M13 : Programmation Événementiel

CS.Net

Formateur : Driouch (cfmoti.driouch@gmail.com)

Etablissement : OFPPT/CFMOTI

04/06/2012

http://www.ista-ntic.net/

Plan

- Introduction
- Objet Form et la Méthode Main
- Explorateur de solution
- Fenêtre de propriété
- Boite outils
- Programmation par événement

Quelques définitions…

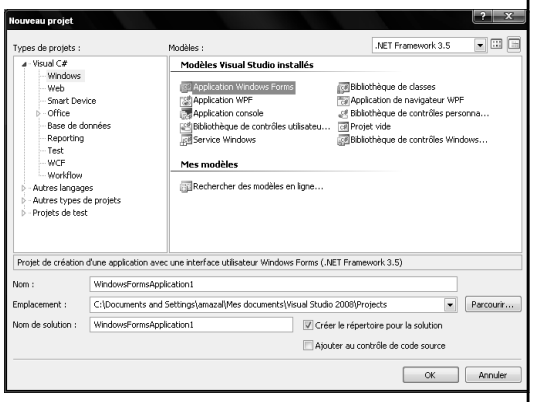
Programmation événementielle :

Une application Windows est essentiellement constituée :

d'une ou de plusieurs fenêtre(s),

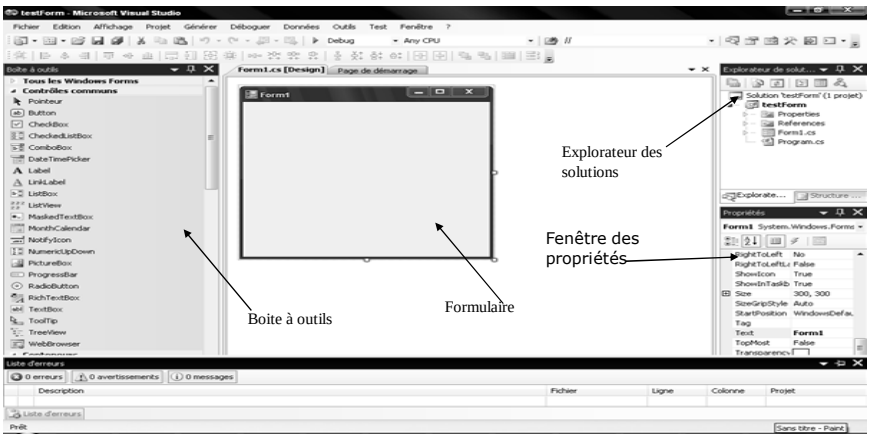
d'un ensemble de composant visuel (bouton de commande, zone de saisie, case a cocher, …).

Création d'un projet «Application Windows Froms»



Objet Form

Un formulaire est créé automatiquement par Visual Studio (VS) ainsi que son code associé.



Le Formulaire

La classe forme est la classe de base pour créer des interfaces dans une application Windows Form.

Lors de la création d'un projet Windows Forms, un formulaire par défaut est créé (Form1). Le code généré par VS est le suivant (double cliquez sur le formulaire)

```
public partial class Form1 : Form
  {  public Form1()
     {   InitializeComponent();
     }
  }
```

Le mot clé ***Partial*** indique que la classe est partielle : Il est possible de fractionner la définition d'une classe sur deux fichiers sources ou plus. Chaque fichier source contient une section de la définition de classe, et toutes les parties sont combinées lorsque l'application est compilée.

Le formulaire créé est dérivé de la classe Form qui se trouve dans l'espace de nom System.Windows.Forms.

Le constructeur appelle la méthode InitializeComponent() qui crée et initialise tous les contrôles du formulaire. Cette méthode est générée automatiquement par le Form Designer.

La méthode Main

Lors de la création d'un projet Windows Forms, un fichier source est créé : Program.cs. On y trouve notamment la fonction Main qui est le point d'entrée de tout programme, y compris des programmes Windows :

```
static class Program
  {  /// <summary>
     /// Point d'entrée principal de l'application.
     /// </summary>
     [STAThread]
     static void Main()
     {  Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());}
  }
```

Au démarrage de l'application, la fonction Main est automatiquement exécutée. Dans celle-ci :

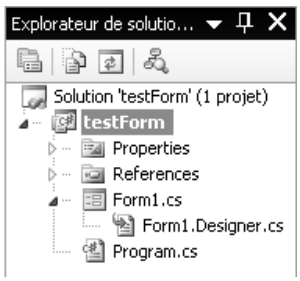
- EnableVisualStyles()) : autorise le style XP de manière à retrouver dans notre programme des boutons et autres composants au look XP.

- SetCompatibleTextRenderingDefault(false) : force les contrôles à s'afficher en utilisant le GDI+ de .NET (interface graphique Windows permettant de créer des graphiques, de dessiner du texte et de manipuler des images graphiques en tant qu'objets)

- Run : lance l'application graphique (On crée l'objet fenêtre avec new Form1)

L'explorateur des solutions

L'explorateur des solutions présente de manière arborescence et visuelle les objets composant l'application chargée. La figure suivante montre que le projet de nom 'testForm' est composé d'un seul formulaire de nom 'Form1'.



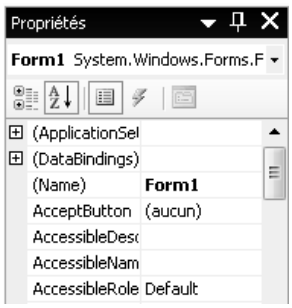
La fenêtre des propriétés

La fenêtre **Propriétés** présente les propriétés (ou attributs) de l'objet sélectionné. La figure suivante présente les propriétés de l'objet (sélectionné) Form1.

On peut citer quelques propriétés de 'Form1' et leurs valeurs respectives :

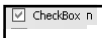
- Name = Form1 : nom logique utilisé pour référencer l'objet dans du code VB
- Text = Form1 : nom qui apparaît visuellement sur l'objet (celui-ci peut être différent de la propriété Name)

Notez que la valeur de chaque propriété peut être modifiée en cliquant sur la colonne de droite de la fenêtre **Properties**

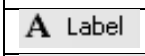
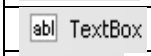
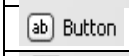
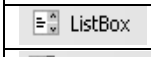
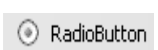
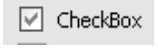


La boîte à outils et les contrôles standards

- La partie graphique de votre application va contenir un (ou plusieurs) formulaire(s). Sur un formulaire, on peut placer un ou plusieurs objets graphiques ou ce qu'on appellera des **contrôles** (Bouton à cliquer, Champ libellé (texte statique), Champ texte à saisir au clavier, Menu, etc.).
- Tous les contrôles héritent de la classe **Control**. Chaque contrôle correspond à une classe de l'espace de nom **System.Windows.Forms**. On cite à titre d'exemple le contrôle bouton qui correspond à la classe System.Windows.Forms.Button.
- Ces contrôles sont des objets pré-programmés dont l'utilité principale est de faciliter l'interaction avec l'utilisateur. Chacun de ces objets graphiques a une fonctionnalité bien précise. Le tableau suivant résume les contrôles standards de base les plus utilisés:



La boîte à outils et les contrôles standards

Contrôle	Nom du contrôle	Utilité
	Label	Afficher un texte statique : un libellé
	Text Box	Afficher et saisir une valeur au clavier
	Button	Lancer l'exécution une procédure événementielle
	ListBox	Afficher une liste statique de valeur
	ComboBox	Afficher une liste déroulante
	RadioButton	Sélectionner une option. Si utilisé en plusieurs instances, une seule option peut être choisie
	Check Box	Sélectionner une option. Si utilisé en plusieurs instances, une ou plusieurs options peuvent être choisies

La boîte à outils et les contrôles standards

Les controles ont des propriétés en commun parmi lesquelles, on cite :

Name : permet d'attribuer un nom à un contrôle.

Text : défini le texte qui apparaît sur un contrôle (dépend du contexte).

Enabled : permet d'activer ou de désactiver un contôle (True :activé, False :Désactivé)

Visible : donne la possibilité de cacher ou rendre visible un contrôle

Left : détermine l'abscisse du bord gauche du contrôle par rapport au contrôle conteneur

Top : détermine l'ordonnée du bord supérieur du contrôle par rapport au contrôle conteneur

=> La combinaison des deux, positionne le coin supérieur gauche d'un contrôle par rapport au conteneur

Height : retourne ou définit la hauteur d'un contrôle

Width : retourne ou définit la largeur d'un contrôle

Remarque : La propriété PasswordChar (du contrôle TextBox) permet de spécifier le caractère de masquage pour la saisie des mots de passe.

Programmation par événements

- C# est un langage qui permet de réaliser de la programmation par événements, c'est à-dire programmer des méthodes qui s'exécutent quand un événement est déclenché.
- La plupart du temps, l'événement est déclenché par l'utilisateur du programme (click sur la souris, appuie sur une touche du clavier, fermeture d'une fenêtre). La programmation par événements consiste à programmer ce que le programme doit faire quand un événement particulier survient.
- A chaque objet C# (contrôle, formulaire, etc.) peut être associé une ou plusieurs procédures événementielles écrites avec le langage de programmation C#. Ci-dessous quelques exemples d'événements :

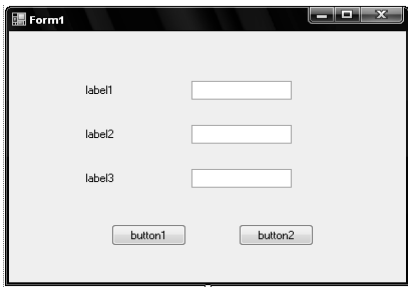
Programmation par événements

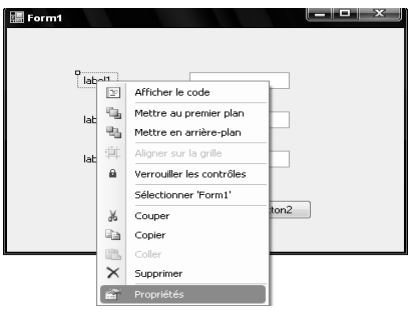
Evénement	Se produit quand
Click	On clique sur le bouton gauche de la souris
DblClick	On double clique sur le bouton gauche de la souris
Load	L'objet est chargé
MouseDown	On clique sur la souris sans relâcher le bouton
MouseUp	On a relâché le bouton de la souris
MouseMove	On a bougé la souris

Programmation par événements

Exemple : Calcul de la somme de deux nombres

Personnaliser le formulaire ainsi que ses différents contrôles





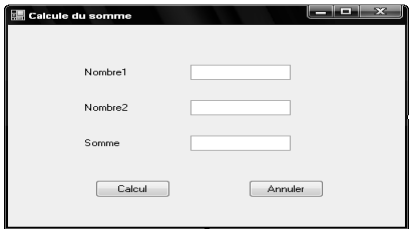
Programmation par événements

Effectuer les modifications suivantes :

Formulaire ou Contrôle	**Propriétés à modifier**
Form1	Text : Calcul du somme
label1	Text : Nombre1
label2	Text : Nombre2
label3	Text : Nombre3
textBox1	Name : nombre1TextBox
textBox2	Name : nombre2TextBox
textBox3	Name : CalculTextBox enabled : false
button1	Name : CalculButton Text : Calcul
button2	Name : AnnulerButton Text : Annuler

Programmation par événements

L'interface obtenue est la suivante :



Il faut par la suite définir les actions à exécuter lorsqu'on clique sur le bouton Calcul. Pour cela, double cliquer sur ce bouton et écrire le code suivant :

```
private void CalculButton_Click(object sender, EventArgs e)
    {
      if (Nombre1TextBox.Text != "" && Nombre2TextBox.Text != "")
      {
        float res = float.Parse(Nombre1TextBox.Text) + float.Parse(Nombre2TextBox.Text);
        CalculTextBox.Text = res.ToString();
      }
      else MessageBox.Show("Veuillez saisir deux nombres","Erreur");
    }
```

Programmation par événements

Le paramètre **Object** (sender) : représente le contrôle qui a déclenché l'événement.

Le paramètre **EventArgs** : contient des informations supplémentaires concernant l'événement qui est déclenché.

La classe **MessageBox** permet l'affichage d'informations à l'utilisateur ou la demande d'une confirmation. Un message est affiché via l'appel à la méthode statique **Show**. Ses Principaux paramètres sont : le texte à afficher et le titre de la fenêtre

De même, Il faut définir les actions à exécuter lorsqu'on clique sur le bouton Annuler. Pour cela, double cliquer sur ce bouton et écrire le code suivant :

```
private void AnnulerButton_Click(object sender, EventArgs e)
    {
      Nombre1TextBox.ResetText();
      Nombre2TextBox.ResetText();
      CalculTextBox.ResetText();
    }
```

La méthode ResetText Rétablit la valeur par défaut de la propriété Text.