Configurer les contrôles Windows Form

Sommaire

1 Introduction

Visual Studio 2008 vous propose un large éventail de contrôles pouvant composer votre application. Ils sont disponibles depuis la *ToolBox* dans la partie designer de votre formulaire et ils permettent ainsi d'interagir avec l'utilisateur au moment de l'exécution de l'application (un bouton, une *TextBox*, …). En plus de la gamme de contrôles que vous propose .NET Framework Windows Form, vous avez la possibilité de créer vos propres contrôles ou d'améliorer à votre guise ceux qui sont déjà présents. Il existe donc deux types de création de contrôle : les contrôles composés, qui sont constitués d'un ou plusieurs contrôles Windows Form (combinaison des contrôles existant) et les contrôles personnalisés, qui sont une création totalement personnelle. En résumé, vous verrez dans ce chapitre les deux types de création ou d'amélioration de contrôles en *Windows Form* ainsi que les outils permettant leur création.

Bon cours .Net

L'équipe *Windows Form*

2 Gestion des contrôles composés

2.1 Qu'est ce qu'un contrôle composé

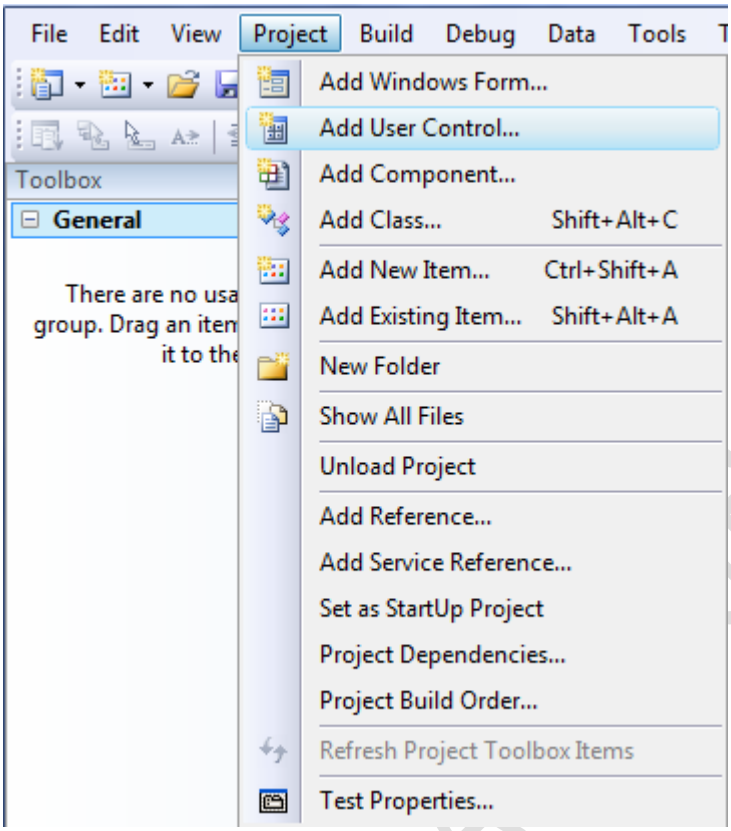
Un contrôle composé ou encore appelé contrôle utilisateur vous permet de combiner plusieurs contrôles existants afin d'en créer un nouveau. Par exemple vous pourrez afficher un *Timer* dans un label qui affichera l'heure. Les contrôles qui composent les contrôles composés sont appelés contrôles constitutif.

Les contrôles composés héritent de toutes les fonctionnalités et les propriétés de chaque contrôle Windows Form qui les compose. Mais vous aurez aussi la possibilité d'ajouter des méthodes, des propriétés et/ou des événements à vos nouveaux contrôles composés.

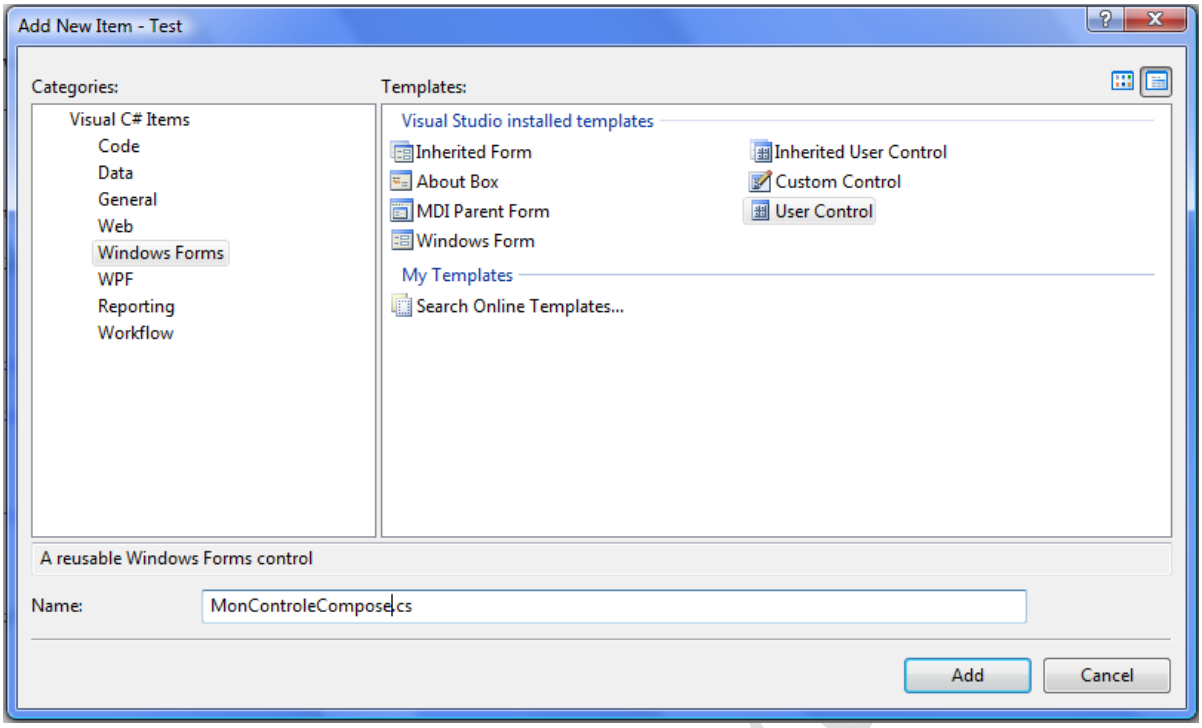
2.2 Création d'un contrôle composé

2.2.1 Créer le contrôle composé

Une fois votre projet Application Windows Form créé, vous pourrez ajouter un contrôle composé. Pour cela vous devez aller dans le menu Projet et cliquez sur "Ajouter un Contrôle Utilisateur…", ou bien vous faites un clique-droit sur le nom de votre projet dans l'Explorateur de Solution puis glisser votre souris sur Ajouter et enfin sélectionnez Contrôle Utilisateur….

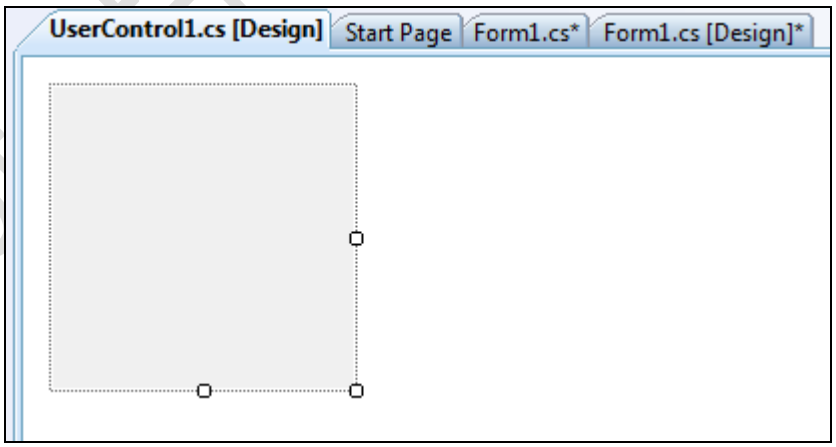


Une fois la boîte de dialogue Ajouter un Nouvel Élément ouverte, cliquez sur Windows Forms dans la partie Modèles puis sélectionnez Contrôle Utilisateur.



Aperçu de la boite de dialogue Ajouter Un Nouvel Élément.

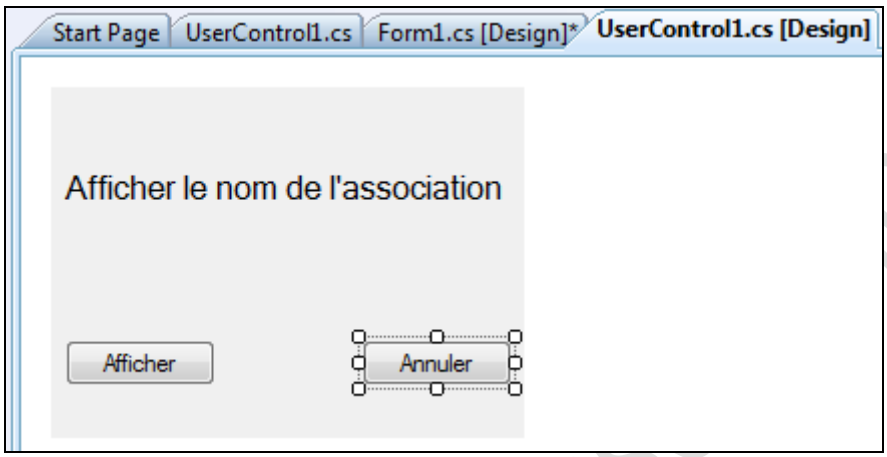
La partie Design de la classe de votre Contrôle Utilisateur s'affichera.



Vue d'ensemble de la partie Design de la classe Contrôle Utilisateur.

2.2.2 Personnaliser le contrôle composé

Pour personnaliser vos contrôles composés il vous faudra ajouter des contrôles constitutifs en réalisant un *Drag-and-Drop* depuis la *ToolBox* comme l'affiche l'imprime écran suivant :



Ensuite double-cliquez sur le bouton nommé "Afficher" afin de générer l'événement *Click*. Et ajoutez dans cet événement la propriété *Text* du Label qui prendra la valeur DotNet-France :

```
'VB

Private Sub btAfficher_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btAfficher.Click

    lbAffichage.Text = "DotNet-France"

  End Sub
```

```
//C#

private void btAfficher_Click(object sender, EventArgs e)

    {

       lbAffichage.Text = "DotNet-France";

    }
```

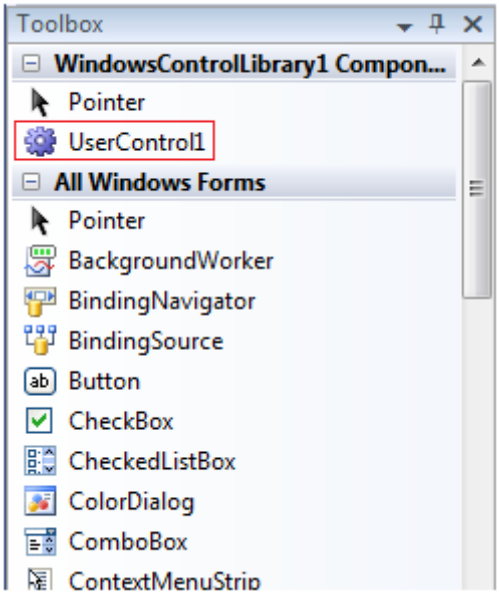
Procédez de la même manière pour le bouton Annuler en définissant la valeur de la propriété *Text* "Afficher le nom de l'association". Ensuite vous devrez ajouter votre contrôle composé à votre Form

depuis la *ToolBox* en faisant un *Drag-and-Drop* (voir la partie 2.3 Utiliser un contrôle composé dans votre application). Enfin vous pouvez débugger votre application.

Voici un aperçu de l'application au moment de l'exécution.

2.3 Utiliser un contrôle composé dans votre application

Pour utiliser vos contrôles composés vous devez suivre plusieurs indications qui sont les suivantes : tout d'abord enregistrer tout votre travail (Fichier->Tout Sauvegarder ou utilisez le raccourcis clavier Ctrl+Shift+S ou encore cliquez sur l'icône correspondant dans la barre des outils de Visual Studio 2008) puis vous devrez générer la solution depuis le menu Générer ou bien en appuyant directement sur la touche F6. Vous aurez ainsi la possibilité de réaliser un Drag-and-Drop de votre contrôle composé dans votre formulaire depuis la *ToolBox*.

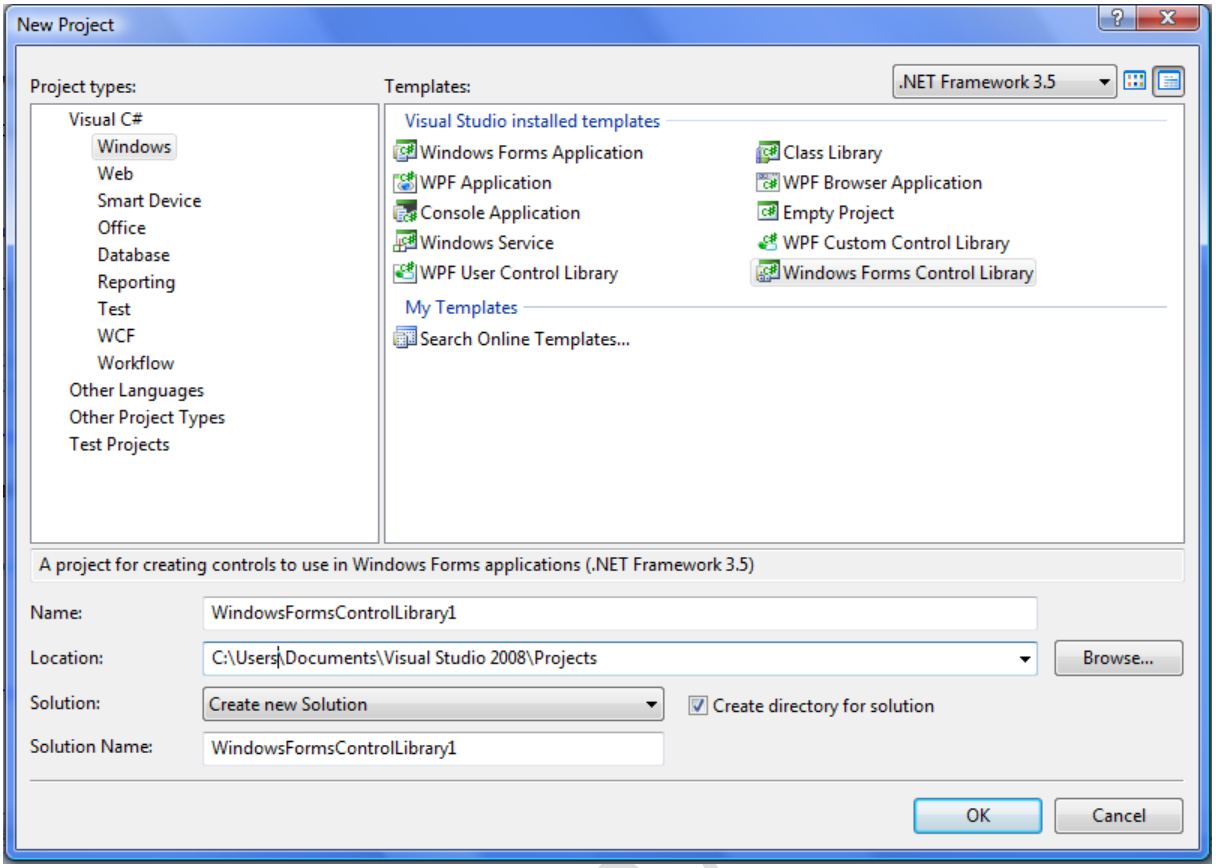


Il ne vous plus qu'à débugger votre application.

2.4 Créer une bibliothèque de contrôle

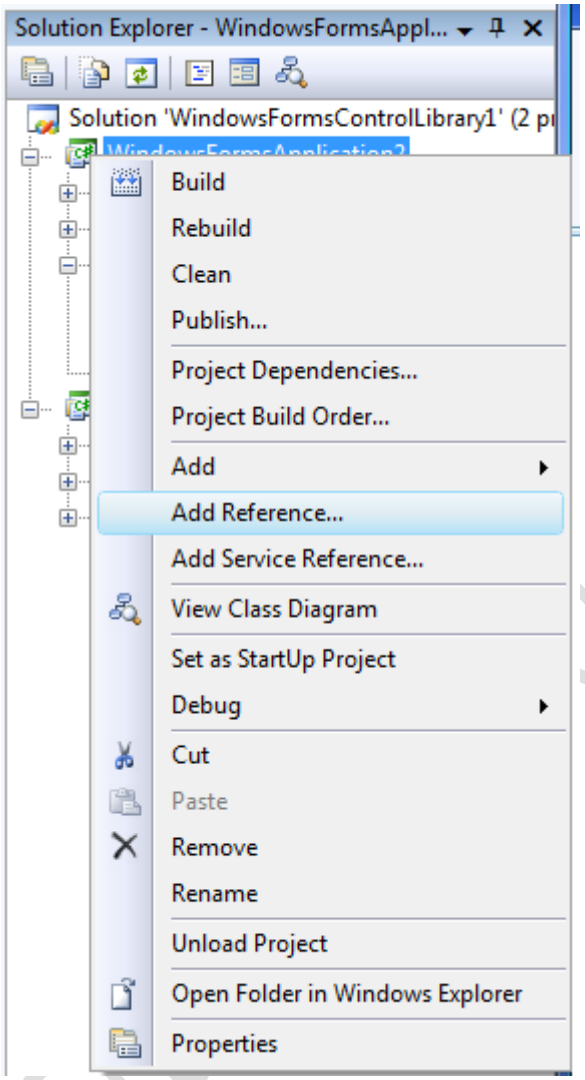
Vous avez la possibilité de créer une bibliothèque de contrôle qui contiendrait plusieurs contrôles composés. Cela vous permettra de les associer à d'autres projets et vous évitez ainsi de les recréer pour chaque projet que vous concevrez.

Donc vous devrez créer un nouveau projet (Fichier **->** Nouveau **->** Projet) puis cliquer, dans la partie Modèles, sur Windows et enfin sélectionnez "Bibliothèque de Contrôle Windows Forms". Ainsi vous pourrez produire tout les contrôles composés qui vous seront nécessaires dans d'autres projets. Pour terminer générez votre bibliothèque en faisant un clique-droit sur le nom de votre projet dans l'Explorateur de Solution puis en sélectionnant Générer.

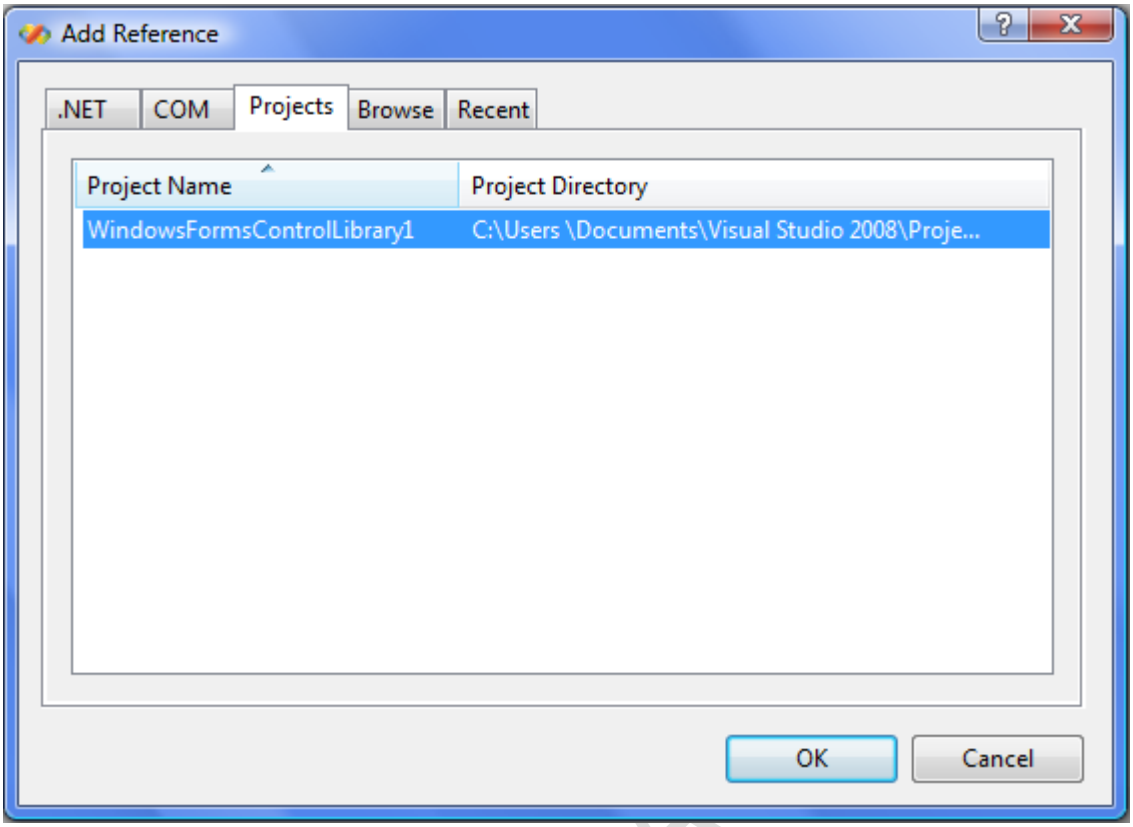


Aperçu de la boite de dialogue pour créer un nouveau projet.

Une fois votre projet Bibliothèque de Contrôles Windows Forms terminé, ajoutez un nouveau projet (Fichier **->** Ajouter **->** Nouveau Projet…) et sélectionnez “Application Windows Forms”. Ensuite vous pourrez utiliser dans votre nouveau projet les contrôles composés que vous avez créés.
Il vous faut alors faire une référence à la bibliothèque pour votre projet Windows Forms. Ceci permettra à votre projet d’hériter des contrôles issus de la bibliothèque. Vous devrez donc aller dans l’Explorateur de Solution puis faire un clique-droit sur le nom de votre projet Windows Forms et sélectionner “Ajouter Une Référence…”.

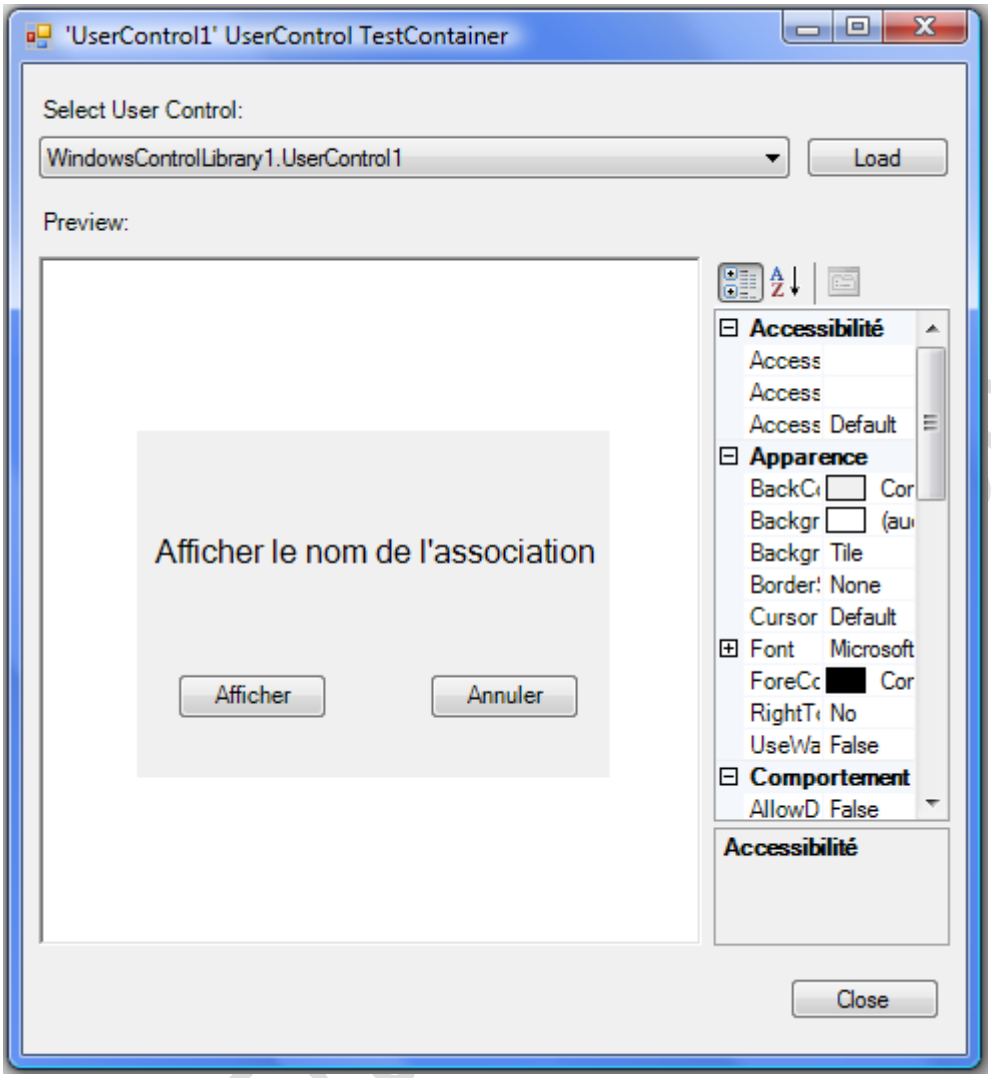


Puis lorsque la boîte de dialogue Ajouter une Référence est ouverte allez dans l'onglet Projet et sélectionnez votre bibliothèque de contrôles.



Enfin dans le menu Générer cliquez sur Générer la Solution ou bien appuyez directement sur la touche F6 de votre clavier. Vous pourrez ainsi utiliser les contrôles issus de vos bibliothèques de contrôles dans votre application Windows Forms en faisant un Drag-and - Drop sur votre Formulaire.

Remarque : Définissez votre projet Application Windows Forms comme étant le projet de démarrage (clique-droit sur votre projet et sélectionnez “Définir comme Projet de Démarrage”). Si votre projet Application Windows Forms n’est pas défini comme étant le projet de démarrage votre contrôle composé s’exécutera dans un Conteneur de test mais pas votre formulaire.



3 Les contrôles étendus

3.1 Présentation des contrôles étendus

Les contrôles étendus sont des types de contrôles auxquels vous ajoutez des propriétés, des méthodes ou encore des événements. Vous améliorez donc votre contrôle qu'il soit de type contrôle commun, contrôle utilisateur ou bien contrôle personnalisé. Dans l'exemple de la partie 3.2 vous verrez que nous ajoutons une propriété à un contrôle utilisateur qui renvoie une chaine de caractère que nous afficherons dans un label.

3.2 Exemple d'un contrôle étendu : ajout d'une propriété

Commencez donc par créer un projet contenant un formulaire et un label et ajoutez à votre projet un contrôle utilisateur. Dans la partie code de votre contrôle utilisateur, importez l'espace de nom *System.Windows.Forms* puis définissez votre classe *UserControl* comme étant un bouton pour cela remplacez *UserControl* par *Button* comme dans la ligne de code ci-dessous :

```
'VB
Public Class UserControl1
    Inherits Button
End Class
```

```
//C#
public partial class MonControleUtilisateur : Button
```

Remarque : dans la partie code du designer de votre contrôle utilisateur vous devrez supprimer la propriété *AutoScaleMode* car celle-ci n'existe pas pour un Bouton.

Ensuite établissez le code nécessaire pour établir une nouvelle propriété, toujours dans votre classe User Control, comme ci-dessous :

```
'VB
Dim maChaine As String
Public Property AfficherChaine() As String
    Get
        Return maChaine
    End Get
    Set(ByVal value As String)
        maChaine = value
    End Set
End Property
```

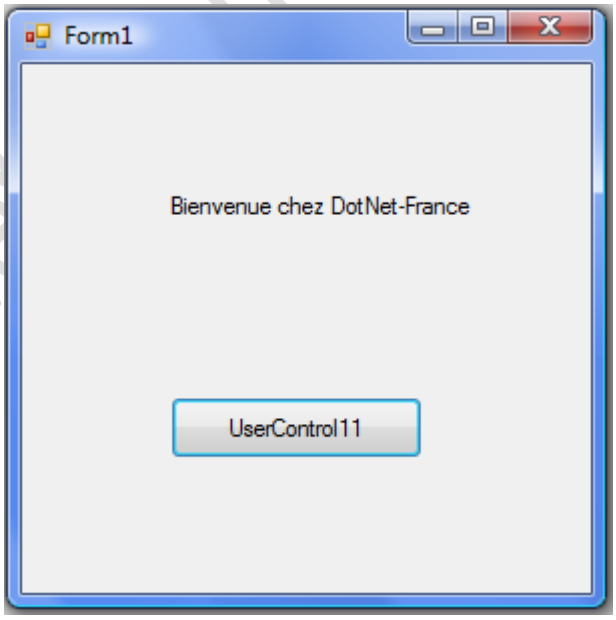
```
//C#
private string maChaine;
public string AfficherChaine
{
        get
        {
           return maChaine;
        }
        set
        {
          maChaine = value;
        }
}
```

Il ne vous reste plus qu'à sauvegarder votre projet puis à générer la solution. Ensuite allez dans la partie design de votre formulaire et faites un *Drag-and-Drop* de votre contrôle utilisateur sur votre formulaire (qui s'affiche dans votre *ToolBox*). Enfin faites un double clique sur votre contrôle utilisateur afin de générer l'évènement *Click* de votre contrôle et incorporez le code permettant d'afficher une chaine de caractère (que vous définissez soit dans la partie design de votre formulaire en faisant un clique-droit sur le contrôle puis en tapant la chaine que vous voulez dans la propriété AfficherChaine, soit en la définissant dans le code comme ci-dessous) comme le montre le code suivant :

```
'VB

Private Sub UserControl11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles UserControl11.Click

    UserControl11.ValeurBouton = " Bienvenue chez DotNet-France "

    Label1.Text = UserControl11.ValeurBouton.ToString()

  End Sub
```

```
//C#

private void userControl11_Click(object sender, EventArgs e)

    {

      userControl11.ValeurBouton = " Bienvenue chez DotNet-France ";

      label1.Text = userControl11.ValeurBouton.ToString();

    }
```

Ainsi lorsque vous déboguerez votre application vous devriez obtenir le résultat qui suit :



Si vous voulez introduire une nouvelle méthode pour votre contrôle, vous devrez, si elle ne retourne pas de valeur, utiliser *Sub* (Visual Basic) ou *void* (en C#), et si elle retourne une valeur, créer une fonction en Visual Basic avec *Function* ou spécifier un type de retour de la méthode en C#.

Et si vous voulez ajouter un nouvel évènement, vous devrez introduire le mot-clé Event dans votre code en Visual Basic, et en C# vous aurez l'obligation de déléguer l'évènement (avec le mot-clé *delegate*) avant de créer votre évènement avec le mot-clé *event*.

4 Les contrôles personnalisés, personnaliser ses contrôles

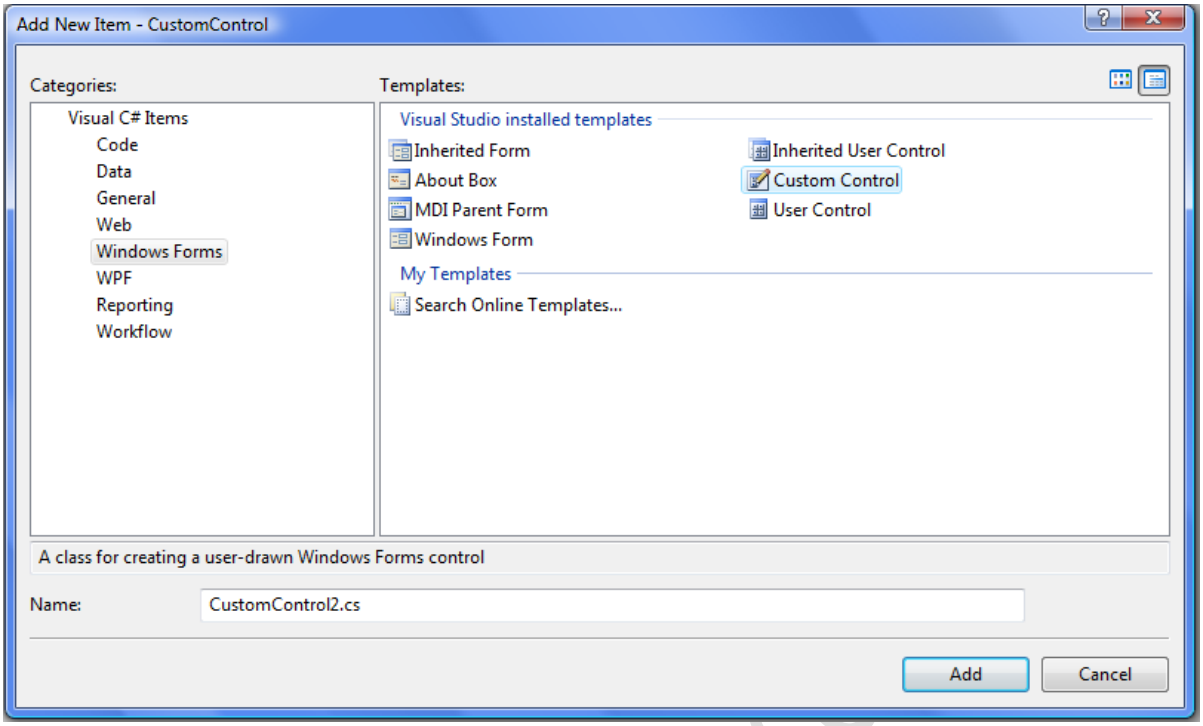
4.1 Introduction aux contrôles personnalisés

Les contrôles personnalisés ou *Custom Controls* fournissent le plus haut niveau de configuration presque entièrement personnalisable. Ils héritent de la classe *Control* qui permet d'obtenir toutes les fonctionnalités élémentaires pour les contrôles telles que les évènements. Cependant ils ne possèdent pas d'interface graphique ou encore de fonctionnalités spécifiques.

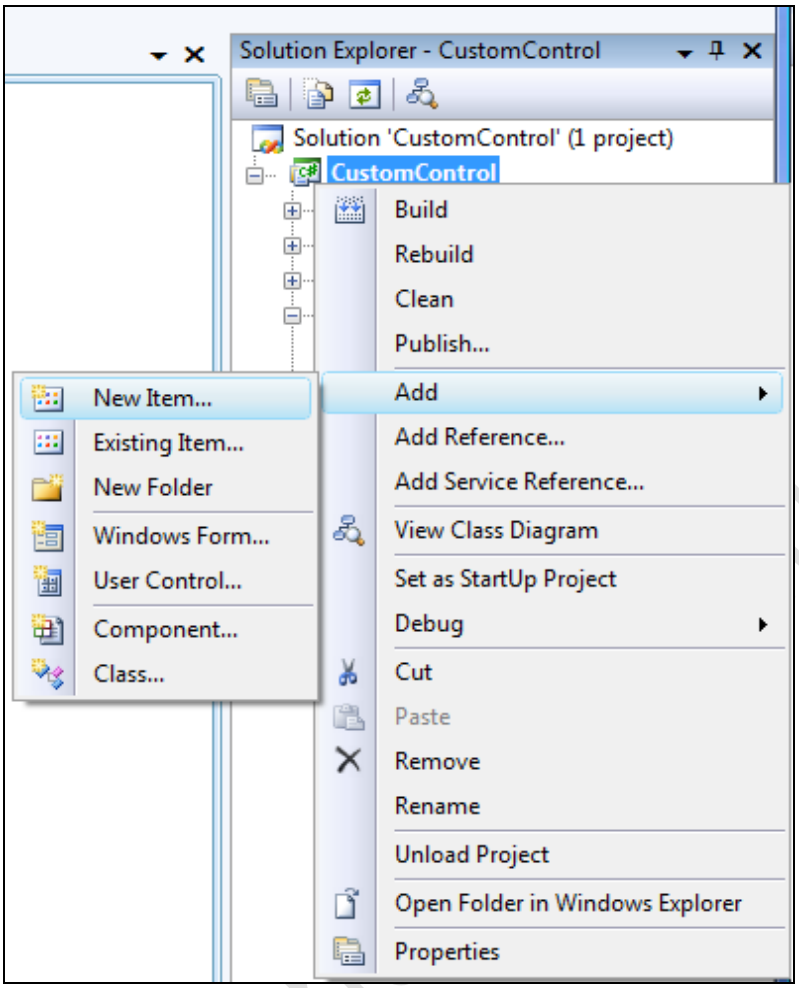
Ces contrôles exigent une plus grande réflexion sur l'héritage de la classe *Control* et leur développement que les contrôles utilisateurs.

4.2 Créer un contrôle personnalisé

Pour créer un contrôle personnalisé vous devez ajouter un nouvel élément permettant sa création. Vous devez donc aller dans le menu Projet -> Ajouter un Nouvel Élément puis dans la partie Modèles *Windows Forms* vous sélectionnerez Contrôle Personnalisé (*Custom Control*).



Vous pouvez aussi faire un clique-droit sur le nom de votre projet dans l'Explorateur de Solution puis sélectionner Ajouter…->Nouvel Élément.



4.3 Les Stylos et les Brosses

Lorsque vous redessinerez vos contrôles vous aurez surement besoin d'outils tel que le Stylo (*Pen*) ou la brosse/le pinceau (*Brush*).

4.3.1 Créer un Stylos

Un stylo ou *Pen* est un objet nécessaire dans le dessin des lignes. Pour créer un pinceau vous devez instancier la classe *Pen* et définir la couleur ainsi que la largeur de la ligne qui sera dessinée (en pixel).

Voici la structure à suivre pour instancier la classe *Pen* :

```
'VB

    Dim monStylo As New Pen(Color.CornflowerBlue, 2)
```

```
//C#
    Pen monStylo = new Pen(Color.CornflowerBlue, 2);
```

4.3.2 Créer une Brosse/un Pinceau

Contrairement aux stylos, la brosse/le pinceau ou *brush* permet de remplir les formes dessinées. La classe de base est la classe *Brush* qui comporte elle-même plusieurs sous-classes : *SolidBrush*, *TextureBrush*,… . La classe la plus utilisée et la plus simple est *SolidBrush* car elle nécessite comme paramètre qu'une seule couleur. Ci-dessous, vous trouverez un exemple utilisant cette classe :

```
'VB
    Dim maBrosse As New SolidBrush(Color.CornflowerBlue)
```

```
//C#
    SolidBrush maBrosse = new SolidBrush(Color.CornflowerBlue);
```

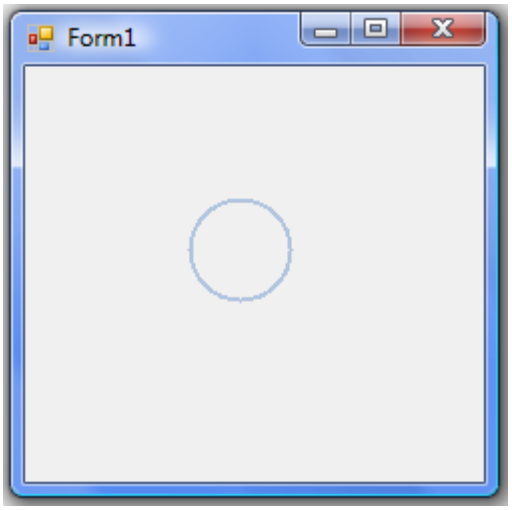
4.3.3 Exemple

Nous allons vous donner un exemple où nous dessinons une ellipse grâce à la classe *Graphics* issue de l'espace de nom *System.Drawing*. Insérer votre code dans la méthode "surchargée" *OnPaint* comme ci-dessous :

```
'VB
Public Class CustomControl1
```

```
    Inherits Control

    Protected Overrides Sub OnPaint(ByVal e As System.Windows.Forms.PaintEventArgs)

        MyBase.OnPaint(e)

'On dessine l'ellipse en définissant un stylo de couleur LightSteelBlue d'une largeur de trait 2 pixels puis on indique les caractéristiques de l'ellipse

        e.Graphics.DrawEllipse(New Pen(Color.LightSteelBlue, 2), 70, 40, 50, 50)

    End Sub

End Class
```

```
//C#

    public partial class CustomControl1 : Control

  {

    public CustomControl1()

    {

      InitializeComponent();

    }

    protected override void OnPaint(PaintEventArgs pe)

    {

//On dessine l'ellipse en définissant un stylo de couleur LightSteelBlue d'une largeur de trait 2 pixels puis on indique les caractéristiques de l'ellipse

      pe.Graphics.DrawEllipse(new Pen(Color.LightSteelBlue, 2), 70, 40, 50, 50);

      base.OnPaint(pe);

    }

  }
```



Voici le résultat que vous devriez obtenir

4.4 Exemple de personnalisation d'un contrôle bouton

Lorsque vous créez un contrôle personnalisé vous remarquez qu'une méthode *OnPaint* est générée (Faites un clique-droit sur la partie design de votre contrôle personnalisé et sélectionnez Afficher le code, ainsi vous accéderez au code de votre contrôle). Cette méthode permettra de dessiner votre contrôle donc vous devrez insérer le code permettant sa création.

Après avoir créé votre contrôle (voir la partie 3.2 de ce chapitre), accédez au code de celui-ci. Insérez ensuite le code suivant qui permet de créer un bouton qui change de couleur lorsque vous ferez un focus sur celui-ci :

```
'VB

Public Class CustomControl1

  Inherits Button

  Protected Overrides Sub OnPaint(ByVal e As System.Windows.Forms.PaintEventArgs)
```

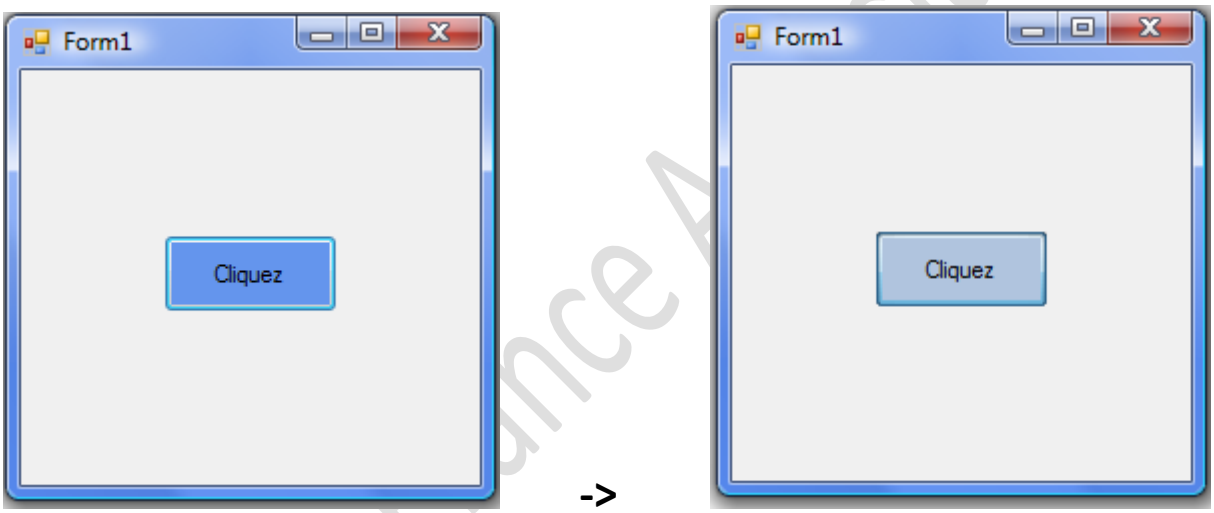
```
    MyBase.OnPaint(e)
    Me.Text = "Cliquez"
    If (Me.Capture = True) Then
      Me.BackColor = Color.LightSteelBlue
    Else
      Me.BackColor = Color.CornflowerBlue
    End If
  End Sub
End Class
```

```
//C#
public partial class CustomControl1 : Button
  {
    public CustomControl1()
    {
      InitializeComponent();
    }
    protected override void OnPaint(PaintEventArgs pe)
    {
      this.Text = "Cliquez";
      if (this.Capture == true)
        this.BackColor = Color.LightSteelBlue;
      else
        this.BackColor = Color.CornflowerBlue;
      base.OnPaint(pe);
    }
  }
```

Remarque : vous devrez remplacer "`public partial class CustomControl1 : Control`" par "`public partial class CustomControl1 : Button`" afin de créer un bouton personnalisé.

Nous définissons donc une propriété *Text* du contrôle puis nous créons une condition. Si le bouton a le focus alors la couleur changera (*LightSteelBlue*) sinon elle gardera sa couleur d'origine (*CornflowerBlue*).

Ensuite générez la solution depuis le menu Générer (ou la touche F6) puis ajoutez votre contrôle personnalisé dans votre formulaire puis débuggez l'application. Vous devrez obtenir ainsi :



Voici une représentation visuelle de votre formulaire et de votre contrôle personnalisé lors du débogage de l'application (Deux état : l'un n'est pas focus et l'autre l'est).

4.5 La méthode *OnPaint*

C'est la méthode qui vous permet de redessiner votre contrôle personnalisé. Celle-ci est automatiquement "surchargée" grâce au mot-clé *override* (en C#) et *Overrides* (Visual Basic). Tout le code que vous incorporerez pour cette méthode devra être hérité de l'espace de nom *System.Drawing* et *System.Windows.Forms* pour les contrôles.

5 Conclusion

Vous arrivez ainsi à la fin de ce chapitre. Entrainez-vous le plus possible afin de comprendre les nombreuses méthodes permettant de créer vos propres contrôles, d'étudier les contrôles composites ou encore d'améliorer vos contrôles.

L'équipe *Windows Form*.