



Beignets aux courgettes

 90 minutes

 Bon marché



Cake au saumon fumé et au citron

 60 minutes

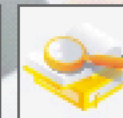
 Prix moyen



Carrot cake

 70 minutes

 Bon marché



Contrôles utilisateur

Véronique RICHARD

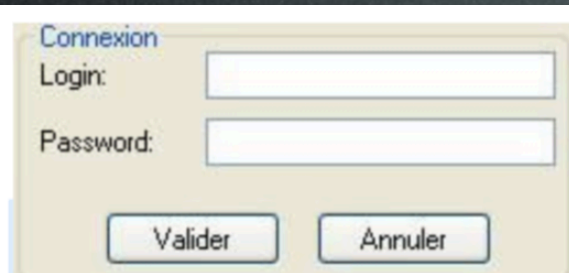
Problématique

- Tous les contrôles utilisés jusqu'à présent (TextBox, Button, etc) dérivent de la classe de base `System.Windows.Forms.Control`.
- Visual Studio permet l'intégration de contrôles supplémentaires par l'ajout à la boîte à outils (clic droit « Choisir les éléments »)
- Si le besoin est spécifique, il est également possible de créer ses propres contrôles (appelé « user controls »).

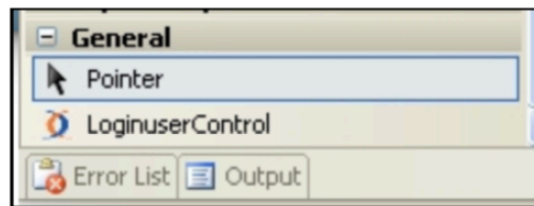
La création de contrôles s'inscrit dans le principe de réutilisation du code : la logique est créée en un seul endroit et peut être utilisée plusieurs fois.

Exemples

Un User Control est une classe à part entière et à ce titre peut posséder des propriétés, des méthodes, des événements, le tout publics ou privés.



—> Contrôle « LoginuserControl », contenant deux labels, deux zones de texte et deux boutons.



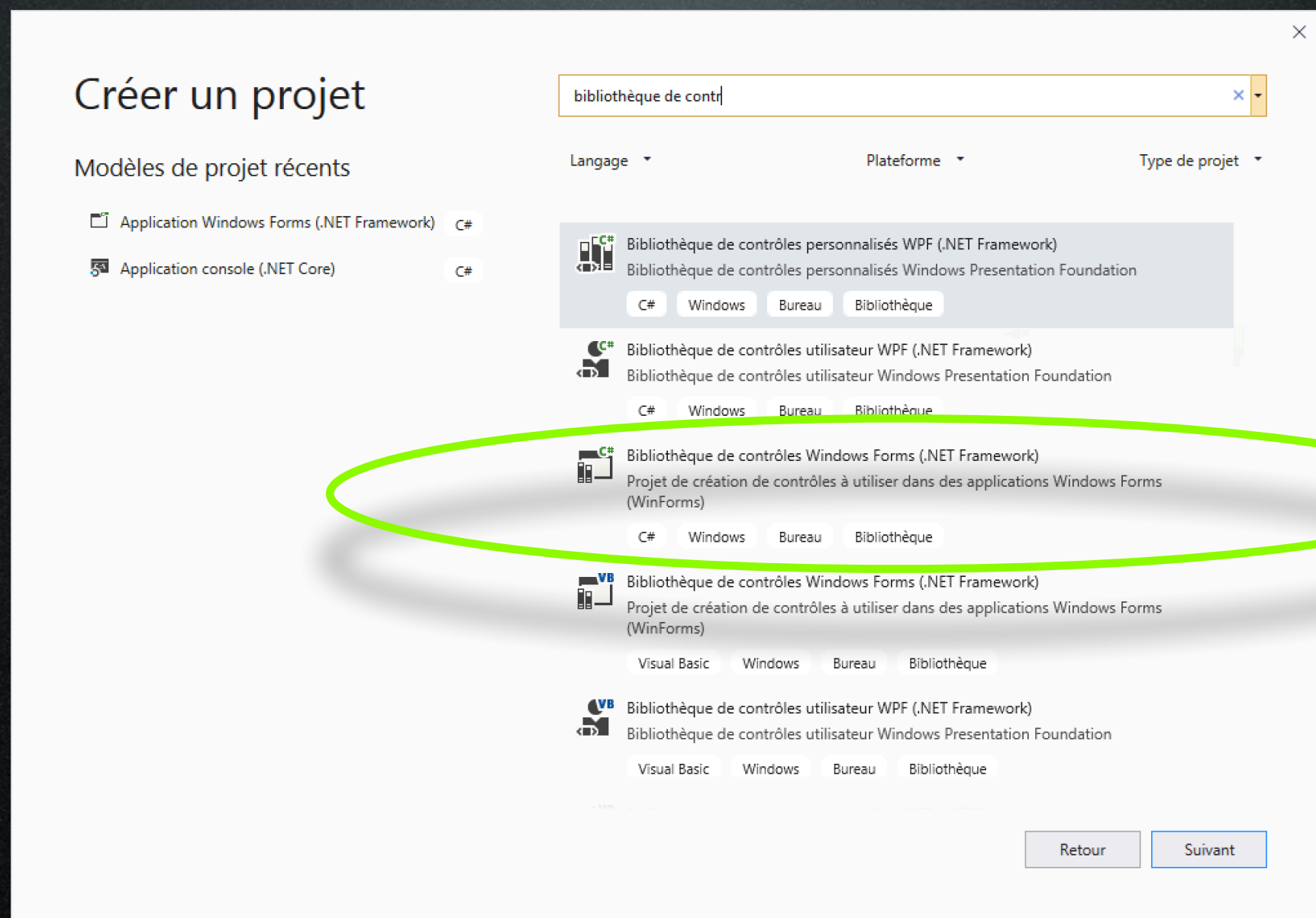
—> Le contrôle utilisateur pourra être sélectionné comme un contrôle classique, dans la boîte à outils.



—> Si l'on veut pouvoir modifier certaines propriétés du contrôle utilisateur via la fenêtre "Properties" (Propriétés) de Visual Studio, il faut utiliser les attributs (en général des propriétés Get/Set).

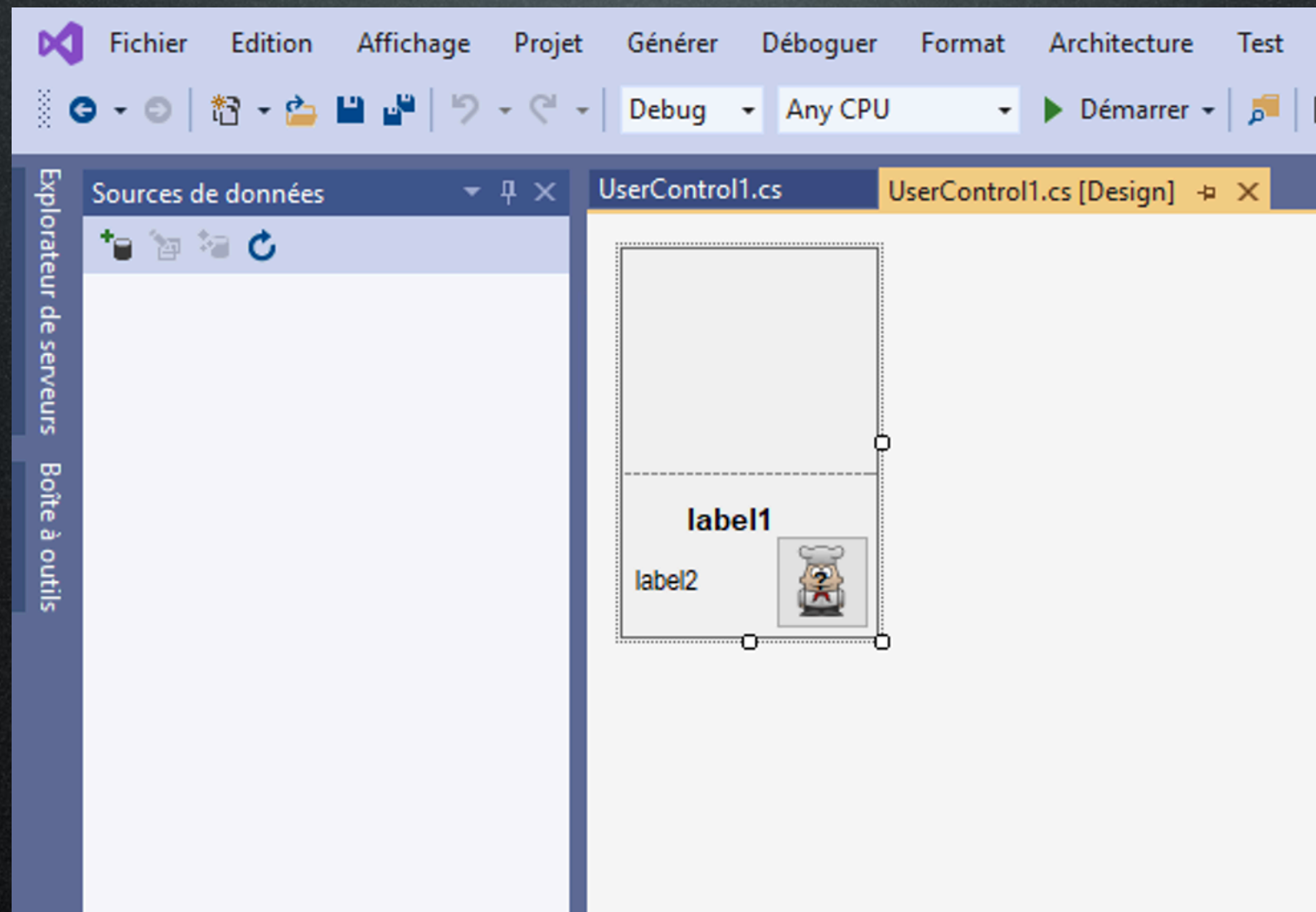
Création du UserControl (1)

- La première étape consiste à « dessiner » l'interface du nouveau contrôle et l'enregistrer dans une dll.
- Création d'une bibliothèque de contrôles Windows Forms.



Création du UserControl (2)

- Conception de l'interface graphique du composant.
- Fenêtre Windows classique, mais sur laquelle il n'y a pas de bordures.



Création du UserControl (3)



Paramétrage du UC

- Tout projet utilisant le UC personnalisé (via la boîte à outils par exemple) doit disposer d'une technique permettant de lui communiquer les trois éléments :
 - chemin de l'image à afficher dans le PictureBox
 - texte à afficher dans le premier label
 - texte à afficher dans le second label
- Comme pour le transfert d'informations entre formulaires, la solution la plus simple est la **surcharge du constructeur de UC**.

Surcharge du constructeur

- On conçoit autant de surcharges du constructeur que de possibilités d'instanciation que l'on offre au futur utilisateur du contrôle.

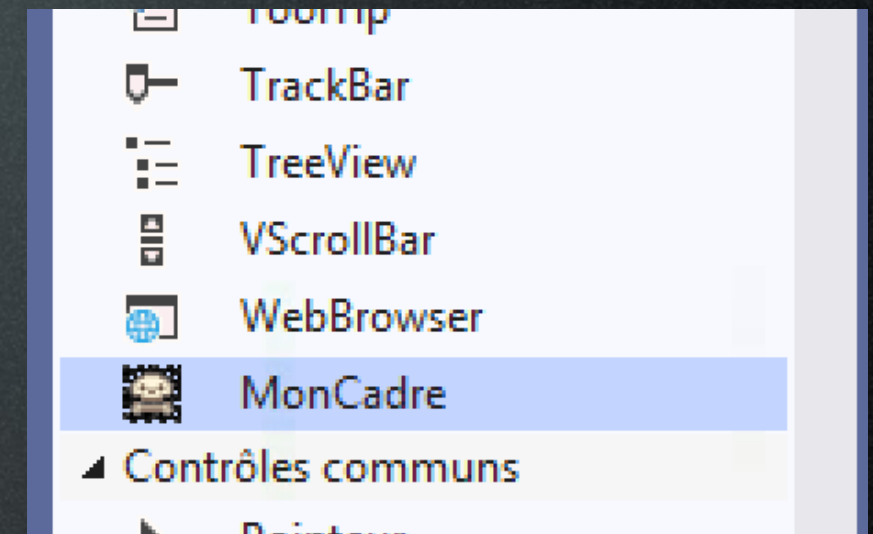
0 references

```
public MonCadre(string nomEtud, string commentaire, string nomImage)
{
    InitializeComponent();
    this.lblNom.Text = nomEtud;
    this.lblComment.Text = commentaire;

    // Affichage de l'image
    if (nomImage != string.Empty)
    {
        //nomImage = "@" + nomImage;
        pictureBox1.Image = Image.FromFile(nomImage);
    }
}
```


Association d'une image

- A utiliser si l'on souhaite que le UC soit plus facilement identifiable dans la boîte à outils
- Utiliser une image de type .bmp ou .ico (taille de 16 sur 16 pixels)
- Ajouter l'image au projet via l'explorateur de solution (**Clic droit - Ajouter un élément existant**)
- La déclarer en ressource incorporée (**Propriétés - Actions de génération**)
- Ajouter l'attribut à la déclaration de la classe



```
namespace mesContrôles
{
    [ToolboxBitmap(typeof(MonCadre), "crane.jpg")]
    public partial class MonCadre : UserControl
    {
        //
        // 0 references
        //
        public MonCadre()
        {
            InitializeComponent();
        }
    }
}
```


Génération de la dll

- Une fois la conception de l'UC achevée, génération de la solution
- Une dll (Dynamic Link Library) apparaît dans le dossier Debug.

bin	10/02/2020 15:34	Dossier de fichiers	
obj	10/02/2020 15:34	Dossier de fichiers	
Properties	10/02/2020 15:33	Dossier de fichiers	
mesControles	10/02/2020 15:35	Fichier CSPROJ	3 Ko
mesControles.sln	10/02/2020 15:34	Visual Studio Solu...	2 Ko
UserControl1.cs	11/02/2020 09:09	Visual C# Source F...	2 Ko
UserControl1.Designer.cs	11/02/2020 08:50	Visual C# Source F...	4 Ko
UserControl1.resx	11/02/2020 08:50	Microsoft .NET M...	6 Ko

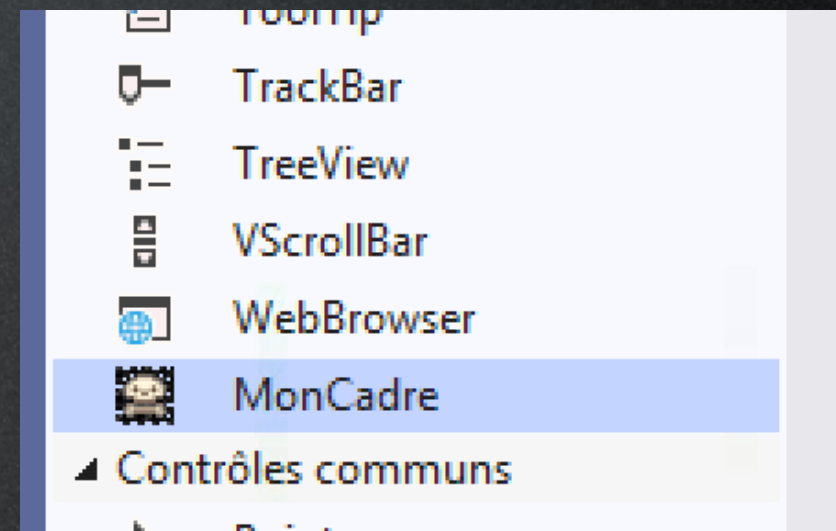
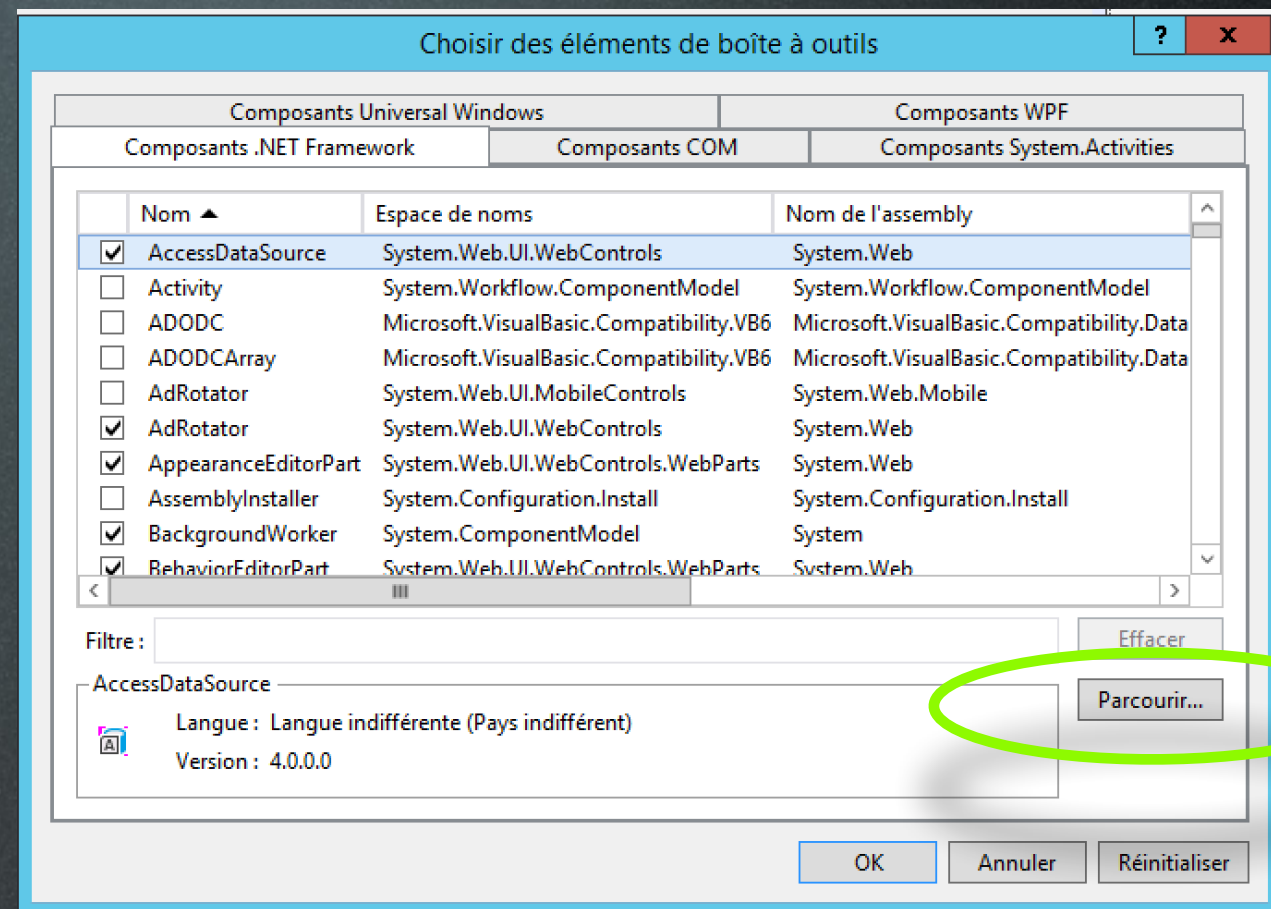
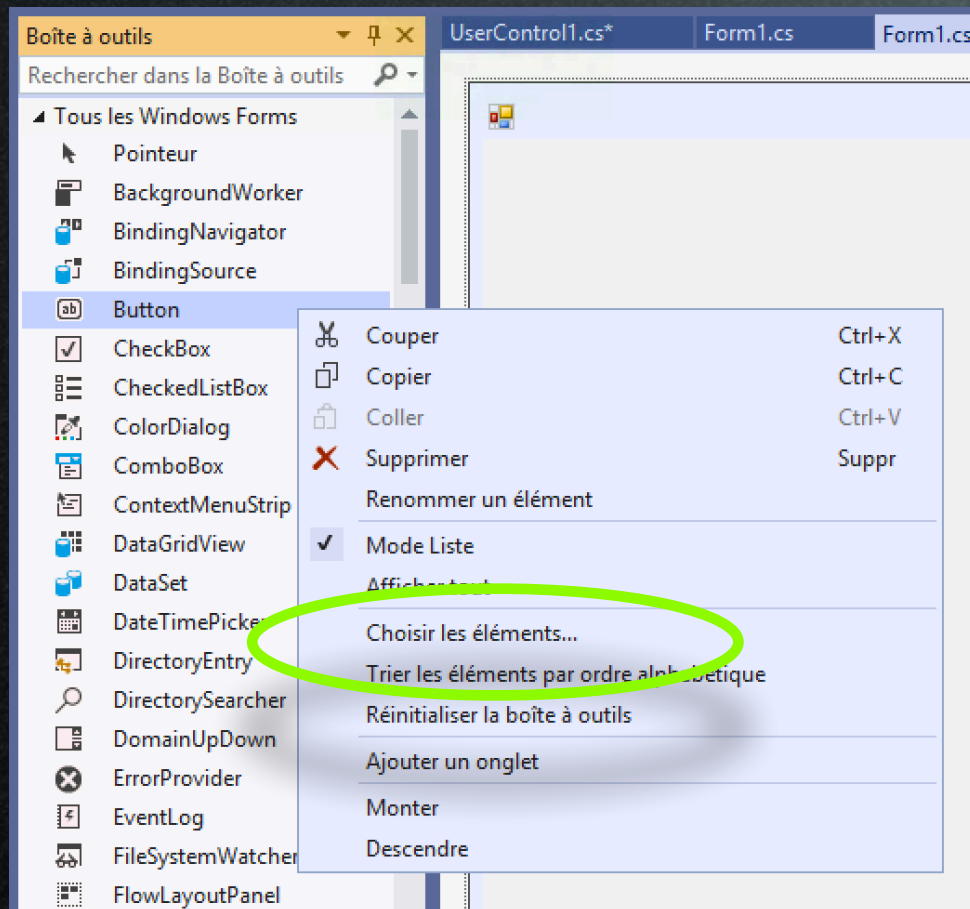
mesControles.dll	11/02/2020 09:10	Extension de l'app...	7 Ko
mesControles.pdb	11/02/2020 09:10	Program Debug D...	18 Ko

Du côté du projet test

- Création d'un projet test de type application Windows Form.
- Ajout du contrôle utilisateur à la boîte à outils :
 - clic droit puis « Choisir les éléments »
 - dans l'onglet « Composants du .Net Framework », clic sur le bouton Parcourir, puis recherche de la dll de l'UC
 - l'UC apparaît alors dans la boîte à outils

Une opération identique sert à ajouter la référence au projet UC via l'explorateur de solutions...

Ajout à la boîte à outils



Paramétrage via le code

- L'instanciation d'un contrôle utilisateur se fait de la même manière que pour les contrôles standards (après rajout d'une directive Using) :

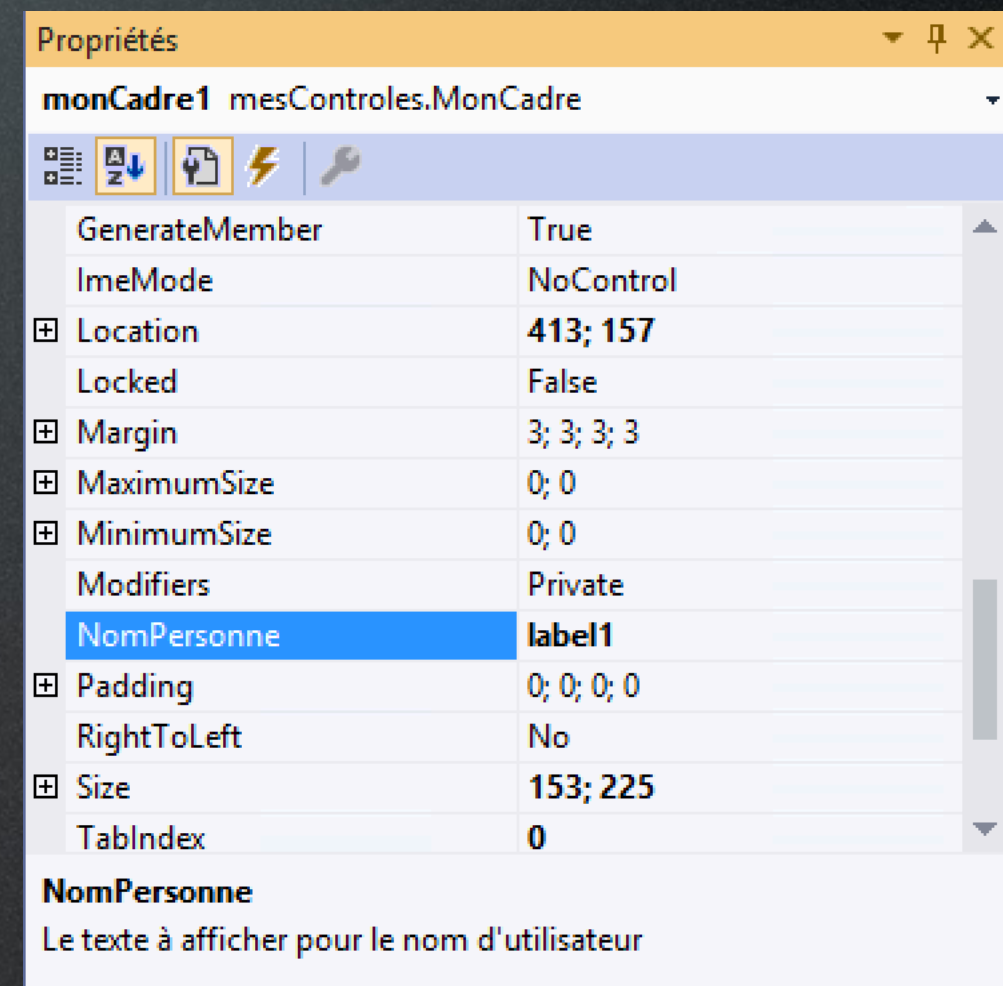
```
1 reference
private void Form1_Load(object sender, EventArgs e)
{
    int bordGauche = 80;
    int bordHaut = 50;
    string nomImage;
    string nomEtudiant;

    // Création du premier minion
    for(int i=1;i<=3;i++)
    {
        nomImage = @"..\..\Images\personne" + i.ToString()+".jpg";
        nomEtudiant = "Robert" + i.ToString();
        mesControles.MonCadre u = new MonCadre(nomEtudiant, "S1 G1", nomImage);
        u.afficheur = FicheSignaletique;
        u.Left = bordGauche;
        u.Top = bordHaut;
        u.Width = 150;
        u.Height = 220;
        bordGauche = bordGauche + u.Width + 30;
        this.Controls.Add(u);
    }
}
```


Pour aller plus loin...

- Pour le paramétrage du UC, il existe une autre méthode que la surcharge du constructeur : celle de **propriétés paramétrables** via la **fenêtre de propriétés**.
- Définition d'**attributs** de type Get/Set.

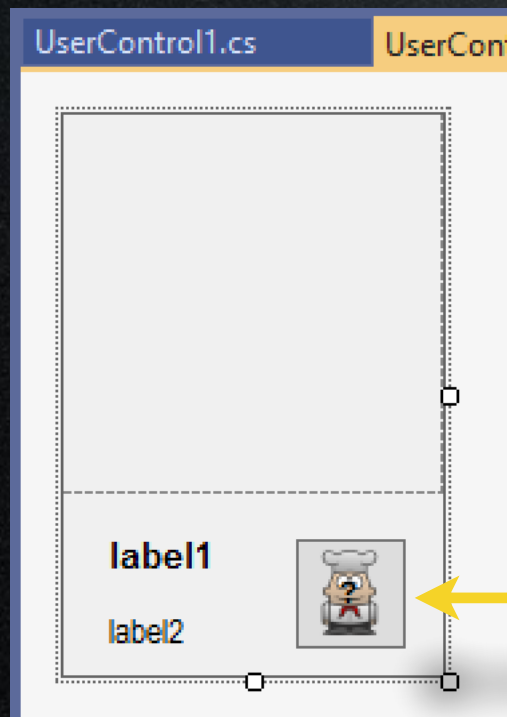
```
[Description("Le texte à afficher pour le nom d'utilisateur")]
public String NomPersonne
{
    get
    {
        return this.lblNom.Text;
    }
    set
    {
        this.lblNom.Text = value;
    }
}
```



Evénements d'un UC

- Option n°1 : prévoir un événement Clic « universel », qui effectuera la même action quel que soit le formulaire où je place l'UC.

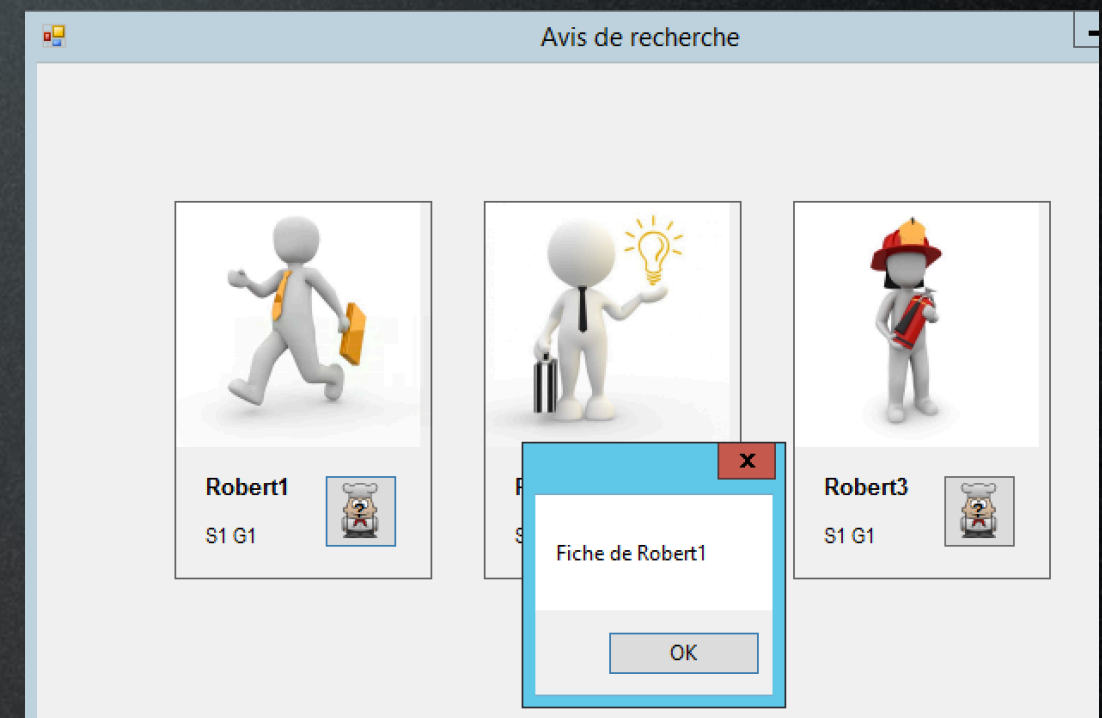
Côté UserControl



bouton btnInfos

```
1 reference
private void BtnInfos_Click(object sender, EventArgs e)
{
    MessageBox.Show("Fiche de "+lblNom.Text);
}
```

Côté projet test



Evénements d'un UC

- Option n°2 : laisser à l'utilisateur final le soin de décider ce qu'il fera en cas de clic sur le bouton btnInfos.
- Possibilité de coder des comportements différents suivant le contexte d'utilisation du UC.
- Utilisation du concept de Delegate : un délégué permet, comme son nom l'indique, de déléguer l'exécution d'une méthode à quelqu'un d'autre.

La notion de Delegate (1)

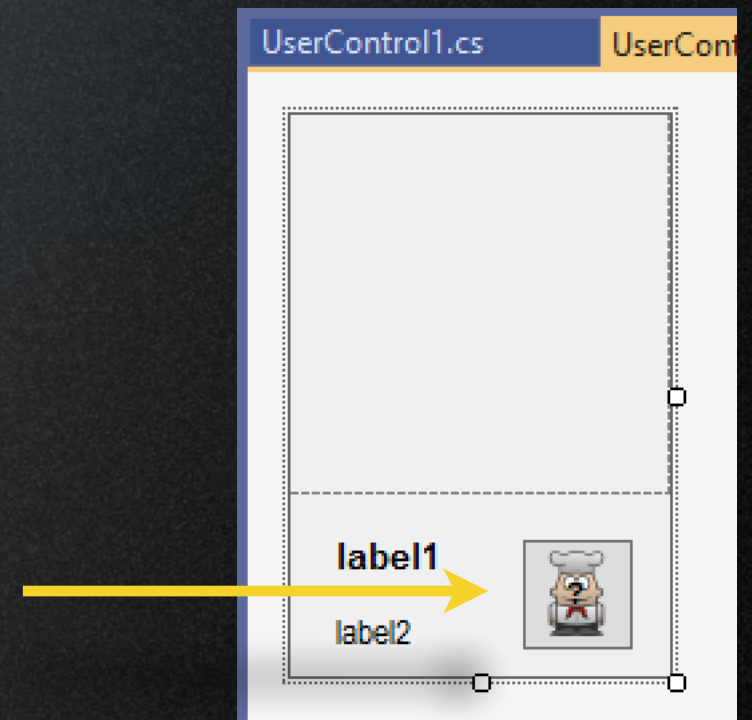
Pour le bouton btnInfos, on ne sait pas si l'utilisateur va l'utiliser :

- dans une application WinForms (et laquelle...)
- dans une application asp.Net

Le bouton btnInfos va déléguer l'affichage des informations à quelqu'un d'autre car il ne sait pas si l'affichage se fera :

- dans un label
- dans un MessageBox
- dans une ListBox...

bouton btnInfos



La notion de Delegate (2)

- Marche à suivre pour déléguer le clic à celui qui aura implémenté l'UC :

1. création de la signature du délégué

```
namespace mesControles
{
    // déclaration de la signature du délégué
    public delegate void afficherInfos(object sender, EventArgs e);
}
```

2. création d'une instance du délégué

```
public partial class MonCadre: UserControl
{
    0 references
    public MonCadre()
    {
        InitializeComponent();
    }

    // Déclaration d'une instance du délégué
    public afficherInfos afficheur;
}
```

procédure qui a la même signature que le clic sur un bouton

propriété publique accessible depuis l'extérieur


La notion de Delegate (2)

```
8      using System.Threading.Tasks;
9      using System.Windows.Forms;
10     using System.IO;
11
12     namespace mesControles
13     {
14         // déclaration de la signature du délégué
15         public delegate void afficherInfos(object sender, EventArgs e);
16
17         [ToolboxBitmap(typeof(MonCadre), "crane.jpg")]
18
19         4 références
20         public partial class MonCadre: UserControl
21         {
22             0 références
23             public MonCadre()
24             {
25                 InitializeComponent();
26             }
27
28             // Déclaration d'une instance du délégué
29             public afficherInfos afficheur;
30
31             0 références
32             public MonCadre(string nomEtud, string commentaire, string nomImage)
33             {
34             }
```


La notion de Delegate (3)

3. Codage de l'événement côté UC :

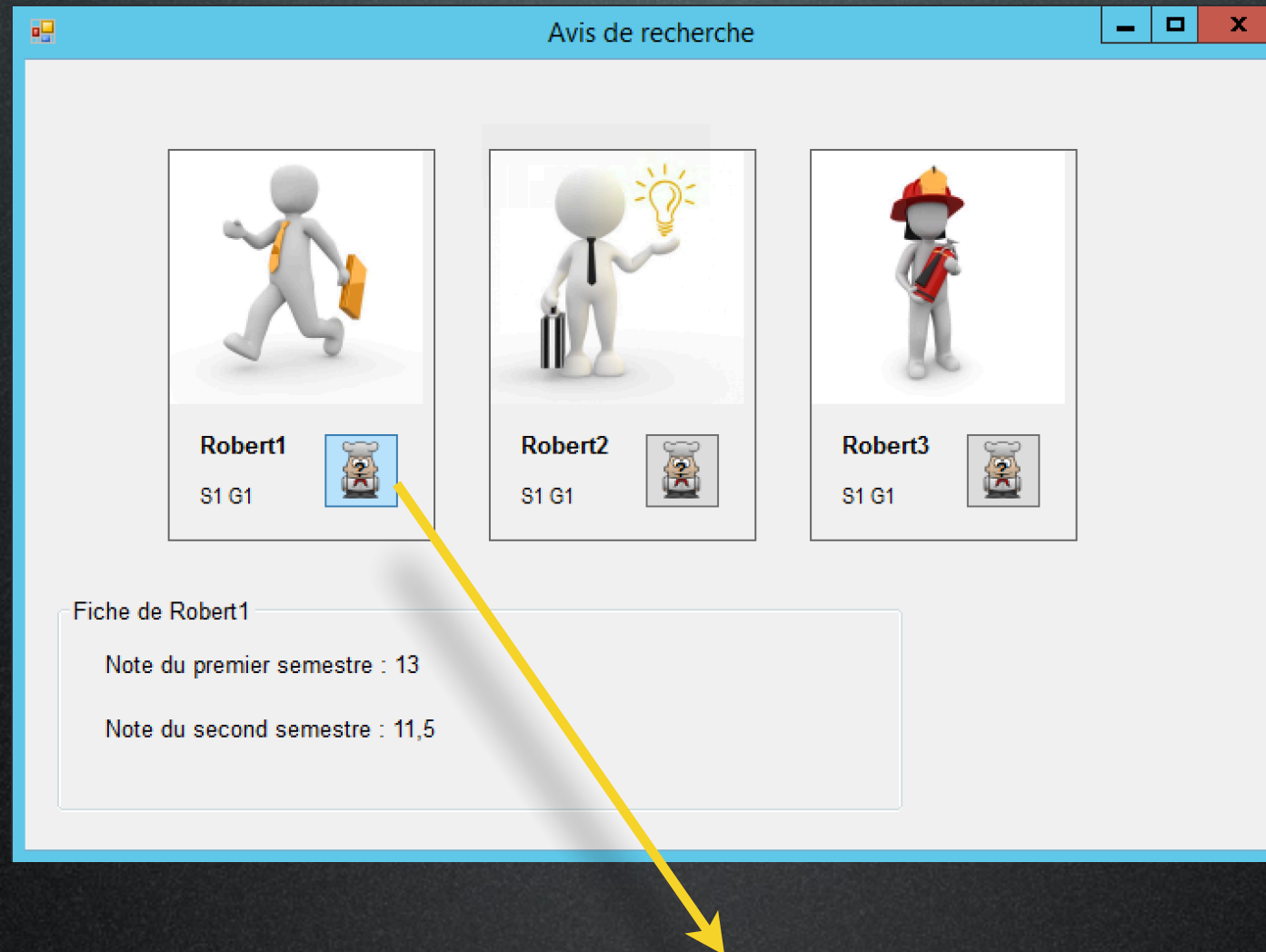
```
private void BtnInfos_Click(object sender, EventArgs e)
{
    // Affichage des informations souhaitées dans le contexte "appelant"
    if (this.afficheur != null)
        this.afficheur(sender, e);
}
```



si quelqu'un clique sur le bouton du UC, celui-ci « passe la main » à la procédure qui reçoit délégation dans le programme appelant

La notion de Delegate (4)

4. Codage côté application de test :



Le clic sur le bouton intégré au UC provoque l'apparition d'une GroupBox avec les résultats de la personne sélectionnée.

La notion de Delegate (5)

4. Codage côté application de test :

```
for(int i=1;i<=3;i++)  
{  
    nomImage = "personne" + i.ToString()+".jpg";  
    nomEtudiant = "Robert" + i.ToString();  
    mesControles.MonCadre u = new MonCadre(nomEtudiant, "S1 G1", nomImage);  
    u.afficheur = FicheSignaletique;  
    u.Left = bordGauche;
```

```
1 reference  
private void FicheSignaletique(object sender, EventArgs e)  
{  
    grpInfos.Visible = true;  
    // Cast du sender  
    Button btn = (Button)sender;  
    mesControles.MonCadre mc = (mesControles.MonCadre) btn.Parent;  
    // Récupération du nom de la personne sélectionnée  
    grpInfos.Text = "Fiche de " + mc.NomPersonne;  
}
```