Création d'applications multifenêtres en C#

Formulaire d'affichage
Boîte de dialogue

Ricco Rakotomalala
Université Lumière Lyon 2

Pourquoi plusieurs fenêtres ?

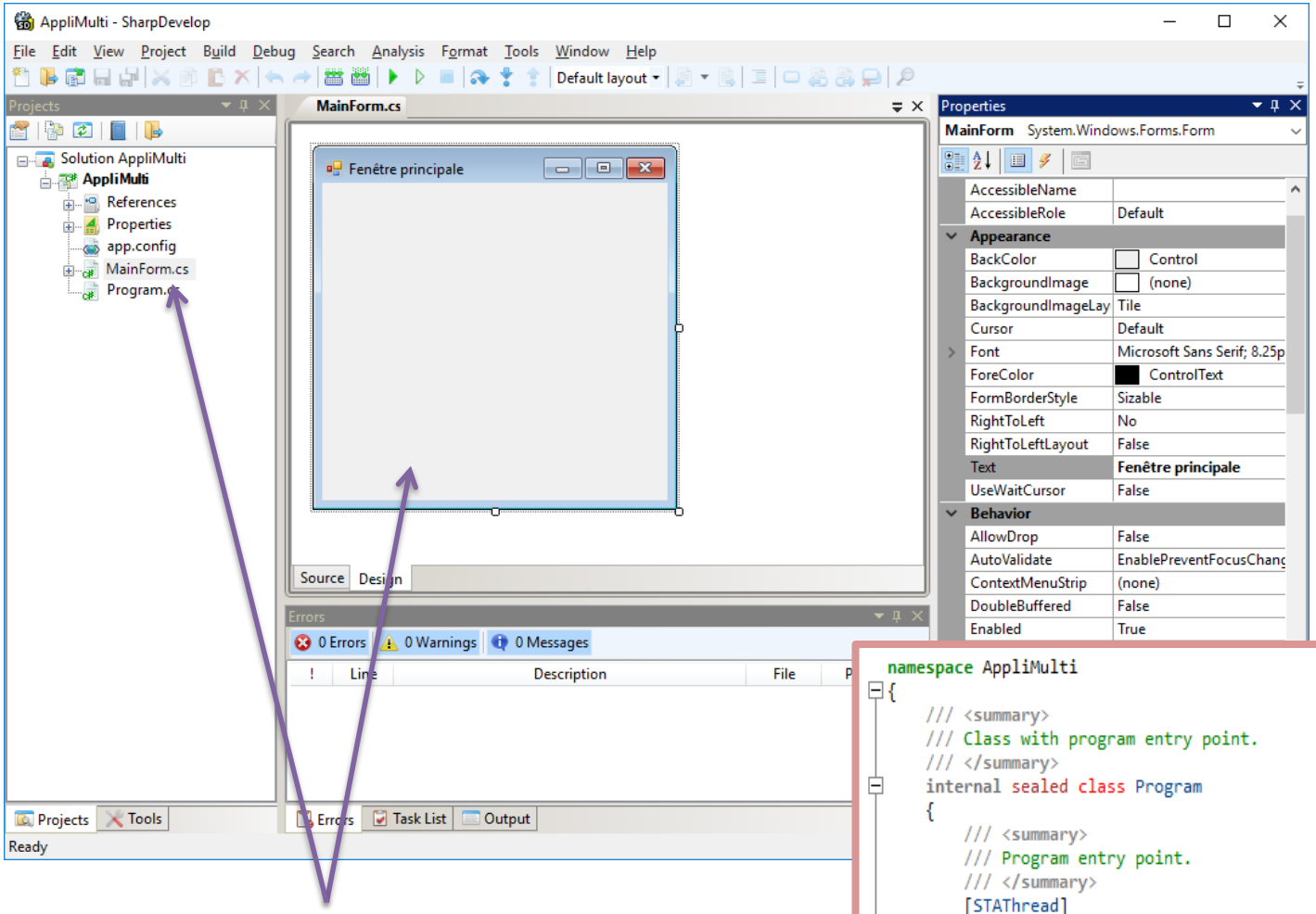
Meilleure organisation de l'interface, échange d'informations entre formulaires : (1) certaines fenêtres servent à l'affichage de résultats : formulaires d'affichage, fenêtre non modale ; (2) d'autres à interagir avec l'utilisateur : boîtes de dialogues, fenêtre modale.

- Dans tous les cas il s'agit de formulaires C#
- On s'intéresse aux fenêtres créées par l'utilisateur dans ce support. Il existe par ailleurs des fenêtres standards Windows (ex. OpenFileDialog, SaveFileDialog, …)
- On s'intéresse à l'organisation SDI (Single Document Interface) dans ce support. Un formulaire joue le rôle de fenêtre principale, les autres gravitent autour.

Affichage d’informations

FORMULAIRE D’AFFICHAGE

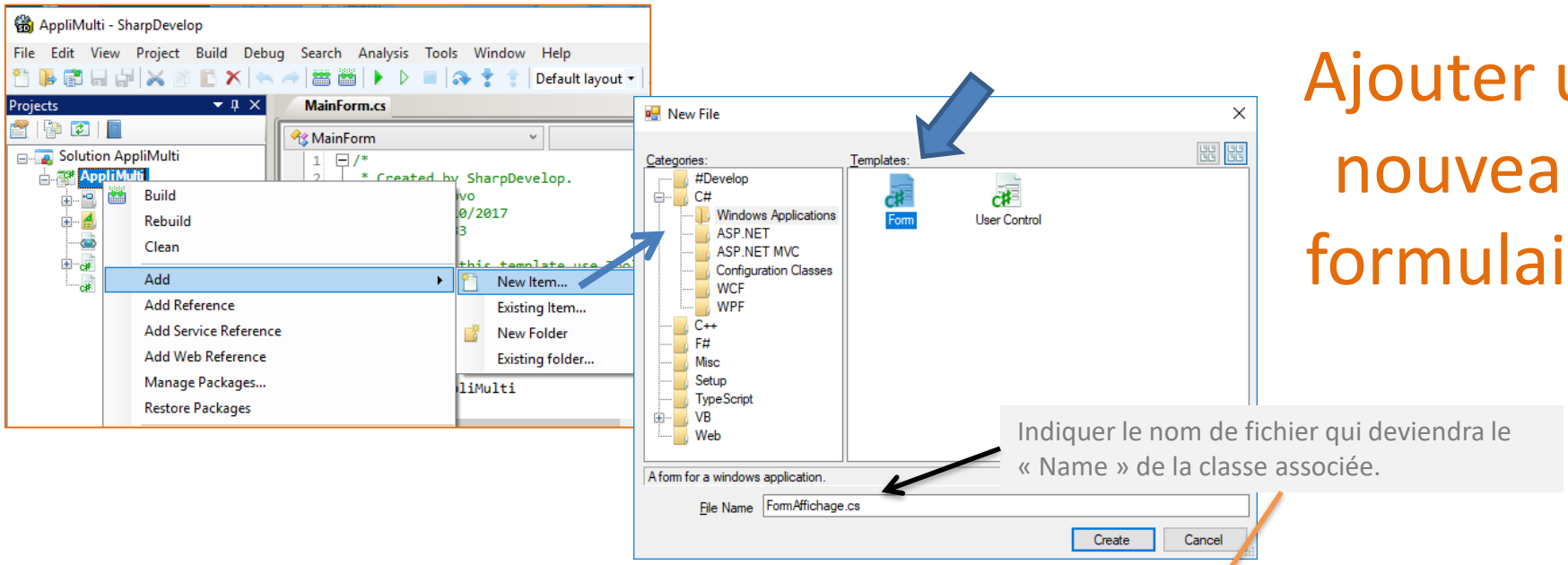
Point de départ, formulaire principal



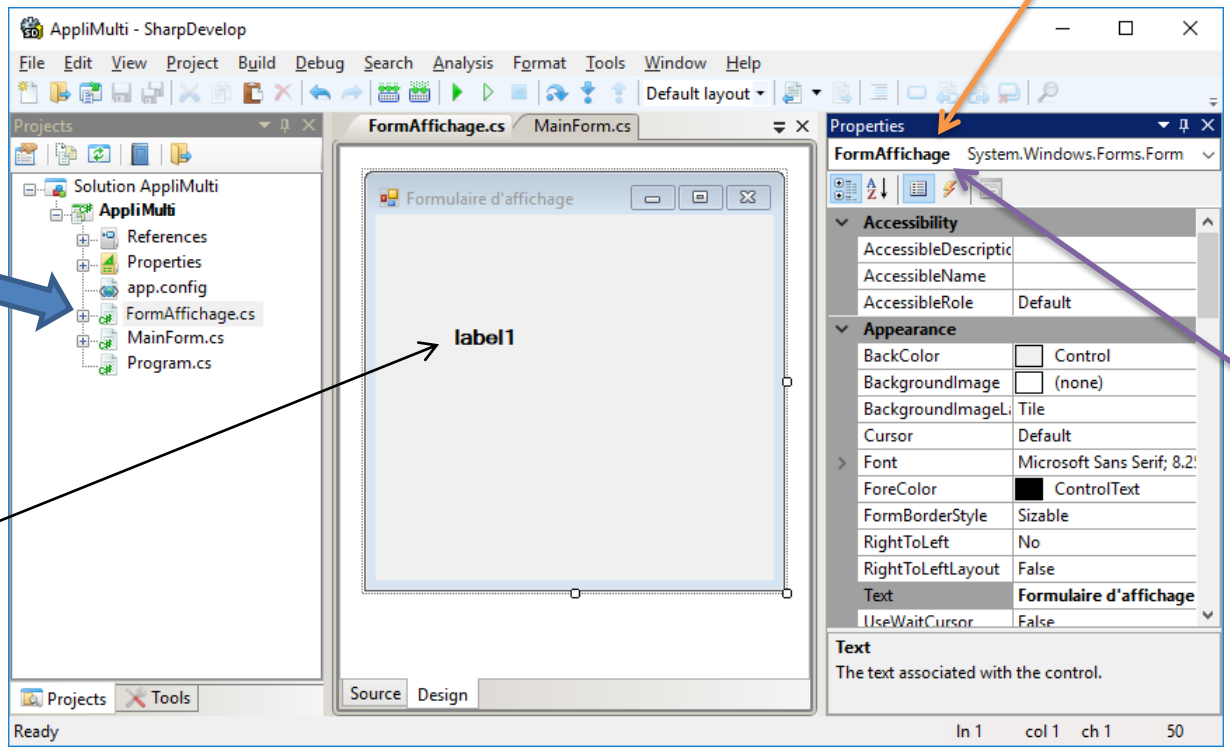

```
namespace AppliMulti
{
    /// <summary>
    /// Class with program entry point.
    /// </summary>
    internal sealed class Program
    {
        /// <summary>
        /// Program entry point.
        /// </summary>
        [STAThread]
        private static void Main(string[] args)
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }

    }
}
```

Le premier formulaire créé fait office de fenêtre principale (va contenir les menus, permet de piloter l'application). Il est créé automatiquement lors du démarrage de l'application.

Ajouter un nouveau formulaire



Indiquer le nom de fichier qui deviendra le « Name » de la classe associée.

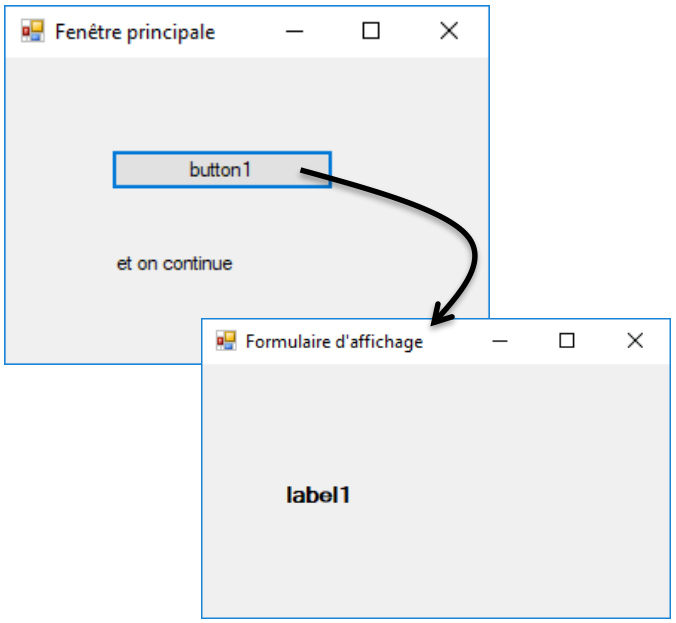


Nouvelle classe dans le projet.

Un « label » est mis pour l'affichage d'informations

Bien noter la propriété « Name » qui définit le nom de la classe du formulaire

Appel du formulaire d'affichage à partir de la fenêtre principale



Programmation du « clic » sur le bouton « Button1 »

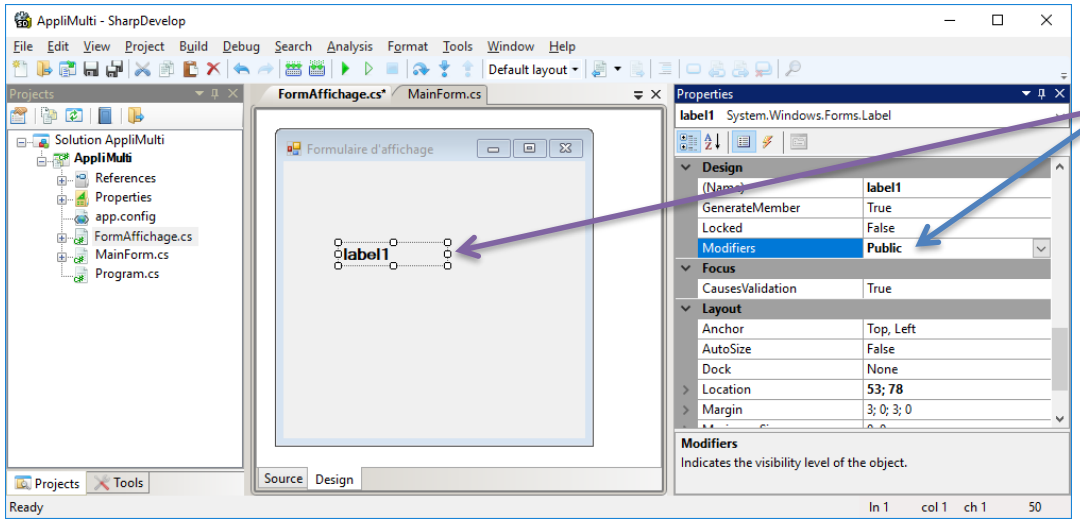
```
MainForm.cs
MainForm
14  namespace AppliMulti
15  {
16      /// <summary>
17      /// Description of MainForm.
18      /// </summary>
19      public partial class MainForm : Form
20      {
21          public MainForm()...
32
33          void Button1Click(object sender, EventArgs e)
34          {
35              FormAffichage fiche = new FormAffichage();
36
37              fiche.Show();
38
39              this.labelPrincipal.Text = "et on continue";
40          }
41      }
42  }
Source  Design
```

Création et affichage (Show) du formulaire d'affichage à partir du formulaire principal.

Dès le formulaire affiché, les instructions suivantes sont exécutées ! Ici, le label du formulaire principal est modifié. Show() est non bloquant. On a une **fenêtre non modale**.

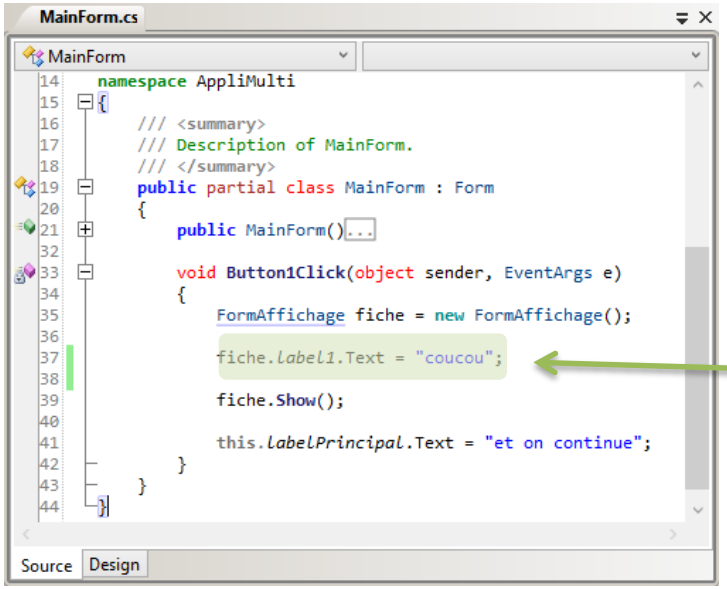
L'application est pilotée par le formulaire principal c.-à-d. fermer la fenêtre principale = arrêter l'application. Ce n'est pas le cas pour le formulaire d'affichage.

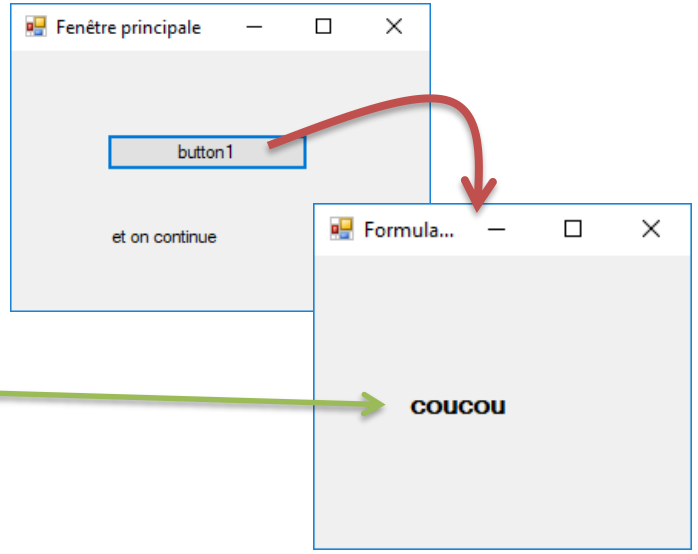
Accès à un composant dans le formulaire d'affichage



Pour rendre un composant « public » i.e. visible sur l'instance, il faut modifier sa propriété Modifiers.

Accès à **Label1** du formulaire d'affichage à partir du formulaire principal.

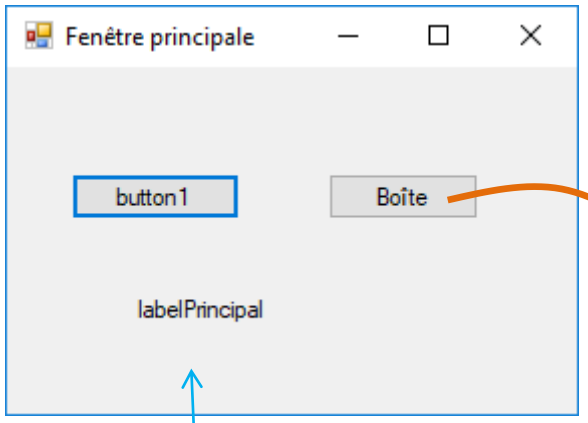


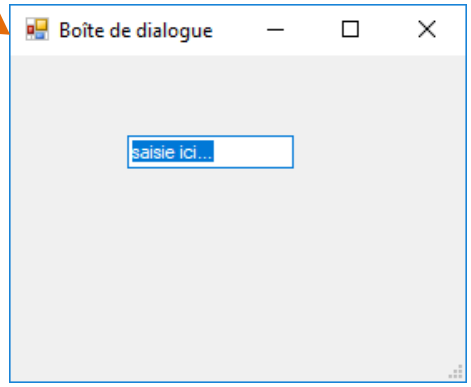


Récupération d’informations saisies par l’utilisateur

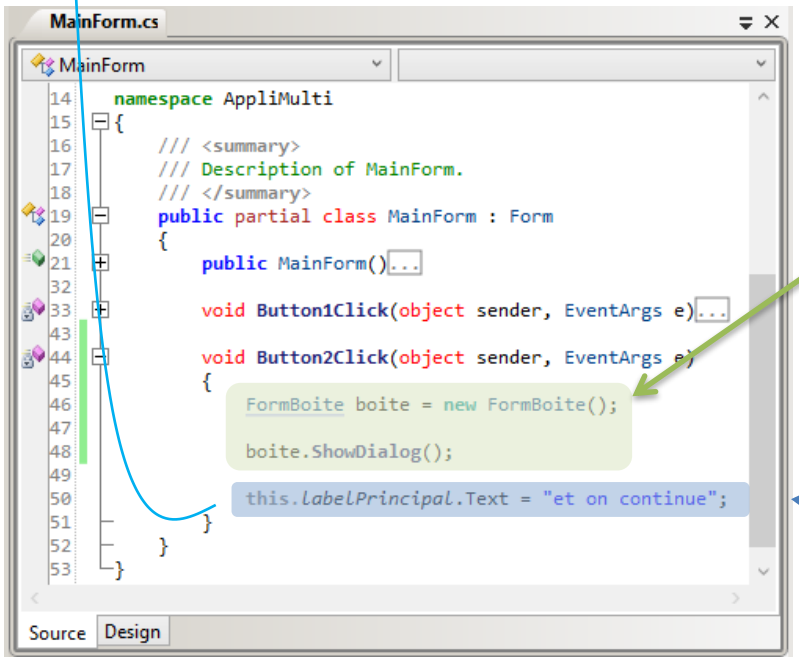
BOÎTE DE DIALOGUE

Une boîte de dialogue est aussi un formulaire. C'est l'appel ShowDialog() qui modifie son comportement.





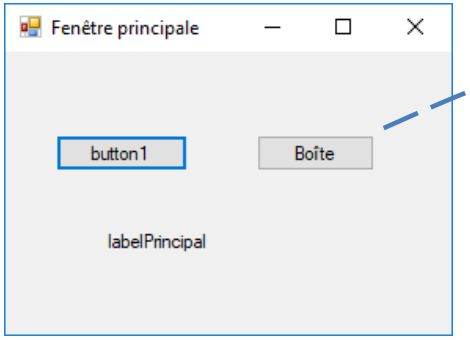
On ne peut pas accéder (cliquer sur) au formulaire principal tant que la boîte de dialogue n'est pas fermée.

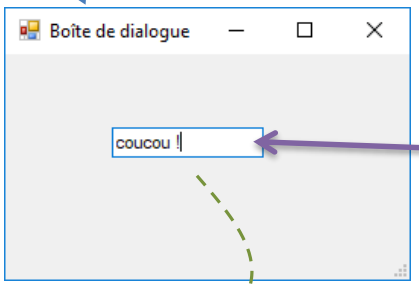


Création de l'instance formulaire, et affichage en tant que fenêtre modale avec ShowDialog()

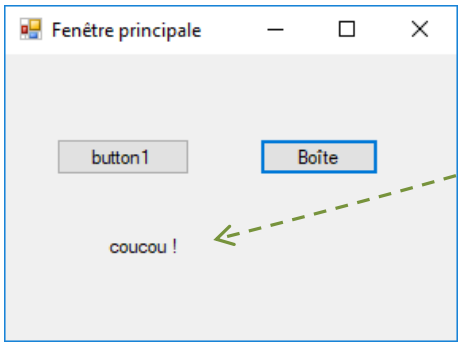
Ce code n'est pas exécuté tant que la boîte de dialogue n'est pas fermée

Récupération d'informations en provenance d'une boîte de dialogue.

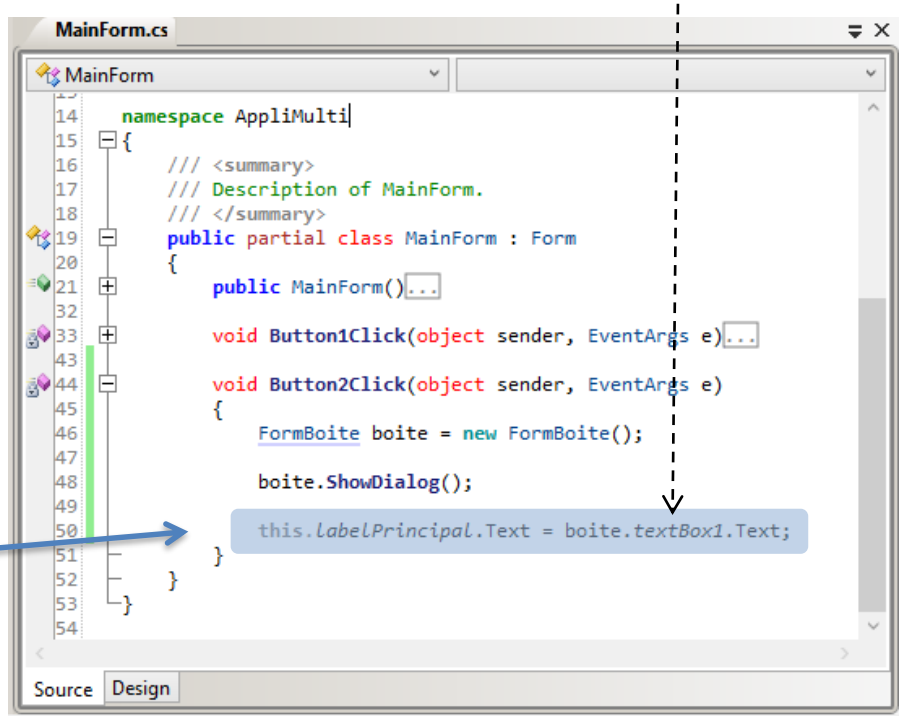




Il faut que la propriété Modifiers soit mise à Public pour que l'objet TextBox1 soit accessible directement.

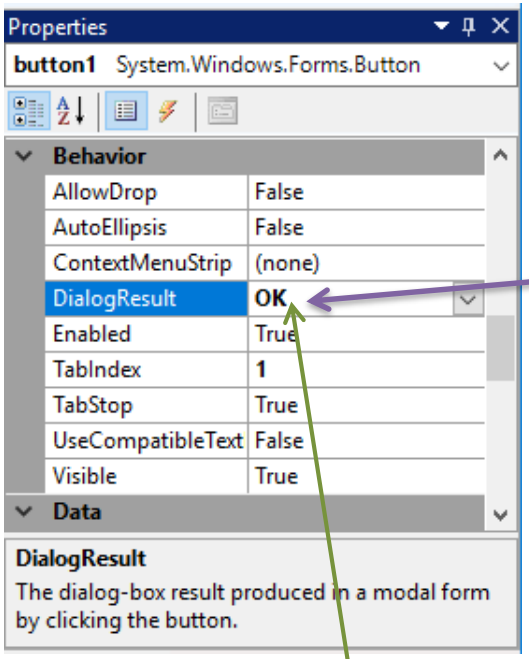


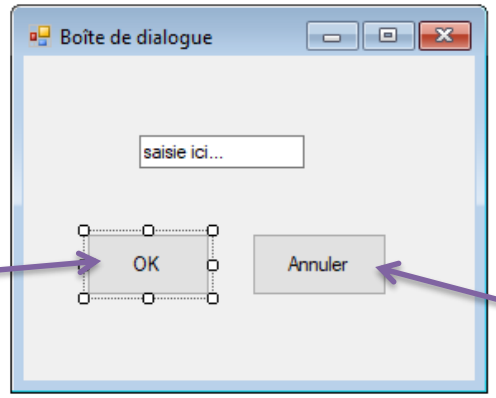
Le contenu de la zone d'édition peut être récupéré, une fois seulement la boîte fermée.



Boîte de dialogue : gestion des boutons de réponse, la propriété DialogResult

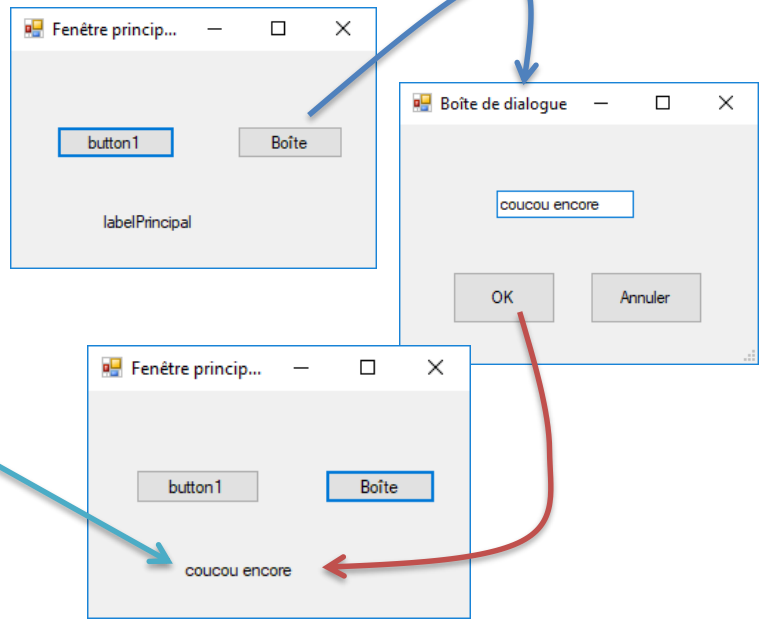
La propriété DialogResult des boutons définit le code de retour renvoyée par ShowDialog() qui est en réalité une fonction





DialogResult = Cancel

```
void Button2Click(object sender, EventArgs e)
{
    FormBoite boite = new FormBoite();

    if (boite.ShowDialog() == DialogResult.OK)
    {
        this.labelPrincipal.Text = boite.textBox1.Text;
    }
    else
    {
        this.labelPrincipal.Text = "il n'a pas cliqué sur OK";
    }
}
```



Appel et gestion de la réponse de la boîte de dialogue dans le formulaire principal !

Bilan des objets visuels composant un projet C#

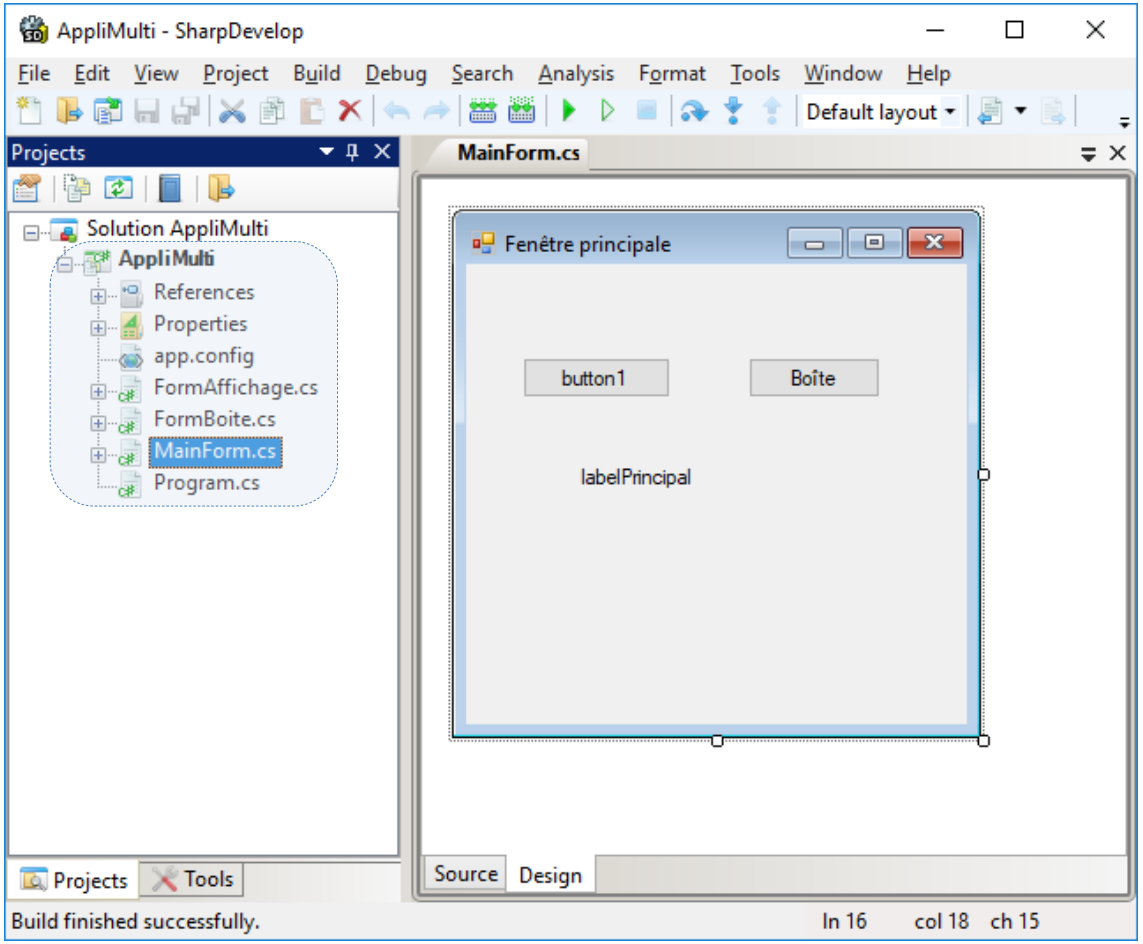
GESTIONNAIRE DE PROJET SHARPDEVELOP

Vision globale d'un projet

Gestionnaire de projet

Avec le gestionnaire de projet, on obtient une vision globale des éléments constitutifs d'un projet :

- 1 Programme principal (Program.cs)
- Formulaires et boîtes de dialogue (dont le formulaire principal MainForm)
- Classes de calcul



FIN…

Les mêmes concepts sont – à peu de choses près – présents dans tous les langages de programmation…