Assistant Visual Studio

réalisé par :

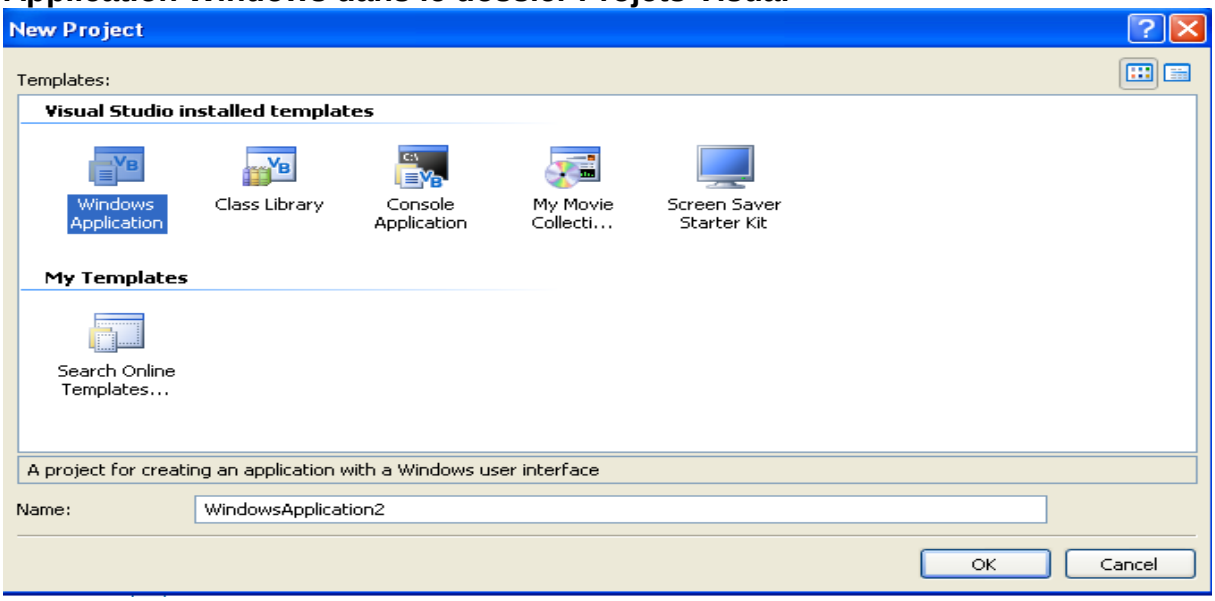
Chamlal Noureddine
Douama Sanae
Khnifira Mohammed
Norri Sanae

Encadré par :

Mr Azzi Hamid

Introduction

Une fois Visual Studio lancé, créez un nouveau projet :
Fichier >> Nouveau >> Projet… ou CTRL + MAJ + N
La fenêtre vous permettant de créer de nouveaux projets va alors s'ouvrir. Sélectionnez Application Windows dans le dossier Projets Visual

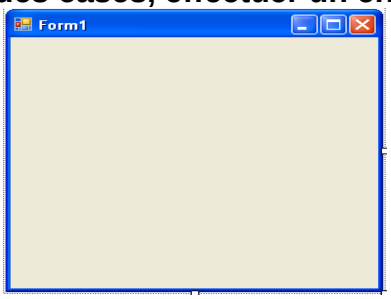


Donnez un nom à votre projet, puis choisissez l'emplacement où sera créé votre projet. Par défaut, celui-ci est créé dans le répertoire Visual Studio Projects qui est créé lors de l'installation de Visual Studio .Net. Vous devriez trouver ce répertoire dans Mes Documents.

formulaire

1-Définition du formulaire

En VB, un formulaire est un espace de saisie dans l'interface utilisateur, pouvant comporter plusieurs zones, ou « champs » : on peut y saisir du texte, cocher des cases, effectuer un choix dans une liste de termes prédéfinis, appuyer sur des boutons, etc.



la barre d'outils

Les Boutons

Modifier ses propriétés:

On peut modifier les propriétés dans la fenêtre des propriétés en bas à droite:

On peut aussi modifier les propriétés par du code.

Name est utilisé pour lui donner un nom explicite (BoutonOk BoutonCancel)

FlatStyle donne un aspect au bouton (Flat, standard, System, pop Up)

Standard et System utilisent le thème d'affichage de Windows que vous avez choisi dans le panneau de configuration. (Thème Windows XP, personnel..)

Quand on utilise Flat on peut choisir dans FlatStyle l'épaisseur du bord et sa couleur et rouge dans notre premier bouton).

Enfin on peut choisir la position du texte avec TextAlign. Il a la valeur TopLeft dans le dernier bouton.

*** Exemple:**

button1.Text="Ok" affiche 'Ok' dans le bouton.

Si on y inclut un « & » la lettre qui suit sera soulignée et servira de raccourci clavier.

Button.Text= "&Ok" donne sur le bouton Ok et crée le raccourci clavier 'Ctrl O' qui est l'équivalent d'un click sur le bouton.

TextAlign permet de positionner le texte dans le bouton.

Image contient le nom de l'image à afficher sur le bouton (si on veut afficher une image, on le fait en mode Design; noter que quand on distribue l'application, il n'y a pas besoin de fournir le fichier contenant l'image avec l'application). (AlignImage permet de positionner l'image sur le bouton.)

On peut aussi puiser une image dans une ImageList grâce à la propriété ImageList et ImageIndex, on peut ainsi changer d'image.

La propriété BackGroundImage permet de mettre une image de fond.

Font contient la police de caractère, sa taille, son enrichissement (gras, italique..)

Truc: quand vous travaillez sur de très petits boutons, changer la propriété Font et choisir une petite taille de caractère (8 par exemple)

Utiliser les évènements:

Les événements vont vous permettre d'effectuer des tâches lors d'actions précises de l'utilisateur. I sont donc à la base de l'interaction entre l'utilisateur et l'application.

Par exemple, lorsque vous cliquez sur un bouton pour afficher une boite de dialogue, vous

déclenchez l'évènement Click du control Button qui contiendra le code pour afficher la boite de dialogue.

Le mécanisme de gestions d'évènements repose sur deux composants :

• Un gestionnaire d'évènement : Méthode qui sera appelée lorsque l'évènement sera déclenché. Elle devra donc contenir le code implémentant la tâche à effectuer en réponse à l'événement.

• Un délégué : Cet objet permet de faire correspondre un gestionnaire d'évènement à l'événement qu vous voudrez exploiter

L'évènement principalement utilisé est Click() : quand l'utilisateur clique sur le bouton la procédure

```
Private Sub Button_Click(..)
End Sub
```

est traitée.

Cette procédure contient le code qui doit être exécuté lorsque l'utilisateur clique sur le bouton.

Le bouton peut être sélectionné grâce à un

clic de souris, à la touche ENTRÉE ou à la BARRE d'espacement si le bouton a le focus.

Créer un bouton Ok ou Cancel:

Parfois il faut permettre aux utilisateurs de sélectionner un bouton en appuyant sur la touche ENTRÉE même si le bouton n'a pas le focus.

Exemple: Il y a sur la fenêtre un bouton "Ok" qui doit être enfoncé quand l'utilisateur tape 'Entrée' au clavier, c'est le bouton qui 'valide' le questionnaire (et qui le ferme souvent).

Comment faire?

Définissez la propriété AcceptButton de la Form en lui donnant le nom du bouton.

Cela permet au formulaire d'avoir le comportement d'une boîte de dialogue.
La propriété CancelButton de la Form permet de la même manière de créer un bouton 'Annuler'(qui répond à la touche 'Echap'(ESC).

Création des raccourcis à partir du clavier :

Au lieu de cliquer par la souris sur le bouton, on peut mettre des raccourcis à partir du clavier.

Pour le faire, on a besoin d'écrire ce code dans la méthode KeyPress

Utilisation avancée: Création d'un bouton par code:

L'exemple suivant crée un Button nommé Button1 sur lequel on voit "Ok", on modifie certaines de ses propriétés et on l'ajoute à Form.

```
Private Sub InitializeMonButton()
Dim button1 As New Button
button1.Text="Ok" ' Ajouter le bouton à la Form
Controls.Add(button1)
```

End Sub

Il faut par code créer aussi les évènements liés à ce bouton: dans ce cas il faut déclarer le bouton plutôt avec la syntaxe contenant WithEvents et en haut du module. Private WithEvents Button1 As New Button

(dans ce cas on ne remet pas la ligne Dim button1 dans la Sub InitializeMonButton)

Puis écrire la sub évènement. Sub OnClique (sender As Objet, EvArg As EventArgs) Handles Button1

End Sub

Ainsi VB sait que pour un évènement sur le Button1 , il faut déclencher la Sub OnClique. (On reviendra sur cela)

Pour deplacer un button pendant l'execution

```
Me.Top = Me.Top – 20
Me.Left = Me.Left – 20
Me.Top = Me.Top +20
Me.Left = Me.Left+ 20
```

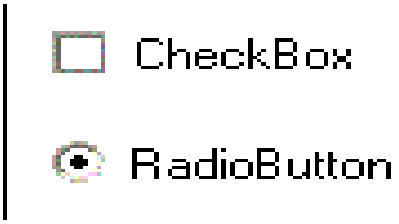
Les Cases à cocher

Il y a 2 sortes de case à cocher :

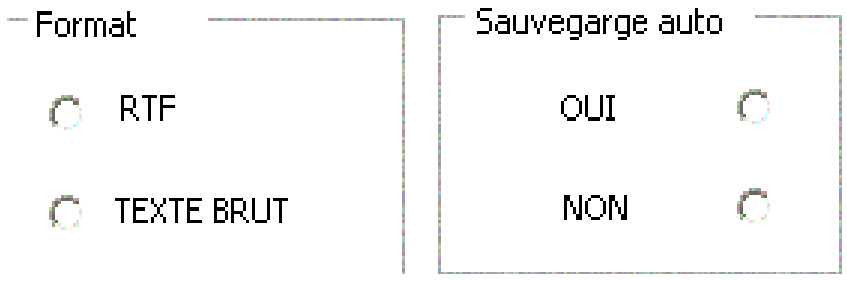
- **Les CheckBox**
- **Les RadioButton**

• Les " cases à cocher " (CheckBox) : Elles sont carrées, et indépendantes les unes des autres, si l'utilisateur coche une case , cela n'a pas d'influence sur les autres cases du formulaire, qu'elles soient regroupées dans un cadre pour faire plus joli ou non.

• Les " boutons radio " (RadioButton) : Ils sont ronds et font toujours partie d'un groupe (Ils sont dans une fenêtre ou dessinés dans un objet GroupBox). Ce groupe est indispensable, car au sein d'un groupe de RadioButton, un seul bouton peut être coché à la fois : si l'utilisateur en coche un, les autres se décochent.



Il faut regrouper les radios boutons dans des 'GroupBox' par exemple pour rendre les groupes indépendants :



Ici si je clique sur le bouton 'OUI' à droite, cela décoche 'NON' mais n'a pas d'influence sur le cadre Format.

La propriété Text, bien sur, permet d'afficher le libellé à coté du bouton, on peut aussi mettre une image avec la propriété Image. CheckAlign permet de mettre la case à cocher à droite ou à gauche du texte, TextAlign permet d'aligner le texte.

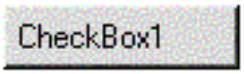
Exemple pour le bouton en haut à droite :

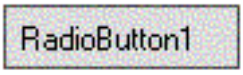
```
RadioButton3.Text = "OUI"
RadioButton3.TextAlign = MiddleCenter 'Middle=hauteur, center = horizontale
RadioButton3.CheckAlign = MiddleRight
```

La propriété la plus intéressante de ces cases est celle qui nous permet de savoir si elle est cochée ou non. Cette propriété s'appelle Checked. Sa valeur change de False à True si la case est cochée.

```
RadioButton.Checked = True 'Coche le bouton
If RadioButton.Checked = True Then ' Teste si le bouton est coché.
End If
```

La procédure RadioButton.CheckedChange() permet d'intercepter le changement d'état d'un bouton. Pour le CheckButton ThreeState permet de définir 3 états au lieu de 2 (coché, indéterminé = grisé, non coché) CheckedState indique un des 3 états (alors que Checked n'en indique que deux.) Appearance peut aussi donner une apparence de bouton à la case à cocher. Il est enfoncé ou pas en fonction de la valeur de Checked.



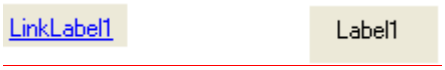


Ici les 2 boutons ont une Appearance = Button, celui du haut n'est pas coché, l'autre est coché (enfoncé).

Les Label

Il y a 2 sortes de Label:

- **Les 'Label'**
- **Les 'LinkLabel'**



Définitiondu label

Les contrôles Label sont généralement utilisés pour fournir un texte descriptif à un contrôle. Vous pouvez par exemple utiliser un contrôle Label pour ajouter un texte descriptif à un contrôle TextBox. Ceci a pour but d'informer l'utilisateur du type de donnée attendu dans le contrôle.

Les propriétés du label

Après avoir déposé le 'Label' sur le formulaire, on peut modifier le texte affiché à partir de la fenêtre de propriétés, en passant par la propriété 'Text':

ForeColor le couleur du text

Pour modifier le texte du label1 par du code:

```
Label1.Text="Donner votre Prénom"
```

Pour modifier le colore de texte du label1 par du code:

```
Label1.Text="Label avec une bordure et un fond jaune"
Label1.BorderStyle=BorderStyle.FixedSingle
Label1.ForeColor=Color.Red
Label2.BackColor=Color.Yellow
```

Il est également possible d'y afficher une image avec la propriété .Image La propriété AutoSize autorise le label, si elle est égale à True, a se redimensionner pour afficher la totalité du texte.

La mise à jour de l'affichage:

La mise à jour de l'affichage du Label autres contrôles d'ailleurs) est effectuée en fin de Sub: Si on écrit:

```
Dim i As Integer

For i = 0 To 100

   Label1.Text = i.ToString

Next i
```

La variable i prend les valeurs 1 à 100, mais à l'affichage rien ne se passe pendant la boucle, VB affiche uniquement 100 à la fin; si on désire voir les chiffres défiler avec affichage de 0 puis 1 puis 2 Il faut rafraîchir l'affichage à chaque boucle avec la méthode Refresh ():

```
Dim i as Integer
For i = 0 To 100
   Label1.Text = i.ToString: Label1.Refresh()
Next i
```

Une alternative est de mettre un Application.DoEvents() qui donne à Windows le temps de traite les messages et de rafraîchir l'affichage.

Les LinkLabel

Définition

Permettent de créer un lien sur un label Text Indique le texte qui apparaît.

LinkArea défini la zone de texte qui agira comme un lien; dans la fenêtre de propriété taper 11 ;4 (on verra que c'est plus simple que de le faire par code)

Les 4 caractères à partir du 11ème seront le lien, ils seront soulignés

L'événement LinkClicked est déclenché quand l'utilisateur clique sur le lien. Dans cette procédure o peut permettre le saut vers un site Internet par exemple ou toute autre action.

Exemple :

```
LinkLabel1.text= "Visitez le site LDF"
LinkLabel1.LinkArea = New System.Windows.Forms.LinkArea(11, 4)
```

'Pourquoi faire simple !!

Si l'utilisateur clique sur le mot 'site', la procédure suivante est déclenchée :

```
Private Sub LinkLabel1.LinkClicked…
End Sub
```

Il est possible de modifier la couleur du lien pour indiquer qu'il a été utilisé:

Si VisitedLinkColor contient une couleur `e.visited=True modifie la couleur.`

(e est l'élément qui a envoyé l'évènement, j'en modifie la propriété Visited.)

On peut y inclure une action quelconque, en particulier un saut vers un site Web:

```
System.diagnostics.Process.Start(" http://google.com/ ")
```

'correspond au code qui ouvre un browser Internet (Internet Explorer ou Netscape) et qui charge la page dont l'adresse est indiquée.

La collection Links permet d'afficher plusieurs liens dans un même texte, mais cela devient vite très compliqué.