Les menus et événements

Sommaire

1 Introduction

Dans ce quatrième chapitre, vous découvrirez deux contrôles (*ToolStrip* et *MenuStrip*) qui s'avèrent importants dans la création d'un programme en *Windows Form*. Vous apprendrez aussi à gérer les évènements et utiliser les *Events Handler*.

Bon cours .Net

L'équipe *Windows Form*

2 Gérer les *ToolStrips*

2.1 Présentation du *ToolStrip*

Le *ToolStrip* est un nouveau contrôle créé pour .NET Framework 2.0. Il permet de configurer facilement des *ToolBars* qui ont l'apparence des barres d'outils de Microsoft Internet Explorer ou encore Microsoft office et d'allure professionnel. Il permet d'utiliser les *ToolStripItems* (éléments du contrôle) qui fournissent une grande variété de fonctionnalité.

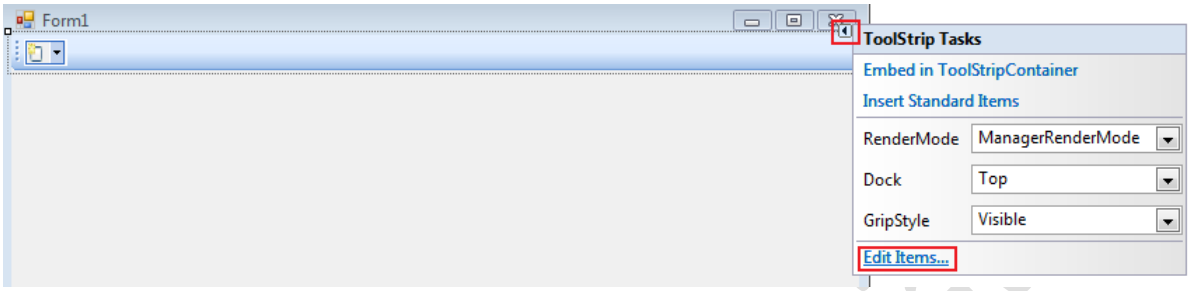
Voici les principales propriétés d'un *ToolStrip* :

Types de Propriétés	Description
AllowItemReorder	Permet d'indiquer si les *Item* peuvent être réorganisés par l'utilisateur. Si sa valeur est *True* l'utilisateur peut déplacer les *Item* en maintenant la touche *Alt*.
AllowMerge	Permet d'indiquer si votre *ToolStrip* peut se fusionner avec un autre *ToolStrip*.
CanOverFlow	Permet de définir si l'utilisateur peut mettre des *Item* sous des menus déroulant.
Dock	Permet d'indiquer où est le *ToolStrip*. En effet il est mieux de mettre le *ToolStrip* sur un bord de votre *Formulaire*.
IsDropDown	Indique si le *ToolStrip* est un contrôle *ToolStripDropDown*.
LayoutStyle	Permet de définir l'emplacement des éléments dans votre *ToolStrip*, par exemple les classer verticalement ou horizontalement.
RenderMode	Permet de définir un style visuel de design, par exemple, un style professionnel ou système.
ResizeRedraw	Indique si le *ToolStrip* se redessine lors d'un redimensionnement.
ShowItemToolTips	Permet de définir si les éléments du *ToolStrip* peuvent afficher des infos bulle.
Stretch	Permet de définir si le *ToolStrip* prend toute la longueur du *ToolStripContainer*.
TextDirection	Permet d'indiquer la direction du texte dans vos éléments de votre *ToolStrip*. Par exemple, vous pouvez mettre votre texte verticalement.
VScroll	Définit une valeur pour qu'une barre de défilement verticale s'affiche.

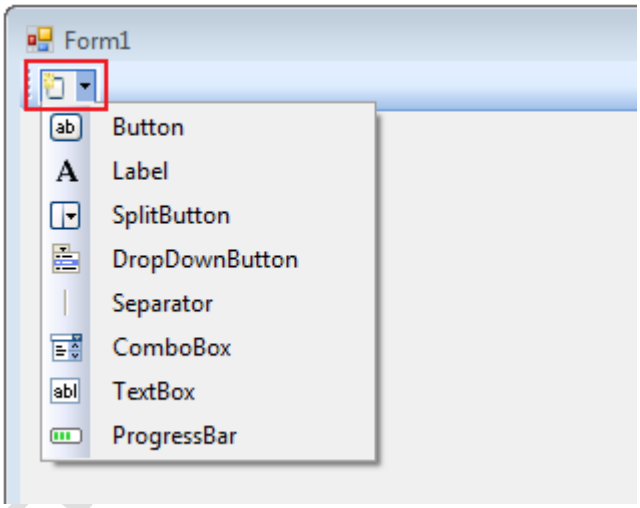
2.2 Ajouter des éléments dans votre *ToolStrip* en utilisant l'interface utilisateur ou le code de programmation

Au moment de la conception de votre *ToolStrip*, vous avez deux façons d'ajouter des éléments :

- Soit en cliquant sur le petit onglet *ToolStrip Tasks* puis *Edit Items…* :



- Soit directement avec un onglet qui s'affiche lorsque que le *ToolStrip* est sélectionné :

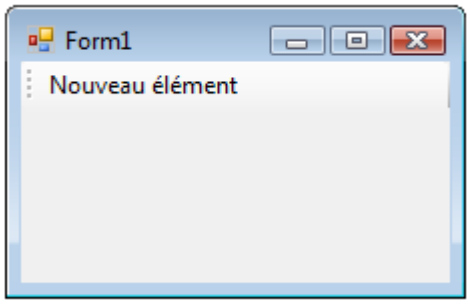


Vous pouvez aussi ajouter ces éléments directement dans le code, par exemple, vous pouvez ajouter un bouton « Nouveau élément » :

```
'VB

Public Sub New()

InitializeComponent()
Dim nouveauElement As ToolStripItem
nouveauElement = toolStripTest.Items.Add("Nouveau élément")

End Sub
```

```
//C#

public Form1()

{

   InitializeComponent();
ToolStripItem nouveauElement;
nouveauElement = toolStripTest.Items.Add("Nouveau élément");

}
```

Ce qui donne :



Remarque : Implémenter votre code dans la classe de votre Formulaire après " InitializeComponent ( ) ; ", afin que votre élément soit ajouté dans votre contrôle.

2.3 Les différents éléments composant le *ToolStrip*

Le *ToolStrip* a été créé dans le but d'accueillir plusieurs éléments déjà existant hors *ToolStrip* comme par exemple des boutons. Certains contrôles sont par contre spécifiques au *ToolStrip*. Voici les éléments d'un *ToolStrip* :

Types d'éléments	Description
ToolStripLabel	La particularité de cet élément est qu'il peut être à la fois un *Label* mais aussi un *LinkLabel*. Il suffit de changer la propriété de *IsLink* pour avoir l'un des deux.
ToolStripButton	Il s'agit d'un bouton comme un *Button* normal. A la base il est juste représenté par une image mais il est configurable. Vous pouvez associer une icône avec un texte, etc…
ToolStripSeparator	Il s'agit d'un élément particulier, il permet surtout de mettre une sorte de « barre » permettant de séparer visuellement nos éléments.
ToolStripComboBox	Cet élément fonctionne comme une *ComboBox* classique.
ToolStriptextBox	Cet élément est identique à une *TextBox* classique. En revanche il est impossible d'utiliser la propriété *Multiline*.
ToolStripProgressBar	Cet élément est similaire à une *ProgressBar*. Il est par exemple possible de définir son style avec un avancement en blocs ou en continue.
ToolStripDropDownButton	Cet élément permet d'ouvrir un menu déroulant lorsque que l'utilisateur clique sur l'icône ou la flèche à droite.
ToolStripSplitButton	Cet élément fonctionne comme le *ToolStripDropButton*, c'est-à-dire qu'il permet d'ouvrir un menu déroulant. Mais l'utilisateur peut en revanche cliquer sur l'icône pour une autre action. C'est-à-dire que pour ouvrir le menu déroulant, il faut cliquer sur la flèche.

2.4 Gérer les images dans un *ToolStrip*

Les éléments *ToolStripButton*, *ToolStripDropDownButton* et *ToolStripSplitButton* peuvent être personnalisés par des images et/ou texte.

Voici les propriétés principales pour gérer les images et textes :

Types de Propriétés	Description
DisplayStyle	Permet de définir si le contrôle affiche du texte, une image ou les deux.
Image	Permet d'associer une image au contrôle.
ImageAlign	Permet de définir comment l'image est placée sur le contrôle.
ImageScaling	Permet de définir si l'image sera redimensionnée pour être adaptée au contrôle.
ImageTransparentColor	Permet de définir la couleur de fond lorsque l'image (ou une partie) est transparente.

2.5 Le container *ToolStripContainer*

Le *ToolStripContainer* est une classe permettant de contenir un ou plusieurs *ToolStrip*. De plus, avec cette option, l'utilisateur du logiciel pourra déplacer librement son/ses *ToolStrip* sur tous les bords de la *Form*.

Vous pouvez définir quels bords seront disponibles pour l'utilisateur avec les propriétés (*Top* ou *Right*/*Left*/*Bottom*)*ToolStripPanelVisible*.

2.6 La fusion de *ToolStrip*

Les *ToolStrip* peuvent être fusionnés lors de l'exécution de votre logiciel avec la méthode *ToolStripManager.Merge*.

Lorsque des *ToolStrip* sont fusionnés, chaque élément des *ToolStrip* sources est comparé avec chaque élément des *ToolStrip* visés. Pour pouvoir fusionner correctement vos ToolStrip, la propriété *AllowMerge* doit être sur *True*.

3 Configurer et créer des menus

3.1 Présentation du contrôle *MenuStrip*

Le contrôle *MenuStrip* est un regroupage de commandes qui les rend plus accessible et qui est optimisé pour l'affichage de *ToolStripMenuItems*. Les *ToolStripMenuItems* sont des contrôles qui permettent une meilleure visualisation dans les menus. Ils peuvent être sous une ou plusieurs formes (texte ou image). Ainsi grâce au *MenuStrip* vous pourrez créer un menu d'éléments pour votre programme.

Remarque : Un *MenuStrip* est le menu que l'on retrouve dans pratiquement tous les programmes tels que Microsoft Word ou encore Adobe.

Les propriétés importantes du contrôle *MenuStrip* sont affichées ci-dessous :

Propriétés	Description
AllowItemReoder	Permet d'indiquer si les *Item* qui peuvent être réorganisés par l'utilisateur. Si sa valeur est *True* l'utilisateur peut déplacer les *Item* en maintenant la touche *Alt*.
AllowMerge	Indique si le *MenuStrip* peut être fusionné avec un autre *MenuStrip*.
Dock	Détermine à quelles bordures sera associé le contrôle ainsi que le redimensionnement avec le contrôle parent (ex : la *Form*).
LayoutStyle	Définit selon des valeurs comment les éléments du contrôle *MenuStrip* seront disposés.
RenderMode	Indique le style de peinture du contrôle appliqué au *MenuStrip* (System pour un style basique du système, Professional pour une apparence ressemblant à l'Office de Microsoft, et *ManagerRenderMode* pour un style de base de Visual Studio).
ShowItemToolTips	Définit si les *ToolStripItems* du contrôle *MenuStrip* ont des info-bulles.
Strech	Définit une valeur si le *MenuStrip* s'étend d'un bout à l'autre de son container (*Form*, *Panel*…)
TextDirection	Indique la direction du texte dans les contrôles accueillant des *MenuStrip*.

Bien sûr nous pouvons remarquer une grande similitude entre les propriétés du contrôle *MenuStrip* et du contrôle *ToolStrip* car le *MenuStrip* dérive du *ToolStrip*.

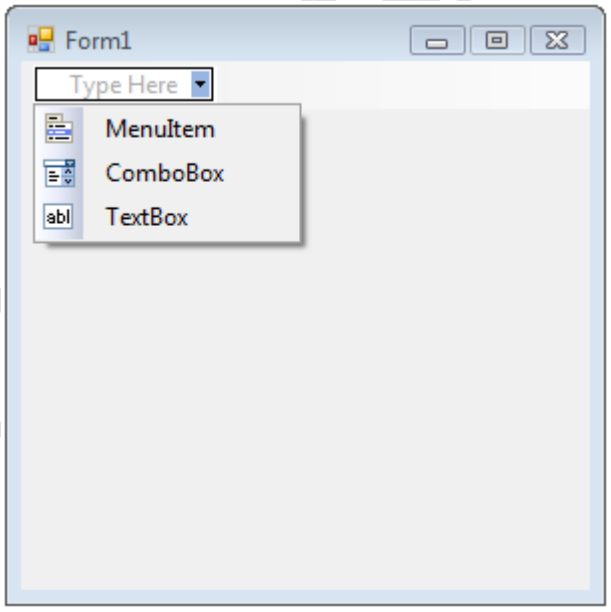
Ensuite nous avons *ToolStripMenusItems* qui fonctionne avec le contrôle *MenuStrip* et voici ses principales propriétés :

Propriétés	Description
AutoSize	Détermine si le menu d'élément redimensionne automatiquement le texte.
Checked	Définit si le *ToolStripMenuItems* apparaît comme étant activé ("*checked*").
CheckOnClick	Détermine une valeur pour indiquer si le *ToolStripMenuItems* doit apparaître automatiquement activé ou désactivé lorsque l'utilisateur clique dessus.
CheckState	Définit dans quel état se trouve initialement le *ToolStripMenuItems* (activé, désactivé ou indéterminé).
DisplayStyle	Indique si le(s) texte(s) et/ou le(s) image(s) sont affiché(e)(s) dans le *ToolStripMenuItems*.
DoubleClickEnabled	Indique si le *ToolStripMenuItems* peut être activé grâce au double clic de la souris.
DropDownItems	Obtient la collection d'élément dans *ToolStripDropDown* auquel la propriété *DropDown* fait référence.
Enable	Détermine si le contrôle est activé.
Image	Indique l'image définit dans le *ToolStripItem.*
MergeAction	Détermine les actions prises lorsque les menus enfants sont fusionnés avec les menus parents.
MergeIndex	Définit la position des éléments après que les menus ont fusionné.
ShortcuKeyDisplayString	Indique le texte associé à la touche de raccourci.
ShortcutKeys	Détermine la touche de raccourci liée au *ToolStripMenuItem*.
ShowShortcutKeys	Détermine si les touches raccourcis associée aux *ToolStripMenuItems* sont affichées à coté de ces

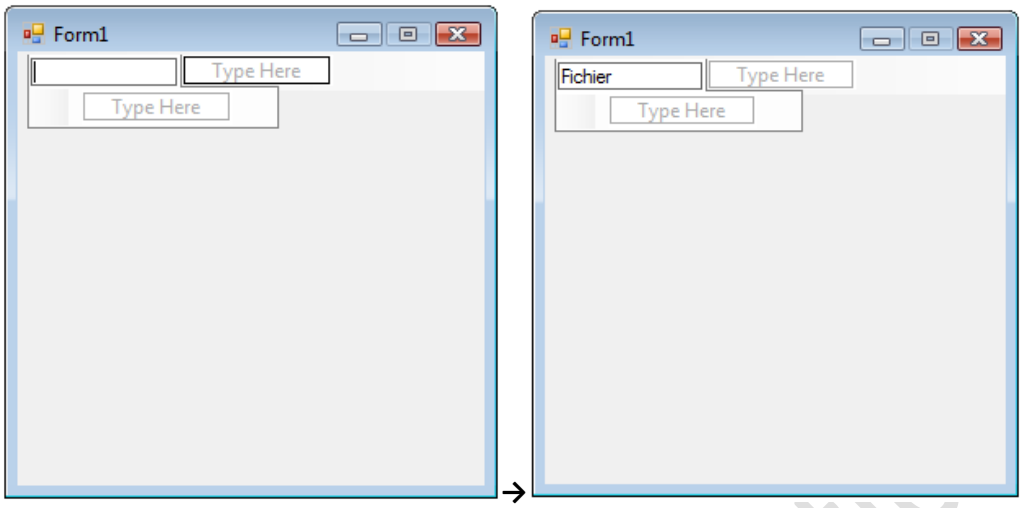
	même *ToolStripMenuItem*.
Text	Indique le texte qui sera affiché dans l'élément.
TextImageRelation	Définit la position du texte et de l'image du *ToolStripMenuItem* par rapport à l'un et l'autre, lorsque la propriété *DisplayStyle* est définie sur *ImageAndText*.

3.2 Concevoir et gérer des *MenuStrips* et des *ToolStripMenuItem*

- La création du contrôle *MenuStrip* se fait de la même façon que tous les autres contrôles : dans la partie design vous faites glisser votre contrôle dans votre *Formulaire* depuis la *Toolbox*. Ensuite vous pouvez ajouter des éléments en cliquant sur votre contrôle puis en renommant votre élément et/ou en choisissant son type (*MenuItem*, *ComboBox*…). Si vous renommez votre élément sans choisir votre type d'élément, il sera alors par défaut un *MenuItem*.



Ici nous choisissons le type d'élément que nous voulons insérer dans le contrôle.



Ici nous renommons directement l'élément en double-cliquant sur son emplacement, nous obtiendrons ainsi par défaut un MenuItem.

Remarque : Par défaut le nom des éléments ajouté sera du type nom_de_votre_élémentToolStripMenuItem ou encore ToolStripComboBox1.

Vous ajoutez tout d'abord votre *MenuStrip* (ici le nom de référence du contrôle est MenuStrip1) puis vous insérez vos éléments comme dans le code suivant :

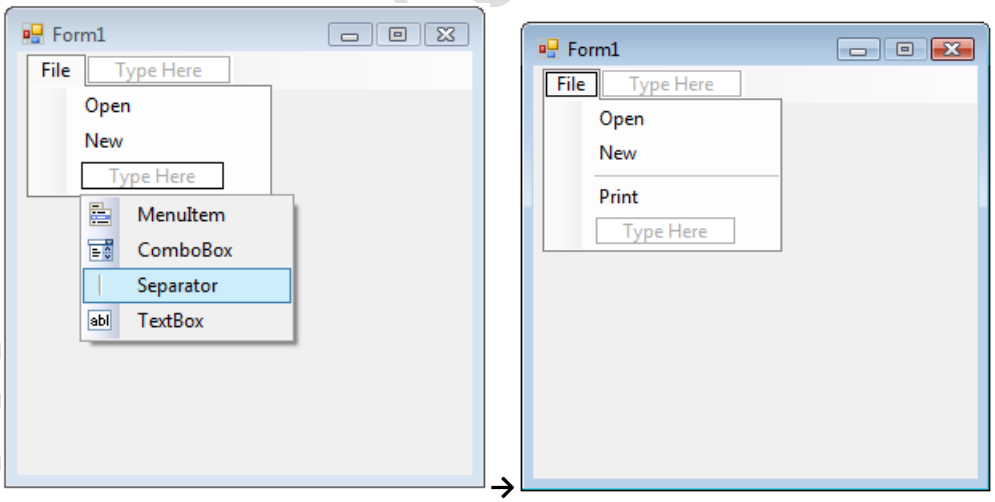
```
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

'Nous créons un nouvel élément MenuItem appelé "Fichier"

Dim FichierToolStripMenuItem As New ToolStripMenuItem("Fichier")

'puis nous l'ajoutons dans le contrôle MenuStrip

MenuStripTest.Items.Add(FichierToolStripMenuItem)

End Sub
```

```
//C#

private void Form1_Load(object sender, EventArgs e)

{

// Nous créons un nouvel élément MenuItem appelé "Fichier"

ToolStripMenuItem FichierToolStripMenuItem = new
ToolStripMenuItem("Fichier");

//puis nous l'ajoutons dans le contrôle MenuStrip

menuStripTest.Items.Add(FichierToolStripMenuItem);

}
```

- Ensuite vous pouvez améliorer votre *MenuStrip* depuis ses propriétés, par l'interface de l'utilisateur ou encore depuis sa partie code. Ci-dessous vous retrouverez différents exemple d'améliorations de votre contrôle :

Le séparateur

Interface utilisateur :



Code :

```vb
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

'nous créons un nouveau séparateur

Dim Separateur As New ToolStripSeparator

'puis nous l'insérons dans le contrôle après le premier élément de celui-
ci

MenuStripTest.Items.Insert(2, aSeparator)

End Sub
```

```csharp
//C#

private void Form1_Load(object sender, EventArgs e)

{

// nous créons un nouveau séparateur

ToolStripSeparator Separateur = new ToolStripSeparator();

//puis nous l'insérons dans le contrôle après le premier élément de celui-
ci

menuStripTest.Items.Insert(2, aSeparator);

}
```

Désactiver, cacher ou supprimer des éléments du menu :

Il est parfois utile de désactiver certaines options de votre menu. Pour cela il vous faut définir la méthode *Enabled* sur *False*. L'élément sera visible par l'utilisateur mais ne pourra être invoqué par le clic ou par les touches de raccourcis.

Vous pouvez aussi cacher les éléments de votre menu en définissant la propriété *Visible* sur *False*. Celui-ci sera invisible à l'utilisateur mais cependant il peut encore l'invoquer grâce aux touches de raccourcis. Ainsi il vous faudra désactiver l'élément si vous voulez qu'il ne puisse être invoqué.

Enfin si vous voulez supprimer un élément définitivement après plusieurs manipulations de l'utilisateur, il vous sera nécessaire d'utiliser les méthodes *MenuStrip.Items.Remove* ou *MenuStrip.Items.RemoveAt* pour un menu, ou bien les méthodes *ToolStripMenuItem.DropDown.Remove* ou *ToolStripMenuItem.DropDown.RemoveAt* pour un sous-menu. Vous trouverez ci-dessous un exemple comportant les différentes méthodes de suppression (menu et sous-menu):

```vb
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load


'nous supprimons dans le code suivant l'élément Fichier du MenuStrip

MenuStripTest.Items.Remove(FichierToolStripMenuItem)

'nous supprimons ici un élément du menu Editer (le sous-menu Couper)

EditerToolStripMenuItem.DropDownItems.Remove(CouperToolStripMenuItem)

'nous supprimons le premier élément du MenuStrip

MenuStripTest.Items.RemoveAt(0)

'nous supprimons l'élément Couper du menu Editer qui se trouve en première
position (index 1)

EditerToolStripMenuItem.DropDownItems.RemoveAt(0)

End Sub
```

```csharp
//C#

private void Form1_Load(object sender, EventArgs e)

{

//nous supprimons dans le code suivant l'élément Fichier du MenuStrip

menuStripTest.Items.Remove(FichierToolStripMenuItem) ;

//nous supprimons ici un élément du menu Editer (le sous-menu Couper)

EditerToolStripMenuItem.DropDownItems.Remove(CouperToolStripMenuItem) ;

//nous supprimons le premier élément du MenuStrip (index 0)

menuStripTest.Items.RemoveAt(0) ;

//nous supprimons l'élément Couper du menu Editer qui se trouve en
première position (index 0)

EditerToolStripMenuItem.DropDownItems.RemoveAt(0);

}
```

Déplacer vos éléments

➢ Vous pouvez déplacer vos éléments de votre contrôle lors de l'exécution en utilisant la méthode *menuStrip.Items.Insert* avec laquelle vous aurez la possibilité de choisir la position ou la méthode menuStrip.Items.Add avec laquelle vous ajoutez seulement l'élément donc celui-ci se placera en dernière position. Cela vous permettra de personnaliser dynamiquement vos menus. Par exemple vous pouvez déplacer un sous-élément d'un menu pour en faire un menu (le sous-élément sera supprimé automatiquement dans le menu où il se trouvait) ou changer l'ordre de vos menus :

```vb
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

'nous ajoutons l'élément Editer dans le contrôle

MenuStripTest.Items.Add(EditerToolStripMenuItem)

'nous insérons l'élément Editer en indiquant sa position(index 3)

MenuStripTest.Items.Insert(3, EditerToolStripMenuItem)

End Sub
```

```csharp
//C#

private void Form1_Load(object sender, EventArgs e)

{

//nous ajoutons l'élément Editer dans le contrôle
menuStripTest.Items.Add(editerToolStripMenuItem);

//nous insérons l'élément Editer en indiquant sa position(index 3)
menuStripTest.Items.Insert(3, editerToolStripMenuItem);

}
```

➢ Vous pouvez également utiliser la propriété ToolStripMenuItem.DropdownItems pour déplacer un menu dans un autre ou encore un sous-élément d'un menu dans un autre menu. Vous utiliserez les méthodes Add ou Insert de la même façon que l'exemple précédent :

```vb
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

'nous ajoutons l'élément Couper du menu Fichier que nous avions créé dans
le menu Editer

EditerToolStripMenuItem.DropDownItems.Add(CouperToolStripMenuItem)

'nous insérons l'élément Couper, en indiquant sa position(index 0),du menu
Fichier que nous avions créé dans le menu Editer

EditerToolStripMenuItem.DropDownItems.Insert(0, CouperToolStripMenuItem)

End Sub
```

```csharp
//C#

private void Form1_Load(object sender, EventArgs e)

{

//nous ajoutons l'élément Couper du menu Fichier que nous avions créé dans
le menu Editer
editerToolStripMenuItem.DropDownItems.Add(couperToolStripMenuItem);

//nous insérons l'élément Couper, en indiquant sa position(index 0),du
menu Fichier que nous avions créé dans le menu Editer
editerToolStripMenuItem.DropDownItems.Insert(0, couperToolStripMenuItem);

}
```

Le raccourci clavier

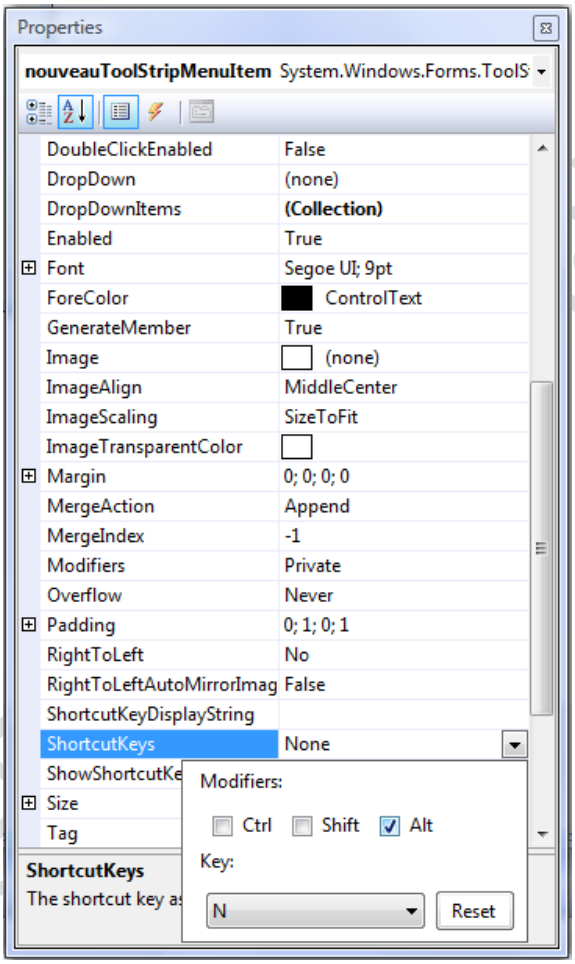
Vous pouvez aussi créer des raccourcis pour vos contrôles. Pour cela vous pouvez vous référer aux précédents chapitres (exemple : Chapitre 2 le contrôle *Label*).

Les touches de raccourcis

A la différence du raccourci clavier, les touches de raccourcis permettent d'exécuter une commande et ne peuvent être utilisés que dans les sous-menus. Par exemple dans le menu Fichier de

votre programme vous pourrez définir les touches de raccourcis Ctrl+N pour accéder au sous-menu Nouveau et ainsi créer un nouveau fichier.

Vous pouvez créer les touches de raccourcis lors de la conception en sélectionnant la propriété *ShortcutKeys* dans les propriétés du *ToolStripMenuItem* de votre choix. Ensuite si vous voulez afficher vos touches de raccourcis dans le menu vous attribuerez alors la valeur *True* dans la propriété *ShownShortcutKeys* ou bien grâce à la propriété *ShortcutKeyDisplayString* vous pourrez afficher un texte à la place de la combinaison des touches que vous avez défini à votre élément.



Nous définissons ici les touches de raccourcis Alt+N à notre élément "Nouveau" dans notre menu "Fichier".

Fusionner les menus :

Les menus peuvent être fusionnés au moment de l'exécution et les éléments de ces menus sont intégrés dans un menu unique. Vous pouvez fusionner des *MenuStrip*, des *ContextMenuStrip* (nous verrons ce contrôle en détails plus loin dans la leçon), ou bien les deux. Pour cela vous

utiliserez la méthode *ToolStripManager.Merge* en indiquant le menu source, le menu de destination, et l'action à entreprendre par la propriété MergeAction. Voici les différentes options que propose cette propriété :

Propriétés	Description
Append	Ajoute l'élément à la fin de la collection
Insert	Insert l'élément à l'emplacement spécifié par la propriété *MergeIndex*.
MatchOnly	Permet de créer une arborescence et accéder correctement aux éléments imbriqués
Remove	Supprime l'élément spécifié.
Replace	Remplace l'élément identifié par l'élément source.

3.3 Le contrôle *ContextMenuStrip*

Le contrôle *ContextMenuStrip* est un menu contextuel qui s'affiche lorsque l'utilisateur réalise un clic-droit sur un contrôle (*TextBox*, Bouton…) ou sur une zone du *Formulaire*. Ce contrôle est similaire au *MenuStrip*, cependant la principale différence entre ces deux contrôles est le fait que le *ContextMenuStrip* ne possède pas de menu de haut niveau c'est-à-dire qu'il ne détient pas une arborescence comme dans les *MenuStrip* (ex : un haut niveau menu appelé Fichier contenant des sous-menus).

- Ajouter ou supprimer un *ContextMenuStrip* :

Vous utiliserez tout simplement les méthodes ContextMenuStrip.Items.Add pour ajouter et ContextMenuStrip.Items.Remove pour supprimer, comme dans l'exemple de code suivant :

```vb
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

'nous ajoutons l'élément Annuler dans le contrôle ContextMenuStripTest

ContextMenuStripTest.Items.Add(AnnulerToolStripMenuItem)

'nous supprimons cet élément du contrôle ContextMenuStripTest

ContextMenuStripTest.Items.Remove(AnnulerToolStripMenuItem)

End Sub
```

```csharp
//C#

private void Form1_Load(object sender, EventArgs e)

{

\\nous ajoutons l'élément Annuler dans le contrôle ContextMenuStripTest

contextMenuStripTest.Items.Add(annulerToolStripMenuItem) ;

\\nous supprimons cet élément du contrôle ContextMenuStripTest

contextMenuStripTest.Items.Remove(annulerToolStripMenuItem) ;

}
```

- Associer un ContextMenuStrip à un contrôle ou à une Form:

Pour associer un *ContexMenuStrip* à un contrôle ou à votre *Form*, il vous faut simplement sélectionner votre *ContextMenuStrip* dans la propriété *ContextMenuStrip* du contrôle ou de la Form auquel vous voulez l'associer. Vous avez alors la possibilité soit de le faire lors de la conception de votre programme soit lors de l'exécution comme ci-dessous :

```vb
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

'nous ajoutons quelques élément dans le ContextMenuStripTest

ContextMenuStripTest.Items.Add(AffichageToolStripMenuItem)

ContextMenuStripTest.Items.Add(NouveauToolStripMenuItem)

'puis nous associons ce ContextMenuStrip au contrôle TextBox

TextBox1.ContextMenuStrip = ContextMenuStripTest

End Sub
```

```csharp
//C#

private void Form1_Load(object sender, EventArgs e)

{

'nous ajoutons l'élément Annuler dans le contrôle ContextMenuStripTest

contextMenuStripTest.Items.Add(affichageToolStripMenuItem);

contextMenuStripTest.Items.Add(nouveauToolStripMenuItem);

'puis nous associons ce ContextMenuStrip au contrôle TextBox

textBox1.ContextMenuStrip = ContextMenuStripTest;

}
```

➢ Copie des éléments d'un menu à partir d'un *Menustrip* au moment de l'exécution :

Le plus souvent vous voulez créer des menus contextuels qui contiennent les mêmes éléments qu'un menu contextuel ordinaire. Pour cela il vous suffit d'appeler le constructeur *ToolStripMenuitem* qui vous permettra de préciser le texte, l'image et l'événement clic d'un gestionnaire d'événement. Ainsi vous ajoutez le *MenuStrip* choisi à votre contrôle *ContextMenuStrip*. Vous trouverez un exemple ci-dessous manipulant ces différentes méthodes :

```
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

'nous appelons le constructeur ToolStripMenuItem puis ses méthodes

Dim  anItem As ToolStripMenuItem

anItem = New ToolStripMenuItem(EditerToolStripMenuItem.Text,
_EditerToolStripMenuItem.Image, New EventHandler(addressof
_EditerrToolStripMenuItem_Click))

'puis nous copions dans le contrôle ContextMenuStrip

ContextMenuStripTest.Items.Add(anItem)

End Sub
```

```
//C#

private void Form1_Load(object sender, EventArgs e)

{

//nous appelons le constructeur ToolStripMenuItem puis ses méthodes

ToolStripMenuItem anItem

anItem = new ToolStripMenuItem(editerToolStripMenuItem.Text,
_editerToolStripMenuItem.Image, new
EventHandler(editerToolStripMenuItem_Click))

//puis nous copions dans le contrôle ContextMenuStrip

contextMenuStripTest.Items.Add(anItem)

}
```

4 Gérer les évènements et utiliser le *Event Handlers*

Un évènement est une sorte de message qui permet de signaler que quelque chose se passe dans votre application. Lorsque qu'un évènement est déclenché, cela offre la possibilité de réagir avec ces évènements grâce aux méthodes *Event Handlers* (gestionnaire d'évènement).

4.1 Les évènements en général

Les évènements appartiennent au contrôle ou à la classe qui les contiennent. En effet, lorsque que quelque chose se passe, un contrôle ou une classe peut appeler un évènement comme par exemple lorsque qu'un utilisateur clique sur un bouton, cela lance un *Click* qui par la suite exécute les méthodes qu'il contient. Les évènements sont générés grâce à la souris, le clavier, ou bien d'autres actions du système (exemple : envoie d'un mail lors d'une nouvelle particulière à laquelle vous êtes abonné.

Remarque : Une méthode peut gérer plusieurs évènements et un évènement peut être gérer par plusieurs méthodes !

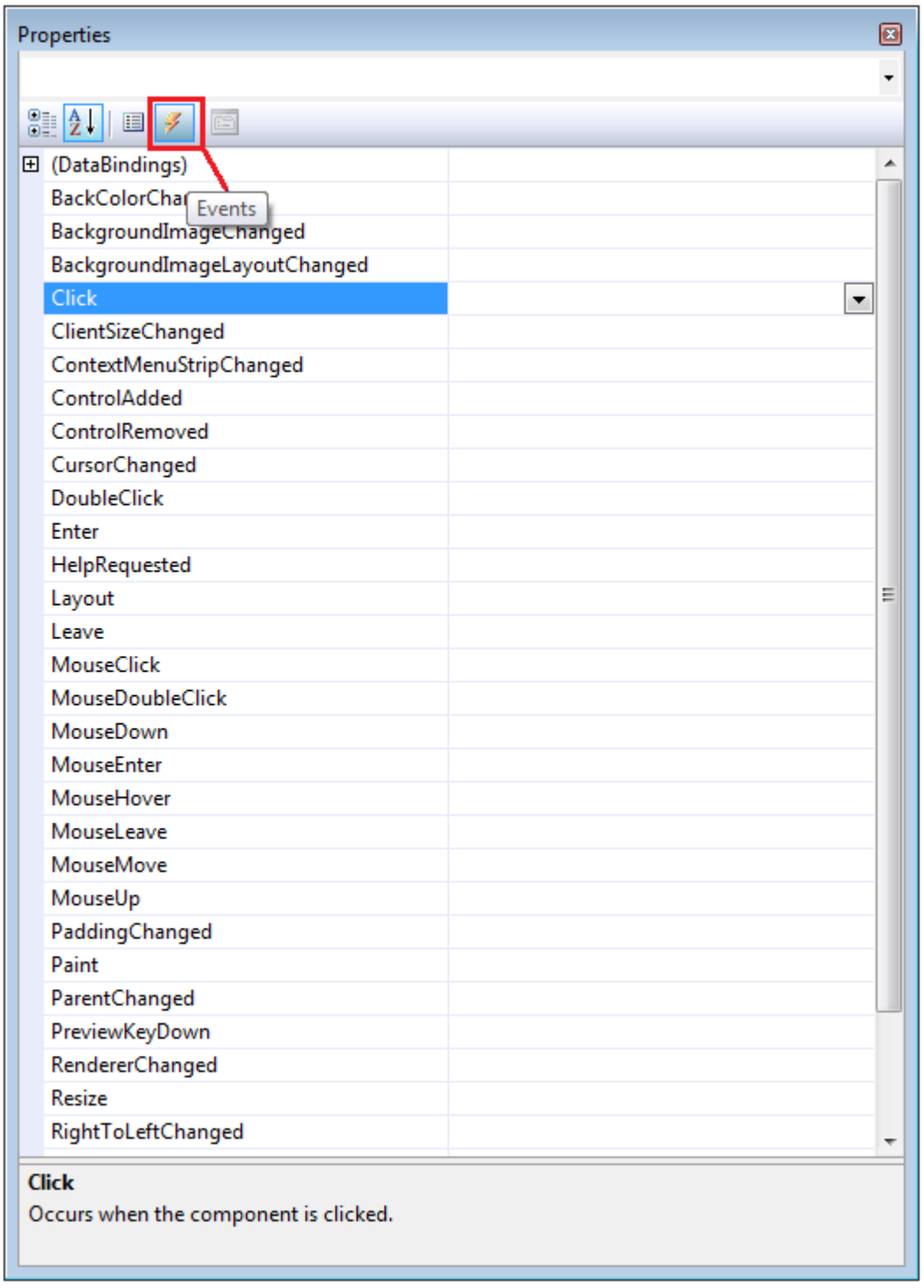
Voici les évènements les plus communs :

Types d'évènements	Description
Click	Généré lors du clic gauche de la souris.
DoubleClick	Généré lors d'un double clic rapide du clic gauche de la souris.
KeyDown	Généré lors le la pression d'une touche quand le control a le focus. Il contient des informations différentes du *KeyPress*.
KeyPress	Généré lors le la pression d'une touche quand le contrôle a le focus. Il contient des informations différentes du *KeyDown*.
KeyUp	Généré lors du relâchement d'une touche quand le contrôle a le focus.
MouseClick	Généré lorsque qu'une touche quelconque de la souris est pressée.
MouseDoubleClick	Généré lorsque qu'une touche quelconque de la souris est pressée deux fois rapidement.
MouseDown	Généré lorsque le curseur de la souris est sur le

	contrôle et qu'un bouton de la souris est pressé.
MouseEnter	Généré lorsque le curseur de la souris se trouve dans le contrôle.
MouseHover	Généré lorsque le curseur se positionne sur le contrôle.
MouseLeave	Généré lorsque le curseur s'écarte du contrôle.
MouseMove	Généré lorsque le curseur de la souris est placé sur le contrôle.
MouseUp	Généré lorsque le curseur de la souris se trouve sur le contrôle et que le bouton de la souris est relâché.
MouseWheel	Généré lorsque la roulette de la souris bouge alors qu'un contrôle a le focus.

4.2 Créer un gestionnaire d'évènements (*Event Handlers*) à la conception

Vous avez accès au gestionnaire des évènements d'un contrôle en allant sur la propriété de celui-ci et en cliquant sur l'icône (éclair encadré en rouge ci-dessous).



4.3 Créer un évènement pour un contrôle

Il suffit d'aller dans le gestionnaire d'évènement de votre contrôle puis de double-cliquer sur l'évènement que vous souhaitez. Cela vous mènera directement dans le code où vous pourrez associer le contrôle et votre évènement aux tâches que vous souhaitez que votre contrôle fasse.

Vous pouvez associer plusieurs évènements à un même contrôle ainsi que plusieurs contrôles au même évènement.

4.4 Gérer les évènements au clavier et à la souris

La majorité des événements sont déclenchés par la souris ou le clavier :

- **Evénement déclenché par le clavier :**

Nous avons vu précédemment les trois évènements principaux du clavier : le *KeyDown*, le *KeyPress* et le *KeyUp*.

Les informations permettant de connaître les activités des ces « *Key* » se trouvent dans le *KeyEventArgs*. En effet de base, les informations les plus communes se trouvent dans l'*EventArgs ;* dans le cas du clavier, d'autres informations sont susceptibles de servir, par exemple savoir quelle touche a été pressée ! Par conséquent le *KeyEventArgs* remplace l'*EventArgs* tout en héritant de ses informations.

Voici les propriétés du *KeyEventArgs* :

Types de Propriétés	Description
Alt	Récupère la valeur indiquée quand la touche *Alt* est pressée.
Control	Récupère la valeur indiquée quand la touche *Ctrl* est pressée.
Handled	Récupère ou définit une valeur indiquant si l'évènement a été géré.
KeyCode	Retourne le code de la touche pressée.
KeyData	Retourne les données de la touche pressée.
KeyValue	Retourne une représentation de la propriété *KeyData*.
Modifiers	Récupère les indicateurs de touches de modification d'un évènement. Les indicateurs indiquent la combinaison de touches *Ctlr*, *Alt* et *Maj* activées.
Shift	Récupère la valeur indiquée quand la touche *Shift* est pressée.
SuppressKeyPress	Récupère ou définit une valeur qui indique si l'évènement d'une touche doit être transmit au contrôle sous-jacent.

- **Evènement déclenché par la souris :**

Les deux actions les plus communes avec la souris sont le clique et le double-clique (*Click* et *DoubleClick*). Par rapport au clavier les informations permettant de connaître leurs activités se trouvent dans le simple *EvenArgs*.

Voici par exemple l'association d'un *Button* avec un évènement *Click* :

```
'VB
private Sub button1_Click(ByVal sender As System. Object, e As
System.EventArgs e)



            //Vous écrivez ici l'action qui va se dérouler au moment de
l'évènement.
```

```
//C#
private void button1_Click(object sender, EventArgs e)
        {
            //Vous écrivez ici l'action qui va se dérouler au moment de
l'évènement.
        }
```

En revanche les autres propriétés (vu précédemment au 4.1) « Mouse » ont un *MouseEventArgs* à la place de *EventArgs*. Le *MouseEventArgs* fonctionne comme le *KeyEventArgs* avec des informations différentes et ayant un rapport avec la souris. En effet, il est important de pouvoir connaître la position du curseur, les pressions, etc.…

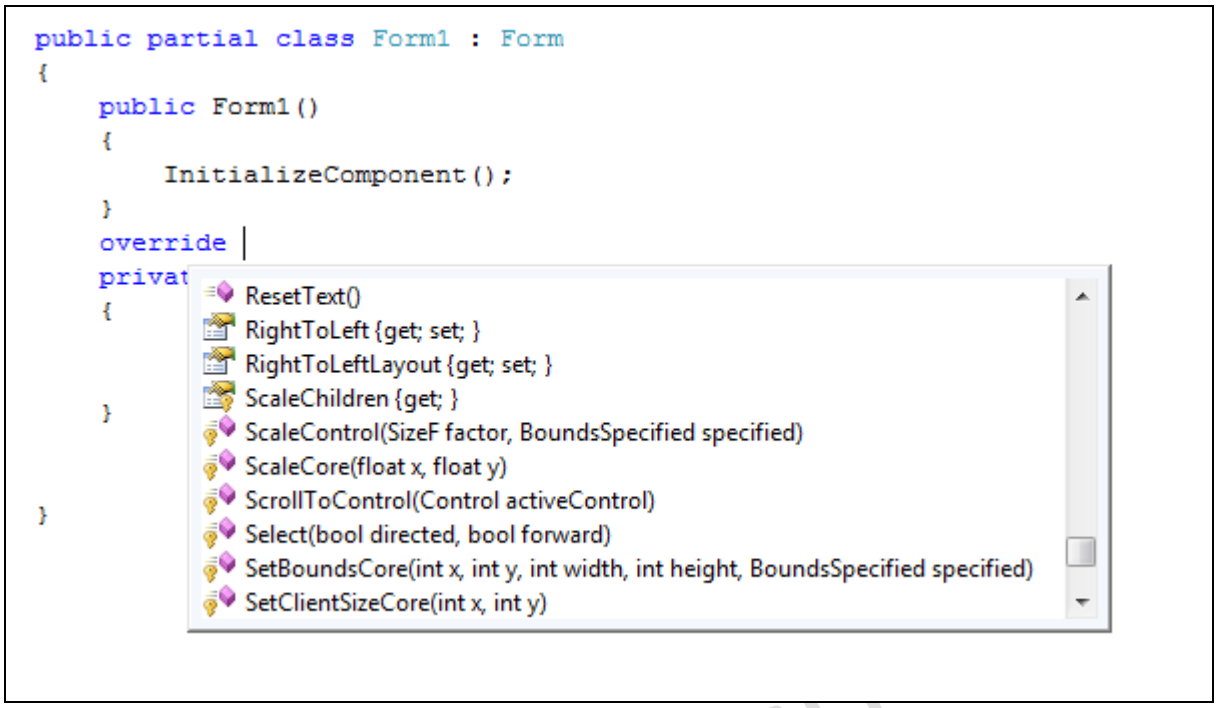
Voici les propriétés du *MouseEventArgs* :

Types de Propriétés	Description
Button	Indique quel bouton a été pressé.
Clicks	Indique combien de fois un bouton a été pressé.
Delta	Indique le nombre de fois où la molette a été touchée.
Location	Indique la position actuelle de la souris.
X	Indique la coordonnée X de la position de la souris.
Y	Indique la coordonnée Y de la position de la souris.

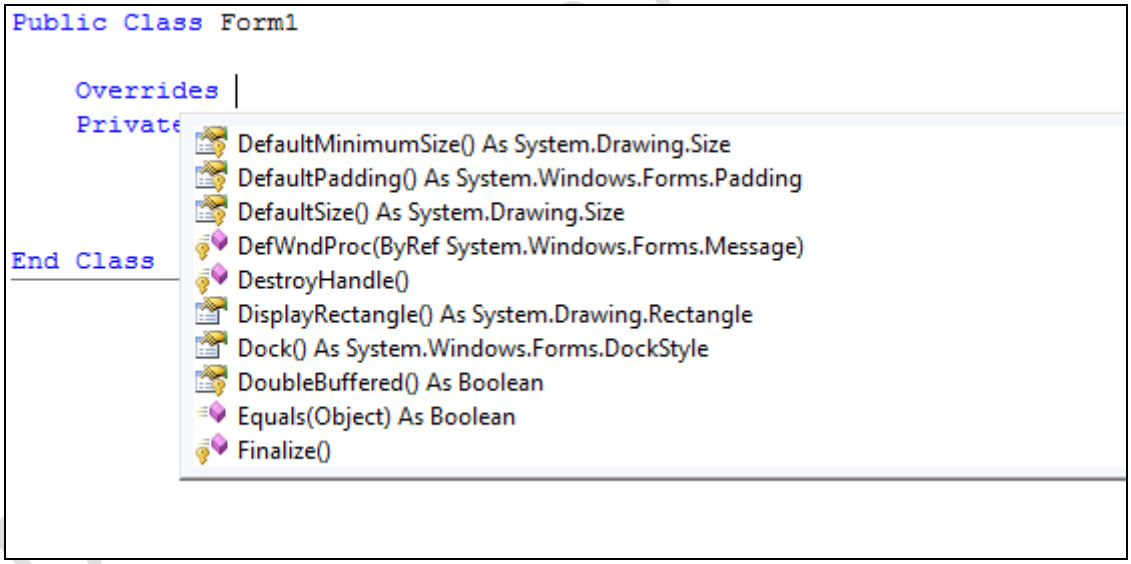
4.5 Charger des méthodes avec le *Code Editor*

Le modificateur *override* en C# et *Override* en Visual Basic est nécessaire pour étendre ou modifier l'implémentation abstraite ou virtuelle d'une méthode, d'une propriété, d'un indexeur ou d'un événement hérité(e).

Le Code Editor vous permet de charger ces méthodes facilement dans votre classe de base. Vous devez donc dans votre éditeur, à l'intérieur de votre classe corps mais en dehors d'une méthode, taper *override* en C# ou *Override* en Visual Basic. Un menu déroulant s'affiche et vous permet de choisir la méthode que vous voudrez grâce à la saisie automatique.



En C#



En Visual Basic

5 Conclusion

Vous avez terminé ce 4ème chapitre de *Windows Form*. Comme d'habitude, nous vous conseillons de pratiquer ce que vous venez de voir et également de vous aider du MSDN pour avoir un support complet des éléments Winforms.

L'équipe *Windows Form.*