

# Imprimer avec VB.NET (2005)

J-M Rabilloud

[www.developpez.com](http://www.developpez.com). Publication sur un autre site Web interdite sans l'autorisation de l'auteur.

## Remerciements

J'adresse ici tous mes remerciements à l'équipe de rédaction de "developpez.com" et tout particulièrement à Cécile Muno et à Emilie Guittier pour le temps qu'elles ont bien voulu passer à la correction et à l'amélioration de cet article.

<b>IMPRIMER AVEC VB.NET (2005).....</b>	<b>1</b>
<b>INTRODUCTION .....</b>	<b>3</b>
<b>CONCEPTS GENERAUX.....</b>	<b>3</b>
<b>L'imprimante .....</b>	<b>3</b>
Imprimantes accessibles .....	3
Imprimante par défaut .....	3
Communication .....	3
Aperçu avant impression .....	4
<b>Gestionnaire d'impressions.....</b>	<b>4</b>
L'unité de mesure .....	4
Zone d'impression.....	4
<b>LES CLASSES DU FRAMEWORK 2.0.....</b>	<b>5</b>
<b>PrintDocument.....</b>	<b>5</b>

Propriétés.....	5
Méthodes.....	5
Evènement PrintPage.....	5
<b>PrintDialog.....</b>	<b>6</b>
Propriétés.....	6
<b>PageSetupDialog.....</b>	<b>6</b>
Propriétés.....	6
<b>PrintPreviewDialog.....</b>	<b>7</b>
<b>PrinterSettings.....</b>	<b>7</b>
Propriétés.....	7
Méthodes.....	9
<b>PageSettings.....</b>	<b>10</b>
Propriétés.....	10
<b>PREPARATION DE L'IMPRESSION.....</b>	<b>11</b>
<b>L'imprimante.....</b>	<b>11</b>
Restreindre les possibilités.....	13
<b>La mise en page.....</b>	<b>16</b>
<b>TECHNIQUES D'IMPRESSION.....</b>	<b>19</b>
Impression d'un formulaire.....	19
Fonctionnement général.....	19
<b>Impression d'une grille de données.....</b>	<b>19</b>
<b>Impression d'un texte.....</b>	<b>25</b>
L'approche empirique.....	26
Impression ajustée.....	27
<b>Imprimer des images.....</b>	<b>28</b>
Cas des images existantes.....	28
Cas des images créées.....	34
<b>Imprimer des contrôles particuliers.....</b>	<b>36</b>
ListBox avec élément sélectionné.....	36
Le contrôle ListView.....	38
Le contrôle TreeView.....	44
<b>Impression d'un formulaire.....</b>	<b>50</b>
Exemple 1.....	50
Exemple 2.....	55
<b>COMMUNIQUER AVEC L'IMPRIMANTE.....</b>	<b>59</b>
<b>CONCLUSION.....</b>	<b>63</b>

# Introduction

Dans cet article, nous allons examiner les fonctionnalités d'impression de Visual Basic 2005. Cette approche se divisera en deux parties, la découverte et l'utilisation des composants et des objets du Framework concernant l'impression, et les techniques couramment utilisées pour réaliser un Gestionnaire d'impression.

Généralement sur ce sujet, on distingue deux groupes parmi les développeurs, ceux qui utilisent les générateurs d'états pour imprimer et les malades comme moi qui ne peuvent pas voir un générateur d'états sans avoir immédiatement une poussée d'urticaire. Le choix d'utiliser un générateur d'états est tout à fait respectable, mais pour ma part j'ai toujours préféré créer mes gestionnaires d'impressions. Il est bien certain que l'écriture est un peu fastidieuse, du moins tant qu'on ne s'est pas constitué une bibliothèque de code pour accélérer la programmation, mais on obtient ainsi des impressions parfaitement maîtrisées, rapides et légères au regard des usines à gaz que sont les générateurs d'états.

Nous verrons dans la deuxième partie de ce document qu'il existe des astuces pour se simplifier grandement la tâche.

Bonne lecture.

## Concepts généraux

### *L'imprimante*

Evidemment sans imprimante, la suite de cet article est beaucoup moins intéressante. La plupart des langages ne fournissent pas d'objet pour simuler une imprimante vu la diversité des modèles existant mais plutôt un objet mappant les propriétés couramment implémentées par les imprimantes. Ceci est d'autant plus logique que les ordinateurs sont de plus en plus souvent connectés à de nombreuses imprimantes ayant des capacités différentes.

### Imprimantes accessibles

Sauf cas particulier extrêmement rare, tout gestionnaire d'impressions doit, à un moment ou à un autre, permettre à l'utilisateur de choisir l'imprimante sur laquelle il désire imprimer. Du fait des légères différences qui existent entre les imprimantes, ce choix doit être fait suffisamment tôt pour pouvoir récupérer les capacités de l'imprimante choisie afin d'imprimer correctement. Dans Visual Basic 2005, il existe des composants prenant en charge cette liste, il peut cependant parfois être utile d'en restreindre la portée lorsqu'on désire utiliser des imprimantes ayant certaines capacités (imprimer du A3 par exemple).

### Imprimante par défaut

Lorsqu'il existe plusieurs imprimantes, l'utilisateur en a désigné une comme imprimante par défaut. Si l'on n'opère pas de nouvelle sélection, c'est celle-ci qui est utilisée. Ce principe est partiellement dangereux puisque si l'on récupère les propriétés de l'imprimante trop tôt, on peut mapper l'imprimante par défaut avant le choix de l'utilisateur. Cette erreur, pourtant bien connue, reste une erreur fréquente. N'oubliez donc pas de relire les propriétés de l'imprimante dès lors que l'utilisateur a modifié l'imprimante cible.

### Communication

Les composants existants dans le Framework, ne gèrent pas réellement la communication avec l'imprimante. Quand il est nécessaire de connaître l'état d'une imprimante, il est obligatoire de passer par des appels de plate-forme. Ceux-ci ne sont pas forcément sans risque, puisque toutes les imprimantes n'établissent pas une liaison bidirectionnelle avec l'ordinateur. Nous verrons un exemple de ce type de code à la fin du présent document.

## **Aperçu avant impression**

L'aperçu avant impression, est généralement un choix de l'utilisateur. Il vous revient cependant de décider si vous souhaitez ou non l'implémenter. Le Framework met à votre disposition un composant gérant cet aperçu. Sauf dans le cas d'une impression standardisée, l'aperçu reste une fonctionnalité extrêmement utile. Il demande cependant un codage légèrement différent puisque l'aperçu sert principalement pour affiner quelques réglages. Vous devez évidemment prendre en compte ceux-ci avant d'imprimer.

## ***Gestionnaire d'impressions***

Un gestionnaire d'impressions au sens large du terme contient deux groupes de code : un code de gestion de l'imprimante (choix, mise en page) et un code d'impression.

Le code d'impression se divise en deux grandes familles, les impressions figées de type formulaire et les impressions à paramètres variables c'est-à-dire ayant des éléments inconnus au moment de l'écriture du code (fichier texte, tableaux, etc.).

Le code d'impression est un code graphique, dans le sens où il n'y a aucune différence selon le périphérique visé (écran, imprimante, zone graphique) si ce n'est le système de coordonnées.

J'ai écrit de nombreux gestionnaire d'impressions en Visual Basic. Bien que ce soit parfois fastidieux, ce n'est jamais complexe dès que les deux problèmes principaux, l'unité de mesure et la zone d'impression, sont bien paramétrés.

## **L'unité de mesure**

C'est toujours là qu'on se plante lorsqu'on débute l'écriture d'un gestionnaire d'impressions. Assez rapidement, on mélange millimètre, pouce et pixel avec un bonheur inégal. On se lance alors dans des conversions audacieuses ce qui finit par engendrer un gestionnaire d'impressions long comme l'œuvre de Zola pour imprimer trois cadres et deux textes. Il y a donc une règle qui doit être votre unique base de travail : la propagation du choix de l'unité au gestionnaire tout entier.

Bien que l'immonde Twips ait disparu dans VB.NET, il reste encore bien des occasions de se tromper. Par défaut, les contrôles et les paramètres d'impressions gèrent des centièmes de pouces, alors que les méthodes graphiques invoquées renvoient des pixels. Il est donc impératif d'harmoniser cela. Comme nous allons le voir dans les exemples, VB.NET donne accès à des convertisseurs assez explicites qui rendent le code nettement plus lisible. N'oubliez cependant jamais que chaque objet devant être imprimé doit correspondre au bon système d'unité.

## **Zone d'impression**

C'est le deuxième loup de cette jungle car elle demande une compréhension correcte de ce qu'elle désigne. Cette expression recouvre plusieurs concepts dans lesquels on se mélange généralement les pincesaux. Par ordre de taille nous avons successivement :

La taille de la page

La zone imprimable

La zone d'impression

La taille de la page, tout le monde se représente bien ce que c'est. Pour le format A4 par exemple, c'est 297x210 mm.

La zone imprimable est autrement plus complexe. Elle est inhérente à chaque imprimante et n'est pas forcément centrée, ou si vous préférez, les marges ne sont pas forcément symétriques. L'obtention de cette zone n'est pas évidente.

La zone d'impression, c'est la zone délimitée par les marges par défaut du gestionnaire de mise en page ou par les marges que vous avez fixées le cas échéant.

# Les classes du Framework 2.0

L'espace de nom qui gère l'impression est **System.Drawing.Printing**.

## ***PrintDocument***

Bien qu'assez simple, c'est cet objet qui gère l'impression à proprement parler.

### **Propriétés**

#### ***DefaultPageSettings (PageSettings)***

Renvoie ou définit les paramètres de page pour le PrintDocument. Ces critères s'appliqueront à toutes les pages imprimées par le biais de ce PrintDocument, bien qu'ils soient modifiables dynamiquement.

#### ***DocumentName (String)***

Donne un nom au document à imprimer. Ce nom est repris par les composants liés au PrintDocument.

#### ***OriginAtMargins (Boolean)***

Détermine si le système de coordonnées a pour origine les marges utilisateurs ou le point (0,0) de la zone imprimable.

#### ***PrinterSettings (PrinterSettings)***

Renvoie ou définit les paramètres d'imprimantes pour le PrintDocument. Ces paramètres s'appliqueront à toutes les pages imprimées par le biais de ce PrintDocument.

### **Méthodes**

#### ***Print***

Déclenche le début du travail d'impression. La cible peut être l'imprimante ou l'aperçu.

### **Evènement PrintPage**

C'est le gestionnaire d'évènement de PrintPage qui contiendra le code graphique de l'impression. Nous verrons en détail le fonctionnement plus loin dans le présent document. Les éléments nécessaires se trouvent dans l'argument PrintPageEventArgs.

#### ***Cancel (Boolean)***

Annule la tâche d'impression en cours.

#### ***Graphics (graphics)***

C'est cet objet graphics qu'il faut récupérer pour gérer les travaux d'impression.

#### ***HasMorePages (Boolean)***

Vrai s'il reste encore au moins une page à imprimer, faux sinon.

#### ***MarginBounds (Rectangle)***

Renvoie le rectangle intérieur défini par les marges. C'est la zone imprimable réelle.

#### ***PageBounds (Rectangle)***

Renvoie le rectangle intérieur défini par les limites de la page.

#### ***PageSettings (PageSettings)***

Permet de modifier les paramètres de page pour chaque page imprimée.

## **PrintDialog**

### **Propriétés**

#### **AllowCurrentPage, AllowPrintToFile, AllowSelection, AllowSomePages**

Autorise ou restreint certaines fonctionnalités de l'imprimante, respectivement :

- Imprimer la page courante
- Imprimer dans un fichier
- Imprimer la sélection
- Imprimer certaines pages

#### **Document (PrintDocument)**

Désigne le PrintDocument associé.

#### **PrinterSettings (PrinterSettings)**

Renvoie ou définit les valeurs de la boîte de dialogue.

#### **PrintToFile (Booléen)**

Indique la valeur de la case à cocher "Imprimer dans un fichier".

#### **ShowHelp (Booléen)**

Indique si le bouton d'aide est affiché.

#### **ShowNetwork (Booléen)**

Indique si le bouton "réseau" est affiché.


## **PageSetupDialog**

### **Propriétés**

#### **AllowMargins, AllowOrientation, AllowPrinter, AllowPaper**

Active ou non des parties de la boîte de dialogue, respectivement :

- Les marges
- L'orientation
- L'accès à la sélection des imprimantes
- La sélection du papier.

 Méfiez-vous d'une certaine redondance. Les boîtes de dialogues peuvent s'invoquer différemment selon les habitudes des utilisateurs. Vous devez soit imposer un ordre, en désactivant AllowPrinter par exemple, ou lire les settings au dernier moment.

#### **Document (PrintDocument)**

Désigne le PrintDocument associé

#### **EnableMetric (Booléen)**

**new** Permet d'automatiser la conversion des marges des millimètres vers les centièmes de pouce et réciproquement.

#### **MinMargins (Margins)**

Définit les marges minimales que l'utilisateur peut donner.

### **PageSettings (PageSettings)**

Renvoie ou définit les valeurs de la boîte de dialogue.

### **PrinterSettings (PrinterSettings)**

Renvoie ou définit les valeurs de la boîte de dialogue imprimante accessible par AllowPrinter.

### **ShowHelp (Booléen)**

Indique si le bouton d'aide est affiché

### **ShowNetwork (Booléen)**

Indique si le bouton "réseau" est affiché

## **PrintPreviewDialog**

La boîte de dialogue d'aperçu avant impression est en fait la boîte de dialogue qui encapsule le contrôle d'aperçu. Vous n'avez rien d'autre à faire que d'attribuer votre PrintDocument à la propriété Document de la boîte de dialogue pour vous en servir.

## **PrinterSettings**

N.B : L'appel du constructeur réinitialise toutes les propriétés à leur valeur par défaut.

### **Propriétés**

#### **CanDuplex (Booléen)**

Renvoie vrai si l'imprimante prend en charge l'impression recto verso.

#### **Collate (Booléen)**

Renvoie ou définit l'option copies assemblées.

#### **Copies (Short)**

Renvoie ou définit le nombre de copies à imprimer

#### **DefaultPageSettings (PageSettings)**

Renvoie les paramètres par défaut de la page pour l'imprimante sélectionnée.

#### **Duplex (Duplex)**

Renvoie ou définit le paramètre de l'impression recto verso. Peut prendre les valeurs :

Nom	Description
Default	Configuration recto-verso par défaut de l'imprimante.
Horizontal	Impression recto-verso horizontale.
Simplex	Impression recto.
Vertical	Impression recto-verso verticale.

#### **FromPage, ToPage (Integer)**

Renvoie ou définit le numéro de la première page ou de la dernière page à imprimer

### **InstalledPrinters (StringCollection) Partagée**

Renvoie la collection des noms de toutes les imprimantes installées sur l'ordinateur.

### **IsDefaultPrinter (Booléen)**

Renvoie vrai si la propriété PrinterName désigne l'imprimante par défaut. Attention, dès lors qu'on définit explicitement la propriété PrinterName, IsDefaultPrinter renvoie toujours faux.

### **IsPlotter (Booléen)**

Renvoie vrai si l'imprimante est un traceur.

### **IsValid (Booléen)**

Renvoie une valeur indiquant si la propriété PrinterName désigne une imprimante valide.

### **MaximumCopies (Integer)**

Renvoie le nombre maximum de copies supporté par l'imprimante

### **MinimumPage, MaximumPage (Integer)**

Permet de limiter les choix minimum et maximum de FromPage et ToPage.

### **PaperSizes ( PaperSizeCollection)**

Renvoie la liste des tailles de papier supportées par l'imprimante

### **PaperSources ( PaperSourceCollection)**

Renvoie la liste des bacs de papier disponibles pour l'imprimante sélectionnée.

### **PrinterName (String)**

Renvoie ou définit le nom de l'imprimante sélectionnée.

### **PrinterResolutions (PrinterResolutionCollection)**

Renvoie la liste de toutes les résolutions prises en charge par l'imprimante sélectionnée.

### **PrintFileName (String)**

Définit le nom du fichier d'impression lorsqu'on a sélectionné la case "impression dans un fichier"

### **PrintRange (PrintRange)**

Définit ou renvoie la zone à imprimer. Peut prendre les valeurs suivantes :

Nom	Description
AllPages	Imprime toutes les pages du document.
CurrentPage	Imprime uniquement la page affichée.
Selection	Imprime la sélection de l'utilisateur
SomePages	Imprime les pages définies par FromPage et ToPage

### **PrintToFile (Booléen)**

Renvoie ou définit si l'impression doit avoir lieu vers un fichier

### **SupportsColor (Booléen)**

Renvoie vrai si l'imprimante gère la couleur.



## Méthodes

### CreateMeasurementGraphics

**Public Function** CreateMeasurementGraphics([pageSettings] **As** PageSettings, [honorOriginAtMargins] **As Boolean**) **As Graphics**

Permet d'obtenir un Graphics pour l'imprimante spécifiée sans créer de travail d'impression.

### GetHdevmode

**Public Function** GetHdevmode (PageSettings **As** PageSettings) **As IntPtr**

Exécute un appel de plate-forme pour récupérer un handle vers une structure DEVMODE du périphérique.

Une structure DEVMODE est de la forme :

```
typedef struct _devicemode { // dvmd
    BCHAR dmDeviceName[CCHDEVICENAME];
    WORD dmSpecVersion;
    WORD dmDriverVersion;
    WORD dmSize;
    WORD dmDriverExtra;
    DWORD dmFields;
    short dmOrientation;
    short dmPaperSize;
    short dmPaperLength;
    short dmPaperWidth;
    short dmScale;
    short dmCopies;
    short dmDefaultSource;
    short dmPrintQuality;
    short dmColor;
    short dmDuplex;
    short dmYResolution;
    short dmTTOption;
    short dmCollate;
    BCHAR dmFormName[CCHFORMNAME];
    WORD dmLogPixels;
    DWORD dmBitsPerPel;
    DWORD dmPelsWidth;
    DWORD dmPelsHeight;
    DWORD dmDisplayFlags;
    DWORD dmDisplayFrequency;
#ifdef WINVER >= 0x0400 // Windows 95 only
    DWORD dmICMMethod;
    DWORD dmICMIntent;
    DWORD dmMediaType;
    DWORD dmDitherType;
    DWORD dmReserved1;
    DWORD dmReserved2;
#endif /* WINVER >= 0x0400 */
} DEVMODE;
```

Attention, il s'agit d'un appel de plate-forme, vous devez donc désallouer explicitement en appelant GlobalFree sur le pointeur retourné.

## **GetHdevnames**

### **Public Function GetHdevnames As IntPtr**

Identique à la fonction précédente pour une structure DEVNAMES. Cette structure est de la forme :

```
typedef struct tagDEVNAMES { // dvnm
    WORD wDriverOffset;
    WORD wDeviceOffset;
    WORD wOutputOffset;
    WORD wDefault;
    // driver, device, and port name strings follow wDefault
} DEVNAMES;
```

## **IsDirectPrintingSupported**

### **Public Function IsDirectPrintingSupported(image As Image) As Boolean**

### **Public Function IsDirectPrintingSupported(imageFormat As ImageFormat) As Boolean**

Renvoie vrai si l'image ou le format d'image spécifié peut être imprimé.

## **PageSettings**

### **Propriétés**

#### **Bounds (Rectangle)**

Renvoie la taille de la page en fonction de l'orientation.

#### **Color (Boolean)**

Vrai si la page doit être imprimée en couleur

#### **HardMarginX, HardMarginY (Single)**

Obtient les coordonnées X ou Y des marges physiques de l'imprimante. En centième de pouce.

#### **Landscape (Booléen)**

Renvoie ou définit si l'impression doit être en mode paysage

#### **Margins (Margins)**

Renvoie ou définit les marges de la page (en centième de pouce)

#### **PaperSize ( PaperSize)**

Renvoie ou définit la taille de papier de la page

#### **PaperSource ( PaperSource)**

Renvoie le bac de papier sélectionné pour imprimer la page.

#### **PrintableArea As RectangleF**

Renvoie la zone imprimable de la page.

#### **PrinterResolution (PrinterResolution)**

Renvoie ou définit la résolution de l'imprimante pour la page.

#### **PrinterSettings (PrinterSettings)**

Renvoie le paramétrage de l'imprimante pour la page.

# Préparation de l'impression

## L'imprimante

Dans la majorité des cas, un gestionnaire d'impressions est indépendant de l'imprimante choisie. Pour imprimer, il est cependant nécessaire de récupérer les informations de l'imprimante. Par ailleurs, il est souvent utile de proposer le choix de l'imprimante à l'utilisateur.

Comme nous l'avons vu, VB.NET vous propose deux contrôles standards pour le choix et/ou le réglage de l'imprimante : PrintDialog et PageSetupDialog. Pour ma part, je préfère le contrôle PageSetupDialog qui permet d'accéder aux propriétés de mise en page mais aussi à la définition de l'imprimante.

La manipulation de ces contrôles est assez aisée, mais vous ne devez pas oublier qu'ils demandent une instance d'un objet PrintDocument. Ceci implique que les sélections réalisées dans les boîtes de dialogues ne ciblent que le document en cours et n'affectent pas le paramétrage de l'imprimante pour d'autres documents et/ou applications.

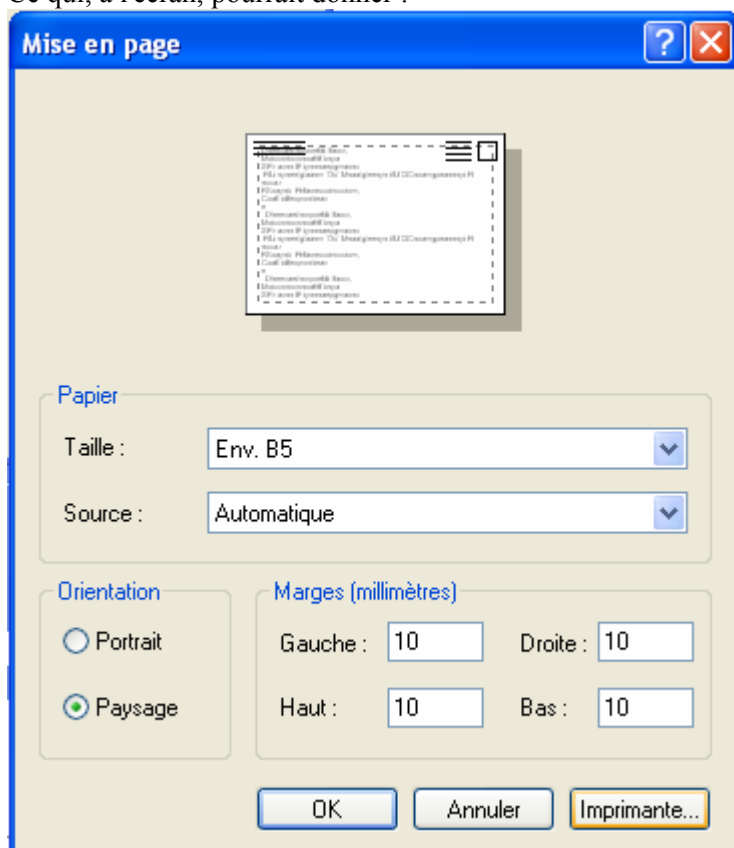
Un exemple simple de récupération d'informations serait par exemple:

```
Imports System.Drawing.Printing
Public Class Form1

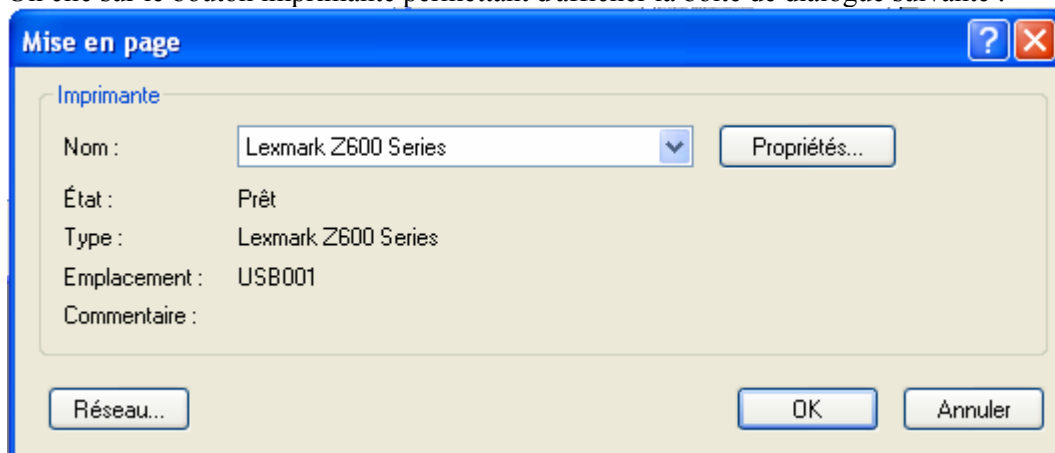
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim Doc As New PrintDocument
        Dim PageSetup As New PageSetupDialog
        With PageSetup
            .Document = Doc
            .ShowDialog(Me)
        End With
        With Me.TextBox1
            .Text = Doc.PrinterSettings.PrinterName & vbCrLf
            .Text = .Text & "Nb de copies : " &
Doc.PrinterSettings.Copies.ToString & vbCrLf
            .Text = .Text & "Assemblées : " &
Doc.PrinterSettings.Collate.ToString & vbCrLf
            .Text = .Text & "Couleur : " &
Doc.DefaultPageSettings.Color.ToString & vbCrLf
            .Text = .Text & "Orientation : " &
IIf(Doc.DefaultPageSettings.Landscape.ToString, "Paysage", "Portrait") & vbCrLf
            .Text = .Text & "Format : " &
Doc.DefaultPageSettings.PaperSize.ToString & vbCrLf
        End With

    End Sub
End Class
```

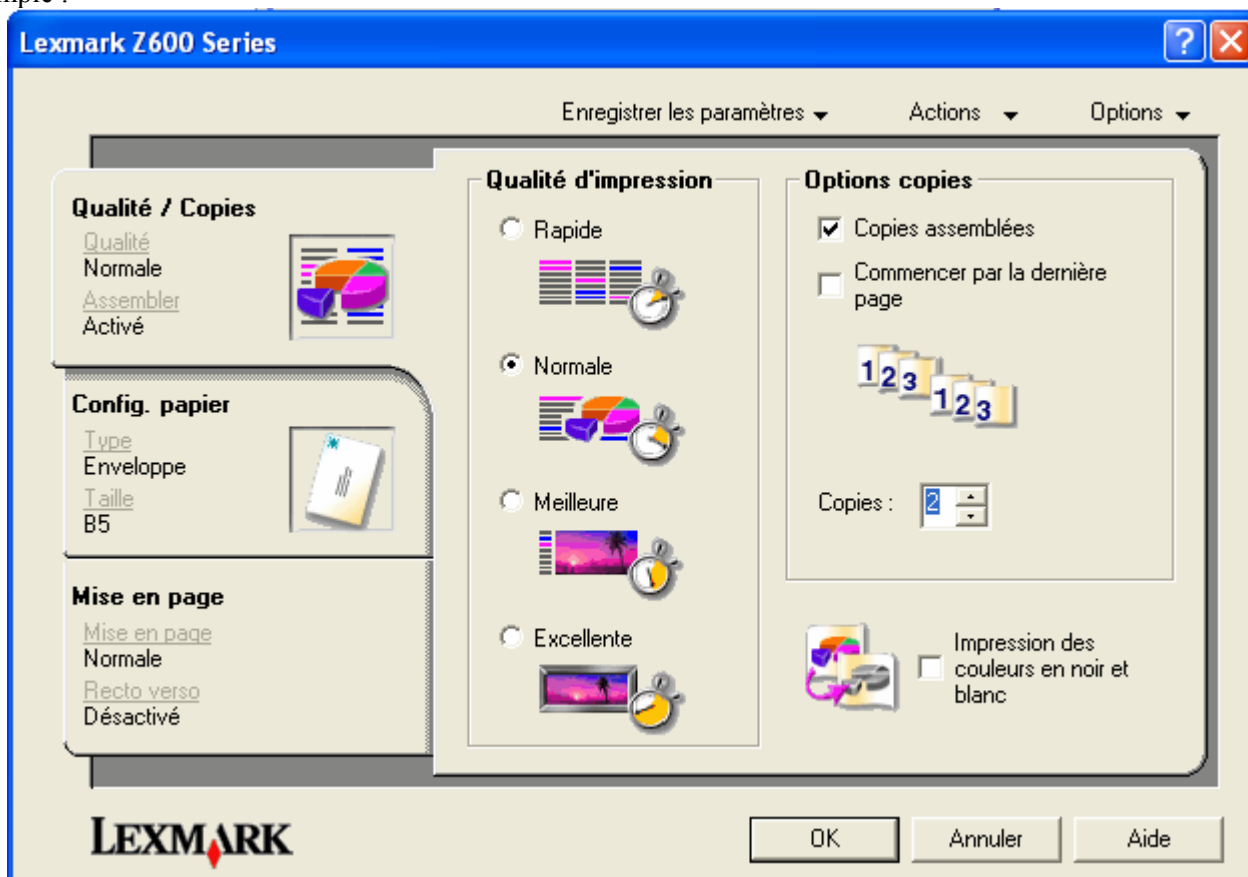
Ce qui, à l'écran, pourrait donner :



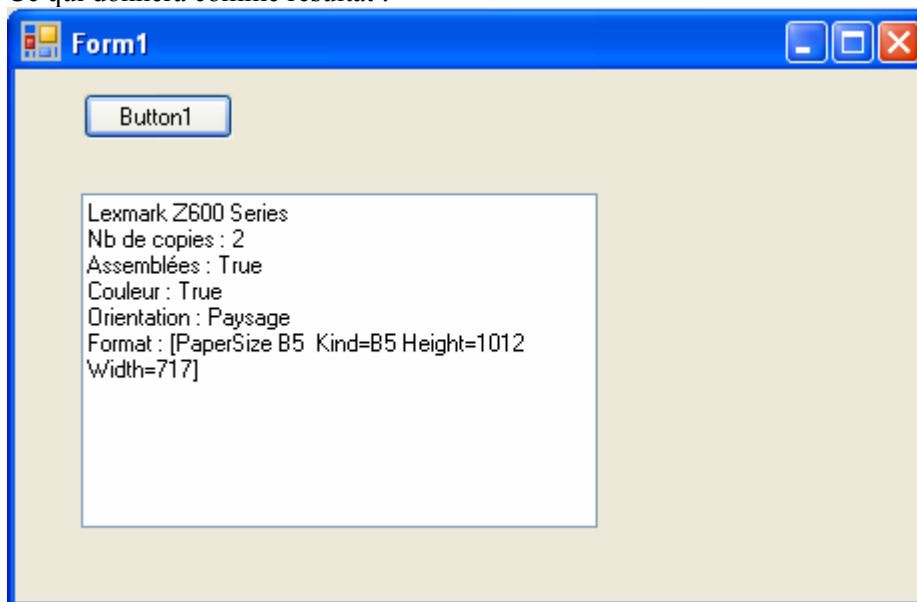
Un clic sur le bouton imprimante permettant d'afficher la boîte de dialogue suivante :



Un clic sur le bouton propriétés permettant d'accéder à la configuration de l'imprimante choisie, par exemple :



Ce qui donnera comme résultat :



Comme je vous l'ai dit précédemment, il peut y avoir une certaine redondance des boîtes de dialogue. C'est pour cela que pour ma part, j'évite d'enchaîner les boîtes de dialogue `PrintDialog` et `PageSetupDialog`. Cependant un tel enchaînement est envisageable.

## Restreindre les possibilités

Par le biais des propriétés `Allow...`, il est possible de restreindre, dans une certaine mesure, les choix de l'utilisateur. Cependant cette possibilité reste très limitée. Si votre application le nécessite, vous devez envisager de créer une boîte de dialogue adaptée. Dans l'exemple suivant, nous allons créer une boîte de dialogue permettant de limiter le choix des imprimantes à celles permettant l'impression recto verso et la couleur.

Nous pourrions donc créer un formulaire nommé imprimante, qui contiendra le code suivant :

```
Imports System.Management

Public Class Imprimante

    Private CapImprimante As Printing.PrinterSettings

    Private Sub Imprimante_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Dim WMIQuery As New ManagementObjectSearcher("Select * from Win32_Printer")
        CapImprimante = New Printing.PrinterSettings
        For Each Printer As ManagementObject In WMIQuery.Get
            CapImprimante.PrinterName = Printer.GetProperty("Name").ToString
            If CapImprimante.CanDuplex AndAlso CapImprimante.SupportsColor Then
                Me.CmbImprimante.Items.Add(CapImprimante.PrinterName)
            End If
        Next
        If Me.CmbImprimante.Items.Count > 0 Then
            Me.CmbImprimante.SelectedIndex = 0
        Else
            MsgBox("Aucune imprimante couleur recto verso détectée", MsgBoxStyle.Critical + MsgBoxStyle.OkOnly)
            Me.DialogResult = Windows.Forms.DialogResult.Cancel
        End If
    End Sub

    Private Sub CmbImprimante_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CmbImprimante.SelectedIndexChanged
        CapImprimante.PrinterName = Me.CmbImprimante.Text
        With CapImprimante
            For Each Taille As Printing.PaperSize In .PaperSizes
                Me.cmbFormat.Items.Add(Taille.PaperName)
            Next
            Me.cmbFormat.SelectedIndex = 0
            Me.NumericUpDown1.Maximum = .MaximumCopies
            For Each Resol As Printing.PrinterResolution In .PrinterResolutions
                Me.cmbResol.Items.Add(Resol.Kind.ToString)
            Next
            Me.cmbResol.SelectedIndex = 0
        End With
    End Sub

    Public ReadOnly Property NomImpr() As String
        Get
            Return Me.CmbImprimante.Text
        End Get
    End Property

    Public ReadOnly Property Resol() As Integer
        Get
            Return Me.cmbResol.SelectedIndex
        End Get
    End Property
End Class
```

```

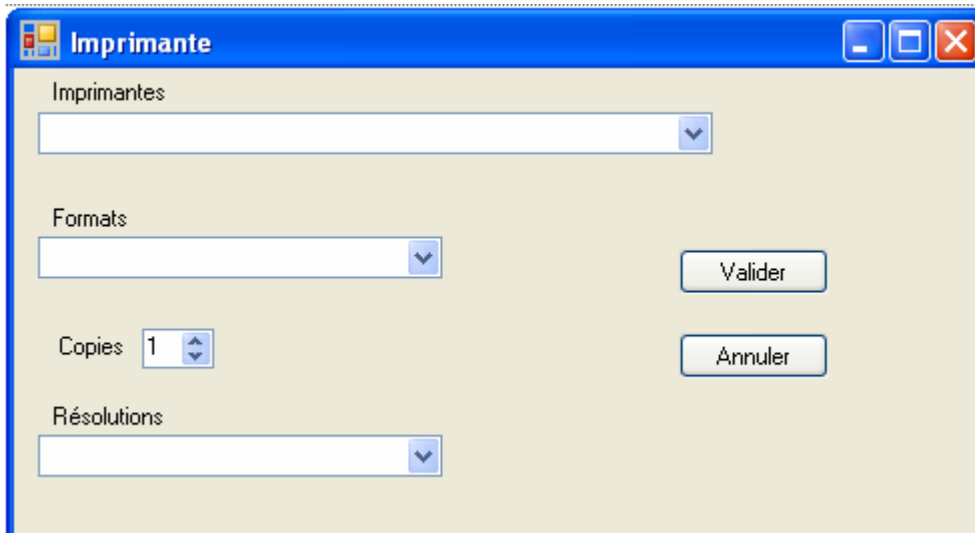
Public ReadOnly Property Taille() As Integer
    Get
        Return Me.cmbFormat.SelectedIndex
    End Get
End Property

Public ReadOnly Property NbCopie() As Integer
    Get
        Return Me.NumericUpDown1.Value
    End Get
End Property
End Class

```

Notez que je récupère les imprimantes par une requête WMI, mais nous pourrions obtenir le même résultat par la propriété InstalledPrinters ou par un appel de plate-forme EnumPrinters.

Notre formulaire ressemblera à cela :



Le code de récupération serait alors :

```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

    Dim DialImpr As New Imprimante
    DialImpr.ShowDialog(Me)
    If DialImpr.DialogResult = Windows.Forms.DialogResult.OK Then
        Dim Config As New PrinterSettings
        With Config
            .PrinterName = DialImpr.NomImpr
            .Copies = DialImpr.NbCopie
            .DefaultPageSettings.PaperSize =
.PaperSizes(DialImpr.Taille)
            .DefaultPageSettings.PrinterResolution =
.PrinterResolutions(DialImpr.Resol)
        End With
        MsgBox(Config.Copies.ToString + vbCrLf +
Config.DefaultPageSettings.PaperSize.ToString)
    End If

End Sub

```

Cet exemple est assez mal programmé pour une raison qui doit vous sauter aux yeux en lisant le code ci-dessus. Sinon cela tient à ce qui découle des PrinterSettings et ce qui est du domaine des PageSettings.

Normalement, on ne définit pas le type de papier ou la résolution au niveau de la sélection de l'imprimante mais au niveau de la configuration de l'impression du document. Vous noterez d'ailleurs que je dois appeler DefaultPageSettings pour valoriser les résultats de la boîte de dialogue. En soit, ce n'est pas une faute grave, mais essayer autant que faire se peut de bien séparer les choix afin de détecter plus simplement les problèmes éventuels.

## La mise en page

Le travail de mise en page est généralement fixé par la nature de ce que vous voulez faire. Dans la plupart des cas, il est restreint par l'application qui définit la taille du papier, l'orientation, les marges, etc....

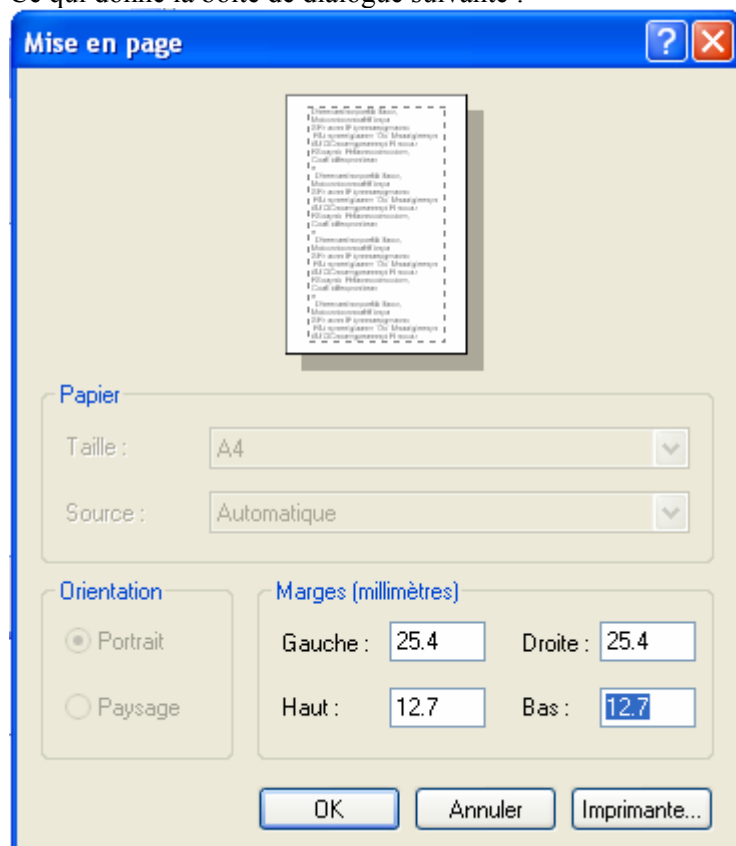
Dans le cadre d'une impression 'libre', le composant PageSetupDialog fait parfaitement l'affaire, dans les autres cas, cela dépend de ce que vous avez besoin de restreindre. Le code suivant par exemple permet à l'utilisateur de spécifier uniquement ses marges en respectant des marges minimum et de sélectionner une imprimante.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

    Dim MiseEnPage As New PageSetupDialog
    Dim MonDoc As New PrintDocument
    With MiseEnPage
        .Document = MonDoc
        .AllowOrientation = False
        .AllowPaper = False
        .EnableMetric = True
        .MinMargins = New Margins(100, 100, 50, 50)
        .ShowDialog(Me)
        .Dispose()
    End With

End Sub
```

Ce qui donne la boîte de dialogue suivante :





Là encore, l'exemple est mal programmé du fait de la redondance des boîtes de dialogue. En effet, certaines imprimantes permettent de choisir la taille du papier par leur boîte de propriétés. Ce qui fait que le blocage au niveau de la boîte de dialogue PageSetupDialog de la taille du papier n'est pas une sécurité en soit. Dès lors que vous autorisez un paramétrage de l'impression par l'utilisateur, vous devez être extrêmement vigilant aux possibilités effectivement offertes pour éviter que votre travail d'impression ne fasse n'importe quoi.

### Appréhender la zone d'impression

Je n'ai pas trouvé dans le Framework une classe permettant de lire les contraintes de l'imprimante. Pour atteindre les paramètres physiques de l'imprimante, j'ai donc appelé une API Windows, GetDeviceCaps, tout comme je le faisais en VB6.

```
Private Declare Function GetDeviceCaps Lib "gdi32" (ByVal hdc As
IntPtr, ByVal nIndex As Integer) As Integer

Private Const HORZRES As Short = 8
Private Const VERTRES As Short = 10
Private Const PHYSICALWIDTH As Short = 110
Private Const PHYSICALHEIGHT As Short = 111
Private Const PHYSICALOFFSETX As Short = 112
Private Const PHYSICALOFFSETY As Short = 113

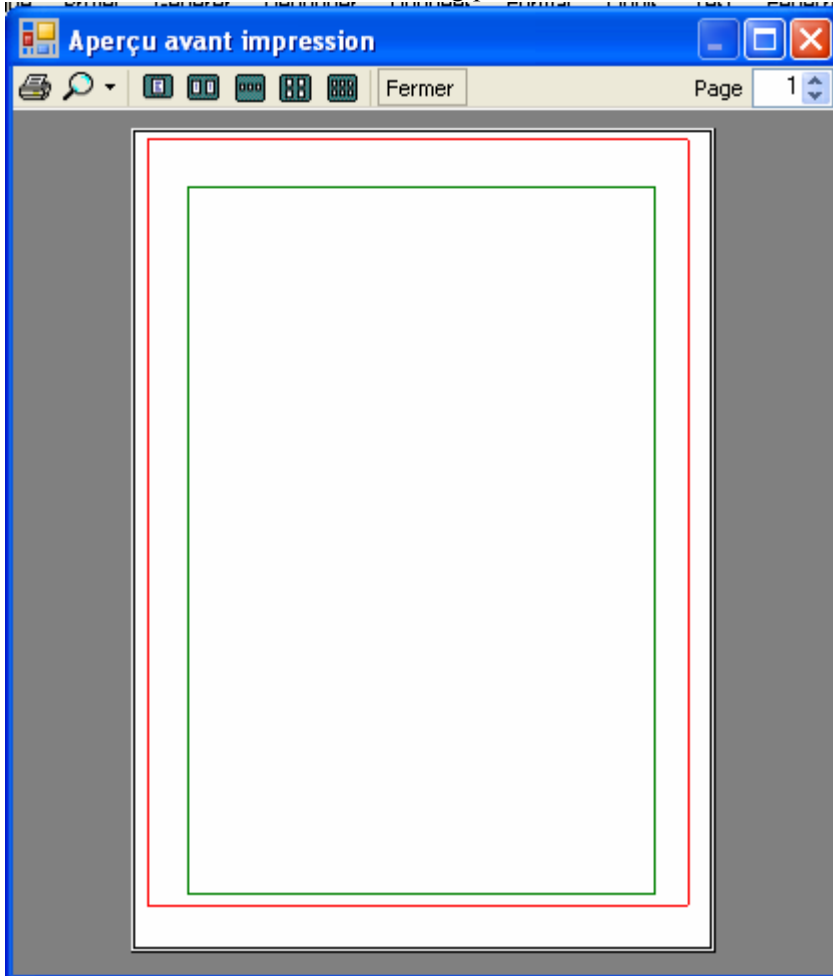
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Me.PageSetupDialog1.ShowDialog()
    Me.PrintPreviewDialog1.ShowDialog()
End Sub

Private Sub PrintDocument1_PrintPage(ByVal sender As Object, ByVal e As
System.Drawing.Printing.PrintPageEventArgs) Handles
PrintDocument1.PrintPage

    Const ConvTomm As Single = 25.4
    Dim HDC As IntPtr = e.Graphics.GetHdc
    Dim MargeGauche, MargeDroite, MargeHaute, MargeBasse As Single
    Dim Hauteur, Largeur, HImprim, LImprim As Single
    MargeGauche = GetDeviceCaps(HDC, PHYSICALOFFSETX)
    MargeHaute = GetDeviceCaps(HDC, PHYSICALOFFSETY)
    LImprim = GetDeviceCaps(HDC, HORZRES)
    HImprim = GetDeviceCaps(HDC, VERTRES)
    Hauteur = GetDeviceCaps(HDC, PHYSICALHEIGHT)
    Largeur = GetDeviceCaps(HDC, PHYSICALWIDTH)
    e.Graphics.ReleaseHdc(HDC)
    MargeGauche = MargeGauche * ConvTomm / e.Graphics.DpiX
    MargeHaute = MargeHaute * ConvTomm / e.Graphics.DpiY
    LImprim = LImprim * ConvTomm / e.Graphics.DpiX
    HImprim = HImprim * ConvTomm / e.Graphics.DpiY
    Largeur = Largeur * ConvTomm / (e.Graphics.DpiX)
    Hauteur = Hauteur * ConvTomm / (e.Graphics.DpiY)
    MargeDroite = Largeur - LImprim - MargeGauche
    MargeBasse = Hauteur - HImprim - MargeHaute
    e.Graphics.DrawRectangle(Pens.Blue, e.PageBounds)
    e.Graphics.DrawRectangle(Pens.Green, e.MarginBounds)
    e.Graphics.PageUnit = GraphicsUnit.Millimeter
    e.Graphics.DrawRectangle(New Pen(Color.Red, 1),
Rectangle.Ceiling(RectangleF.FromLTRB(MargeGauche + 1, MargeHaute + 1,
LImprim - 2, HImprim - 2)))
    e.Graphics.Dispose()
```

End Sub

J'utilise le contrôle PrintPreviewDialog pour visualiser mon résultat.



Je vois trois rectangles imbriqués, l'un représentant les limites de la page, l'autre les limites de la zone imprimable et enfin les limites de la zone d'impression. Notez que si l'affichage est correct, l'impression sur papier ne donnerait pas le même résultat. Cela est dû au fait que le rectangle PageBounds, bien tracé sur le bord de la feuille dans l'aperçu avant impression, sera tracé aux limites de la zone imprimable. Donc, le point (0,0) de la feuille est déterminé par les marges physiques de l'imprimante.

# Techniques d'impression

Nous allons voir maintenant plusieurs techniques d'impression classiques. Une bonne maîtrise de celles-ci permet sensiblement d'imprimer tout ce que l'on veut, même des documents extrêmement complexes.

## Impression d'un formulaire

La méthode PrintForm n'existe plus en VB.NET. Elle est remplaçable par une impression de l'image du formulaire. Le code est d'ailleurs donné dans l'aide au chapitre « Code : impression d'un Windows Form ». Une technique couramment utilisée en VB6 consistait à manipuler le formulaire le temps de l'impression pour ne pas avoir à gérer l'impression. Cette approche est toujours possible mais, comme en VB 6, je ne la recommande pas.

## Fonctionnement général

A la base, vous avez obligatoirement une instance de PrintDocument. Tous vos contrôles de configuration ou d'impression attendront ce PrintDocument pour pouvoir fonctionner puisque l'impression dans VB.NET passe obligatoirement par lui.

Normalement, si votre programme laisse le choix à l'utilisateur, vous devez lui afficher une boîte de dialogue de choix d'imprimante et/ou de paramétrage. Cependant, vous pouvez configurer celle-ci afin d'interdire certaines options. Tout peut évidemment être géré par le code.

Ne perdez pas de vue que les boîtes de dialogues ont un bouton "Annuler" et qu'il convient donc de gérer complètement la réponse de l'utilisateur.

L'impression va alors se déclencher que ce soit pour un aperçu ou pour une impression. Il y a pour cela appel à un objet PrintController. Généralement, on ne l'utilise pas explicitement, mais il peut être utile de le dériver pour des impressions un peu complexes.

Ce PrintController va gérer une séquence d'appels vers les événements de PrintDocument. C'est généralement ceux-ci que l'on intercepte.

Dans la plupart des cas, il suffit d'écrire le code dans l'évènement PrintPage. Cependant, il peut parfois être intéressant d'utiliser BeginPrint pour vérifier la configuration demandée par l'utilisateur ou QueryPageSettings si les feuilles ne doivent pas toutes avoir le même paramétrage.

## ***Impression d'une grille de données***

Dans cet exemple, nous allons imprimer une grille de données. Je vais utiliser le code de remplissage suivant :

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    Dim MaConn As New OleDbConnection("Data
Source=D:\User\BIBLIO.MDB;Provider=Microsoft.Jet.OLEDB.4.0")
    Dim MonAdapt As New OleDbDataAdapter("SELECT * From Publishers",
MaConn)
    Dim MaTable As New DataTable
    MonAdapt.Fill(MaTable)
    Me.DataGrid1.DataSource = MaTable
End Sub
```

Et je veux pouvoir imprimer une grille contenant quelques champs de ma visualisation écran. Sur mon formulaire, j'ajoute donc :

- Un contrôle PrintDocument
- Un contrôle PageSetupDialog
- Un contrôle PrintPreviewDialog

En fait, je ne vais pas travailler en me basant sur le Datagrid mais uniquement sur la Table. Il est parfois mieux de se baser sur la grille si on veut prendre en compte la mise en forme utilisateur, tout dépend de ce que l'on veut obtenir.

N'oubliez pas que les deux boîtes de dialogues doivent avoir le PrintDocument1 défini dans leur propriété Document.

Il existe toujours plusieurs façons d'envisager un gestionnaire d'impression. Par habitude du VB6, j'utilise toujours une structure. Il serait cependant plus adapté maintenant de passer par une classe. Cependant je vais garder mon approche traditionnelle puisqu'elle est parfaitement valide.

Mon code va commencer comme suit :

```
Imports System.Data.OleDb
Imports System.Drawing.Printing
Public Class Form1

    Private Enum Champ
        PubId
        Name
        Company_Name
        Address
        City
        State
        Zip
        Telephone
        Fax
        Comments
    End Enum

    Private Structure Tableau
        Dim NbRow As Int32
        Dim RowPerPage As Int32
        Dim Hauteur As Int32
        Dim Largeur As Int32
        Dim TabLarg() As Int32
    End Structure

    Private TabImpr As New Tableau
    Private NombrePage As Int32 = 0
    Private PageNum As Int32 = 0

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim MaConn As New OleDbConnection("Data
Source=D:\svg\tutoriel\BIBLIO.MDB;Provider=Microsoft.Jet.OLEDB.4.0")
        Dim MonAdapt As New OleDbDataAdapter("SELECT * From Publishers",
MaConn)
        Dim MaTable As New DataTable
        MonAdapt.Fill(MaTable)
        Me.DataGridView1.DataSource = MaTable

    End Sub

    Private Function ConvTo(ByVal ValueMM As Integer) As Integer
        'convertit les millimètres en unités imprimantes (0.01 inch)
        Return CInt(Printing.PrinterUnitConvert.Convert(ValueMM * 10.0R,
Printing.PrinterUnit.TenthsOfAMillimeter, Printing.PrinterUnit.Display))
    End Function
```

La première énumération me permettra d'atteindre la colonne de mon choix en utilisant un index mais avec un code plus lisible. En effet, on peut toujours appeler une colonne par son nom ou par son index. L'appel par le nom est plus lent mais plus explicite, l'énumération me permet d'avoir le meilleur des deux modes.

Ma structure va contenir les éléments dont j'ai besoin pour l'impression, tels que :

- NbrRow → Nombre de lignes de la table
- RowPerPage → Nombre de lignes par page
- Hauteur → Hauteur d'une ligne (1/100 inch)
- Largeur → Largeur de toutes les colonnes réunies
- TabLarg() → Tableau des positions des colonnes (1/100 inch – cumulé)

Je déclare enfin deux variables globales pour gérer le nombre de page et la page en cours

La fonction ConvTo permet juste la conversion d'unité entre les millimètres et les centièmes de pouce.

Je vais donc ensuite coder le bouton qui gère l'enchaînement de l'impression, en l'occurrence les deux boîtes de dialogue. Son code sera :

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

    'paramétrage
    With Me.PrintDocument1.DefaultPageSettings
        .Landscape = True
        .Color = False
        .Margins = New Printing.Margins(ConvTo(10), ConvTo(10),
ConvTo(10), ConvTo(10))
    End With
    'bloque le bouton de sélection d'imprimante et l'orientation
    With Me.PageSetupDialog1
        .AllowMargins = True
        .AllowOrientation = False
        .AllowPaper = True
        .AllowPrinter = False
        If System.Globalization.RegionInfo.CurrentRegion.IsMetric
Then
            .MinMargins = New Printing.Margins(100, 100, 100, 100)
        Else
            .MinMargins = New Printing.Margins(ConvTo(10),
ConvTo(10), ConvTo(10), ConvTo(10))
        End If
    End With
    If Me.PageSetupDialog1.ShowDialog(Me) =
Windows.Forms.DialogResult.Cancel Then
        If MessageBox.Show("Voulez vous abandonnez l'impression",
"Annulation", MessageBoxButtons.OKCancel, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1, MessageBoxOptions.DefaultDesktopOnly) =
Windows.Forms.DialogResult.OK Then
            Exit Sub
        End If
    End If
    ReDim TabImpr.TabLarg(4)
    Dim MaConn As New OleDbConnection("Data
Source=D:\svg\tutoriel\BIBLIO.MDB;Provider=Microsoft.Jet.OLEDB.4.0")
    Dim MaCommand As New OleDbCommand("SELECT Max(Len([Name])) AS
lNom, Max(Len([Address])) AS lAdresse, Max(Len([City])) AS lVille,
Max(Len([State])) AS lEtat FROM Publishers", MaConn)
    MaConn.Open()
    'Récupération des longueurs de chaines
    Dim dtrTaille As OleDbDataReader =
MaCommand.ExecuteReader(CommandBehavior.SingleRow)
    dtrTaille.Read()
    For cmpt As Int32 = 0 To 3
        TabImpr.TabLarg(cmpt) = dtrTaille.GetInt32(cmpt)
    Next
End Sub
```

```

dtrTaille.Close()
'récupération du nombre de lignes
MaCommand.CommandText = "SELECT COUNT(PubID) FROM Publishers"
TabImpr.NbRow = CInt(MaCommand.ExecuteScalar)
MaConn.Close()
Me.PrintPreviewDialog1.ShowDialog()

End Sub

```

La première partie de ce code me sert uniquement à paramétrer le document et la boîte de dialogue PageSetup. Notez que je vérifie les paramètres de CurrentRegion pour les marges car il y a un bug dans la boîte de dialogue si vous êtes en système métrique (et oui, je sais...).

Dans la seconde partie, je vais commencer à remplir ma structure.

En effet, pour gérer correctement mon encadrement je dois connaître la taille maximale des chaînes de chaque champ et le nombre de lignes. Je pourrais récupérer ces informations de ma table mais, dans ce cas, je passe par des commandes. Cela nous permet de voir rapidement l'appel de commandes en VB.NET.

Notez bien que pour l'instant, ma structure TabImpr.TabLarg() stocke des longueurs de chaîne.

Tout le reste va se passer au niveau de l'évènement principal de l'impression PrintPage.

```

Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal
e As System.Drawing.Printing.PrintPageEventArgs) Handles
PrintDocument1.PrintPage
    Dim PHauteur As Single = CSng(e.MarginBounds.Height)
    Dim ImprFont As New Font("Arial", 10)
    Dim ImprBrush As New SolidBrush(Color.Black)
    Dim ImprFormat As New StringFormat
    ImprFormat.Alignment = StringAlignment.Center
    ImprFormat.LineAlignment = StringAlignment.Center
    'récupération de l'objet graphics
    Dim gr As Graphics = e.Graphics
    'définit le point 0,0
    Dim XBase As Int32 = CInt(e.MarginBounds.Left)
    Dim YBase As Int32 = CInt(e.MarginBounds.Top)
    'Fin du remplissage de la structure Tableau
    If NombrePage = 0 Then
        With gr
            'pour chaque colonne, détermine la largeur de la case la plus
grande
            For cmpt As Int32 = 0 To 3
                TabImpr.TabLarg(cmpt) = CInt(.MeasureString(New
String(CType("X", Char), TabImpr.TabLarg(cmpt)), ImprFont).ToPointF.X)
                TabImpr.Largeur = TabImpr.Largeur + TabImpr.TabLarg(cmpt)
                If cmpt > 0 Then TabImpr.TabLarg(cmpt) =
TabImpr.TabLarg(cmpt) + TabImpr.TabLarg(cmpt - 1)
            Next
            TabImpr.Hauteur = CInt(.MeasureString("X", ImprFont).Height)
            TabImpr.RowPerPage = CInt(PHauteur / TabImpr.Hauteur) - 1
            NombrePage = TabImpr.NbRow \ TabImpr.RowPerPage
            If TabImpr.NbRow Mod TabImpr.RowPerPage = 0 Then NombrePage
+= 1
        End With
    End If
    Dim Stylo As New Pen(Color.Black, 1)
    'Impression des titres sous la forme
    '1) definition du rectangle de la case
    '2) drawstring écrit la chaine, imprformat servant à centrer le texte
    '3) drawRectangle dessine la case
    Dim TableCase As New Rectangle(XBase, YBase, TabImpr.TabLarg(0),
TabImpr.Hauteur)

```

```

    gr.DrawString("Editeur", ImprFont, ImprBrush, TableCase, ImprFormat)
    gr.DrawRectangle(Stylo, TableCase)
    TableCase = New Rectangle(XBase + TabImpr.TabLarg(0), YBase,
TabImpr.TabLarg(1) - TabImpr.TabLarg(0), TabImpr.Hauteur)
    gr.DrawString("Adresse", ImprFont, ImprBrush, TableCase, ImprFormat)
    gr.DrawRectangle(Stylo, TableCase)
    TableCase = New Rectangle(XBase + TabImpr.TabLarg(1), YBase,
TabImpr.TabLarg(2) - TabImpr.TabLarg(1), TabImpr.Hauteur)
    gr.DrawString("Ville", ImprFont, ImprBrush, TableCase, ImprFormat)
    gr.DrawRectangle(Stylo, TableCase)
    TableCase = New Rectangle(XBase + TabImpr.TabLarg(2), YBase,
TabImpr.TabLarg(3) - TabImpr.TabLarg(2), TabImpr.Hauteur)
    gr.DrawString("Etat", ImprFont, ImprBrush, TableCase, ImprFormat)
    gr.DrawRectangle(Stylo, TableCase)
    'impression des données sous la forme
    '1) definition du rectangle de la case
    '2) drawstring écrit la chaine, imprformat servant à centrer le texte
    '3) drawRectangle dessine la case
    Dim MaTable As DataTable = CType(Me.DataGridView1.DataSource,
DataTable)
    Dim PosLigne As Int32 = (PageNum * TabImpr.RowPerPage)
    Try
        For cmptRow As Int32 = PosLigne To PosLigne + TabImpr.RowPerPage
- 1
            TableCase = New Rectangle(XBase, YBase + ((cmptRow - PosLigne
+ 1) * TabImpr.Hauteur), TabImpr.TabLarg(0), TabImpr.Hauteur)

gr.DrawString(MaTable.Rows(cmptRow).Item(Champ.Name).ToString, ImprFont,
ImprBrush, TableCase, ImprFormat)
            gr.DrawRectangle(Stylo, TableCase)
            TableCase = New Rectangle(XBase + TabImpr.TabLarg(0), YBase +
((cmptRow - PosLigne + 1) * TabImpr.Hauteur), TabImpr.TabLarg(1) -
TabImpr.TabLarg(0), TabImpr.Hauteur)

gr.DrawString(MaTable.Rows(cmptRow).Item(Champ.Address).ToString,
ImprFont, ImprBrush, TableCase, ImprFormat)
            gr.DrawRectangle(Stylo, TableCase)
            TableCase = New Rectangle(XBase + TabImpr.TabLarg(1), YBase +
((cmptRow - PosLigne + 1) * TabImpr.Hauteur), TabImpr.TabLarg(2) -
TabImpr.TabLarg(1), TabImpr.Hauteur)

gr.DrawString(MaTable.Rows(cmptRow).Item(Champ.City).ToString, ImprFont,
ImprBrush, TableCase, ImprFormat)
            gr.DrawRectangle(Stylo, TableCase)
            TableCase = New Rectangle(XBase + TabImpr.TabLarg(2), YBase +
((cmptRow - PosLigne + 1) * TabImpr.Hauteur), TabImpr.TabLarg(3) -
TabImpr.TabLarg(2), TabImpr.Hauteur)

gr.DrawString(MaTable.Rows(cmptRow).Item(Champ.State).ToString, ImprFont,
ImprBrush, TableCase, ImprFormat)
            gr.DrawRectangle(Stylo, TableCase)
        Next
    Catch ex As Exception

    Finally
        gr.Dispose()
        PageNum += 1
        If PageNum < NombrePage Then e.HasMorePages = True

```

```
End Try
```

```
End Sub
```

Regardons ce code de plus près. Je crée d'abord des objets Font, Brush, StringFormat dont je vais avoir besoin pour mes appels de méthode DrawString et DrawRectangle.

Je finis ensuite de remplir ma structure. Pour l'instant, j'ai des tailles de chaînes, mais il me faut des positions géométriques, en 1/100<sup>ème</sup> de pouce, pour pouvoir imprimer correctement. Pour faire cette conversion, je vais invoquer la méthode MeasureString de l'objet graphique, tel que :

```
TabImpr.TabLarg(cmpt) = CInt(.MeasureString(New String(CType("X", Char),  
CInt(TabImpr.TabLarg(cmpt))), New Font("Arial", 10)).ToPointF.X)
```

Ceci revient à dire :

- crée une chaîne de n caractères "X"
- considère une police Arial 10
- renvoie-moi la largeur en centième de pouce.

Je récupère aussi la hauteur de la même façon.

Vous noterez que je cumule mes positions dans mon tableau TabImpr.TabLarg(). Autrement dit on peut le lire tel que :

TabImpr.TabLarg(0) = Largeur de la première colonne

TabImpr.TabLarg(1) = Position droite de la deuxième colonne

TabImpr.TabLarg(2) = Position droite de la troisième colonne

Etc...

Ceci implique que pour obtenir une largeur d'une colonne autre que la première, je devrai faire la soustraction entre sa position droite et celle de la colonne précédente, que pour obtenir sa position gauche je n'aurais qu'à lire la position droite de l'élément précédent, etc...

Enfin je calcule le nombre de lignes par page et le nombre de pages.

Ensuite nous rentrons dans le travail d'impression proprement dit. A chaque page, j'imprimerai la ligne de titre. Pour chaque colonne, je trace le texte et le cadre. Les deux méthodes, DrawString et DrawRectangle acceptent de nombreuses surcharges. Pour ma part j'utilise :

Graphics.DrawString(String, Font, Brush, Rectangle, StringFormat)

et

Graphics.DrawRectangle(Pen, Rectangle)

Je n'ai plus ensuite qu'à écrire mes lignes et mes rectangles sur autant de pages que nécessaire.

Notez que j'utilise un bloc Try...Catch...Finally pour l'impression des données.

En effet, il n'y a quasiment aucune chance que le nombre de lignes tombe juste à la dernière page. Ce qui fait que ma boucle risque d'invoquer des appels sur ma table avec un numéro de ligne qui n'existe pas. C'est pour éviter ce problème que j'utilise le bloc.



## Impression d'un texte

Il s'agit finalement de la partie la plus complexe à appréhender, non qu'elle soit très complexe mais parce qu'elle demande de bien comprendre la gestion de la géométrie de l'impression.

Commençons par imprimer un fichier en utilisant un code similaire à celui fourni dans MSDN.

```
Imports System.Drawing
Imports System.Drawing.Printing
Imports System.IO

Public Class Form1

    Private FichierTexte As StreamReader

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim ValMarge As Integer =
        Printing.PrinterUnitConvert.Convert(100, PrinterUnit.TenthsOfAMillimeter, PrinterUnit.Display)
        With Me.PrintDocument1.DefaultPageSettings
            .Landscape = False
            .Color = False
            .Margins = New Printing.Margins(ValMarge, ValMarge, ValMarge, ValMarge)
        End With
        FichierTexte = New StreamReader("d:\svg\tutoriel\impression\win32api.txt")
        Me.PrintPreviewDialog1.ShowDialog()
        FichierTexte.Close()
    End Sub

    Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage
        Dim LigneParPage As Integer = 0
        Dim YPos As Single = 0
        Dim cmpt As Integer = 0
        Dim MargeG As Single = e.MarginBounds.Left
        Dim MargeH As Single = e.MarginBounds.Top
        Dim LigneEnCours As String = Nothing
        Dim Police As New Font("Arial", 11)
        Dim InterLigne As Single = Police.GetHeight(e.Graphics)

        LigneParPage = e.MarginBounds.Height / InterLigne
        While cmpt < LigneParPage
            LigneEnCours = FichierTexte.ReadLine()
            If LigneEnCours Is Nothing Then
                Exit While
            End If
            YPos = MargeH + cmpt * InterLigne
            e.Graphics.DrawString(LigneEnCours, Police, Brushes.Black, MargeG, YPos, New StringFormat())
            cmpt += 1
        End While
        If Not (LigneEnCours Is Nothing) Then
            e.HasMorePages = True
        Else
            e.HasMorePages = False
        End If
    End Sub
End Class
```

```
End If
```

```
End Sub
```

```
End Class
```

Ce code va donc imprimer un fichier texte ligne après ligne en utilisant des pages A4 en mode portrait.

Si vous regardez le résultat obtenu vous verrez que l'impression ligne par ligne fonctionne bien, mais que toutes les lignes plus larges que la taille de la page sont tronquées. En effet, l'imprimante ne gère pas les retours à la ligne automatiquement. Nous allons donc devoir adapter notre code pour gérer ceux-ci. Cette problématique va concerner évidemment l'impression de n'importe quel texte dans un rectangle défini.

## L'approche empirique

Cette approche approximative consiste à déterminer un nombre de caractères fixe par ligne et à couper les chaînes en sous chaînes de ce nombre de caractère en ajoutant un tiret à la fin des sous chaînes. Cela donnerait un code du type :

```
Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage
    Dim LigneParPage As Integer = 0
    Dim YPos As Single = 0
    Dim cmpt As Integer = 0
    Dim MargeG As Single = e.MarginBounds.Left
    Dim MargeH As Single = e.MarginBounds.Top
    Dim LigneEnCours As String = Nothing
    Dim Police As New Font("Arial", 11)
    Dim InterLigne As Single = Police.GetHeight(e.Graphics)
    Dim CaracParLigne As Integer = CInt(e.MarginBounds.Width / 11)
    LigneParPage = e.MarginBounds.Height / InterLigne
    While cmpt < LigneParPage
        LigneEnCours = FichierTexte.ReadLine()
        If LigneEnCours Is Nothing Then
            Exit While
        End If
        If LigneEnCours.Length < CaracParLigne Then
            YPos = MargeH + cmpt * InterLigne
            e.Graphics.DrawString(LigneEnCours, Police, Brushes.Black, MargeG, YPos, New StringFormat())
            cmpt += 1
        Else
            Dim tmpLigne As String = Nothing
            Dim NbSubstring As Int32 = LigneEnCours.Length \ CaracParLigne
            If LigneEnCours.Length Mod CaracParLigne <> 0 Then
                NbSubstring += 1
            For cmpt1 As Int32 = 1 To NbSubstring
                If cmpt1 < NbSubstring Then
                    tmpLigne = LigneEnCours.Substring(CaracParLigne * (cmpt1 - 1), CaracParLigne) + "-"
                Else
                    tmpLigne = LigneEnCours.Substring(CaracParLigne * (cmpt1 - 1))
                End If
                YPos = MargeH + cmpt * InterLigne
                e.Graphics.DrawString(tmpLigne, Police, Brushes.Black, MargeG, YPos, New StringFormat())
                cmpt += 1
            Next
        End If
    End While
End Sub
```

```

End While
If Not (LigneEnCours Is Nothing) Then
    e.HasMorePages = True
Else
    e.HasMorePages = False
End If

```

```
End Sub
```

On retrouve donc le code de découpage sous la forme :

```

Dim tmpLigne As String = Nothing
Dim NbSubstring As Int32 = LigneEnCours.Length \ CaracParLigne
If LigneEnCours.Length Mod CaracParLigne <> 0 Then NbSubstring += 1
    For cmpt1 As Int32 = 1 To NbSubstring
        If cmpt1 < NbSubstring Then
            tmpLigne = LigneEnCours.Substring(CaracParLigne * (cmpt1 - 1),
CaracParLigne) + "-"
        Else
            tmpLigne = LigneEnCours.Substring(CaracParLigne * (cmpt1 - 1))
        End If
        YPos = MargeH + cmpt * InterLigne
        e.Graphics.DrawString(tmpLigne, Police, Brushes.Black, MargeG,
YPos, New StringFormat())
        cmpt += 1
    Next
End If

```

Si vous regardez le résultat obtenu, vous allez constater que ce code fonctionne mais que le résultat est assez mauvais. Les lignes sont coupées bien avant la largeur de la page, la coupure peut avoir lieu au milieu de la ponctuation, etc. Cela vient du fait qu'on ne peut pas prédire la taille d'une chaîne à l'impression en définissant uniquement un nombre arbitraire de caractères. Nous nous trouvons donc face à un problème beaucoup plus complexe. En effet, il va donc falloir mesurer la taille des chaînes systématiquement.

Par ailleurs la coupure des mots est de moins en moins admise dans les documents actuels où l'on préfère travailler sur les espaces d'une phrase, ce qui n'est pas toujours adapté non plus mais nous ne sommes pas là pour remettre en question les règles typographiques.

## Impression ajustée

Au doux temps du VB6, il fallait coder un tableau de décomposition de chaînes puis faire des tests de mesure de chaînes par ajouts successifs de mots. Le Framework 2.0 nous affranchit de ce travail puisque nous pouvons utiliser des surcharges des méthodes MeasureString et DrawString qui vont prendre en charge l'impression multi lignes à notre place.

En effet, avec VB 2005, on peut forcer la remise à la ligne d'une chaîne longue en précisant un rectangle à la méthode DrawString. Dans ce cas, il y a gestion automatique des retours à la ligne pour respecter le rectangle si possible. Pour connaître la hauteur nécessaire à l'impression, si on souhaite travailler avec une largeur maximale, on utilise la surcharge suivante de la méthode MeasureString :

**Public Function MeasureString (text As String, font As Font, width As Integer, format As StringFormat) As SizeF**

La valeur SizeF retournée va être ajustée en hauteur en tenant compte de la largeur maximale passée en argument.

Cependant, nous allons devoir gérer le code légèrement différemment puisqu'on ne peut pas savoir avant d'avoir mesuré si la chaîne avec renvoi ne va pas déborder en hauteur. Nous utiliserons un code d'impression de la forme :

```

Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal
e As System.Drawing.Printing.PrintPageEventArgs) Handles
PrintDocument1.PrintPage

    Dim YPos As Single = 0
    Dim MargeG As Single = e.MarginBounds.Left
    Dim MargeH As Single = e.MarginBounds.Top

```

```

Dim Hauteur As Single = e.MarginBounds.Height
Dim Largeur As Single = e.MarginBounds.Width
Static LigneEnCours As String
Static SautForce As Boolean
Dim Police As New Font("Arial", 11)
Dim InterLigne As Single = Police.GetHeight(e.Graphics)
Dim RectSortie As New SizeF
YPos = MargeH
'début de l'imprssion ligne par ligne
While YPos < MargeH + e.MarginBounds.Height
    If SautForce Then
        SautForce = False
    Else
        LigneEnCours = FichierTexte.ReadLine
    End If
    If LigneEnCours Is Nothing Then 'fin du fichier
        Exit While
    End If
    RectSortie = e.Graphics.MeasureString(LigneEnCours, Police,
Largeur, New StringFormat())
    If RectSortie.Height = 0 Then 'cas de la ligne vide
        YPos = YPos + InterLigne
    Else 'ligne non vide
        'controle de hauteur
        If YPos + RectSortie.Height > Hauteur Then
            SautForce = True
            Exit While
        Else
            e.Graphics.DrawString(LigneEnCours, Police,
Brushes.Black, New RectangleF(MargeG, YPos, Largeur, RectSortie.Height),
New StringFormat())
            YPos = YPos + RectSortie.Height
        End If
    End If
End While
If Not (LigneEnCours Is Nothing) Then
    e.HasMorePages = True
Else
    e.HasMorePages = False
End If
End Sub

```

## ***Imprimer des images***

Nous allons traiter l'impression des images en imaginant quelques scénarios classiques. Il n'y a pas de véritable complexité à la chose sauf si vous désirez appliquer des modifications sur les images pendant le travail d'impression.

### **Cas des images existantes**

L'objet Graphics propose une série de méthode telle que :

**DrawIcon** → imprime une icône à la position spécifiée et éventuellement ajustée au rectangle spécifié.

**DrawIconUnstretched** → imprime une icône sans ajustement éventuellement tronquée

**DrawImage** → imprime tout ou partie d'une image avec ajustement ou non. Voir la liste des surcharges

**DrawImageUnscaled** → imprime une image avec sa taille d'origine

**DrawImageUnscaledAndClipped** → imprime une image avec sa taille d'origine éventuellement tronquée

Nous verrons le travail avec des icônes un peu plus loin, commençons donc par traiter des images.

La première et quasiment la seule difficulté consiste à savoir comment gérer les dimensions de l'image. En effet, on peut vouloir avoir soit des images ajustées à une taille définie, soit avoir une taille fluctuante pour avoir les dimensions originales de l'image ou un rapport de celle-ci, puisqu'il est assez rare qu'on travaille sur des images tronquées.

### Ajustement des images

Nous allons commencer par imprimer une image de grande taille, supérieure à la taille d'une page.

Partons du code suivant :

```
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Drawing.Printing
Imports System.IO

Public Class Form1
    Private MonImage As Bitmap
    Friend WithEvents PrintPreviewDialog1 As PrintPreviewDialog

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

        Try
            MonImage = New Bitmap("d:\user\tutos\impression\baron.jpg")
            Dim ImprPage As New PrintDocument()
            AddHandler ImprPage.PrintPage, AddressOf Me.pd_PrintPage
            Me.PrintPreviewDialog1 = New PrintPreviewDialog
            With Me.PrintPreviewDialog1
                .ClientSize = New System.Drawing.Size(400, 300)
                .Location = New System.Drawing.Point(50, 50)
                .UseAntiAlias = True
                .Document = ImprPage
                .ShowDialog(Me)
            End With
            Catch ex As Exception
                MsgBox("Erreur lors de l'impression", MsgBoxStyle.Critical
And MsgBoxStyle.OkOnly)
            End Try

        End Sub

        Private Sub pd_PrintPage(ByVal sender As Object, ByVal ev As PrintPageEventArgs)

            ev.Graphics.DrawImage(MonImage, ev.MarginBounds)

        End Sub
    End Class
```

Dans ce cas, j'ajuste mon image à la taille de la zone imprimable. En l'état, il y a relativement peu de chance pour que la proportion de l'image ait été respectée. C'est toujours un problème lors de l'ajustement des images, on prend toujours un risque de modifier fortement l'image sur des problèmes d'ajustement. Une approche plus propre serait d'utiliser le code suivant :

```
Private Sub pd_PrintPage(ByVal sender As Object, ByVal ev As PrintPageEventArgs)

    Using MyGr As Graphics = ev.Graphics
        Dim LargImage As Single = MonImage.Width /
MonImage.HorizontalResolution * 100
```

```

        Dim RapX As Single = ev.MarginBounds.Width / LargImage
        Dim HautImage As Single = MonImage.Height /
MonImage.VerticalResolution * 100
        Dim RapY As Single = ev.MarginBounds.Height / HautImage
        RapX = CSng(IIf(RapX < RapY, RapX, RapY))
        Dim RectSortie As New Rectangle(0, 0, CInt(LargImage * RapX),
CInt(HautImage * RapX))
        RectSortie.X = ev.MarginBounds.X + CInt((ev.MarginBounds.Width -
RectSortie.Width) / 2)
        RectSortie.Y = ev.MarginBounds.Y + CInt((ev.MarginBounds.Height -
RectSortie.Height) / 2)
        MyGr.DrawImage(MonImage, RectSortie)
        'MyGr.DrawRectangle(Pens.Red, ev.MarginBounds)
    End Using
End Sub

```

Dans ce cas là, je calcule la proportion ajustée sur une des deux dimensions et je réduis l'autre dimension avec cette proportion. J'ai donc une image réduite qui tient dans ma page et dont le ratio a été respecté.

Invertissons maintenant le problème en prenant une image petite que nous souhaitons agrandir, en utilisant le code précédent. L'image va rapidement subir une grosse déperdition de qualité si le facteur de grossissement est important. Dans une certaine mesure, on peut éventuellement jouer avec la propriété InterpolationMode de l'objet Graphics, mais ceci n'aura guère d'effet si votre facteur de grossissement est supérieur à 100%.

### **Fichier Metafile**

Les images Metafile sont des images partiellement vectorielles, c'est-à-dire définie par des groupes de données géométriques plutôt que par des tableaux de pixels. Ces images ont généralement l'avantage de mieux subir l'épreuve de l'ajustement de taille quand elles sont fortement vectorisées.

Dans le cadre de l'impression, elles se gèrent comme les bitmaps si ce n'est qu'on utilise le Type MetaFile pour l'instance de l'image. Prenons l'exemple suivant :

```

Private MonImage As Metafile

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

    Try
        MonImage = New
Metafile("d:\user\tutos\impression\B20G9Tecmulti.wmf")
        Dim ImprPage As New PrintDocument()
        AddHandler ImprPage.PrintPage, AddressOf Me.pd_PrintPage
        Me.PrintPreviewDialog2.Document = ImprPage
        Me.PrintPreviewDialog2.ShowDialog(Me)
    Catch ex As Exception
        MsgBox("Erreur lors de l'impression", MsgBoxStyle.Critical And
MsgBoxStyle.OkOnly)
    End Try

End Sub

Private Sub pd_PrintPage(ByVal sender As Object, ByVal ev As
PrintPageEventArgs)

    ev.Graphics.DrawImage(MonImage, New PointF(ev.MarginBounds.X +
CInt((ev.MarginBounds.Width - (MonImage.Width /
MonImage.HorizontalResolution * 100)) / 2), ev.MarginBounds.Y +
CInt((ev.MarginBounds.Height - (MonImage.Height /
MonImage.VerticalResolution * 100)) / 2)))

```

```
End Sub
```

### **Modification à l'impression**

Il est parfaitement possible de modifier l'image avant de l'imprimer, de ne prendre qu'une partie de l'image, de zoomer une partie spécifique et toute sorte de traitement divers et variés. La seule contrainte est in fine de passer l'image désirée à la méthode DrawImage. Voyons rapidement quelques exemples.

#### **Rotation**

En utilisant la méthode RotateFlip sur l'image :

```
Private Sub pd_PrintPage(ByVal sender As Object, ByVal ev As
PrintPageEventArgs)

    MonImage.RotateFlip(RotateFlipType.Rotate90FlipNone)
    ev.Graphics.DrawImage(MonImage, New PointF(ev.MarginBounds.X +
CInt((ev.MarginBounds.Width - (MonImage.Width /
MonImage.HorizontalResolution * 100)) / 2), ev.MarginBounds.Y +
CInt((ev.MarginBounds.Height - (MonImage.Height /
MonImage.VerticalResolution * 100)) / 2)))

End Sub
```

Ou en utilisant une surcharge de DrawString :

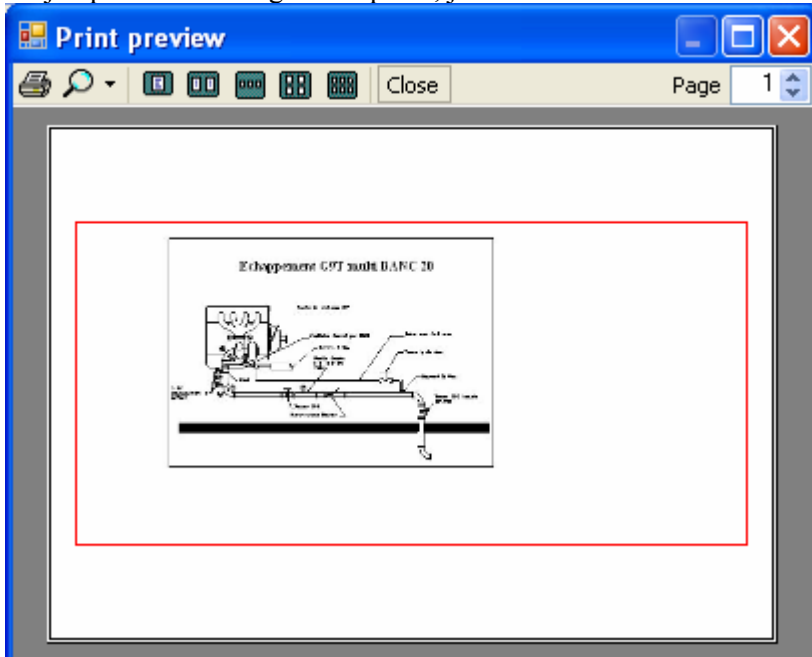
```
Private Sub pd_PrintPage(ByVal sender As Object, ByVal ev As
PrintPageEventArgs)

    Using MyGr As Graphics = ev.Graphics
        Dim LargImage As Single = MonImage.Width /
MonImage.HorizontalResolution * 100
        Dim HautImage As Single = MonImage.Height /
MonImage.VerticalResolution * 100
        Dim RectSortie As New Rectangle(0, 0, CInt(LargImage),
CInt(HautImage))
        RectSortie.X = ev.MarginBounds.X + CInt((ev.MarginBounds.Width -
RectSortie.Width) / 2)
        RectSortie.Y = ev.MarginBounds.Y + CInt((ev.MarginBounds.Height -
RectSortie.Height) / 2)
        Dim destinationPoints As Point() = {New Point(RectSortie.X +
RectSortie.Height, RectSortie.Y), New Point(RectSortie.X +
RectSortie.Height, RectSortie.Y + RectSortie.Width), New
Point(RectSortie.X, RectSortie.Y)}
        MyGr.DrawImage(MonImage, destinationPoints)
    End Using

End Sub
```

## Partie d'une image

Prenons le cas suivant. J'ai une image assez grande qui contient l'image qui m'intéresse dans un cadre noir. Si j'imprime mon image telle quelle, j'obtiendrais la sortie suivante :



Où le rectangle rouge représente les limites de l'image.

Je souhaite donc récupérer la partie dans le cadre noir pour centrer correctement mon image. Je vais donc travailler sur mon image afin de trouver les limites de ce cadre.

```
Private Sub pd_PrintPage(ByVal sender As Object, ByVal ev As
PrintPageEventArgs)

    'traitement de l'image (travail en pixel)
    Dim x, y As Integer
    Dim X1, X2, Y1, Y2 As Int32
    ' Recherche les limites du cadre noir
    '1) recherche rapide d'un pixel non blanc au milieu de chaque
ligne en partant du haut
    For y = 1 To MonImage.Height - 1
        If MonImage.GetPixel(CInt(MonImage.Width / 2), y).ToArgb <> -
1 Then
            Y1 = y
            Exit For
        End If
    Next
    '2) recherche rapide d'un pixel non blanc au milieu de chaque
ligne en partant du bas
    For y = MonImage.Height - 1 To 0 Step -1
        If MonImage.GetPixel(CInt(MonImage.Width / 2), y).ToArgb <> -
1 Then
            Y2 = y
            Exit For
        End If
    Next
    '3) recherche rapide d'un pixel non blanc au milieu de chaque
colonne en partant de la gauche
    For x = 1 To MonImage.Width - 1 'To 0 Step -1
        If MonImage.GetPixel(x, CInt(MonImage.Height / 2)).ToArgb <>
-1 Then
            X1 = x
```



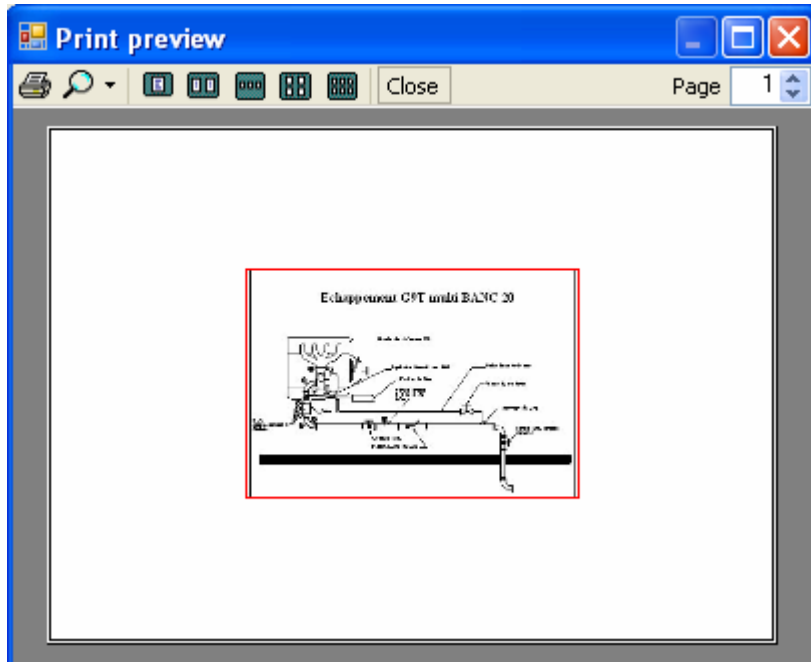
```

        Exit For
    End If
Next
'4) recherche rapide d'un pixel non blanc au milieu de chaque
colonne en partant de la droite
For x = MonImage.Width - 1 To 0 Step -1
    If MonImage.GetPixel(x, CInt(MonImage.Height / 2)).ToArgb <>
-1 Then
        X2 = x
        Exit For
    End If
Next
Dim NouvelleTaille As New Rectangle(X1, Y1, X2 - X1, Y2 - Y1)
Dim NouvelleImage As Bitmap = MonImage.Clone(NouvelleTaille,
MonImage.PixelFormat)
MonImage = NouvelleImage
Using MyGr As Graphics = ev.Graphics
    Dim LargImage As Single = MonImage.Width /
MonImage.HorizontalResolution * 100
    Dim HautImage As Single = MonImage.Height /
MonImage.VerticalResolution * 100
    Dim RectSortie As New Rectangle(0, 0, CInt(LargImage),
CInt(HautImage))
    RectSortie.X = ev.MarginBounds.X +
CInt((ev.MarginBounds.Width - RectSortie.Width) / 2)
    RectSortie.Y = ev.MarginBounds.Y +
CInt((ev.MarginBounds.Height - RectSortie.Height) / 2)
    MyGr.DrawImage(MonImage, RectSortie)
    MyGr.DrawRectangle(Pens.Red, RectSortie)
End Using

End Sub

```

J'obtiendrais alors la sortie suivante :



## Cas des images créées

Imprimons maintenant une image créée par les méthodes graphiques de l'objet Graphics. Dans l'exemple suivant, nous allons tracer l'éternel camembert avec une petite légende. Nous allons pour cela utiliser le code suivant :

```
Imports System.Drawing
Imports System.Drawing.Printing

Public Class Form1

    Private TabResult(4) As Int32
    Private Total As Int32

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        Try
            Randomize()
            For cmpt As Int32 = 0 To 4
                TabResult(cmpt) = CInt(Int((100 * Rnd()) + 1))
                Total = Total + TabResult(cmpt)
            Next
            Dim ImprPage As New PrintDocument()
            AddHandler ImprPage.PrintPage, AddressOf Me.pd_PrintPage
            ImprPage.DefaultPageSettings.Landscape = True
            Me.PrintPreviewDialog2.Document = ImprPage
            Me.PrintPreviewDialog2.ShowDialog(Me)
        Catch ex As Exception
            MsgBox("Erreur lors de l'impression", MsgBoxStyle.Critical
And MsgBoxStyle.OkOnly)
        End Try

    End Sub

    Private Sub pd_PrintPage(ByVal sender As Object, ByVal ev As
PrintPageEventArgs)

        ' Crée un stylo
        Dim BlackPen As New Pen(Color.Black, 3)
        'crée cinq brosses
        Dim TabBrush(4) As Drawing2D.HatchBrush
        TabBrush(0) = New
Drawing2D.HatchBrush(Drawing2D.HatchStyle.DarkVertical, Color.Blue,
Color.White)
        TabBrush(1) = New
Drawing2D.HatchBrush(Drawing2D.HatchStyle.DarkHorizontal, Color.Red,
Color.White)
        TabBrush(2) = New
Drawing2D.HatchBrush(Drawing2D.HatchStyle.DarkUpwardDiagonal,
Color.Green, Color.White)
        TabBrush(3) = New
Drawing2D.HatchBrush(Drawing2D.HatchStyle.ForwardDiagonal, Color.Yellow,
Color.White)
        TabBrush(4) = New
Drawing2D.HatchBrush(Drawing2D.HatchStyle.NarrowHorizontal, Color.Violet,
Color.White)
```

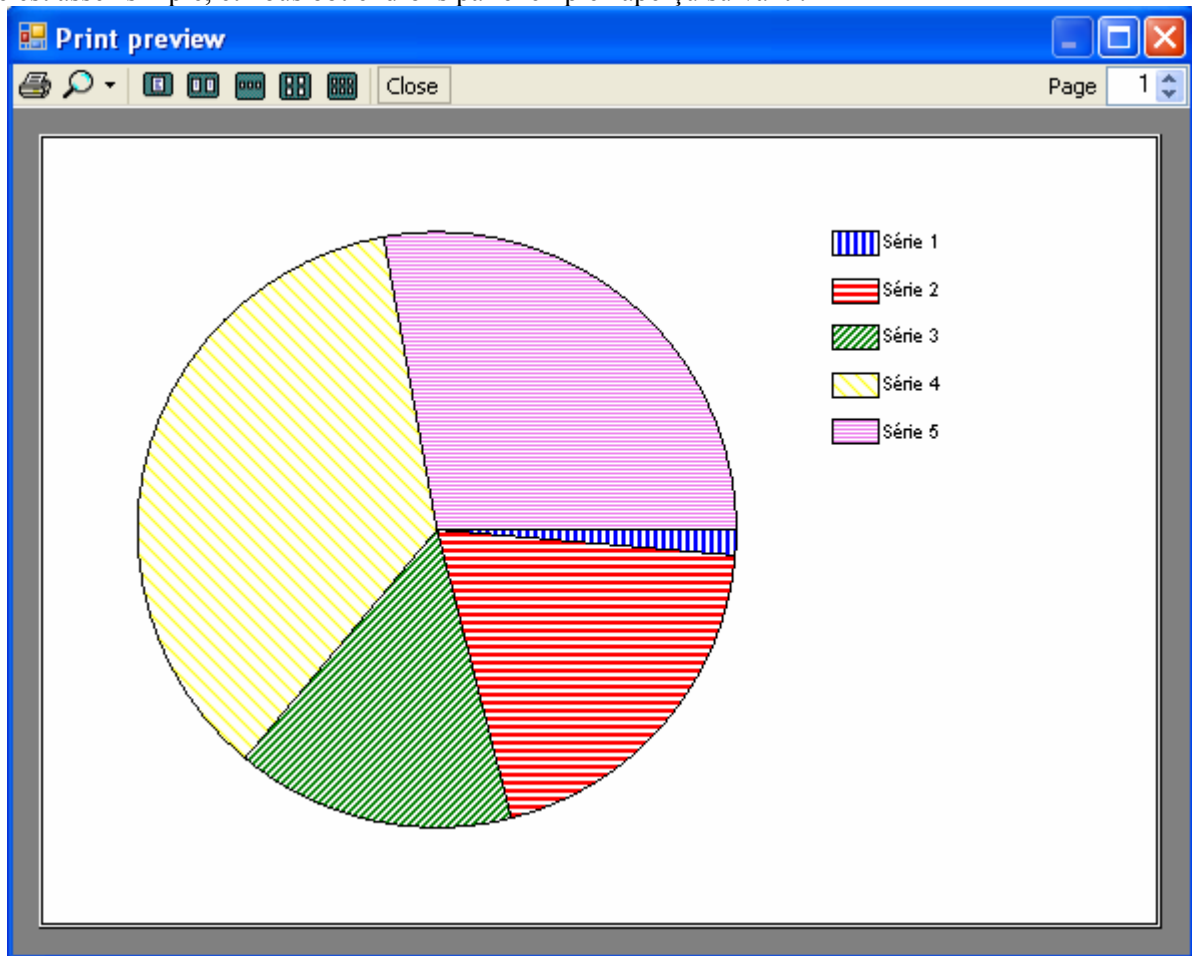
```

        ' Crée le rectangle du cercle
        Dim rect As New Rectangle(CInt((ev.PageBounds.Width -
ev.MarginBounds.Width) / 2), CInt((ev.PageBounds.Height -
ev.MarginBounds.Height) / 2), ev.MarginBounds.Width,
ev.MarginBounds.Height)
        Dim startAngle As Single = 0.0F
        Dim sweepAngle As Single = 0.0F
        Using MyGr As Graphics = ev.Graphics
            Dim Ligne As Single = MyGr.MeasureString("Serie", New
Font("Arial", 14)).Height
            For cmpt As Int32 = 0 To 4
                ' calcule les angles
                sweepAngle = CSng(TabResult(cmpt) / Total) * 360.0F
                'remplit et trace la zone
                MyGr.FillPie(TabBrush(cmpt), rect, startAngle,
sweepAngle)
                MyGr.DrawPie(BlackPen, rect, startAngle, sweepAngle)
                startAngle = startAngle + sweepAngle
                'trace le rectangle de légende puis le texte
                MyGr.FillRectangle(TabBrush(cmpt), rect.Right + 100,
ev.MarginBounds.Top + (cmpt * 2) * Ligne, 2 * Ligne, Ligne)
                MyGr.DrawRectangle(Pens.Black, rect.Right + 100,
ev.MarginBounds.Top + (cmpt * 2) * Ligne, 2 * Ligne, Ligne)
                MyGr.DrawString("Série " + (cmpt + 1).ToString, New
Font("Arial", 14), Brushes.Black, rect.Right + 150, ev.MarginBounds.Top +
(cmpt * 2) * Ligne)
            Next
        End Using

    End Sub
End Class

```

J'utilise une fonction aléatoire pour générer les données à tracer dans ce cas. Comme nous le voyons, le code est assez simple, et nous obtiendrons par exemple l'aperçu suivant :



## ***Imprimer des contrôles particuliers***

Avant de réunir tout ce que nous avons vu dans un exemple d'impression un peu plus touffu, arrêtons-nous un peu sur des impressions spécifiques de contrôles. Généralement il n'y a pas de lien particulier entre l'interface utilisateur et la sortie à l'impression. Toutefois il arrive dans certains cas que l'on souhaite imprimer des contrôles de façon assez similaire à ce qu'ils sont à l'écran.

### **ListBox avec élément sélectionné**

Ce code est assez simple puisque ce n'est qu'une variation de l'impression de ligne de texte.

```
Imports System.Drawing
Imports System.Drawing.Printing

Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        For cmpt As Int32 = 1 To 14
            Me.ListBox1.Items.Add("Elément " + cmpt.ToString)
        Next
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Me.PrintDocument1.DefaultPageSettings.Margins = New Margins(50,
50, 50, 50)
```

```

        Me.PrintDocument1.OriginAtMargins = True
        Me.PrintPreviewDialog1.ShowDialog()
    End Sub

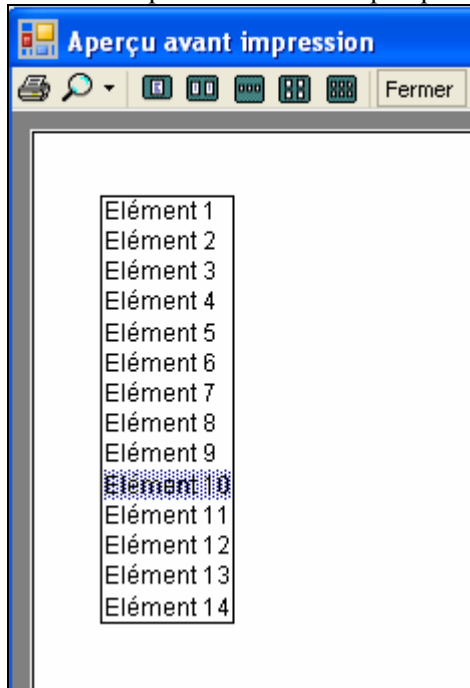
    Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object,
    ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles
    PrintDocument1.PrintPage

        Dim TailleMax, HauteurLigne As Single
        Dim Police As New Font("Arial", 12)
        Dim MyBrush As New
    Drawing2D.HatchBrush(Drawing2D.HatchStyle.Percent05, Color.Blue,
    Color.White)
        Dim YPos As Single = 10.0F
        Dim XPos As Single = 10.0F
        Using MyGr As Graphics = e.Graphics
            HauteurLigne = MyGr.MeasureString("E", Police).Height
            For Each Element As Object In Me.ListBox1.Items
                TailleMax = CSng(IIf(TailleMax <
    MyGr.MeasureString(Element.ToString, Police).Width,
    MyGr.MeasureString(Element.ToString, Police).Width, TailleMax))
            Next
            For cmpt As Int32 = 0 To Me.ListBox1.Items.Count - 1
                YPos = cmpt * HauteurLigne
                If Me.ListBox1.SelectedIndex = cmpt Then
                    MyGr.FillRectangle(New
    Drawing2D.HatchBrush(Drawing2D.HatchStyle.Percent25, Color.DarkBlue,
    Color.White), XPos, YPos, TailleMax, HauteurLigne)
                    End If
                    MyGr.DrawString(Me.ListBox1.Items(cmpt).ToString, Police,
    Brushes.Black, XPos, YPos)
                Next
                MyGr.DrawRectangle(New Pen(Color.Black, 2), 10, 0, TailleMax,
    YPos + HauteurLigne)
            End Using

        End Sub
    End Class

```

La sortie imprimante donnera quelque chose du style :



## **Le contrôle ListView**

### **La copie d'image**

La méthode marche pour presque tous les contrôles. Comme nous allons le voir, elle possède quand même quelques inconvénients mais a le mérite de répondre facilement à quelques cas aisés comme l'impression d'une case à cocher par exemple.

Les contrôles possèdent une méthode DrawToBitmap qui permet de capturer l'image du contrôle. Imaginons tout d'abord le code suivant :

```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        With Me.ListView1
            .View = View.LargeIcon
            .LargeImageList = Me.ImageList1
            For cmpt As Int32 = 1 To 6
                Me.ListView1.Items.Add("Elément " + cmpt.ToString, 0)
            Next
        End With
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Me.PrintDocument1.DefaultPageSettings.Margins = New Margins(50, 50, 50, 50)
        Me.PrintDocument1.OriginAtMargins = True
        Me.PrintPreviewDialog1.ShowDialog()
    End Sub

    Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage
```

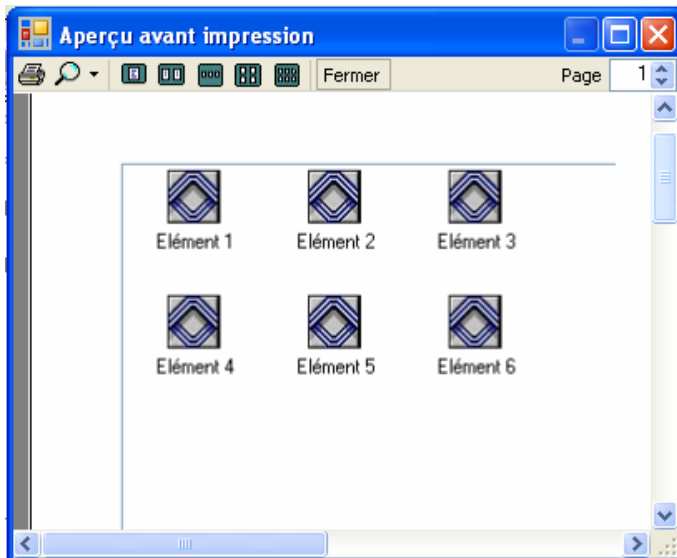
```

    Dim Rect As New Rectangle(New Point(10, 10),
Me.ListView1.PreferredSize)
    Dim MonImage As New Bitmap(Rect.Width, Rect.Height)
    Me.ListView1.DrawToBitmap(MonImage, Rect)
    e.Graphics.DrawImage(MonImage, New Point(10, 10))

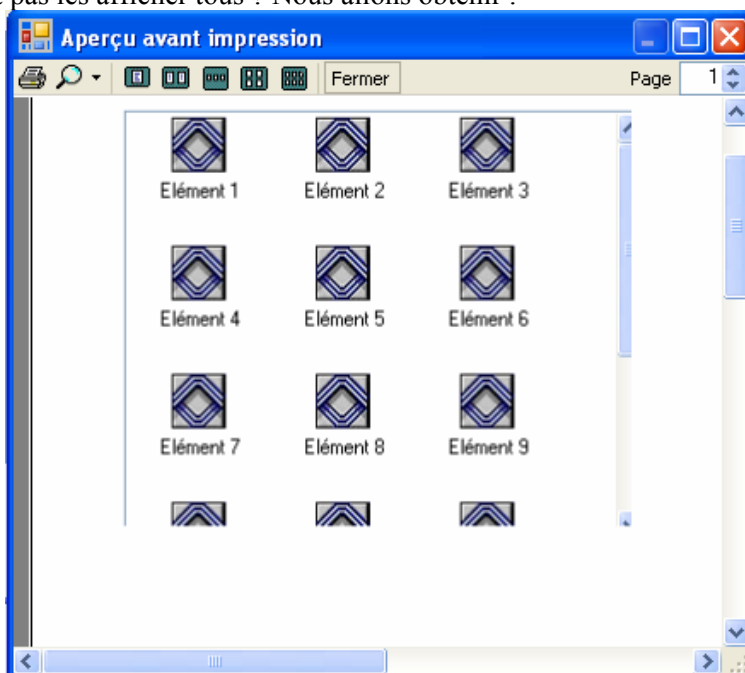
End Sub
End Class

```

Nous obtiendrons la sortie suivante :



A l'exception du demi-cadre que nous pourrions masquer en traçant un rectangle, la sortie correspond à nos souhaits. Cependant que va-t-il se passer si au lieu de six éléments, j'en trace seize, et que ma liste ne peut pas les afficher tous ? Nous allons obtenir :



Ce qui, admettons-le, est plutôt vilain. Evidemment je pourrais ruser et agrandir temporairement la liste pour faire rentrer les éléments sans avoir les barres de défilement. Encore faudrait-il que la feuille puisse supporter la taille du contrôle sans elle-même se mettre en mode scroll. Par ailleurs, le problème se reposerait avec un plus grand nombre d'éléments. Bref, cette méthode est adaptée dans certains cas mais pas pour les cas où le contrôle peut se mettre en mode défilement.

## L'impression gérée

Nous voilà donc prêt à gérer l'impression par le code. Bien que cela puisse paraître complexe de prime abord, ce n'est qu'un petit mix de ce que nous avons déjà vu. Dans le cas où l'on ne connaît pas le nombre d'éléments, on fixe une dimension, par exemple la largeur, et on module sur l'autre dimension.

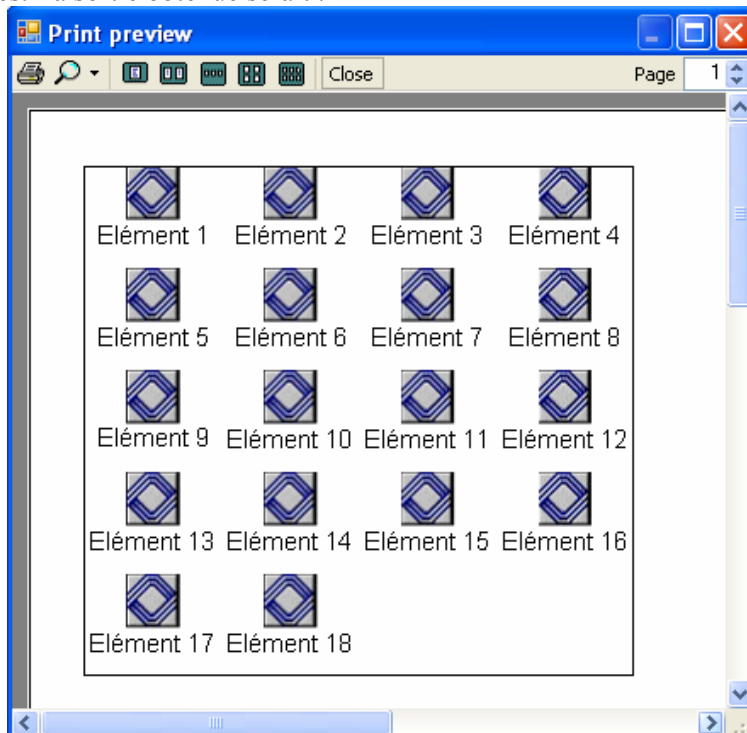
Ensuite, il s'agit juste d'un tableau où chaque case va contenir l'image et le texte. Reprenons le code précédant en créant dix-huit éléments, et en fixant la largeur du contrôle à quatre éléments par ligne.

```
Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage

    Dim TailleCase As New SizeF(0.0F, 0.0F)
    Dim Police As New Font("Arial", 11)
    Dim LargImage As Single = 0.0F
    Dim HautImage As Single = 0.0F
    Using MyGr As Graphics = e.Graphics
        'calcul de la taille d'une case
        Dim MonImage As New
        Bitmap(Me.ImageList1.Images.Item(Me.ListView1.Items(0).ImageIndex))
        HautImage = MonImage.Height / MonImage.VerticalResolution * 100
        TailleCase.Height = CSng(1.5 *
        MyGr.MeasureString(Me.ListView1.Items(0).Text, Police).Height +
        HautImage)
        For Each Elem As ListViewItem In Me.ListView1.Items
            Dim CompTaille As Single
            LargImage = MonImage.Width / MonImage.HorizontalResolution *
100
            CompTaille = CSng(IIf(MyGr.MeasureString(Elem.Text,
Police).Width < LargImage, LargImage, MyGr.MeasureString(Elem.Text,
Police).Width))
            If TailleCase.Width < CompTaille Then
                TailleCase.Width = CompTaille
            End If
        Next
        Dim MonFormat As New StringFormat()
        MonFormat.Alignment = StringAlignment.Center
        MonFormat.LineAlignment = StringAlignment.Near
        Dim NbLigne As Int32 = (Me.ListView1.Items.Count - 1) \ 4 +
CInt(IIf(Me.ListView1.Items.Count Mod 4 > 0, 1, 0))
        For cmptLigne As Int32 = 0 To NbLigne
            For cmptColonne As Int32 = 0 To 3
                If cmptColonne + (cmptLigne * 4) >
Me.ListView1.Items.Count - 1 Then Exit For
                MyGr.DrawImage(MonImage, cmptColonne * TailleCase.Width +
((TailleCase.Width - LargImage) / 2), cmptLigne * TailleCase.Height)
                MyGr.DrawString(Me.ListView1.Items(cmptColonne +
(cmptLigne * 4)).Text, Police, Brushes.Black, New RectangleF(cmptColonne
* TailleCase.Width, cmptLigne * TailleCase.Height + HautImage,
TailleCase.Width, TailleCase.Height - HautImage), MonFormat)
            Next
        Next
        MyGr.DrawRectangle(Pens.Black, 0, 0, 4 * TailleCase.Width,
NbLigne * TailleCase.Height)
    End Using
End Sub
```



Comme vous le voyez, le code est assez simple. On calcule la taille de la case la plus grande et on imprime un tableau de cases. Notez que pour les très longues listes il faudrait aussi gérer le code des sauts de pages. La sortie obtenue serait :



### ListView en mode détail

Là encore, il s'agit d'une application de méthode connue. J'ai un peu modifié le code de remplissage de la liste pour vous montrer le principe de la mesure en amont, c'est-à-dire en utilisant CreateMeasurementGraphics. Pour le reste c'est assez standard.

```
Imports System.Drawing
Imports System.Drawing.Printing
Imports System.IO

Public Class Form1

    Private LColonne(2) As Single

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Try
            'lecture des fichiers de c:\windows
            Dim Repertoire As DirectoryInfo = New
DirectoryInfo("c:\windows\")
            Dim ListFichiers() As FileInfo = Repertoire.GetFiles()
            Me.ListView1.View = View.Details
            Me.ListView1.SmallImageList = Me.ImageList1
            'récupération des informations pour remplir la liste
            Me.ListView1.Columns.Add("Nom", 100,
HorizontalAlignment.Center)
            Me.ListView1.Columns.Add("Taille", 100,
HorizontalAlignment.Left)
            Me.ListView1.Columns.Add("Créé le", 100,
HorizontalAlignment.Left)
            'récupère un graphics de l'imprimante
```

```

        Using MyGr As Graphics =
Me.PrintDocument1.PrinterSettings.CreateMeasurementGraphics
    'mesure la largeur de l'icone
    Dim TailleImage As Single = Me.ImageList1.ImageSize.Width
/ Me.ImageList1.Images(0).HorizontalResolution * 100
    For Each Fichier As FileInfo In ListFichiers
        Dim Element As New ListViewItem(Fichier.Name)
        'mesure de la colonne nom de fichier + icone
        LColonne(0) =
Math.Max(MyGr.MeasureString(Fichier.Name, Me.ListView1.Font).Width +
TailleImage, LColonne(0))
        Element.SubItems.Add(Fichier.Length.ToString)
        'mesure de la colonne taille de fichier
        LColonne(1) =
Math.Max(MyGr.MeasureString(Fichier.Length.ToString,
Me.ListView1.Font).Width, LColonne(1))

        Element.SubItems.Add(Fichier.CreationTime.ToLongDateString)
        'mesure de la colonne date de création
        LColonne(2) =
Math.Max(MyGr.MeasureString(Fichier.CreationTime.ToLongDateString,
Me.ListView1.Font).Width, LColonne(2))
        'sélectionne l'icone à utiliser
        If (Fichier.Attributes And FileAttributes.Hidden) <>
0 Then
            Element.ImageIndex = 3
        ElseIf (Fichier.Attributes And
FileAttributes.ReadOnly) <> 0 Then
            Element.ImageIndex = 2
        ElseIf (Fichier.Attributes And
FileAttributes.Archive) <> 0 Then
            Element.ImageIndex = 1
        Else
            Element.ImageIndex = 1
        End If
        Me.ListView1.Items.Add(Element)
    Next
    'ajuste les colonnes pour l'affichage à l'écran

Me.ListView1.Columns(0).AutoSize(ColumnHeaderAutoSizeStyle.ColumnCont
ent)

Me.ListView1.Columns(1).AutoSize(ColumnHeaderAutoSizeStyle.ColumnCont
ent)

Me.ListView1.Columns(2).AutoSize(ColumnHeaderAutoSizeStyle.ColumnCont
ent)

        End Using
    Catch ex As Exception
        MsgBox("Erreur dans la lecture du répertoire",
MsgBoxStyle.Critical And MsgBoxStyle.OkOnly)
    End Try

End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

```

```

        Me.PrintDocument1.DefaultPageSettings.Margins = New Margins(50,
50, 50, 50)
        Me.PrintDocument1.OriginAtMargins = True
        Me.PrintPreviewDialog1.ShowDialog()
    End Sub

    Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object,
ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles
PrintDocument1.PrintPage

        Static Position As Int32
        Using MyGr As Graphics = e.Graphics
            Dim HauteurTexte As Single = MyGr.MeasureString("X",
Me.ListView1.Font).Height
            Dim LigneParPage As Int32 = CInt(e.MarginBounds.Height /
HauteurTexte)
            Dim cmpt As Int32
            'crée deux format pour la première ou les autres colonnes
            Dim FormatColl As New StringFormat
            FormatColl.Alignment = StringAlignment.Near
            FormatColl.LineAlignment = StringAlignment.Center
            Dim FormatCenter As New StringFormat
            FormatCenter.Alignment = StringAlignment.Center
            FormatCenter.LineAlignment = StringAlignment.Center
            Dim police As Font = Me.ListView1.Font
            Dim XPos, YPos As Single
            'imprime les titres
            For cmpt = 0 To 2
                MyGr.FillRectangle(New
Drawing2D.HatchBrush(Drawing2D.HatchStyle.Percent25, Color.Gray,
Color.White), XPos, YPos, LColonne(cmpt), HauteurTexte)
                MyGr.DrawString(CStr(Choose(cmpt + 1, "Nom", "Taille",
"Crée le")), police, Brushes.Black, New RectangleF(XPos, YPos,
LColonne(cmpt), HauteurTexte), FormatCenter)
                XPos += LColonne(cmpt)
            Next
            YPos += HauteurTexte
            XPos = 0
            Dim MonIcône As Image
            'imprime la liste
            For cmptLig As Int32 = 0 To Me.ListView1.Items.Count - 1
                If cmptLig > LigneParPage Then 'changement de page
                    e.HasMorePages = True
                    Exit For

                ElseIf Position > Me.ListView1.Items.Count - 1 Then
'travail terminé
                    e.HasMorePages = False
                    Exit For
                End If
                For cmptCol As Int32 = 0 To Me.ListView1.Columns.Count -
1
                    If cmptCol = 0 Then
                        MonIcône =
Me.ImageList1.Images(Me.ListView1.Items(Position).ImageIndex)
                        MyGr.DrawImage(MonIcône, 0, YPos + ((HauteurTexte
- (MonIcône.Height / MonIcône.VerticalResolution * 100)) / 2))

```

```

MyGr.DrawString(Me.ListView1.Items(Position).Text, police, Brushes.Black,
New RectangleF(XPos + (MonIcône.Width / MonIcône.HorizontalResolution *
100), YPos, LColonne(cmptCol), HauteurTexte), FormatColl)
    Else

MyGr.DrawString(Me.ListView1.Items(Position).SubItems(cmptCol).Text,
police, Brushes.Black, New RectangleF(XPos, YPos, LColonne(cmptCol),
HauteurTexte), FormatCenter)
        End If
    XPos += LColonne(cmptCol)
Next
YPos += HauteurTexte
XPos = 0
Position += 1
Next
MyGr.DrawRectangle(Pens.Black, 0, 0, LColonne(0) +
LColonne(1) + LColonne(2), YPos)
MyGr.DrawRectangle(Pens.Black, 0, 0, LColonne(0) +
LColonne(1) + LColonne(2), HauteurTexte)
MyGr.DrawRectangle(Pens.Black, LColonne(0), 0, LColonne(1),
YPos)
    End Using

End Sub

End Class

```

Il y a toujours plusieurs possibilités pour aborder l'impression des tableaux. On peut tout aussi bien décider de faire un tableau de coordonnées que, comme dans notre exemple, de faire bouger deux variables de position. Tout comme on peut traiter la gestion des pages de façon relativement statique, comme dans cet exemple, ou de manière plus dynamique en calculant si la ligne suivante reste dans la zone des marges. Evidemment là, j'essaye de vous montrer quelques petites variations, mais habituellement j'utilise des snippets et mon approche est identique dans presque tous les cas.

## Le contrôle TreeView

Nous allons exécuter ici un exercice nettement plus complexe. L'impression d'un contrôle TreeView demande une bonne gestion de la géométrie d'impression. En effet, il va nous falloir dans le même temps gérer la notion d'impression des icônes selon que le nœud soit sélectionné ou non, les cases plus/minus, les traits de liaisons, l'indentation, bref un bel exercice de style. Cependant vous verrez que même ce type d'impression est assez simple pour peu qu'on l'aborde avec méthode.

Imaginons donc un contrôle TreeView servant à visualiser le schéma d'une table de base de données, en l'occurrence la table "Products" de la base SQL-Server Northwind.

Le code de remplissage sera donc :

```

Imports System.Data
Imports System.Data.SqlClient
Imports System.Drawing
Imports System.Drawing.Printing

Public Class Form1

    Private Structure Gestion
        Dim Niveau As Int32
        Dim HasChild As Boolean
        Dim IndexImage As Int32
        Dim IsExpand As Boolean
        Dim Texte As String
    End Structure

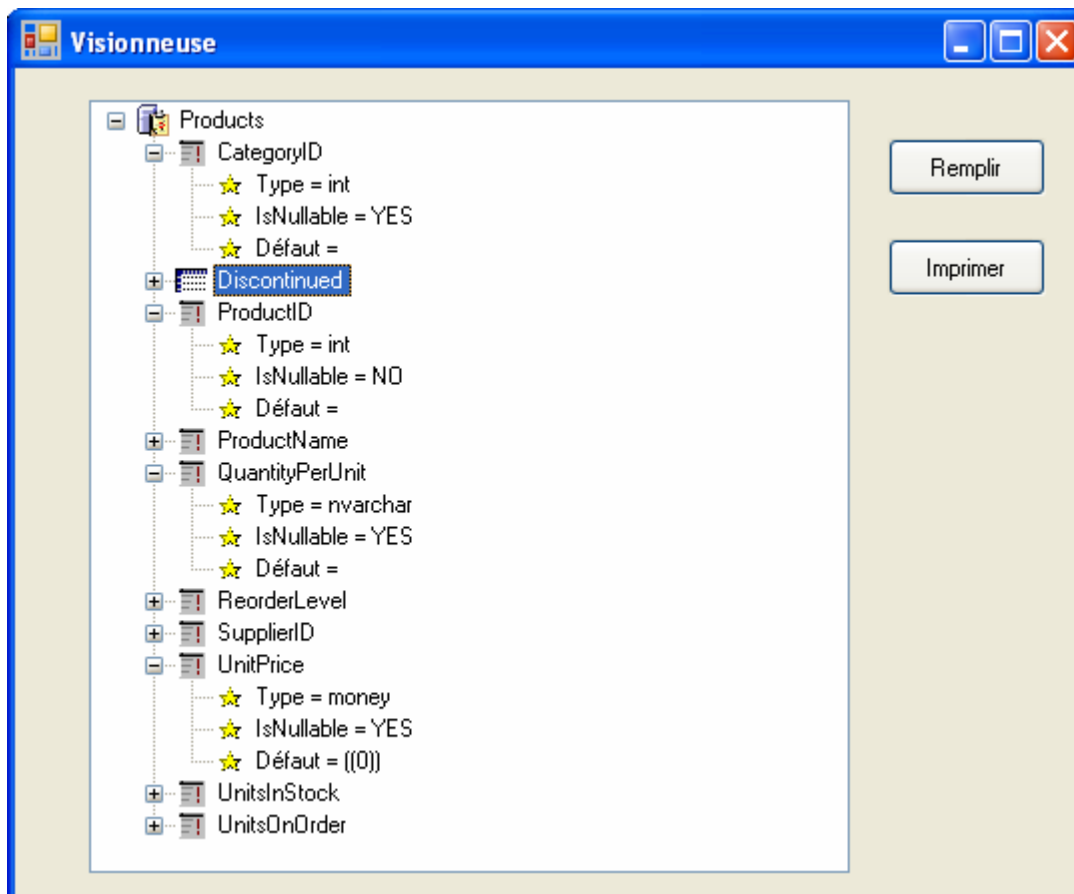
```

```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdFill.Click
    Using SqlConn As New SqlConnection("Data Source=ACER-
0717DB779B\SQLEXPRESS;Initial Catalog=Northwind;Integrated
Security=True")
        SqlConn.Open()
        Dim TableColonne As DataTable = SqlConn.GetSchema("Columns",
New String() {Nothing, Nothing, "Products"})
        SqlConn.Close()
        Dim cmpt As Int32 = 0
        Me.TreeView1.ImageList = Me.ImageList1
        Dim RootNode As TreeNode =
Me.TreeView1.Nodes.Add(cmpt.ToString, "Products", 0)
        cmpt += 1
        For Each Ligne As DataRow In TableColonne.Rows
            Dim ColNode As TreeNode =
RootNode.Nodes.Add(cmpt.ToString, CStr(Ligne("column_name")), 1, 2)
            cmpt += 1
            ColNode.Nodes.Add(cmpt.ToString, "Type = " +
Ligne("data_type").ToString, 3)
            cmpt += 1
            ColNode.Nodes.Add(cmpt.ToString, "IsNullable = " +
Ligne("is_nullable").ToString, 3)
            cmpt += 1
            ColNode.Nodes.Add(cmpt.ToString, "Défaut = " +
Ligne("column_default").ToString, 3)
            cmpt += 1
        Next
    End Using
End Sub

```

Ce qui, après quelques clics sur le TreeView, peut nous donner une sortie écran de ce type :



Nous voulons imprimer cet arbre à peu près tel qu'il est à l'écran.

Analysons un peu la structure de ce que nous souhaitons imprimer. Chaque ligne représentant un nœud ayant des enfants aura un symbole plus/moins puis une icône puis le texte, les autres n'auront que l'icône et le texte. Les traits de liaisons vont de la moitié du bord inférieur de l'icône du père vers la moitié du bord de gauche des icônes fils en passant par le symbole plus/minus le cas échéant.

Nous allons utiliser le code suivant :

```

Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object,
ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles
PrintDocument1.PrintPage
    Dim ImprEnCours As New ArrayList
    Dim Police As New Font("Arial", 11)
    Dim PolicePM As New Font("Arial", 6)
    Using MyGr As Graphics = e.Graphics
        Dim SizePlusMinus As New SizeF(MyGr.MeasureString("+",
PolicePM))
        Dim Interligne As Single = MyGr.MeasureString("X",
Police).Height
        Dim LargIcône As Single = Me.ImageList1.Images(0).Width /
Me.ImageList1.Images(0).HorizontalResolution * 100
        Dim HautIcône As Single = Me.ImageList1.Images(0).Height /
Me.ImageList1.Images(0).VerticalResolution * 100
        AppelTreeViewRec(Me.TreeView1, ImprEnCours)
        Dim XPos, YPos As Single
        Dim MonFormat As New StringFormat()
        MonFormat.Alignment = StringAlignment.Center
        MonFormat.LineAlignment = StringAlignment.Center
        Dim TabYPos(3) As Single
        For cmpt As Int32 = 0 To ImprEnCours.Count - 1
            Dim Elem As Gestion = CType(ImprEnCours.Item(cmpt),
Gestion)

```

```

        XPos = (Elem.Niveau - 1) * (SizePlusMinus.Width + 10 +
(LargIcône - SizePlusMinus.Width) / 2)
        If Elem.HasChild = True Then
            'tracé du caractère plusminus
            If Elem.IsExpand Then
                MyGr.DrawRectangle(Pens.Black, XPos, YPos +
(Interligne - SizePlusMinus.Height) / 2, SizePlusMinus.Width,
SizePlusMinus.Height)
                MyGr.DrawString("-", PolicePM, Brushes.Black, New
RectangleF(XPos, YPos + (Interligne - SizePlusMinus.Height) / 2,
SizePlusMinus.Width, SizePlusMinus.Height), MonFormat)
            Else
                MyGr.DrawRectangle(Pens.Black, XPos, YPos +
(Interligne - SizePlusMinus.Height) / 2, SizePlusMinus.Width,
SizePlusMinus.Height)
                MyGr.DrawString("+", PolicePM, Brushes.Black, New
RectangleF(XPos, YPos + (Interligne - SizePlusMinus.Height) / 2,
SizePlusMinus.Width, SizePlusMinus.Height), MonFormat)
            End If
        End If
        XPos += 10 + SizePlusMinus.Width
        'tracé de l'icône
        MyGr.DrawImage(Me.ImageList1.Images(Elem.IndexImage),
XPos, YPos)
        If Elem.Niveau > 0 Then
            'tracé des lignes
            If Elem.HasChild Then
                MyGr.DrawLine(Pens.Gray, XPos, YPos + HautIcône /
2, XPos - 10, YPos + HautIcône / 2)
            Else
                MyGr.DrawLine(Pens.Gray, XPos, YPos + HautIcône /
2, XPos - 10 - SizePlusMinus.Width / 2, YPos + HautIcône / 2)
            End If
            MyGr.DrawLine(Pens.Gray, XPos - 10 -
SizePlusMinus.Width / 2, YPos + HautIcône / 2, XPos - 10 -
SizePlusMinus.Width / 2, TabYPos(Elem.Niveau - 1))
        End If
        XPos = XPos + 10 + LargIcône
        'impression du texte
        MyGr.DrawString(Elem.Texte, Police, Brushes.Black, XPos,
YPos)
        TabYPos(Elem.Niveau) = YPos + (Interligne -
SizePlusMinus.Height) / 2 + SizePlusMinus.Height
        YPos += Interligne * 2
    Next
End Using

End Sub

Private Sub AppelTreeViewRec(ByVal aTreeView As TreeView, ByRef
TabGestion As ArrayList) 'appel du parcours récursif
    Dim n As TreeNode
    For Each n In aTreeView.Nodes
        Dim EtatNoeud As Gestion
        EtatNoeud.Niveau = 0
        EtatNoeud.Texte = n.Text
        If n.IsSelected Then
            EtatNoeud.IndexImage = n.SelectedImageIndex

```

```

        Else
            EtatNoeud.IndexImage = n.ImageIndex
        End If
        If n.Nodes.Count > 0 Then EtatNoeud.HasChild = True
        If n.IsExpanded AndAlso EtatNoeud.HasChild Then
            EtatNoeud.IsExpand = True
            TabGestion.Add(EtatNoeud)
            Parcours(n, TabGestion)
        Else
            TabGestion.Add(EtatNoeud)
        End If
    Next
End Sub

Private Sub Parcours(ByVal n As TreeNode, ByRef TabGestion As
ArrayList) 'parcours récursif
    Dim aNode As TreeNode
    For Each aNode In n.Nodes
        Dim EtatNoeud As New Gestion
        EtatNoeud.Niveau = aNode.Level
        EtatNoeud.Texte = aNode.Text
        If aNode.IsSelected Then
            EtatNoeud.IndexImage = aNode.SelectedImageIndex
        Else
            EtatNoeud.IndexImage = aNode.ImageIndex
        End If
        If aNode.Nodes.Count > 0 Then EtatNoeud.HasChild = True
        If aNode.IsExpanded AndAlso EtatNoeud.HasChild Then
            EtatNoeud.IsExpand = True
            TabGestion.Add(EtatNoeud)
            Parcours(aNode, TabGestion)
        Else
            TabGestion.Add(EtatNoeud)
        End If
    Next
End Sub

```

Le cheminement est le suivant. Comme vous l'avez peut être remarqué, j'ai implémenté une structure qui va me permettre de modéliser l'arbre tel que je veux qu'il s'imprime. Cette structure a la forme :

```

Private Structure Gestion
    Dim Niveau As Int32
    Dim HasChild As Boolean
    Dim IndexImage As Int32
    Dim IsExpand As Boolean
    Dim Texte As String
End Structure

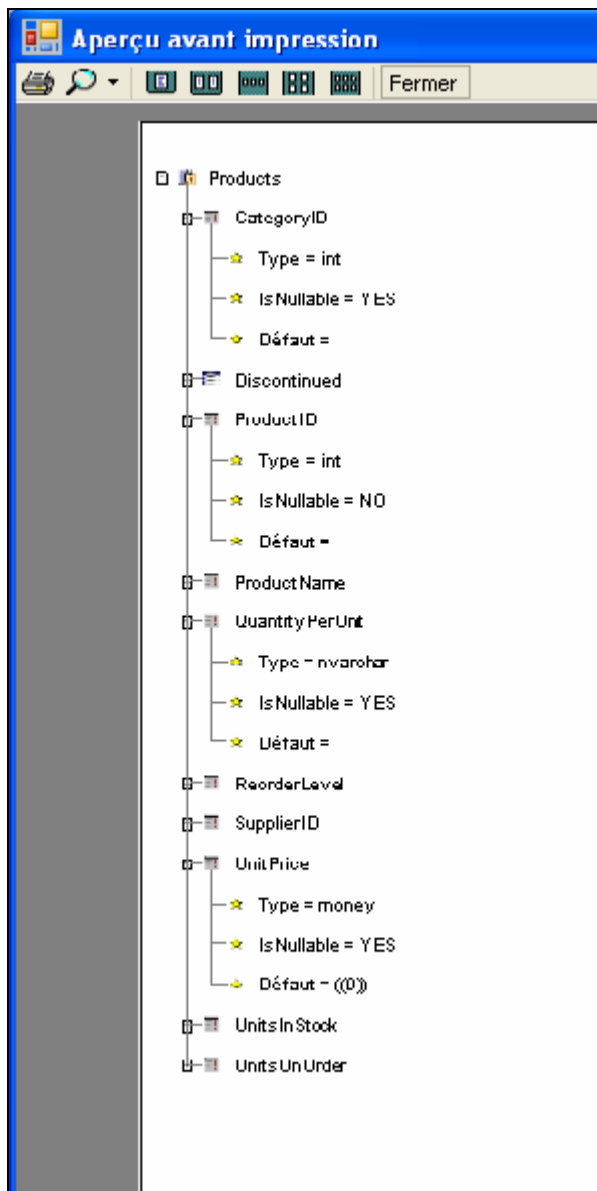
```

Je vais donc stocker le niveau du nœud, s'il a des enfants, ce qui induit l'impression du signe PlusMinus, son image, s'il est déployé et le texte à imprimer.

Pour parcourir un arbre, on utilise une fonction récursive. Pour cela j'utilise donc une procédure AppelTreeViewRec à laquelle je passe un ArrayList que je vais remplir avec des éléments de ma structure au fur et à mesure du parcours de l'arbre.

Lorsque le parcours est terminé, je récupère la liste qui va me permettre d'imprimer. Le code d'impression est assez simple. Je détermine l'indentation en fonction du niveau du nœud, puis j'imprime chaque ligne dans l'ordre c'est-à-dire les caractères PlusMinus le cas échéant, l'icône puis le texte. Enfin, je trace les traits de liaisons. J'obtiendrai alors la sortie suivante :

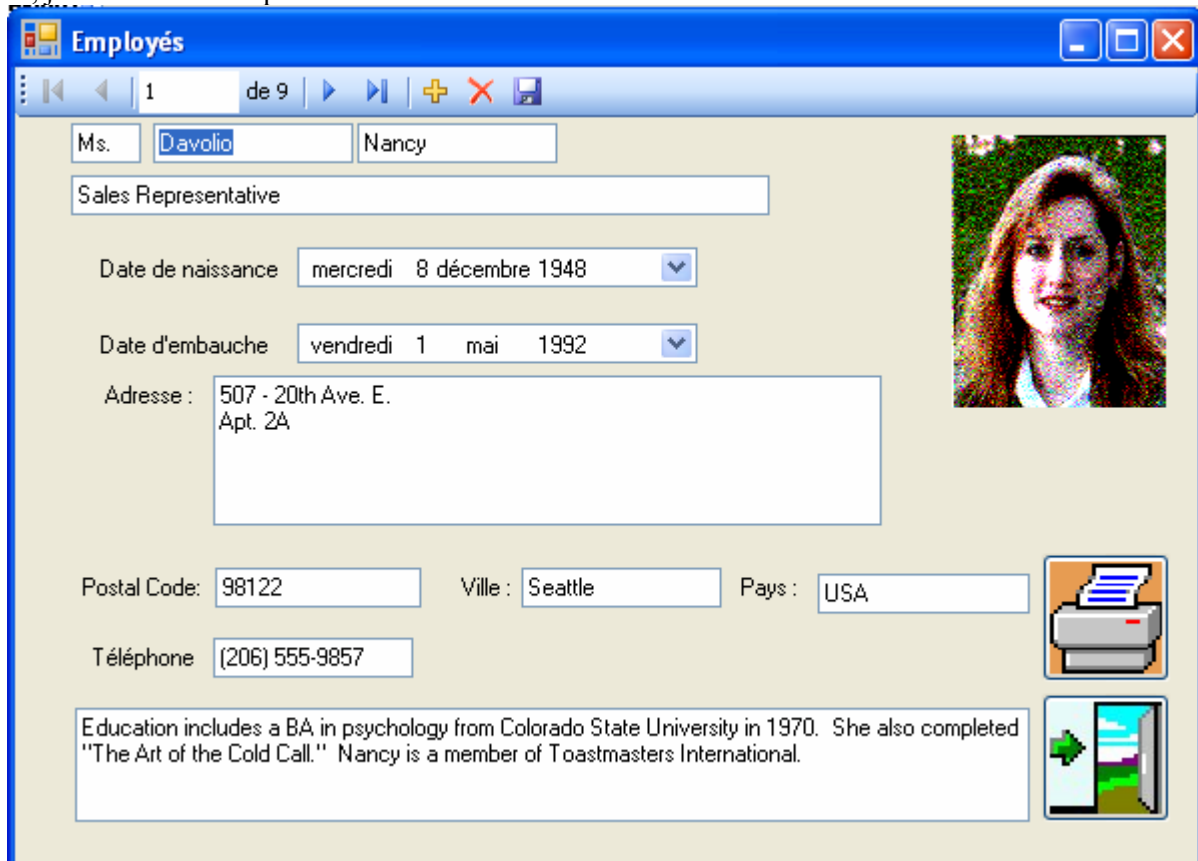




# Impression d'un formulaire

## Exemple 1

Mettons maintenant tout cela en pratique en traitant de l'impression d'un formulaire de base de données. Commençons par un cas standard, l'impression d'un formulaire « Employés » de la base Northwind. A l'écran, j'ai un formulaire qui ressemble à :



The screenshot shows a Windows application window titled "Employés". The window contains a form for an employee named Nancy Davolio. The form fields are as follows:

- Name: Ms. Davolio Nancy
- Title: Sales Representative
- Birth Date: mercredi 8 décembre 1948
- Hire Date: vendredi 1 mai 1992
- Address: 507 - 20th Ave. E. Apt. 2A
- Postal Code: 98122
- City: Seattle
- Country: USA
- Phone: (206) 555-9857

There is a photo of Nancy Davolio on the right side of the form. Below the photo, there is a text box containing the following text: "Education includes a BA in psychology from Colorado State University in 1970. She also completed 'The Art of the Cold Call.' Nancy is a member of Toastmasters International." To the right of the text box, there is a printer icon and a navigation arrow icon.

Évidemment, je ne suis pas tenu de suivre à l'impression la même disposition qu'à l'écran. D'ailleurs dans cet exemple je vais adopter une présentation légèrement différente.

Utilisons le code suivant :

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    With Me.PrintDocument1.DefaultPageSettings
        .Landscape = False
        .Margins = New Margins(50, 50, 50, 50)
        If .PrinterSettings.SupportsColor Then
            .Color = True
        Else
            .Color = False
        End If
    End With
    Me.PrintDocument1.OriginAtMargins = True
    Me.PrintPreviewDialog1.ShowDialog(Me)
End Sub

Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage
    Dim Police As New Font("Arial", 11)
    Dim PoliceG As New Font("Arial", 12, FontStyle.Bold)
    Dim XPos, YPos As Single
```

```

Dim ChaîneImpr As String
Using MyGr As Graphics = e.Graphics
    Dim Photo As New Bitmap(Me.PhotoPictureBox.Image)
    MyGr.DrawImage(Photo, 2.0F * e.MarginBounds.Width / 3.0F, YPos)
    Dim SautLigne As Single = MyGr.MeasureString("X", PoliceG).Height
    ChaîneImpr = Me.TitleOfCourtesyTextBox.Text + " " +
Me.FirstNameTextBox.Text + " " + Me.LastNameTextBox.Text
    MyGr.DrawString(ChaîneImpr, PoliceG, Brushes.Black, XPos, YPos)
    YPos = YPos + SautLigne * 1.5F
    MyGr.DrawRectangle(Pens.Blue, XPos, YPos,
MyGr.MeasureString(Me.TitleTextBox.Text, Police).Width,
MyGr.MeasureString(Me.TitleTextBox.Text, Police).Height)
    MyGr.DrawString(Me.TitleTextBox.Text, Police, Brushes.Blue, New
RectangleF(New PointF(XPos, YPos),
MyGr.MeasureString(Me.TitleTextBox.Text, Police)))
    YPos = YPos + SautLigne * 2.5F
    ChaîneImpr = "Date de naissance : " +
Me.BirthDateDateTimePicker.Value.ToLongDateString
    MyGr.DrawString(ChaîneImpr, Police, Brushes.Black, XPos, YPos)
    YPos = YPos + SautLigne * 1.5F
    ChaîneImpr = "Date d'embauche : " +
Me.HireDateDateTimePicker.Value.ToLongDateString
    MyGr.DrawString(ChaîneImpr, Police, Brushes.Black, XPos, YPos)
    YPos = YPos + SautLigne * 3.0F
    MyGr.DrawString("Adresse : ", Police, Brushes.Black, XPos, YPos)
    ChaîneImpr = Me.AddressTextBox.Text + vbCrLf +
Me.PostalCodeTextBox.Text + vbTab + Me.CityTextBox.Text + vbCrLf +
Me.CountryTextBox.Text
    Dim TailleAdresse As SizeF = MyGr.MeasureString(ChaîneImpr,
Police, CInt(2.0F * e.MarginBounds.Width / 3.0F -
MyGr.MeasureString("Adresse : ", Police).Width - 20.0F), New
StringFormat)
    MyGr.DrawRectangle(Pens.DarkGreen, XPos +
MyGr.MeasureString("Adresse : ", Police).Width, YPos, 2.0F *
e.MarginBounds.Width / 3.0F - MyGr.MeasureString("Adresse : ",
Police).Width - 20.0F, TailleAdresse.Height)
    MyGr.DrawString(ChaîneImpr, Police, Brushes.DarkGreen, New
RectangleF(New PointF(XPos + MyGr.MeasureString("Adresse : ",
Police).Width, YPos), TailleAdresse))
    YPos = YPos + TailleAdresse.Height + SautLigne * 2.0F
    MyGr.DrawString("Téléphone : " + Me.HomePhoneTextBox.Text,
Police, Brushes.Black, XPos, YPos)
    YPos = YPos + SautLigne * 3.0F
    MyGr.DrawString(Me.NotesTextBox.Text, Police, Brushes.DarkCyan,
New RectangleF(XPos, YPos, e.MarginBounds.Width,
MyGr.MeasureString(Me.NotesTextBox.Text, Police,
e.MarginBounds.Width).Height), New StringFormat)
    MyGr.DrawRectangle(Pens.DarkCyan, XPos, YPos,
e.MarginBounds.Width, MyGr.MeasureString(Me.NotesTextBox.Text, Police,
e.MarginBounds.Width).Height)
End Using
End Sub

```

Comme vous le voyez, il s'agit d'un code très simple et assez linéaire.

J'ai modifié l'impression pour regrouper des informations connexes comme le nom et le prénom ou l'ensemble de l'adresse. Nous allons obtenir une sortie qui va ressembler à :

Ms. Nancy Davolio	
<u>Sales Representative</u>	
Date de naissance : mercredi 8 décembre 1948	
Date d'embauche : vendredi 1 mai 1992	
Adresse	
507 - 20th Ave. E. Apt. 2A 98122 Seattle USA	
Téléphone : (206) 555-9857	
<u>Education includes a BA in psychology from Colorado State University in 1970. She also completed "The Art of the Cold Call." Nancy is a member of Toastmasters International.</u>	

Nous pourrions évidemment présenter l'impression sous de nombreuses formes différentes. Si vous avez bien suivi tout ce que nous avons vu jusqu'ici, il ne s'agit jamais que de traiter un problème géométrique assez simple. Le code d'impression utilise rarement plus d'une demi-douzaine de méthodes de l'objet Graphics et la seule question à se poser reste la disposition voulue. Nous avons réalisé une sortie tenant sur une demi page car nous allons enrichir un peu l'exemple. Imaginons maintenant que nous souhaitions imprimer toutes les fiches du personnel avec un seul code.

Nous allons devoir travailler alors non pas sur les contrôles du formulaire mais directement sur le Dataset sous jacent. Ce n'est pas tellement plus compliqué si ce n'est qu'il nous faudra traiter la récupération d'image.

Le code deviendra alors :

```
Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage
    Dim Police As New Font("Arial", 11)
    Dim PoliceG As New Font("Arial", 12, FontStyle.Bold)
    Dim XPos, YPos As Single
    Dim ChaineImpr As String
    Static Cmpt As Int32
    Dim Ligne As NorthwindDataSet.EmployeesRow
    Dim Photo As Bitmap
    Using MyGr As Graphics = e.Graphics
        Do While cmpt < Me.NorthwindDataSet.Employees.Rows.Count
            Ligne = CType(Me.NorthwindDataSet.Employees.Rows(cmpt), NorthwindDataSet.EmployeesRow)
            Dim RecupData() As Byte = Ligne.Photo
            Using Ms As New IO.MemoryStream
                Ms.Write(RecupData, 78, RecupData.Length - 78)
                Photo = New Bitmap(Ms)
                Ms.Close()
            End Using
            If Cmpt Mod 2 = 0 Then
                MyGr.DrawLine(Pens.Gray, XPos, e.MarginBounds.Height / 2.0F + YPos, XPos + e.MarginBounds.Width, e.MarginBounds.Height / 2.0F)
                MyGr.DrawString(Chr(34), New Font("Wingdings", 12, FontStyle.Bold), Brushes.Black, XPos, YPos + e.MarginBounds.Height / 2.0F - MyGr.MeasureString(Chr(34), New Font("Wingdings", 12, FontStyle.Bold)).Height / 2.0F)
            End If
        End While
    End Using
End Sub
```

```

        MyGr.DrawImage(Photo, 2.0F * e.MarginBounds.Width / 3.0F,
YPos, Me.PhotoPictureBox.Image.Width /
Me.PhotoPictureBox.Image.HorizontalResolution * 100,
Me.PhotoPictureBox.Image.Height /
Me.PhotoPictureBox.Image.VerticalResolution * 100)
        Dim SautLigne As Single = MyGr.MeasureString("X",
PoliceG).Height
        ChaîneImpr = Ligne.TitleOfCourtesy + " " + Ligne.FirstName +
" " + Ligne.LastName
        MyGr.DrawString(ChaîneImpr, PoliceG, Brushes.Black, XPos,
YPos)
        YPos = YPos + SautLigne * 1.5F
        MyGr.DrawRectangle(Pens.Blue, XPos, YPos,
MyGr.MeasureString(Ligne.Title, Police).Width,
MyGr.MeasureString(Ligne.Title, Police).Height)
        MyGr.DrawString(Ligne.Title, Police, Brushes.Blue, New
RectangleF(New PointF(XPos, YPos), MyGr.MeasureString(Ligne.Title,
Police)))
        YPos = YPos + SautLigne * 2.5F
        ChaîneImpr = "Date de naissance : " +
Ligne.BirthDate.ToLongDateString
        MyGr.DrawString(ChaîneImpr, Police, Brushes.Black, XPos,
YPos)
        YPos = YPos + SautLigne * 1.5F
        ChaîneImpr = "Date d'embauche : " +
Ligne.HireDate.ToLongDateString
        MyGr.DrawString(ChaîneImpr, Police, Brushes.Black, XPos,
YPos)
        YPos = YPos + SautLigne * 3.0F
        MyGr.DrawString("Adresse : ", Police, Brushes.Black, XPos,
YPos)
        ChaîneImpr = Ligne.Address + vbCrLf + Ligne.PostalCode +
vbTab + Ligne.City + vbCrLf + Ligne.Country
        Dim TailleAdresse As SizeF = MyGr.MeasureString(ChaîneImpr,
Police, CInt(2.0F * e.MarginBounds.Width / 3.0F -
MyGr.MeasureString("Adresse : ", Police).Width - 20.0F), New
StringFormat)
        MyGr.DrawRectangle(Pens.DarkGreen, XPos +
MyGr.MeasureString("Adresse : ", Police).Width, YPos, 2.0F *
e.MarginBounds.Width / 3.0F - MyGr.MeasureString("Adresse : ",
Police).Width - 20.0F, TailleAdresse.Height)
        MyGr.DrawString(ChaîneImpr, Police, Brushes.DarkGreen, New
RectangleF(New PointF(XPos + MyGr.MeasureString("Adresse : ",
Police).Width, YPos), TailleAdresse))
        YPos = YPos + TailleAdresse.Height + SautLigne * 2.0F
        MyGr.DrawString("Téléphone : " + Ligne.HomePhone, Police,
Brushes.Black, XPos, YPos)
        YPos = YPos + SautLigne * 3.0F
        MyGr.DrawString(Ligne.Notes, Police, Brushes.DarkCyan, New
RectangleF(XPos, YPos, e.MarginBounds.Width,
MyGr.MeasureString(Ligne.Notes, Police, e.MarginBounds.Width).Height),
New StringFormat)
        MyGr.DrawRectangle(Pens.DarkCyan, XPos, YPos,
e.MarginBounds.Width, MyGr.MeasureString(Ligne.Notes, Police,
e.MarginBounds.Width).Height)
        Cmppt += 1
        If Cmppt Mod 2 = 0 Then
            YPos = 0

```

```

                e.HasMorePages = True
                Exit Do
            Else
                YPos = e.MarginBounds.Height / 2.0F + e.MarginBounds.Top
+ 50
                e.HasMorePages = False
            End If
        Loop
    End Using
End Sub

```

Je parcours donc les lignes de la table pour récupérer les valeurs du Dataset et accessoirement pour gérer correctement les sauts de pages. Détaillons rapidement quelques éléments de ce code :

Le code suivant permet de convertir le tableau d'octets du champ photo en image

```

Dim RecupData() As Byte = Ligne.Photo
Using Ms As New IO.MemoryStream
    Ms.Write(RecupData, 78, RecupData.Length - 78)
    Photo = New Bitmap(Ms)
    Ms.Close()
End Using

```

78 étant la taille de l'entête d'un bitmap.

Le code suivant trace un trait au milieu de la page avec un symbole de ciseau (oui je sais, ça ne sert à rien mais j'aime bien) :

```

If Cmpt Mod 2 = 0 Then
    MyGr.DrawLine(Pens.Gray, XPos, e.MarginBounds.Height / 2.0F + YPos,
XPos + e.MarginBounds.Width, e.MarginBounds.Height / 2.0F)
    MyGr.DrawString(Chr(34), New Font("Wingdings", 12, FontStyle.Bold),
Brushes.Black, XPos, YPos + e.MarginBounds.Height / 2.0F -
MyGr.MeasureString(Chr(34), New Font("Wingdings", 12,
FontStyle.Bold)).Height / 2.0F)
End If

```

## Exemple 2

Prenons maintenant un autre formulaire beaucoup plus complexe, en terme d'impression s'entend.

Client	OrderDate	Qt	Discount
Ernst Handel	17/07/1996	6	0.2
Rattlesnake Canyon Groc...	25/09/1996	40	0.1
Hungry Owl All-Night Groc...	22/10/1996	6	0.2
Die Wandernde Kuh	22/04/1997	50	0.2
Seven Seas Imports	23/05/1997	24	0.15
Berglunds snabbköp	18/06/1997	10	0.1
HILARION-Abastos	15/10/1997	4	0
Ana Trujillo Emparedados ...	28/11/1997	10	0
Magazzini Alimentari Riuniti	07/01/1998	20	0
Rancho grande	27/02/1998	6	0
Hungry Owl All-Night Groc...	30/03/1998	35	0.1

Dans ce cas, nous avons tout intérêt à gérer une impression assez différente de la sortie écran puisqu'on ne peut pas afficher toute la grille à l'écran. Ce re-zonage est assez courant lorsqu'on fait de l'impression de formulaire. Il faut respecter des règles relativement standards pour le regroupement des informations connexes. Dans notre exemple, nous allons utiliser une page orientée en paysage, le coin supérieur gauche reprendra les informations du produit, le coin inférieur gauche les informations du fournisseur et la partie droite le tableau des commandes. Nous verrons qu'il y aura un petit travail d'adaptation sur le tableau.

Une fois défini notre zonage, il n'y a plus qu'à écrire le code suivant :

```

Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object,
ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles
PrintDocument1.PrintPage
    Dim PoliceS As New Font("Arial", 11)
    Dim PoliceT As New Font("Arial", 10)
    Dim PoliceG As New Font("Arial", 12, FontStyle.Bold)
    Dim XPos, YPos As Single
    Dim Interligne As Single
    Dim ChaineImpr As String = ""
    Using MyGr As Graphics = e.Graphics
        Interligne =
MyGr.MeasureString(Me.ProductNameComboBox.Text.ToString, PoliceG).Height
        'Impression de la partie produit
        MyGr.DrawString(Me.ProductNameComboBox.Text.ToString,
PoliceG, Brushes.DarkBlue, XPos, YPos)
        MyGr.DrawString("Quantité unitaire : " +
Me.QuantityPerUnitTextBox.Text, PoliceS, Brushes.Black, XPos, YPos + 2 *
Interligne)
        MyGr.DrawString("Prix unitaire : " +
Me.UnitPriceNumericUpDown.Value.ToString + " €", PoliceS, Brushes.Black,
XPos, YPos + 4 * Interligne)
    
```



```

        MyGr.DrawString("Stock : " +
Me.UnitsInStockNumericUpDown.Value.ToString, PoliceS, Brushes.Black,
XPos, YPos + 6 * Interligne)
        MyGr.DrawString("Unités commandées : " +
Me.UnitsOnOrderTextBox.Text, PoliceS, Brushes.Black, XPos, YPos + 8 *
Interligne)
        MyGr.DrawString("Niveau bas : " +
Me.ReorderLevelTrackBar.Value.ToString, PoliceS, Brushes.Black, XPos,
YPos + 10 * Interligne)
        'récupération graphique des trackbar et checkBox
        Dim Rect As New Rectangle(New Point(10, 10),
Me.ReorderLevelTrackBar.PreferredSize)
        Dim MonImage As New Bitmap(Rect.Width, Rect.Height)
        Dim BckColor As Color = Me.ReorderLevelTrackBar.BackColor
        Me.ReorderLevelTrackBar.BackColor = Color.White
        Me.ReorderLevelTrackBar.DrawToBitmap(MonImage, Rect)
        Me.ReorderLevelTrackBar.BackColor = BckColor
        MyGr.DrawImage(MonImage, XPos, YPos + 11 * Interligne)
        Rect = New Rectangle(New Point(10, 10),
Me.DiscontinuedCheckBox.Size)
        MonImage = New Bitmap(CInt(Rect.Width / 96.0F * 72.0F *
100.0F), CInt(Rect.Height / 96.0F * 72.0F * 100.0F))
        Me.DiscontinuedCheckBox.BringToFront()
        Me.DiscontinuedCheckBox.BackColor = Color.White
        Me.DiscontinuedCheckBox.DrawToBitmap(MonImage, Rect)
        Me.DiscontinuedCheckBox.BackColor = BckColor
        MyGr.DrawImage(MonImage, XPos, YPos + 13 * Interligne)
        'Impression de la partie fournisseur
        YPos = 2.0F * e.MarginBounds.Height / 3
        MyGr.DrawString("Fournisseur : " +
Me.CompanyNameTextBox.Text, PoliceG, Brushes.DarkGreen, XPos, YPos)
        YPos += Interligne
        ChaineImpr = Me.AddressTextBox.Text + vbCrLf +
Me.PostalCodeTextBox.Text + vbTab + Me.CityTextBox.Text + vbCrLf
        ChaineImpr += Me.CountryTextBox.Text + vbCrLf
        ChaineImpr += "Téléphone : " + Me.PhoneTextBox.Text + vbTab +
"Fax : " + Me.FaxTextBox.Text + vbCrLf
        Dim TailleChaine As SizeF = MyGr.MeasureString(ChaineImpr,
PoliceS, CInt(0.4 * e.MarginBounds.Width))
        MyGr.DrawString(ChaineImpr, PoliceS, Brushes.DarkGreen, New
RectangleF(New PointF(XPos, YPos), TailleChaine))
        MyGr.DrawString(Me.HomePageLinkLabel.Text, New Font("Arial",
11, FontStyle.Underline), Brushes.Blue, XPos, YPos + TailleChaine.Height
+ Interligne)
        MyGr.DrawRectangle(Pens.DarkGreen, XPos, YPos, 0.4F *
e.MarginBounds.Width, TailleChaine.Height + 2 * Interligne)
        'tracé de la grille, doit tenir sur une demi largeur, Wrap
éventuelle sur la première colonne
        XPos = e.MarginBounds.Width / 2.0F
        YPos = 0
        Dim LargCol(3) As Single
        Dim PosCol(3) As Single
        LargCol(3) = MyGr.MeasureString("Remise", PoliceS).Width
        LargCol(2) = MyGr.MeasureString("Quantité", PoliceS).Width
        LargCol(1) = MyGr.MeasureString("00/00/0000", PoliceS).Width
        LargCol(0) = e.MarginBounds.Width / 2.0F - LargCol(1) -
LargCol(2) - LargCol(3)
        PosCol(0) = XPos

```



```

        For cmpt As Int32 = 1 To 3
            PosCol(cmpt) = PosCol(cmpt - 1) + LargCol(cmpt - 1)
        Next
        Dim FormatTitre As New StringFormat
        FormatTitre.Alignment = StringAlignment.Center
        FormatTitre.LineAlignment = StringAlignment.Center
        Dim FormatCase As New StringFormat
        FormatCase.Alignment = StringAlignment.Center
        'impression des titres
        MyGr.FillRectangle(Brushes.Blue, XPos, YPos,
e.MarginBounds.Width / 2.0F, Interligne)
        MyGr.DrawString("Client", PoliceS, Brushes.White, New
RectangleF(PosCol(0), YPos, LargCol(0), Interligne), FormatTitre)
        MyGr.DrawRectangle(Pens.Black, PosCol(0), YPos, LargCol(0),
Interligne)
        MyGr.DrawString("Date", PoliceS, Brushes.White, New
RectangleF(PosCol(1), YPos, LargCol(1), Interligne), FormatTitre)
        MyGr.DrawRectangle(Pens.Black, PosCol(1), YPos, LargCol(1),
Interligne)
        MyGr.DrawString("Quantité", PoliceS, Brushes.White, New
RectangleF(PosCol(2), YPos, LargCol(2), Interligne), FormatTitre)
        MyGr.DrawRectangle(Pens.Black, PosCol(2), YPos, LargCol(2),
Interligne)
        MyGr.DrawString("Remise", PoliceS, Brushes.White, New
RectangleF(PosCol(3), YPos, LargCol(3), Interligne), FormatTitre)
        MyGr.DrawRectangle(Pens.Black, PosCol(3), YPos, LargCol(3),
Interligne)
        YPos += Interligne
        For cmptLigne As Int32 = 0 To
Me.OrdersByProductDataGridView.RowCount - 1
            For cmptCol As Int32 = 0 To 3
                If cmptCol = 0 Then
                    ChaineImpr =
Me.OrdersByProductDataGridView.Item(cmptCol + 1,
cmptLigne).Value.ToString
                    TailleChaine = MyGr.MeasureString(ChaineImpr,
PoliceT, CInt(LargCol(0)))
                    MyGr.DrawString(ChaineImpr, PoliceT,
Brushes.Black, New RectangleF(PosCol(0), YPos, CInt(LargCol(0)),
TailleChaine.Height))
                    MyGr.DrawRectangle(Pens.Black, PosCol(0), YPos,
CInt(LargCol(0)), TailleChaine.Height)
                Else
                    MyGr.DrawString(Me.OrdersByProductDataGridView.Item(cmptCol + 1,
cmptLigne).Value.ToString, PoliceT, Brushes.Black, New
RectangleF(PosCol(cmptCol), YPos, CInt(LargCol(cmptCol)),
TailleChaine.Height))
                    MyGr.DrawRectangle(Pens.Black, PosCol(cmptCol),
YPos, CInt(LargCol(cmptCol)), TailleChaine.Height)
                End If
            Next
            YPos += TailleChaine.Height
        Next
    End Using
End Sub

```

Comme vous le voyez, j'ai utilisé une impression par zone afin que ce soit plus lisible. Il n'y a plus rien dans ce code qui peut vous échapper maintenant, si ce n'est que je fais deux récupérations graphiques pour les contrôles TrackBar et CheckBox.

Par ailleurs, j'ai utilisé une gestion du tableau un peu différente. Dans le cas présent, j'ai défini une taille maximum pour l'ensemble de la grille. Comme la colonne 'Client' peut contenir des noms assez longs, j'ai ajusté les autres colonnes et dimensionné la première pour que la somme des colonnes corresponde à ma largeur maximale. Ensuite, je traite la colonne comme autorisant un saut de ligne éventuel si le texte est plus large que la colonne.

Nous obtiendrons alors une sortie telle que :

**Ma carpepe Fablioi**

Quantité unitaire : 24 - 200 g pks.

Prix unitaire : 32.0000 €

Stock : 9

Unités commandées : 40

Niveau bas : 25

Disponible

Fournisseur : Formaggi Fortini s.r.l.  
 Viale Vanic, 75  
 48100 Ravenna  
 Italy  
 Téléphone : (0544) 60323 Fax : (0544) 60603  
[#FORMAGGI.HTM](#)

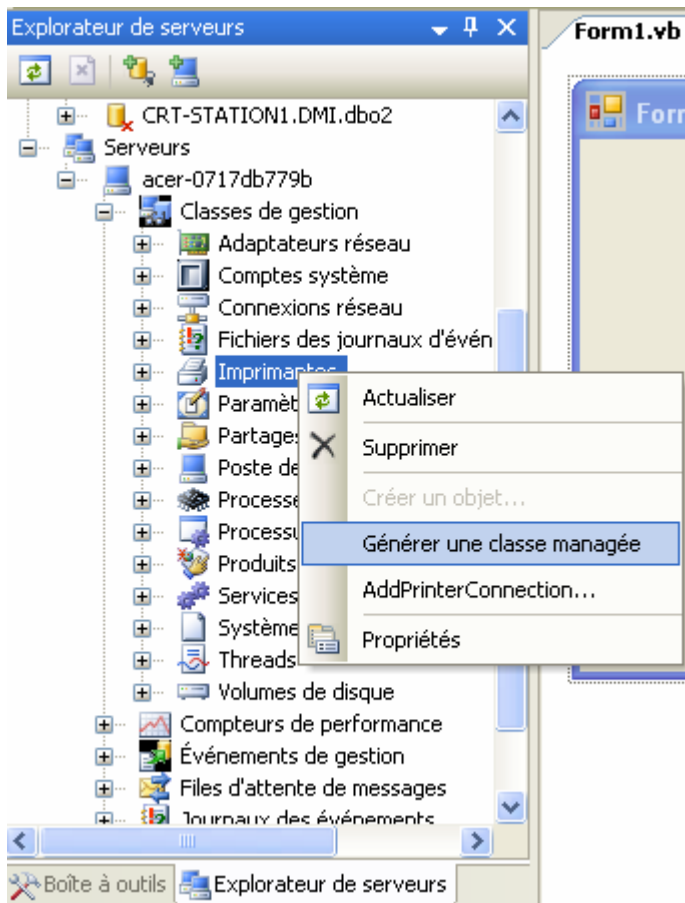
Client	Date	Quantité	Remise
Emis Handel	17/07/1998	5	0.2
Rattlesnake Canyon Grocery	25/09/1998	40	0.1
Hungry Owl All-Night Grocers	22/10/1998	5	0.2
Ole Wandemide Kuh	23/04/1998	50	0.2
Seven Seas Imports	23/05/1998	24	0.15
Berglund's snabbköp	18/06/1998	10	0.1
HTLARIO II-Abas los	15/10/1998	4	0
Pina Trullio Emparedados y helados	28/11/1998	10	0
Magazzini Alimentari Riuniti	07/01/1998	20	0
Rancho grande	27/02/1998	5	0
Hungry Owl All-Night Grocers	30/03/1998	35	0.1
LICA-Supermercado	03/04/1998	50	0
Ole Wandemide Kuh	23/04/1998	15	0.05
Piccolouni mehr	27/04/1998	20	0
Rattlesnake Canyon Grocery	05/05/1998	1	0

Voilà, l'impression n'a maintenant plus de secrets pour nous. Pour finir ce cours sur une petite note technique, nous allons traiter succinctement de la communication bidirectionnelle en temps réel.

## Communiquer avec l'imprimante

Comme nous l'avons vu au début de ce cours, la communication avec l'imprimante fait partie intégrante du processus d'impression. Cependant la communication par le biais des PrinterSettings ne couvre que les informations de mise en page "standard". Or il arrive parfois que l'on souhaite récupérer des informations 'dynamiques' comme l'état de l'imprimante, la liste des travaux en cours ou une notification de travail.

Le Framework DotNet ne gère pas cette communication directement à l'aide des classes d'impression mais il permet toutefois d'y accéder, soit en procédant à des appels de plateforme, soit en pilotant l'instrumentation Windows avec l'espace de nom System.Management.



Vous devez tout d'abord ajouter une référence à cet espace de nom dans votre projet. Ensuite, commençons par récupérer des informations supplémentaires sur l'imprimante choisie. Comme nous sommes un peu fainéants, nous allons faire travailler Visual Studio à notre place. Affichez l'explorateur de serveur, déployer l'arborescence serveur, puis celle de la machine et enfin 'Classes de gestion'. Là, sélectionnez l'élément 'imprimantes', faites un clic droit et sélectionnez 'Générer une classe managée'.

Dans votre explorateur de solution, vous allez voir apparaître un composant intitulé ROOT.CIMV2.WIN32\_Printer. Il s'agit d'un ensemble de classe permettant la gestion WMI des imprimantes.

Attention, le générateur commet deux erreurs que vous allez voir apparaître dans la liste des erreurs si vous travaillez avec Option Strict. En effet il utilise deux fois la notation Value = Nothing alors que vous devriez avoir Value Is Nothing, vous devrez donc faire la correction vous-même.

Maintenant, vous n'avez plus qu'à récupérer les propriétés de l'imprimante qui vous intéresse, comme dans l'exemple suivant :

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim Printers As New ROOT.CIMV2.Printer.PrinterCollection(New Management.Management.ObjectSearcher("SELECT * FROM WIN32_Printer").Get)
    For Each Imprimante As ROOT.CIMV2.Printer In Printers
        Dim TabPg As New TabPage
        Dim MonTrv As New TreeView
        TabPg.Text = Imprimante.Name
        TabPg.Controls.Add(MonTrv)
        MonTrv.Dock = DockStyle.Fill
        Me.TabControl1.Controls.Add(TabPg)
        Dim RootNode As TreeNode = MonTrv.Nodes.Add(Imprimante.Name)
        Dim RecupInfo As String = ""
        If Not Imprimante.IsAttributesNull Then
            With RootNode.Nodes.Add("Attributs")
                For Each Attributs As String In [Enum].Format(GetType(ROOT.CIMV2.Printer.AttributesValues), Imprimante.Attributes, "g").Split(New Char() {"", "c"})
                    .Nodes.Add(Attributs)
                End For
            End With
        End If
    Next
End Sub
```

```

        Next
    End With
End If
If Not Imprimante.IsAvailabilityNull Then
    With RootNode.Nodes.Add("Disponibilité")
        For Each Disponibilite As String In
[Enum].Format(GetType(ROOT.CIMV2.Printer.AvailabilityValues),
Imprimante.Availability, "g").Split(New Char() {"","c"})
            .Nodes.Add(Disponibilite)
        Next
    End With
End If
With RootNode.Nodes.Add("Capacité")
    For Each Capacite As String In Imprimante.CapabilityDescriptions
        .Nodes.Add(Capacite)
    Next
End With
If Not Imprimante.IsConfigManagerErrorCodeNull Then
    With RootNode.Nodes.Add("Erreur ConfigManager")
        For Each ErrorCM As String In
[Enum].Format(GetType(ROOT.CIMV2.Printer.ConfigManagerErrorCodeValues),
Imprimante.ConfigManagerErrorCode, "g").Split(New Char() {"","c"})
            .Nodes.Add(ErrorCM)
        Next
    End With
End If
RootNode.Nodes.Add("Taille Papier actuelle : " +
Imprimante.CurrentPaperType)
RootNode.Nodes.Add("Description : " + Imprimante.Description)
If Not Imprimante.IsDetectedErrorStateNull Then
    With RootNode.Nodes.Add("Erreur détectée")
        .Nodes.Add([Enum].GetName(GetType(ROOT.CIMV2.Printer.DetectedErrorStateValues),
Imprimante.ConfigManagerErrorCode))
    End With
End If
RootNode.Nodes.Add("DeviceID : " + Imprimante.DeviceID)
If Not Imprimante.ErrorDescription Is Nothing Then
    With RootNode.Nodes.Add("Erreur")
        .Nodes.Add(Imprimante.ErrorDescription)
        For Each ErrorDescription As String In
Imprimante.ErrorDescription
            .Nodes.Add(ErrorDescription)
        Next
        For Each ErrorInformation As String In
Imprimante.ErrorInformation
            .Nodes.Add(ErrorInformation)
        Next
        .Nodes.Add([Enum].GetName(GetType(ROOT.CIMV2.Printer.ExtendedDetectedErrorStateV
alues), Imprimante.ExtendedDetectedErrorState))
    End With
End If
If Not Imprimante.IsExtendedPrinterStatusNull Then
    With RootNode.Nodes.Add("Etat étendu")
        .Nodes.Add([Enum].GetName(GetType(ROOT.CIMV2.Printer.ExtendedPrinterStatusValues
), Imprimante.ExtendedPrinterStatus))
    End With
End If
RootNode.Nodes.Add("Résolution : " +
Imprimante.HorizontalResolution.ToString + "*" +
Imprimante.VerticalResolution.ToString)

```

```

        RootNode.Nodes.Add("JobCountSinceLastReset : " +
Imprimante.JobCountSinceLastReset.ToString)
        If Not Imprimante.IsMarkingTechnologyNull Then
            With RootNode.Nodes.Add("Technologie")

.Nodes.Add([Enum].GetName(GetType(ROOT.CIMV2.Printer.MarkingTechnologyValues),
Imprimante.MarkingTechnology))
                End With
            End If
            If Not Imprimante.IsMaxCopiesNull Then RootNode.Nodes.Add("Max copies :
" + Imprimante.MaxCopies.ToString)
            If Not Imprimante.IsMaxSizeSupportedNull Then RootNode.Nodes.Add("Taille
Max : " + Imprimante.MaxSizeSupported.ToString)
            With RootNode.Nodes.Add("Taille papier")
                For Each TaillePapier As String In Imprimante.PrinterPaperNames
                    .Nodes.Add(TaillePapier)
                Next
            End With
            RootNode.Nodes.Add("Port : " + Imprimante.PortName)
            If Not Imprimante.IsPowerManagementSupportedNull Then
                With RootNode.Nodes.Add("Power Management")

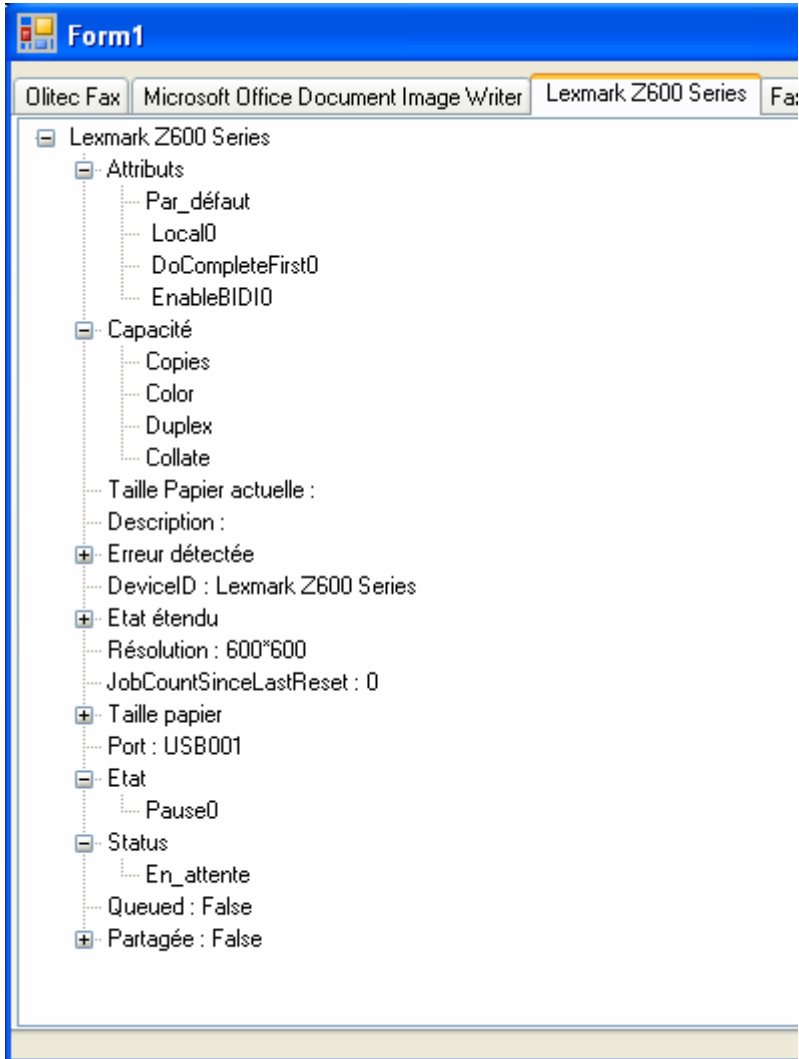
.Nodes.Add([Enum].GetName(GetType(ROOT.CIMV2.Printer.PowerManagementCapabilities
Values), Imprimante.PowerManagementCapabilities))
                    End With
                End If
                If Not Imprimante.IsPrinterStateNull Then
                    With RootNode.Nodes.Add("Etat")

.Nodes.Add([Enum].GetName(GetType(ROOT.CIMV2.Printer.PrinterStateValues),
Imprimante.PrinterState))
                        End With
                    End If
                    If Not Imprimante.IsPrinterStatusNull Then
                        With RootNode.Nodes.Add("Status")

.Nodes.Add([Enum].GetName(GetType(ROOT.CIMV2.Printer.PrinterStatusValues),
Imprimante.PrinterStatus))
                            End With
                        End If
                        RootNode.Nodes.Add("Queued : " + Imprimante.Queued.ToString)
                        If Not Imprimante.IsSharedNull Then
                            With RootNode.Nodes.Add("Partagée : " + Imprimante.Shared.ToString)
                                .Nodes.Add(Imprimante.ShareName)
                            End With
                        End If
                    Next
                End Sub

```

Nous obtiendrons alors une sortie de la forme :



Maintenant, nous voulons aussi pouvoir récupérer des informations sur les travaux d'impressions en cours. Là encore, nous allons utiliser WMI pour récupérer ces informations. Cependant la classe que nous avons créée ne permet pas d'atteindre les travaux de l'imprimante. Nous allons donc utiliser WMI de manière plus dynamique, tel que :

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

    Dim RequJob As New ManagementObjectSearcher("SELECT * FROM
Win32_PrintJob")
    Dim Jobs As ManagementObjectCollection = RequJob.Get()
    Dim Job As ManagementObject
    For Each Job In Jobs
        If Not Job("Caption") Is Nothing Then
Me.ListBox1.Items.Add("Nom: " + Job("Caption").ToString)
        If Not Job("Document") Is Nothing Then
Me.ListBox1.Items.Add("Document: " + Job("Document").ToString)
        If Not Job("JobId") Is Nothing Then
Me.ListBox1.Items.Add("JobId: " + Job("JobId").ToString())
        If Not Job("JobStatus") Is Nothing Then
Me.ListBox1.Items.Add("JobStatus: " + Job("JobStatus").ToString)
        Me.ListBox1.Items.Add("Notify: " + Job("Notify").ToString)
        Me.ListBox1.Items.Add("Pages imprimées: " + _
Job("PagesPrinted").ToString())
    End For
End Sub
```

```

        Me.ListBox1.Items.Add("Priorité: " +
Job("Priority").ToString().ToString)
        Me.ListBox1.Items.Add("Taille: " +
Job("Size").ToString().ToString)
        Me.ListBox1.Items.Add("Status: " + Job("Status").ToString)
        Me.ListBox1.Items.Add("Nombre de Pages: " +
Job("TotalPages").ToString().ToString)
        Next
End Sub

```

Nous obtiendrons alors les informations telles que :

```

Nom: Lexmark Z600 Series, 3
Document: Microsoft Word - impression.doc
JobId: 3
JobStatus: Impression en cours
Notify: jmarc
Pages imprimées: 0
Priorité: 1
Taille: 56469795
Status: OK
Nombre de Pages: 58

```

Il est possible d'aller encore plus loin dans la communication mais cela ne présente pas souvent d'intérêt puisque le système donne accès à la plupart des informations sur ses tâches d'impression par l'accès au Gestionnaire d'impression dans le SysTray.

## Conclusion

Voilà, nous avons vu ensemble les techniques d'impression les plus souvent utilisées. Il existe bien sur des cas un peu plus complexe lorsqu'on souhaite imprimer des textes avec un angle, ou pour construire des images complexes, mais cela dépasse le cadre de ce document.

J'espère vous avoir montré que l'impression n'est ni complexe ni très longue dès lors qu'on connaît les trucs et astuces du parfait petit imprimeur.