visual Studio 2008 fournit deux modèles pour générer des applications graphiques : le modèle Windows Forms Application et le modèle WPF Application. Windows Forms est une technologie qui est apparue avec la version 1.0 du .NET Framework. WPF, qui est l'acronyme de Windows Presentation Foundation, est une technologie améliorée qui est apparue pour la première fois avec la version 3.0 du .NET Framework. WPF fournit de nombreuses fonctionnalités supplémentaires par rapport à Windows Forms, et c'est pourquoi vous devez le privilégier pour tout nouveau développement.

### Création d'une application graphique dans Visual Studio 2008

- Si vous utilisez Visual Studio 2008 Standard Edition ou Visual Studio 2008 Professional Edition, effectuez les opérations suivantes pour créer une nouvelle application graphique :

1. Dans le menu *Fichier*, pointez *Nouveau*, puis cliquez sur *Projet*.  
La boîte de dialogue *Nouveau projet* s'ouvre.
2. Dans le volet *Types de projets*, cliquez sur *Visual C#*.
3. Dans le volet *Modèles*, cliquez sur l'icône *Application WPF*.
4. Assurez-vous que le champ *Emplacement* fait bien référence à votre dossier *Documents\Visual C Sharp Étape par étape\Chapitre 1*.
5. Dans le champ *Nom*, saisissez **WPFBonjour**.
6. Dans le champ *Solution*, assurez-vous que *Créer une nouvelle solution* est sélectionné.  
Cette action crée une nouvelle solution qui contiendra le projet. Vous pouvez aussi employer la commande *Ajouter à la solution*, qui ajoute le projet à la solution Bonjour.

7. Cliquez sur *OK*.

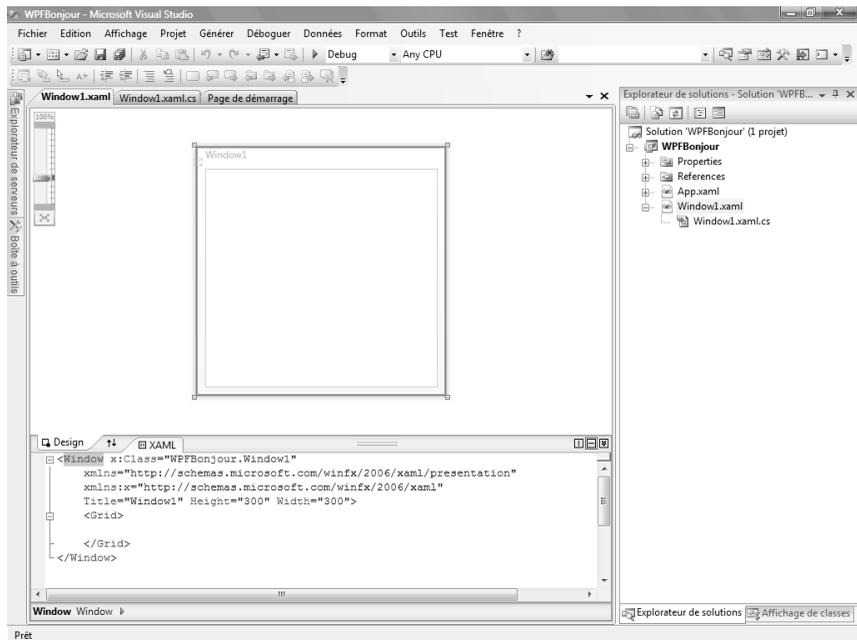
- Si vous utilisez Visual C# 2008 Express Edition, effectuez les tâches suivantes pour créer une nouvelle application graphique.

1. Dans le menu *Fichier*, cliquez sur *Nouveau projet*.
2. Si la boîte de dialogue *Nouveau projet* apparaît, cliquez sur *Enregistrer* pour sauvegarder les modifications du projet Bonjour. Dans la boîte de dialogue *Enregistrer le projet*, vérifiez que le champ *Emplacement* fait référence à *Visual C Sharp Étape par étape\Chapitre 1* sous votre dossier Documents, puis cliquez sur *Enregistrer*.
3. Dans la boîte de dialogue *Nouveau projet*, cliquez sur l'icône *Application WPF*.
4. Dans le champ *Nom*, saisissez **WPFBonjour**.
5. Cliquez sur *OK*.

Visual Studio 2008 ferme votre application en cours et crée une nouvelle application WPF. Il affiche un formulaire WPF vide dans la fenêtre *Concepteur*, avec une autre fenêtre contenant une description XAML du formulaire, comme cela est illustré à la figure suivante :



**Astuce** Si vous fermez les fenêtres *Sortie* et *Liste d'erreurs*, vous obtiendrez alors plus d'espace pour afficher la fenêtre *Concepteur*.



XAML, qui signifie Extensible Application Markup Language, est un langage XML utilisé par les applications WPF pour définir la mise en page d'un formulaire et son contenu. Si vous connaissez déjà XML, XAML devrait vous paraître familier. Vous pouvez en fait définir complètement un formulaire WPF en écrivant une description XAML si vous n'aimez pas utiliser la fenêtre *Concepteur* de Visual Studio ou si vous n'avez pas accès à Visual Studio ; Microsoft fournit un éditeur XAML appelé XMLPad que vous pouvez télécharger gratuitement à partir du site Web MSDN.

Dans l'exercice suivant, vous utiliserez la fenêtre *Concepteur* pour ajouter trois contrôles à un formulaire Windows et vous examinerez une partie du code C# généré automatiquement par Visual Studio 2008 pour implémenter ces contrôles.

### Création de l'interface utilisateur

1. Cliquez sur l'onglet *Boîte à outils* qui apparaît à gauche du formulaire dans la fenêtre *Concepteur*.

La Boîte à outils apparaît, masquant partiellement le formulaire et affichant les divers composants et contrôles que vous pouvez placer dans un formulaire Windows. La section Commun affiche une liste de contrôles qui sont utilisés par la plupart des applications WPF. La section Contrôles affiche une liste plus importante de contrôles.

2. Dans la section Commun, cliquez sur Label, puis cliquez sur la partie visible du formulaire.

Un contrôle Label est ajouté au formulaire (vous le déplacerez à l'emplacement souhaité dans un moment), et la *Boîte à outils* disparaît.



**Astuce** Si vous souhaitez que la *Boîte à outils* reste visible, mais ne masque aucune partie du formulaire, cliquez sur le bouton *Masquer automatiquement* à droite dans la barre de titre de la *Boîte à outils* (il ressemble à une punaise). La *Boîte à outils* apparaît en permanence à gauche de la fenêtre Visual Studio 2008, et la fenêtre *Concepteur* se rétrécit pour s'adapter (vous perdrez certainement beaucoup d'espace si votre écran présente une faible résolution). En cliquant à nouveau sur le bouton *Masquer automatiquement*, vous ferez disparaître la *Boîte à outils*.

3. Le contrôle *Label* du formulaire ne se trouve peut-être pas exactement où vous le souhaitez. Vous pouvez cliquer sur les contrôles que vous avez ajoutés à un formulaire et les faire glisser pour les repositionner. Grâce à cette technique, déplacez le contrôle *Label* pour le placer vers le coin supérieur gauche du formulaire (un positionnement exact n'est pas important pour cette application).



**Note** La description XAML du formulaire dans le volet en bas inclut désormais le contrôle Label, avec des propriétés comme son emplacement dans le formulaire qui est déterminé par la propriété *Margin*. La propriété *Margin* se compose de quatre nombres qui indiquent la distance de chaque coin du Label à partir des coins du formulaire. Si vous déplacez le contrôle dans le formulaire, la valeur de la propriété *Margin* change. Si le formulaire est redimensionné, les contrôles ancrés aux coins du formulaire qui sont déplacés sont redimensionnés pour préserver les valeurs de la propriété *Margin*. Vous pouvez empêcher cela en définissant les valeurs de *Margin* à zéro. Vous en apprendrez davantage sur les propriétés *Margin*, *Height* et *Width* des contrôles WPF dans le chapitre 22, « Introduction à Windows Presentation Foundation ».

4. Dans le menu *Affichage*, cliquez sur *Fenêtre Propriétés*.

La fenêtre *Propriétés* apparaît en bas à droite de l'écran, sous l'*Explorateur de solutions* (si elle n'est pas déjà affichée). La fenêtre *Propriétés* offre un autre moyen de modifier les propriétés des éléments d'un formulaire, ainsi que les autres éléments d'un projet. Elle est sensible au contexte, ce qui signifie qu'elle affiche les propriétés de l'élément actuellement sélectionné. Quand on clique sur la barre de titre du formulaire affiché dans la fenêtre *Concepteur*, la fenêtre *Propriétés* affiche les propriétés du formulaire. Si l'on clique sur le contrôle Label, la fenêtre affiche alors les propriétés du contrôle Label. Si vous cliquez n'importe où ailleurs dans le

formulaire, la fenêtre *Propriétés* affiche les propriétés d'un élément mystérieux appelé *grid* (grille). Un grid agit comme un conteneur pour les éléments d'un formulaire WPF, et vous pouvez utiliser le grid, entre d'autres choses, pour indiquer la manière dont les éléments d'un formulaire doivent être alignés et regroupés.

5. Cliquez sur le contrôle Label du formulaire. Dans la fenêtre *Propriétés*, recherchez la section *Texte*.

En utilisant les propriétés de cette section, vous pouvez spécifier la police et la taille de la police du contrôle Label, mais pas le texte qui est affiché dans le contrôle *Label*.

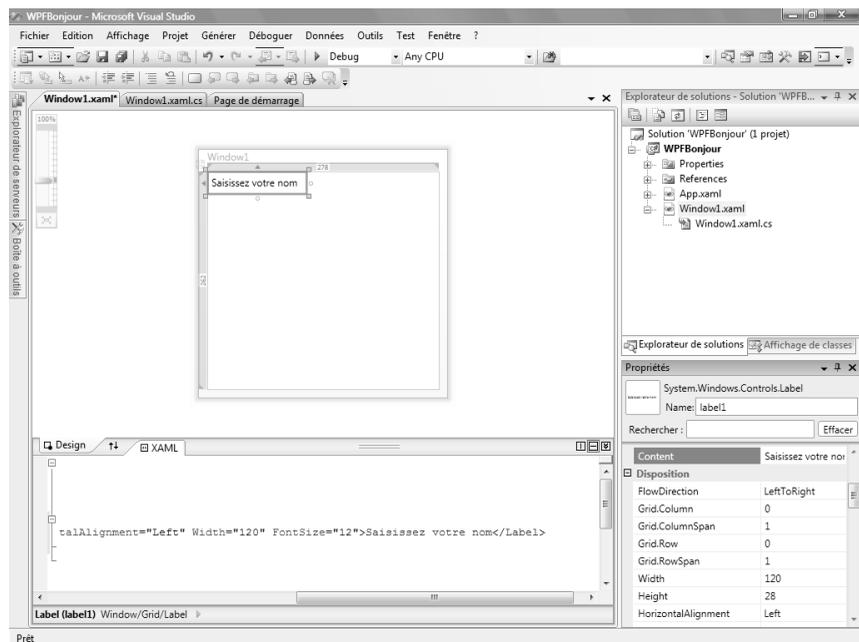
6. Modifiez la propriété *FontSize* en **20**, puis cliquez sur la barre de titre du formulaire. La taille du texte du contrôle Label change, bien que le label ne soit plus assez grand pour afficher le texte. Rétablissez la valeur de la propriété *FontSize* à **12**.

Note Le texte affiché dans le contrôle Label peut ne pas se redimensionner immédiatement dans la fenêtre *Concepteur*. Le problème se réglera lorsque vous générerez et exécuterez l'application, ou lorsque vous fermerez et ouvrirez le formulaire dans la fenêtre *Concepteur*.

7. Faites défiler la description XAML du formulaire dans le volet en bas vers la droite, et examinez les propriétés du contrôle Label.

Le contrôle Label comporte une balise `<Label>` contenant les valeurs des propriétés, suivi du texte du label lui-même ("Label"), puis d'une balise fermante `</Label>`.

8. Modifiez le texte Label (juste avant la balise fermante) en **Saisissez votre nom**, comme le montre l'image suivante.



Notez bien que le texte affiché dans l'étiquette du formulaire change, bien que le contrôle Label soit encore trop petit pour l'afficher correctement.

9. Cliquez sur le formulaire dans la fenêtre *Concepteur*, puis affichez de nouveau la *Boîte à outils*.



**Note** Si vous ne cliquez pas sur le formulaire dans la fenêtre *Concepteur*, la *Boîte à outils* affiche le message « Il n'existe aucun contrôle utilisable dans ce groupe ».

10. Dans la *Boîte à outils*, cliquez sur *TextBox*, puis cliquez sur le formulaire. Un contrôle *TextBox* est ajouté au formulaire. Déplacez le contrôle *TextBox* de manière à ce qu'il soit directement en dessous du contrôle *Label*.



**Astuce** Lorsque vous déplacez un contrôle dans un formulaire, les poignées d'alignement apparaissent automatiquement lorsque le contrôle est aligné verticalement et horizontalement avec d'autres contrôles. Cela vous permet de voir rapidement si les contrôles sont bien alignés.

11. Lorsque le contrôle *TextBox* est sélectionné, modifiez dans la fenêtre *Propriétés* la valeur de la propriété *Name* en **nomUtilisateur**.



**Note** Nous aborderons en détail les conventions de nommage des contrôles et des variables dans le chapitre 2, « Variables, opérateurs et expressions ».

12. Affichez à nouveau la *Boîte à outils*, cliquez sur *Button*, puis cliquez sur le formulaire. Déplacez le contrôle *Button* à droite du contrôle *TextBox* en veillant à ce qu'il soit aligné horizontalement avec la zone de texte.
13. En utilisant la fenêtre *Propriétés*, modifiez la propriété *Name* du contrôle *Button* en **ok**.
14. Dans la description XAML du formulaire, faites défiler le texte vers la droite pour afficher la légende du bouton, puis modifiez-la en changeant *Button* en **OK**. Vérifiez que la légende du bouton est modifiée dans le formulaire.
15. Cliquez sur la barre de titre du formulaire *Window1.xaml* dans la fenêtre *Concepteur*. Dans la fenêtre *Propriétés*, modifiez la propriété *Title* en **Bonjour**.
16. Dans la fenêtre *Concepteur*, vous noterez qu'une poignée de redimensionnement (un petit carré) apparaît dans le coin inférieur droit du formulaire lorsqu'il est sélectionné. Déplacez le pointeur de la souris sur la poignée. Lorsque le pointeur se change en une double flèche, cliquez puis déplacez le pointeur pour redimensionner le formulaire tout en maintenant le bouton de la souris enfoncé. Relâchez le bouton de la souris lorsque l'espace autour des contrôles est à peu près identique.

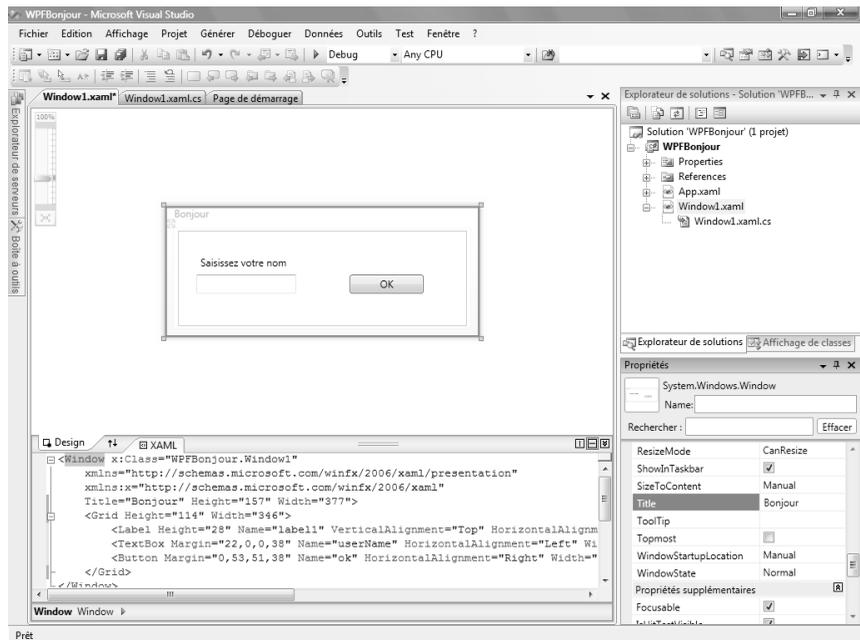


**Important** Cliquez sur la barre de titre du formulaire et non pas sur la bordure du grid du formulaire avant de le redimensionner. Si vous sélectionnez le grid, vous allez modifier la disposition des contrôles dans le formulaire, mais pas la taille du formulaire lui-même.



**Note** Si vous rétrécissez le formulaire, le bouton OK reste à une distance fixe du coin droit du formulaire qui est déterminée par sa propriété *Margin*. Si vous rétrécissez trop le formulaire, le bouton OK écrasera le contrôle TextBox. La marge à droite de l'étiquette est également fixe, et le texte de l'étiquette commencera à disparaître lorsque le contrôle Label se rétracte au fur et à mesure que le formulaire se rétrécit.

Le formulaire devrait désormais ressembler à ceci :



17. Dans le menu *Générer*, cliquez sur *Générer la solution*, et vérifiez que le projet se génère avec succès.

18. Dans le menu *Débuguer*, cliquez sur *Exécuter sans débogage*.

L'application devrait s'exécuter et afficher votre formulaire. Vous pouvez saisir votre nom dans la zone de texte et cliquer sur *OK*, mais rien ne se produit encore. Vous devez ajouter du code pour gérer l'événement *Click* du bouton *OK*, ce dont nous allons nous occuper un peu plus tard.

19. Cliquez sur le bouton *Fermer* (le X dans le coin supérieur droit du formulaire) pour fermer le formulaire et retourner dans Visual Studio.

Vous avez réussi à créer une application graphique sans écrire une seule ligne de code C# car Visual Studio génère en fait beaucoup de code qui gère les tâches courantes que toutes les applications graphiques doivent réaliser, comme le démarrage ou l'affichage d'un formulaire. Avant d'ajouter votre propre code à l'application, il est utile de comprendre ce que Visual Studio a généré pour vous.

Dans l'*Explorateur de solutions*, cliquez sur le signe plus (+) à côté du fichier `Window1.xaml`. Le fichier `Window1.xaml.cs` apparaît. Faites un double-clic sur le fichier `Window1.xaml.cs`. Le code du formulaire s'affiche dans la fenêtre *Code*. Il devrait ressembler à ceci :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WPFBonjour
{
    /// <summary>
    /// Interaction logic for Window1.xaml
    /// </summary>

    public partial class Window1 : Window
    {

        public Window1()
        {
            InitializeComponent();
        }

    }
}
```

Mis à part un bon nombre d'instructions *using* qui placent dans la portée des espaces de noms que la plupart des applications WPF utilisent, le fichier contient la définition d'une classe nommée *Window1*, mais pas grand-chose d'autre. La classe *Window1* qui contient très peu de code est ce que l'on appelle un constructeur ; elle se contente d'appeler une méthode nommée *InitializeComponent*. Un *constructeur* est une méthode spéciale ayant le même nom que la classe. Il est exécuté lorsqu'une instance de la classe est créée et peut contenir du code pour initialiser l'instance. Les constructeurs sont abordés au chapitre 7.

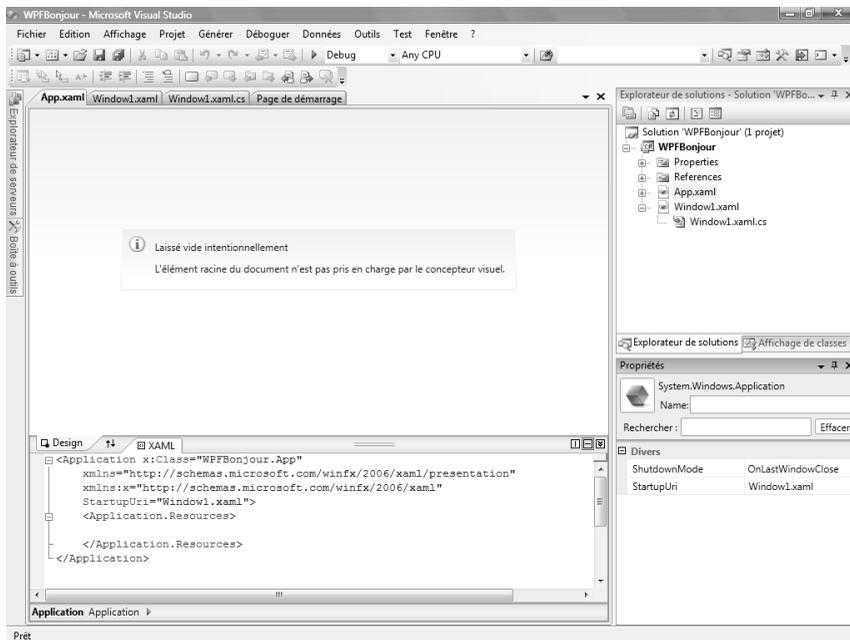
En fait, l'application contient bien plus de code, mais la plupart de ce code qui est généré automatiquement à partir de la description XAML du formulaire est masqué. Ce code masqué réalise des opérations comme la création et l'affichage du formulaire, ainsi que la création et le positionnement des différents contrôles du formulaire.

Le but du code que vous *pouvez* voir dans cette classe est de vous permettre d'ajouter vos propres méthodes pour gérer la logique de votre application, comme ce qui se passe lorsque l'utilisateur appuie sur le bouton OK.



**Astuce** Vous pouvez aussi afficher le fichier code C# d'un formulaire WPF en faisant un clic droit n'importe où dans la fenêtre *Conceuteur* puis en cliquant sur *Afficher le code*.

À partir de là, vous devez vous demander où la méthode *Main* se trouve et comment le formulaire s'affiche lorsque l'application s'exécute ; rappelez-vous que *Main* définit le point de démarrage du programme. Dans l'*Explorateur de solutions*, vous devez remarquer un autre fichier source appelé *App.xaml*. Si vous faites un double-clic sur ce fichier, la fenêtre *Conceuteur* affiche le message "Laissé vide intentionnellement", mais le fichier possède une description XAML. Une des propriétés du code XAML, qui se nomme *StartupUri*, fait référence au fichier *Window1.xaml*, comme cela est illustré ci-dessous :



Si vous cliquez sur le signe plus (+) à gauche de App.xaml dans l'*Explorateur de solutions*, vous verrez qu'il existe aussi un fichier App.xaml.cs. Si vous faites un double-clic sur ce fichier, vous verrez qu'il contient le code suivant :

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Windows;

namespace WPFBonjour
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>

    public partial class App : Application
    {

    }
}
```

Une fois encore, il y a un certain nombre d'instructions *using*, mais pas grand-chose d'autre, la méthode *Main* n'étant même pas mentionnée. En fait, *Main* est bien là, mais elle est aussi cachée. Le code de *Main* est généré en fonction des paramètres du fichier App.xaml ; en particulier, *Main* va créer et afficher le formulaire spécifié par la propriété *StartupUri*. Si vous souhaitez afficher un formulaire différent, vous devez modifier le fichier App.xaml.

Il est temps d'écrire du code par vous-même !

### Écriture du code du bouton OK

1. Cliquez sur l'onglet *Window1.xaml* sous la fenêtre *Code* pour afficher Window1 dans la fenêtre *Concepteur*.
2. Faites un double-clic sur le bouton *OK* du formulaire.

Le fichier Window1.xaml.cs apparaît dans la fenêtre *Code*, mais une nouvelle méthode a été ajoutée, appelée *ok\_Click*. Visual Studio génère automatiquement du code pour appeler cette méthode lorsque l'utilisateur appuie sur le bouton *OK*. C'est un exemple d'événement, et vous en apprendrez davantage sur le fonctionnement des événements au fur et à mesure de la lecture de cet ouvrage.

3. Ajoutez le code imprimé en gras à la méthode *ok\_Click* :

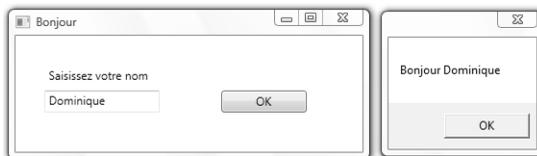
```
void ok_Click(object sender, RoutedEventArgs e)
{
    MessageBox.Show("Bonjour" + nomUtilisateur.Text);
}
```

C'est le code qui va s'exécuter lorsque l'utilisateur appuiera sur le bouton *OK*. Ne vous préoccupez pas trop pour le moment de la syntaxe du code (assurez-vous que vous avez bien recopié le code ci-dessus) car les méthodes sont abordées dans le chapitre 3. La partie intéressante est l'instruction *MessageBox.Show*. Cette instruction affiche une boîte de dialogue contenant le texte "Bonjour" suivi du nom que l'utilisateur a saisi dans la zone de texte du formulaire.

4. Cliquez sur l'onglet *Window1.xaml* sous la fenêtre *Code* pour afficher de nouveau *Window1* dans la fenêtre *Concepteur*.
5. Dans le volet en bas affichant la description XAML du formulaire, examinez l'élément *Button*, mais veillez à ne rien modifier. Notez qu'il contient un élément appelé *Click* qui fait référence à la méthode *ok\_Click* :

```
<Button Height="23" ... Click="ok_Click">OK</Button>
```

6. Dans le menu *Débuguer*, cliquez sur *Exécuter sans débogage*.
7. Lorsque le formulaire apparaît, saisissez votre nom dans la zone de texte, puis cliquez sur *OK*. Une boîte de dialogue apparaît et vous salue en vous appelant par votre nom.



8. Cliquez sur *OK* dans la boîte de dialogue.  
La boîte de dialogue se ferme.
9. Fermez le formulaire.
  - Si vous souhaitez poursuivre votre lecture  
Laissez Visual Studio 2008 ouvert, et passez au chapitre 2.
  - Si vous souhaitez fermer Visual Studio 2008 maintenant  
Dans le menu *Fichier*, cliquez sur *Quitter*. Si vous voyez une boîte de dialogue vous proposant d'enregistrer les modifications, cliquez sur *Oui* (si vous utilisez Visual Studio 2008) ou *Enregistrer* (si vous utilisez Visual C# 2008 Express Edition) et enregistrez le projet.

## Aide-mémoire du chapitre 1

Pour	Accomplissez	Touches
Créer une nouvelle application console avec Visual Studio 2008 Standard ou Professional Edition	Dans le menu <i>Fichier</i> , pointez <i>Nouveau</i> , puis cliquez sur <i>Projet</i> pour ouvrir la boîte de dialogue <i>Nouveau projet</i> . Pour le type de projet, sélectionnez <i>Visual C#</i> . Pour le modèle, sélectionnez <i>Application console</i> . Sélectionnez un répertoire pour les fichiers du projet dans le champ <i>Emplacement</i> . Choisissez un nom pour le projet. Cliquez sur <i>OK</i> .	
Créer une nouvelle application console avec Visual C# 2008 Express Edition	<p>Dans le menu <i>Outils</i>, cliquez sur <i>Options</i>. Dans la boîte de dialogue <i>Options</i>, cliquez sur <i>Projets et solutions</i>. Dans le champ <i>Emplacements des projets Visual Studio</i>, spécifiez un répertoire pour les fichiers du projet.</p> <p>Dans le menu <i>Fichier</i>, cliquez sur <i>Nouveau projet</i> pour ouvrir la boîte de dialogue <i>Nouveau projet</i>. Pour le modèle, sélectionnez <i>Application console</i>. Choisissez un nom pour le projet. Cliquez sur <i>OK</i>.</p>	
Créer une nouvelle application graphique avec Visual Studio 2008 Standard ou Professional Edition	Dans le menu <i>Fichier</i> , pointez <i>Nouveau</i> , puis cliquez sur <i>Projet</i> pour ouvrir la boîte de dialogue <i>Nouveau projet</i> . Pour le type de projet, sélectionnez <i>Visual C#</i> . Pour le modèle, sélectionnez <i>Application WPF</i> . Sélectionnez un répertoire pour les fichiers du projet dans le champ <i>Emplacement</i> . Choisissez un nom pour le projet. Cliquez sur <i>OK</i> .	
Créer une nouvelle application graphique avec Visual C# 2008 Express Edition	<p>Dans le menu <i>Outils</i>, cliquez sur <i>Options</i>. Dans la boîte de dialogue <i>Options</i>, cliquez sur <i>Projets et solutions</i>. Dans le champ <i>Emplacements des projets Visual Studio</i>, spécifiez un répertoire pour les fichiers du projet.</p> <p>Dans le menu <i>Fichier</i>, cliquez sur <i>Nouveau projet</i> pour ouvrir la boîte de dialogue <i>Nouveau projet</i>. Pour le modèle, sélectionnez <i>Application WPF</i>. Choisissez un nom pour le projet. Cliquez sur <i>OK</i>.</p>	
Générer l'application	Dans le menu <i>Générer</i> , cliquez sur <i>Générer la solution</i> .	F6
Exécuter l'application	Dans le menu <i>Débugger</i> , cliquez sur <i>Exécuter sans débogage</i> .	Ctrl+F5