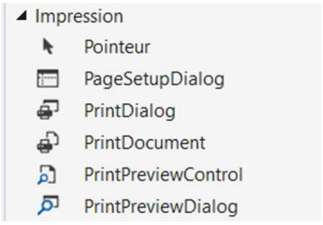
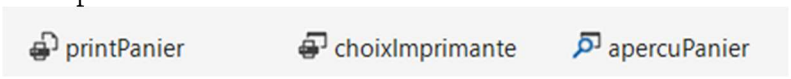
1. Présentation de l'impression

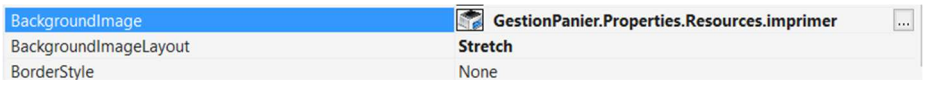


L'impression s'effectue à l'aide d'un composant non visuel PrintDocument
Sa propriété événementielle PrintPage est chargée de générer l'image (sous forme graphique) de la page à imprimer. Elle utilise un objet graphique passé en paramètre
Si l'on souhaite pouvoir sélectionner l'imprimante, on ajoute un composant PrintDialog (nommez le 'choixImprimante' de préférence)
Si l'on souhaite visualiser un aperçu, on ajoute le composant non visuel PrintPreviewDialog.

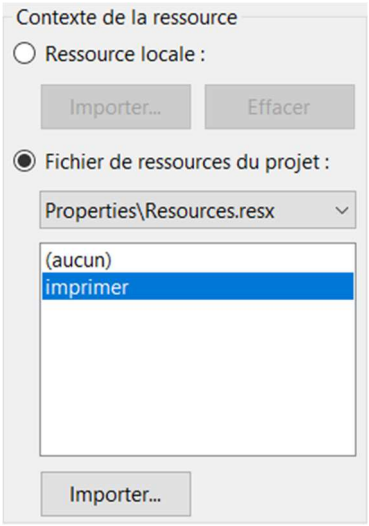
Exemple :



Pour lancer l'impression, on place dans la fenêtre un composant pictureBox (ou un bouton) qui déclenchera l'impression sur son événement clic
Un composant pictureBox est plus pratique pour avoir une image qui s'adapte à la taille du composant



L'image est intégrée dans le projet.



L'événement clic doit essentiellement appeler la méthode Print du composant PrintDocument
Si on utilise un composant PrintDialog, il doit avant appeler la méthode ShowDialog() de ce composant pour pouvoir sélectionner l'imprimante.

```
private void imgImprimer_Click(object sender, EventArgs e)  {
      printPanier.DocumentName = "Panier";
       printPanier.DefaultPageSettings.Landscape = false;   // mode portrait
      choixImprimante.Document = printPanier;
      DialogResult result = choixImprimante.ShowDialog();
      if (result == DialogResult.OK)
            printPanier.Print();
}
```

Si on passe par l'aperçu, la propriété Document du composant PrintPreviewDialog doit être lié au composant PrintDocument gérant l'impression du document.
Il suffit alors d'appeler la méthode ShowDialog du composant pour obtenir l'aperçu :

```
private void imgImprimer_Click(object sender, EventArgs e)       {
      choixImprimante.Document = printPanier;
      apercuPanier.Document = printPanier;
      apercuPanier.WindowState = FormWindowState.Maximized;
      apercuPanier.ShowDialog();
```

La méthode Print va déclencher la procédure évènementielle PrintPage qui réalise concrètement l'impression. **Cette procédure est appelée pour chaque page imprimée**.

```
private void printPanier_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{…}
```

Pour dessiner les éléments composant la page, on dispose d'un grand nombre de méthodes applicables sur l'objet 'e' passé en paramètre. L'objet e représente en quelque sorte la page.

Pour afficher un texte :

e.Graphics.**DrawString**(texteAImprimer, unePolice, uneCouleur, unRectangle, unFormat);

Pour afficher un trait

e.Graphics.**DrawLine**(styleTrait, unPoint, point2);

Pour afficher un rectangle

e.Graphics.DrawRectangle(styleTrait, unRectangle);

Si le texte doit être aligné, il faut l'imprimer dans un cadre.

Pour tracer un rectangle sur la page :

```
Rectangle unRectangle = new Rectangle(x, y, largeur, hauteur);  // x, y coin supérieur gauche
Pen styleTrait = new Pen(Color.Black, 1);
e.Graphics.DrawRectangle(styleTrait, unRectangle);
```

La méthode DrawRectangle accepte la surcharge suivante :

```
e.Graphics.DrawRectangle(styleTrait, x, y, largeur, hauteur);
```

Pour dessiner du texte dans un cadre :

```
e.Graphics.DrawString(texteAImprimer, unePolice, unStyle, unRectangle, unFormat);
```

Pour définir la police

```
Font unePolice = new Font("Courier New", 16);
```

Pour définir le style

```
SolidBrush unStyle = new SolidBrush(Color.Black);
```

Pour définir le format

```
StringFormat unFormat = new StringFormat();
unFormat.Alignment = StringAlignment.Center;  // alignement horizontal
unFormat.LineAlignment = StringAlignment.Center; // alignement vertical
```

Pour dessiner un trait

```
styleTrait = new Pen(Color.Black, 3);
Point point1 = new Point(150, 100);
Point point2 = new Point(600, 100);
e.Graphics.DrawLine(styleTrait, point1, point2);
```

Valeurs possibles pour StringAlignement : far (à droite), Near (à gauche) et center

2. Impression d'un fichier texte sans aucune mise en forme

Le contenu du fichier est transféré dans une chaîne de caractères.
C'est l'impression la plus simple car on dispose d'une méthode MeasureString qui permet de récupérer le nombre de caractères qui peuvent être imprimés sur la page en fonction des paramètres de style utilisé et du contenu à imprimer
De cette façon quand on imprime une page, on retire du contenu à imprimer ce qui a réellement été imprimé et on redemande une impression s'il reste des caractères à imprimer

- private void printTexte_PrintPage(…) {

```
int charactersOnPage = 0;  // nombre de caractères sur une page
int linesPerPage = 0;

// calcul de la valeur de charactersOnPage et de linesPerPage
e.Graphics.MeasureString(texteAImprimer, this.Font, e.MarginBounds.Size,
StringFormat.GenericTypographic, out charactersOnPage, out linesPerPage);

// Représente graphiquement la page à imprimer
e.Graphics.DrawString(texteAImprimer, this.Font, Brushes.Black, e.MarginBounds,
StringFormat.GenericTypographic);

// Retire la partie imprimée du texte à imprimer
texteAImprimer = texteAImprimer.Substring(charactersOnPage);

// Détermine si une autre page doit être imprimer (boucle virtuelle)
e.HasMorePages = (texteAImprimer.Length > 0);
```

- private void btnImprimer_Click(object sender, EventArgs e) {

```
printTexte.DocumentName = "Impression d'un texte sans mise en forme";
choixImprimante.Document = printTexte;
DialogResult result = choixImprimante.ShowDialog();
if (result == DialogResult.OK) {
    texteAImprimer = rtbFichier.Text;   // le composant rtb contient le fichier texte à imprimer
    printTexte.Print();
}
```

La méthode printTexte est relancée automatiquement si e.HasMorePages vaut true.
Il ne faut surtout pas initialiser la variable texteAImprimer dans cette procédure sinon on risque d'obtenir une boucle sans fin.

3. Impression avec mise en forme d'un fichier Xml tenant sur une page

- private void printXml_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)

```
…
// dessine le cadre du titre de la page
e.Graphics.DrawRectangle(styleTrait, x, y, largeur, hauteur);
// affiche le titre centré dans le cadre
StringFormat unFormat = new StringFormat();
unFormat.Alignment = StringAlignment.Center;
unFormat.LineAlignment = StringAlignment.Center;
texteAImprimer = " Liste des étudiants ";
e.Graphics.DrawString(texteAImprimer, unePolice, unStyle, unRectangle, unFormat);

// affiche l'entête du tableau
// Chaque colonne est associée à un rectangle pour permettre un alignement de son contenu
unePolice = new Font("Courier New", 10);
// colonne id
x = 150.0F;
y = 70.0F;
largeur = 30.0F;
hauteur = 20.0F;
texteAImprimer = "Id";
unRectangle = new RectangleF(x, y, largeur, hauteur);
e.Graphics.DrawString(texteAImprimer, unePolice, unStyle, unRectangle, unFormat);
// autre colonne
// Impression des lignes à partir de la lecture du fichier XML etudiant
XmlDocument unDocument = new XmlDocument();
unDocument.Load("etudiant.xml");
XmlNodeList lesEtudiants = unDocument.GetElementsByTagName("etudiant");
foreach (XmlNode unEtudiant in lesEtudiants) {
point1 += new Size(0, 10);
largeur = 30.0F;
unRectangle = new RectangleF(point1.X, point1.Y, largeur, hauteur);
e.Graphics.DrawString(unEtudiant.Attributes[0].InnerText, unePolice, unStyle, unRectangle,
unFormat);
point1 += new Size(30,0);
e.Graphics.DrawString(unEtudiant.ChildNodes[0].InnerText, unePolice, unStyle, point1);
point1 += new Size(150,0);
e.Graphics.DrawString(unEtudiant.ChildNodes[1].InnerText, unePolice, unStyle, point1);
point1 += new Size(150,0);
    // on centre l'option
    largeur = 100.0F;
    unRectangle = new RectangleF(point1.X, point1.Y, largeur, hauteur);
e.Graphics.DrawString(unEtudiant.ChildNodes[2].InnerText, unePolice, unStyle, unRectangle,
unFormat);
    point1 += new Size(-330, 15);
 }
```

4. Impression d'un fichier Xml sur plusieurs pages

La difficulté réside dans la gestion d'une impression sur plusieurs pages. Il faut être capable d'imprimer le contenu de chaque page en utilisant toujours la même procédure.
Pour rappeler cette procédure il suffit de placer à vrai la propriété **HasMorePages** de l'objet e

Dans ce cas on ne peut pas utiliser la boucle foreach car il faut que le rappel de la méthode imprime les données de la deuxième page (le foreach commence toujours par la première ligne) puis la troisième.

Il faut donc utiliser une boucle while avec un itérateur **i déclaré au niveau global** pour conserver sa valeur à chaque appel de la méthode.
Pour savoir si on a atteint la fin d'une page il faut tester la position y atteinte dans la page et la comparer à la position qui doit déclencher le saut de page (taille de la page – marge)

- Déclaration de la variable globale i

```
static int i = 0;
```

- private void btnFichierXml2_Click(object sender, EventArgs e)

```
            printXml.DocumentName = "Etudiant 2";
            choixImprimante.Document = printXml;
            DialogResult result = choixImprimante.ShowDialog();
            if (result == DialogResult.OK)    {
               // initialisation du numéro de ligne
               i = 0;
               printXml2.Print();
            }
```

- private void printXml2_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)

```
// par défaut on considère qu'il n'y aura pas une autre page
e.HasMorePages = false;
…
while (i < lesEtudiants.Count)     {
  XmlNode unEtudiant = lesEtudiants[i];
  point1 += new Size(0, 10);
  // impression de la ligne
        …..
  // on se place au début de la ligne suivante
  point1 += new Size(-330, 15);
  i++;
  // a-t 'on atteint la fin de la page
  if (point1.Y >= e.PageBounds.Height - e.PageSettings.Margins.Top )   {
      e.HasMorePages = true; // on prévoit de relancer la procédure
     // il faut arrêter la procédure en cours
    return;
  }
}
```