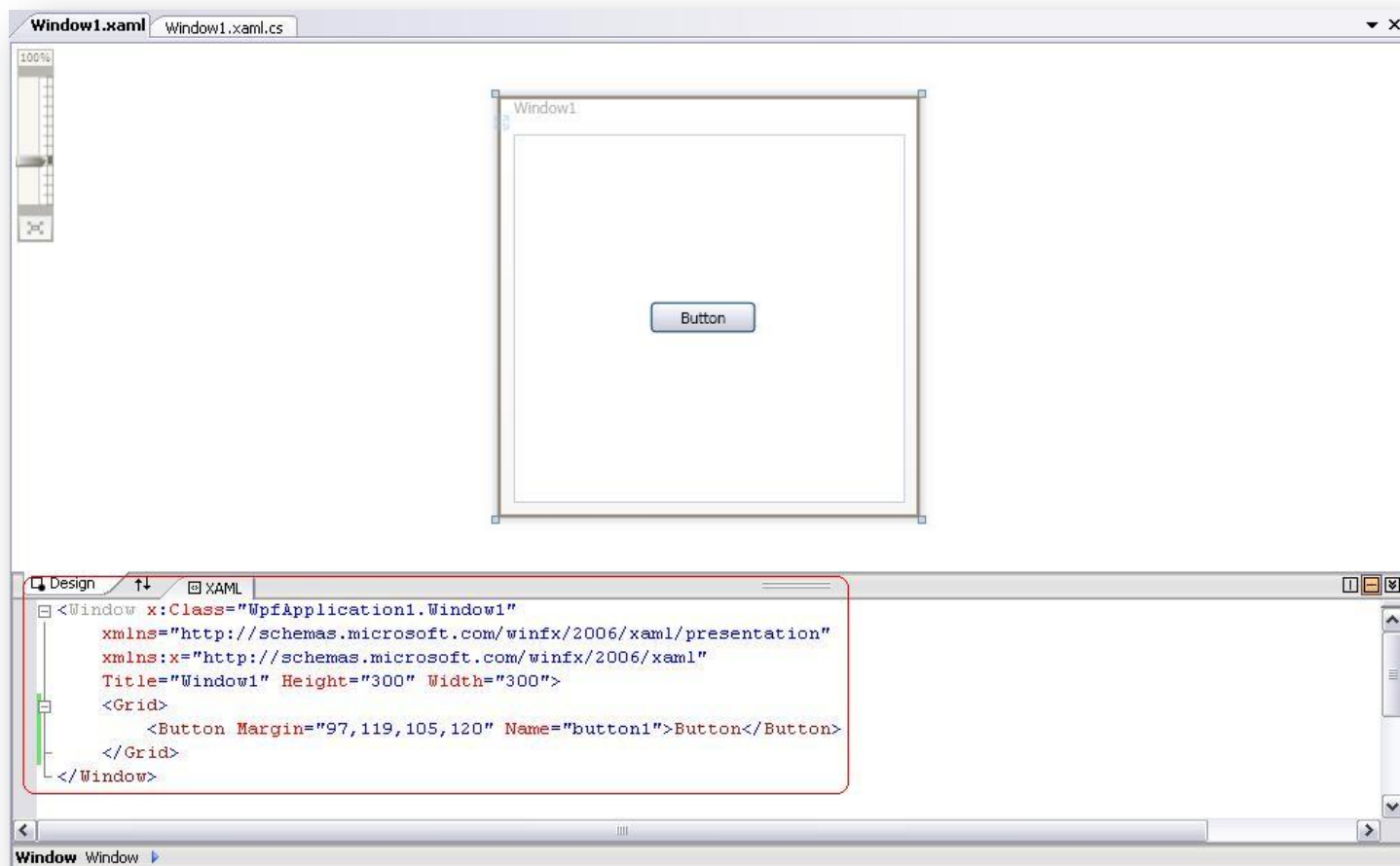


Dans cet exemple nous avons une application WPF contenant juste un bouton, le but est de définir l'action à déclencher lorsque l'utilisateur clique sur le bouton.

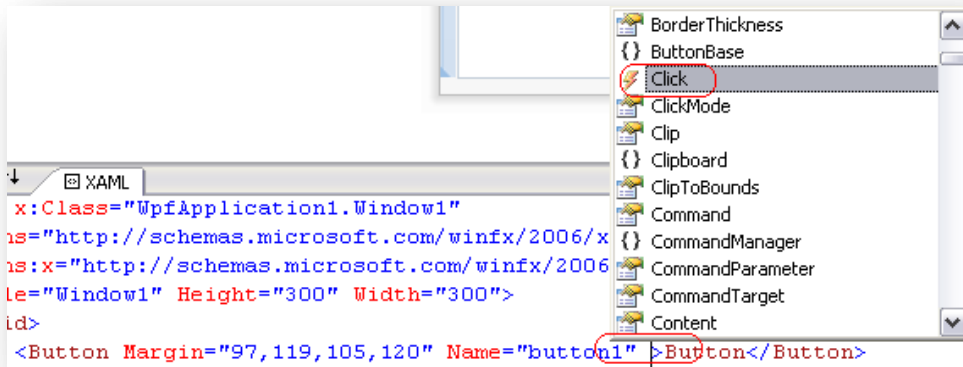
Aperçu du projet sous Visual Studio 2008 :




La partie encadrée en rouge est la partie XAML de la fenêtre contenant le bouton, elle va servir à définir les évènements à attribuer au bouton.

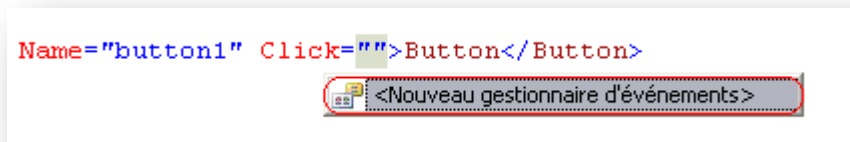
Afin d'ajouter un évènement, on place le curseur à la fin de la balise `<Button ...>` et on appuie sur la touche Espace (voir figure ci-dessous). IntelliSense ouvrira automatiquement le panneau suivant :

(Si le panneau ne s'affiche pas, appuyer sur les touches Ctrl+Espace)



Ce panneau servira à choisir l'évènement que l'on souhaite attribuer au bouton. Les évènements sont reconnaissables avec une icône éclair 

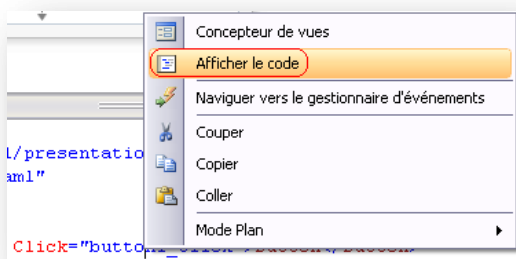
Pour cet exemple nous choisissons d'attribuer l'action « Click » au bouton en la choisissant dans le panneau.



Visual Studio demande alors au développeur un nom de méthode. Par défaut en appuyant sur la touche Entrée nous obtenons le nom « button1\_Click », ce que nous ferons dans cet exemple.

Cette méthode servira à définir l'action(s) qui sera exécutée(s) lors de l'évènement.

Après avoir choisi un nom ou appuyé sur Entrée, la méthode est automatiquement créée. Nous pouvons y accéder par divers moyen, par exemple : clic droit sur le nom de la méthode, puis *Afficher le code*.



Nous sommes alors directement redirigés vers le fichier Window1.xaml.cs (Window1 étant le nom de la fenêtre par défaut). Vous y spécifierez le gestionnaire d'évènement en Microsoft C#, Visual Basic .NET, ou tout autre langage .NET.

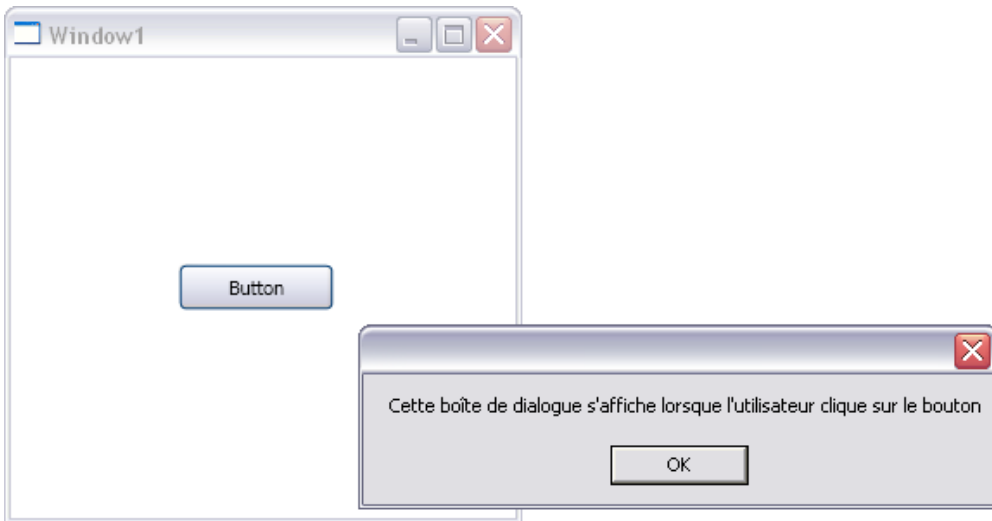
Nous retrouvons dans ce fichier la méthode qui s'exécute lors du clic, écrite en C# :

```
using ...

namespace WpfApplication1
{
    /// <summary> ...
    public partial class Window1 : Window
    {
        public Window1() ...

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            MessageBox.Show("Cette boîte de dialogue s'affiche lorsque l'utilisateur clique sur le bouton");
        }
    }
}
```

Dans l'encadré rouge, nous retrouvons bien le nom de la méthode et l'action exécutée. Nous obtenons ceci à l'exécution :



Quand l'utilisateur clique sur le bouton, une boîte de dialogue s'ouvre. Nous avons donc défini un nouvel évènement pour un objet, et l'action à effectuer lorsqu'il est déclenché.

WPF fournit une fonctionnalité pratique, celle d'attribuer un même gestionnaire d'évènement pour deux boutons différents, c'est-à-dire leur attribuer une même action.

Nous avons deux solutions pour y parvenir :

Pour ces exemples nous ajoutons un bouton au projet crée précédemment.


Première solution : si l'on veut attribuer un gestionnaire aux boutons vous devez attribuer la méthode « button1\_Click » au conteneur (Grid), avec l'évènement ButtonBase.Click.

```
<Grid ButtonBase.Click="button1_Click">
  <Button Height="23" Margin="101,102,102,0" Name="button1">Button</Button>
  <Button Height="23" Margin="101,0,102,97" Name="button2">Button</Button>
</Grid>
```

Quand l'utilisateur cliquera sur un bouton, la boîte de dialogue s'ouvrira.

Deuxième solution : on ajoute l'évènement Click au deuxième bouton, IntelliSense proposera cette fois ceci :

```
<Grid>
  <Button Margin="101,98,102,0" Name="button1" Height="23" Click="button1_Click">Button</Button>
  <Button Height="23" Margin="101,0,102,90" Name="button2" Click="" >Button</Button>
</Grid>
ndow>
```



En choisissant « button1\_Click » dans cette liste, l'évènement pourra être déclenché par l'un ou l'autre bouton, la même action sera exécutée.

Nous avons pu voir que les deux objets distincts sont gérés par un seul gestionnaire.