Gestion de dossiers et fichiers en C#

(J-C Armici, 2006)

D'autres documents et exemples concernant la programmation, sur les sites: www.unvrai.com, www.unfaux.com, jca.developpez.com et www.developpez.com

Ce document basé sur le Framework .NET 2.0 et Visual Studio 2005.

Ce document contient quelques informations concernant la gestion des fichiers, des dossiers, des unités de disques, ainsi que la lecture et l'écriture de fichiers textes.

Informations concernant Fichiers, Dossiers et Disques

Voici un tableau des principaux membres des classes Files, Directories et Drives :

Membre	**S'applique à**
Exists	FileInfo et DirectoryInfo
Attributes	FileInfo et DirectoryInfo
CreationTime, LastAccessTime, LastWriteTime	FileInfo et DirectoryInfo
FullName, Name, Extension	FileInfo et DirectoryInfo
IsReadOnly	FileInfo
Length	FileInfo
DirectoryName, Directory	FileInfo
Parent, Root	DirectoryInfo
CreateSubdirectory	DirectoryInfo
GetDirectories	DirectoryInfo
GetFiles	DirectoryInfo
DriveType	DriveInfo
AvailableFreeSpace	DriveInfo
GetDrives	DriveInfo

Voici un tableau des méthodes permettant de manipuler des objets de type FileInfo :

Méthode	**Description**
CopyTo	Copie de fichiers
Create, CreateText	Diverses créations de fichiers
Open, OpenRead, OpenText, OpenWrite	Ouverture de fichiers
Delete	Suppression d'un fichier (s'il existe)
Encrypt, Decrypt	Crypte et décrypte un fichier (compte de l'utilisateur courant). Pour NTFS seulement
MoveTo	Déplacement d'un fichier
Replace	Remplacement du contenu d'un fichier

Voici un tableau des méthodes permettant de manipuler des objets de type DirectoryInfo :

Méthode	**Description**
Create	Création d'un dossier (et éventuellement des dossiers intermédiaires s'ils n'existent pas)
CreateSubdirectory	Crée un sous-dossier du dossier représenté par le dossier courant (DirectoryInfo)
Delete	Suppression d'un dossier (existe en version récursive pour supprimer plusieurs dossiers en cascade)
MoveTo	Déplace un dossier et son contenu. Permet aussi de renommer un dossier

Parcours des fichiers d'un dossier

Le fragment de programme ci-dessous permet de parcourir tous les fichiers qui se trouvent dans un dossier :

```
DirectoryInfo dir = new DirectoryInfo("d:\temp");
FileInfo[] finfo = dir.GetFiles();

foreach (FileInfo fi in finfo)
{
  ListeFichiers.Items.Add(fi.FullName);
}
```

Voici un exemple de code permettant de créer un fichier texte et d'y écrire la liste des fichiers d'un dossier. En plus du nom du fichier figurera également sa taille ainsi que la date/heure de son dernier accès:

```
StreamWriter sw = new StreamWriter(txtDest.Text, false, Encoding.Default);
DirectoryInfo dir = new DirectoryInfo("d:\temp");

FileInfo[] finfo = dir.GetFiles();          // fichiers du dossier dir
string ligne;

sw.WriteLine("Structure du dossier " + "d:\temp" + ":");    // titre
foreach (FileInfo fi in finfo)
{
  ligne = string.Format ("{0,15} bytes  ", fi.Length) +
          fi.LastAccessTime.ToShortDateString() + "   " + fi.FullName;
  sw.WriteLine (ligne);                  // écriture des infos
}
sw.Close();
```

Parcours récursif d'une arborescence de fichiers

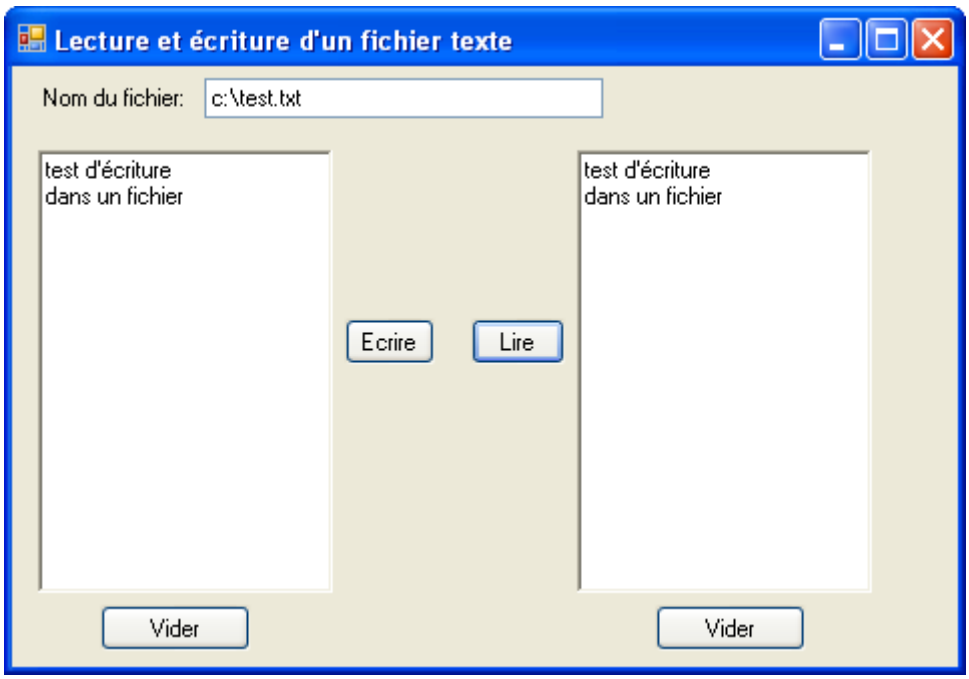
Le programme qui suit permet un parcours récursif du contenu d'un dossier :

```
public static void ListFolder(DirectoryInfo dir)
{
  Console.Write(dir.FullName);
  FileInfo[] fis = dir.GetFiles();   // récupération de la liste des fichiers
  foreach (FileInfo fi in fis)       // parcours des fichiers
    Console.WriteLine(fi.FullName);
  // parcours des sous-dossiers du dossier
  DirectoryInfo[] dirs = dir.GetDirectories();
  foreach (DirectoryInfo subdir in dirs)
    ListFolder(subdir);
}


static void Main(string[] args)
{
  // si aucun chemin n'est passé en paramètre on prend le dossier courant
  if (args.Length == 0)
    args = new string[] { "." };

  // parcours de tous les dossiers passés en ligne de commandes
  foreach (string dir in args)
    ListFolder(new DirectoryInfo(dir));

  Console.ReadLine();
}
```

Lecture et écriture d'un fichier texte

En .NET il est possible de lire et d'écrire dans un fichier texte en utilisant des *streams* : **StreamReader** et **StreamWRiter**. Les méthodes **Write**, **WriteLine**, **Read** et **ReadLine** sont les plus courantes. Elles supportent plusieurs encodages de caractères : ASCII, Unicode (UTF-16), UTF-7, UTF-8,…

Voici un exemple de programme qui lit et écrit du texte dans un fichier texte :



Dans cet exemple nous avons utilisé deux RichTextBox en tant que conteneurs de texte. Bien entendu ce composant dispose de méthodes permettant de lire et d'écrire le contenu de et vers un fichier texte en une seule opération. Nous les utilisons ici uniquement pour travailler sur plusieurs lignes.

Voici le code associé au bouton **Ecrire** :

```
private void btnEcrire_Click(object sender, EventArgs e)
{
  FileStream fs = new FileStream(txtNom.Text, FileMode.Create);
  StreamWriter sw = new StreamWriter(fs, Encoding.UTF8);

  for (int i=0; i < rtEntree.Lines.Length; i++){
    sw.WriteLine(rtEntree.Lines[i]);
  }
  sw.Close();
  fs.Close();
}
```

Ainsi que le code associé au bouton **Lire** :

```
private void btnLire_Click(object sender, EventArgs e)
{
  FileStream fs = new FileStream(txtNom.Text, FileMode.Open);
  StreamReader sr = new StreamReader(fs, Encoding.UTF8);
  string ligne;

  while ((ligne = sr.ReadLine()) != null)
  {
    rtSortie.AppendText(ligne+"\r\n");
  }

  sr.Close();
  fs.Close();
}
```