BD et interopérabilité (SQL - Reformulation et C#)

François Boisson
francois.boisson@gmail.com

ESILV – S06

(2018 - 2019)

ESILV LÉONARD DE VINCI | ÉCOLE D'INGÉNIEURS PARIS-LA DÉFENSE

Les fichiers

Les fichiers stockent l'information de manière persistante

- Un fichier est identifié par un chemin, un nom et un type
- Les données contenues dans un fichier sont organisées
- L'organisation dépend des données (texte, image, son)

Le type (ou extension) donne une indication sur la nature (et l'organisation) des données stockées dans le fichier.
Elle associe le fichier avec un programme qui sait le lire (reconnait l'organisation interne du fichier).

- texte brut (txt)
- document text formaté (doc,docx,odf,tex ...)
- image (jpg,png,gif ...)
- musique (wav,mp3 ...)
- video (avi,vob,mp4 ...)

On s'interresse ici aux fichiers texte (.txt ... ou .CSV)

lisible avec : bloc-notes, WordPad, Notepad et autres ...

Les fichiers txt

- Ces fichiers contiennent l'information sous forme de texte brut
- Le nom complet du fichier (chemin, nom et type) qui sera enregistré dans un *string*
  par exemple pour le fichier c :\temp\fichier.txt

Code

```
1 string nomFichier = "C:\\temp\\fichier.txt" ;
```

- On lira ou écrira une ligne à la fois du début à la fin du fichier
- Chaque ligne lue ou écrite sera contenue dans un *string*
- Les entrées et sorties de données depuis ou vers le fichier constituent un flux de données
- Pour lire ou écrire un flux de données, on utilise de nouveaux *type* d'objets (des classes *StreamWriter et StreamReader*)

Ecriture de fichiers txt (1)

- on définit le nom du fichier (chemin+nom+extension)
- on génére un objet flux de sortie *StreamWriter* vers le fichier
- on écrit dans le flux avec la méthode WriteLine de l'objet flux (comme vers l'écran (la console) avec Console.WriteLine)
- à la fin des écritures, on ferme le fichier (le flux de sortie)

Code

```
1 string nomFich = "C:\\temp\\test.txt" ;
2 StreamWriter fichEcr = new StreamWriter (nomFich,true
    ) ;
3 fichEcr.WriteLine ("1ere ligne") ; // ECRIRE UNE
    LIGNE
4 fichEcr.WriteLine ("1ere ligne") ;
5 \\ etc...
6 fichEcr.Close() ; // ON FERME LE FICHIER
```

ATTENTION :

Ecriture de fichiers txt (2)

Code

```
1 string nomFich = "C:\\temp\\test.txt" ;
2 StreamWriter fichEcr = new StreamWriter (nomFich,true
      ) ;
3 fichEcr.WriteLine ("1ere ligne) ;
4 fichEcr.WriteLine ("1ere ligne) ;
5 \\ etc...
6 fichEcr.Close() ;
```

Attention :

- StreamWriter est une classe => S en majuscule
- fichierSortie est un objet de type StreamWriter donc il propose des méthodes préécrites (par exemple WriteLine)

- ne pas oublier Close() sinon le fichier ne sera pas enregistré

Lecture de fichiers txt (1)

- on définit le nom du fichier (chemin+nom+extension)
- on génére un objet flux de lecture *StreamReader* vers ce fichier
- on lit dans le flux avec la méthode Readline() de l'objet flux (comme depuis le clavier (la console) avec Console.ReadLine())
- à la fin des lectures, on ferme le fichier (le flux de lecture)

Code

```
1 string nomFich = "C:\\tp\\test.txt" ;
2 StreamReader fichLect = new StreamReader (nomFich) ;
3 string ligne = "";
4 While (fichLect.Peek() > 0)
5 {
6   ligne = fichLect.ReadLine() ; // LECTURE D'UNE
        LIGNE
7   Console.WriteLine( "Lu : "+ ligne);
}
```

Lecture de fichiers txt (2)

Code

```
1 string nomFich = "C:\\temp\\test.txt" ;
2 StreamReader fichLect = new StreamReader (nomFich) ;
3 string ligne = "";
4 While (fichLect.Peek() > 0)
5 {
6   ligne = fichLect.ReadLine() ; // LECTURE D'UNE
      LIGNE
7   Console.WriteLine ( "Lu : "+ ligne);
8 }
9 fichLect.Close() ; // ON FERME LE FICHIER
```

- StreamReader est une classe => S en majuscule
- fichLect est un objet de type StreamReader donc il propose des méthodes préécrites (par exemple ReadLine(), Peek())

Compléments

IMPORTANT

- File, StreamWriter, StreamReader font partie du Namespace System.IO

↯ il faut vérifier la présence en en haut du programme (ou l'ajouter) de la ligne de code : using System.IO ;

Les fichiers csv

- Fichiers texte avec séparation des informations par des ' ;'
- Organisation des données dans le fichier csv ligne par ligne : data1 ; data1 ; data3
- Lisibles par les éditeurs de texte comme tout fichier texte
- Reconnu en lecture/ecriture par Excel qui sépare les informations d'une ligne du fichier dans une ligne de cellules

un exemple

Cotillard ;Marion ;37 ;45765 ;rue des vagues ;Lyon
Rochefort ;Jean ;70 ;56678 ;rue des perdrix ;Marseille
Blier ;Bernard ;68 ;33457 ;rue Leonard ;Paris

	A	B	C	D	E	F
1	Cotillard	Marion	37	45765	rue des vagues	Lyon
2	Rochefort	Jean	70	56678	rue des perdrix	Marseille
3	Blier	Bernard	68	33457	rue Leonard	Paris

Ecriture de fichiers csv

- Ce sont des fichiers texte qui s'écrivent donc comme des fichiers texte
- Cconstruction de la ligne à écrire dans le fichier en intercalant un ' ;' entre chacune des données (data1, data2, etc..) qui constituent une ligne du fichier
- Puis on écrit la ligne sur le flux (comme pour les fichiers texte)

Code

```
1 string nomFich = "C:\\temp\\clients.csv" ;
2 StreamWriter fichEcr =
3                     new StreamWriter (nomFich,true) ;
4 string ligne = "Cotillard" + ";" + "Marion" + ";" +
    37 + ";" + 45795 + ";" + "rue des vagues" + ";" +
     "Lyon";
5 fichEcr.WriteLine (ligne) ; // ECRIRE UNE LIGNE
6 \\ etc...
```

Lecture de fichiers csv

- Ce sont des fichiers texte qui se lisent donc comme des fichiers texte
- Chaque lecture dans le fichier lit un string composé d'une série de data séparées par des ' ;'
- Pour séparer ces informations on utilise une méthode des objets string : la méthode Split
- string string.Split(char[]) : On passe en paramètre un tableau de char qui correspondant à la liste des séparateurs à utiliser (ici uniquement le ' ;')

Lecture de fichiers csv

Code

```
1  string nomFich = "C:\\tp\\clients.csv" ;
2  StreamReader fichLect = new StreamReader (nomFich) ;
3  char[] sep = new char[1] {'1'};
4  string ligne = "";
5  string[] datas = new string[6];
6  While (fichLect.Peek() > 0)
7  {
8    ligne = fichLect.ReadLine() ; // LECTURE D'UNE
       LIGNE
9    Console.WriteLine( "ligne lue : "+ ligne);
10   datas = ligne.Split(sep);
11 }
12 etc...
```

La classe File

Une classe d'utilitaires

La classe File (1)

La classe File : des méthodes utiles

- Existence d'un fichier => bool File.Exists (string nomFich)
- Supprimer fichier => void File.Delete (string nomFich)
- Copier fichier => void File.Copy (string source, string dest)
- Deplacer fichier => void File.Move (string source, string dest)

Code

```
1 if ( ! File.Exists ("c:\\Mes Douments\\monFichier.txt
    ") )
2       Console.WriteLine ( "Fichier inexistant") ;
```

- et aussi l'accès aux paramètres des fichiers : attributs, date création, date de dernier accès etc ...

La classe File (2)

La classe File : c'est aussi
Des méthodes d'instanciation d'objet de controle de flux de données (StreamReader, SreamWriter)

- Lecture de fichier
  StreamReader fichLect = File.OpenText (nomFich) ;
- Ecriture de fichier
  StreamWriter fichEcr = File.CreateText (nomFich) ;
  StreamWriter fichEcr = File.AppendText (nomFich) ;
- si nomFich existe déjà, il sera écrasé par CreateText

Code

```
1 StreamReader fichLect = File.OpenText(nomFichier)
2 \\
3 StreamWriter fichEcr = File.CreateText(nomFichier)
4 StreamWriter fichEcr = File.AppendText(nomFichier)
```

Exemple

Code

```
1  static void AppendFichier(string nomfich, string
       ligne)
2  {
3    StreamWriter ecriture = new StreamWriter(nomfich,
         true);
4    ecriture.WriteLine(ligne);
5    ecriture.Close();
6  }
7  static void Main(string[] args)
8  {
9    string fichier = "c:\\temp\\message.txt";
10   string ligne = "";
11   if (File.Exists(fichier)) File.Delete(fichier);
12   while (ligne != "fin")
13   {
```

La classe File (3)

pour plus d'informations :

- voir l'**aide contextuelle de VisualStudio**
- ou/et la documentation de référence de Microsoft C#
  https ://msdn.microsoft.com/fr-fr/library/618ayhy6.aspx