(Parte 9)

Esta parte do documento C# Essencial dá início ao tratamento de Ficheiros.

Sumário:

Figuras:

Considerações

Quando de fala em manipular ficheiros entramos no contexto do chamado “acesso a dados” que, na terminologia .NET poderá abarcar essencialmente Ficheiros e Serialização, XML e Bases de Dados.

Em C# a manipulação de ficheiros não é directa (como acontece em C, Pascal e outras linguagens), ie, operações de escrita e leitura (I/O) são realizadas através de *streams*.

O NameSpace a utilizar para manipular ficheiros é *System.IO*.

Patterns para manipulação de Ficheiros

Manipular um ficheiro implica conseguir criá-lo, definir o tipo de conteúdo que vai possuir e o tipo de acesso que se pretende. Um ficheiro surge em formato texto ou formato binário (embora efectivamente todos os ficheiros sejam guardados em formato binário!). Na prática, num ficheiro de texto, caso se pretenda guardar o valor 42, é guardado o ASCII equivalente, ie, 0x34 e 0x32. No formato binário é usado o valor 42. Qualquer editor de texto consegue apresentar em pleno um ficheiro de texto.

Ficheiros de Texto

Path do ficheiro

- *Indicar o nome e local (a path) do ficheiro:*

```
string Filename = @"C:\Documents and Settings\Employees.spr";
```

ou

```
string Filename = "C:\\Documents and Settings\\Employees.spr";
```

- *Directoria actual*

```
//actual directoria de trabalho
Directory.GetCurrentDirectory();

//directoria onde a aplicação está a executar
Application.StartupPath

//definir nova directoria actual
Directory.SetCurrentDirectory( Application.StartupPath );
```

Criar um ficheiro de texto para escrita:

Classe *StreamWriter* e método *File.CreateText()*

```
        private void SaveText(string text, string file)
        {
            StreamWriter sw = null;
            try
            {
                sw = File.CreateText(file);
            }
            catch (Exception e)
            {
                MessageBox.Show(e.Message);
            }
            finally
            {
                if (sw != null)
                    sw.Close();
            }
        }
```

Exemplo de evocação: *SaveText("aula.txt","Isto é um teste");*

Outra forma de escrever:

```
StreamWriter sw = File.CreateText("aula.txt");
sw.WriteLine("Hello file");
```

Outra forma:

```
TextWriter tw = new StreamWriter("teste.txt");
```

Abrir o ficheiro em modo *append*:

```
StreamWriter asw = new StreamWriter("teste.txt", true);
```

Nota:

A classe *StreamWriter* é adequada para gravar em ficheiros de texto. Caso se pretenda gravar em ficheiros HTML existe o *HtmlTextWriter*.

Abrir um ficheiro para leitura:

Classe *StreamReader* e método *File.OpenTex()t*

```
  public static string ReadFromFile(string filename)
  {
    string s="",aux="";
    try
```

```
    {
      using (StreamReader sr = File.OpenText(filename))
      {
        while ((aux=sr.ReadLine()) != null)
          {
                s += aux + "\r\n";
          }
        sr.Close();
      }
    }
    catch (Exception e)
    {
      throw new FileNotFoundException("Problema ..." + e.Message);
    }
  return s;
  }
```

Outra forma de abrir um ficheiro poderá ser feita com:

```
FileStream fs = new FileStream(fileName, FileMode.Open);
using (StreamReader sr = new StreamReader(fs))
{
      … …
}
```

O exemplo seguinte mostra uma possível implementação do comando Type:

```
        /// <summary>
        /// Mostra conteúdo do ficheiro
        /// </summary>
        /// <param name="fileName">Nome do ficheiro</param>
        public static void Type(string fileName)
        {
            try
            {
                using (StreamReader sr = new StreamReader(fileName))
                {
                    String line;
                    while ((line = sr.ReadLine()) != null)
                    {
                        Console.WriteLine(line);
                    }
                }
            }
            catch (Exception e)
            {

                Console.WriteLine("Ficheiro desconhecido.."+ e.Message);
            }
        }
```

Ficheiro Binários

Criar um ficheiro binário para escrita:

Classe *FileStream* e método *File.Create ()*

```
      private void SaveBinary( byte[] bytes, string file )
       {
           FileStream fs = null;
           try
           {
               if( File.Exists( file ) )
               {
                   File.Delete(file);
               }
               fs = File.Create( file );
               fs.Write( bytes, 0, bytes.Length );
           }
           catch( Exception e )
           {
               MessageBox.Show( e.Message );
           }
           finally
           {
               if( fs != null )
                   fs.Close();
           }
       }
```

Existem outras formas de conseguir o mesmo. A classe *FleStream* permite definir de forma explícita o tipo de acesso que se permite ter ao ficheiro. Veja-se o seguinte exemplo:

```
public static void SaveToFile(string fileName, string t)
 {
   if (File.Exists(fileName))          //se existe, append
   {
     using (StreamWriter sw = new StreamWriter(fileName))
     {
        sw.WriteLine(t);
       sw.Close();
     }
   }
   else                                //senão, cria e grava
   {
     try
     {
       FileStream fs = new FileStream(fileName, FileMode.Append);
      using (StreamWriter sw = new StreamWriter(fs)) {
       sw.WriteLine(t);
       sw.Close();
      }
     }
     catch (FileNotFoundException e)
     {
```

```
            //return erro!
            throw new FileNotFoundException("Problema..." + e.Message);
        }
    }
}
```

Existem várias formas de criar instâncias da classe FileStream:

```
//abre ficheiro para escrita
FileStream fs = File.OpenWrite(fileName);

//abre ficheiro para leitira
FileStream fs = File.OpenRead(fileName);

//cria ficheiro consoante o modo pretendido
FileStream fs = new FileStream(fileName, FileMode,FileAccess, FileShare);
```

Onde os enumerados possíveis são definidos por:

FileMode	Descrição
Create	*FileMode.Create* cria um ficheiro e se existir faz overrite
CreateNew	Cria um ficheiro e se existe gera uma excepção *IOException* .
Append	Abre um ficheiro e prepara para escrever no final. Se o ficheiro não existe, cria-o. Só pode usar *FileMode.Append* com *FileAccess.Write* senão é gerada a excepção *ArgumentException*.
Open	Abre um ficheiro. Se não existir gera a excepção *FileNotFoundException*.
OpenOrCreate	Abre um ficheiro. Se não existir, cria-o.
Truncate	Abre um ficheiro e “limpa-o”.

FileAccess	Descrição
Read	Define acesso para leitura
Write	Define acesso para escrita
ReadWrite	Define acesso para escrita/leitura

O Enumerado *FileShare* sai do âmbito deste documento

Verificar se um ficheiro existe:

Independentemente do formato do ficheiro, verificar se o ficheiro existe ou não é conseguido com o método *File.Exists()*.

Como já vimos no exemplo anterior:

```
if (File.Exists("C:\\aula.txt"))
{
    ……
}
```

Outros métodos disponíveis em *File.IO*

File.FileExists(filename)	*true* se existir
File.Delete(filename)	Apaga o ficheiro
File.AppendText(String)	Acrescenta texto ao final do ficheiro
File.Copy(fromFile, toFile)	Copia o ficheiro
File.Move(fromTile, toFile)	Move o ficheiro, apagando o original
File.GetExtension(filename)	Devolve a extensão do ficheiro
File.HasExtension(filename)	True se o ficheiro tem extensão

As classe *FileInfo*, *Directory* e *DirectoryInfo* possuem vários objectos para manipular as propriedades de directorias e ficheiros.

Manipular ficheiros em Windows Forms

Para manipular directorias e ficheiros os objectos mais comuns são o *OpenFileDialog* e *FolderBrowserDialog*.

Quando se utiliza o *OpenFileDialog* é necessário definir o filtro (extensões de ficheiros), no sentido de filtrar os ficheiros que se pretendem. A sintaxe é:

[Texto] [Lista de extensões separadas por “;”]

Exemplo:

```
openFileDialog1.Filter = "Word (*.doc) |*.doc;*.rtf|Text (*.txt) |*.txt|All (*.*) |*.*";
```

No exemplo seguinte, uma textbox é preecnhida com o nome do ficheiro seleccionado.

```
      private void button1_Click(object sender, EventArgs e)
        {
```

```
            OpenFileDialog dlg = new OpenFileDialog();
            dlg.Title = "Ficheiro a ler...";
            dlg.DefaultExt = "txt";
            dlg.InitialDirectory=@"c:\temp";
            dlg.Filter = "Word (*.doc) |*.doc;*.rtf|Text (*.txt) |*.txt|All
(*.*) |*.*";
            dlg.FilterIndex = 2 ;
            dlg.RestoreDirectory = true;
            dlg.FileName = "";
            if (dlg.ShowDialog() == DialogResult.OK)
            {
                textBox1.Text = dlg.FileName;
            }

        }
```

Figura 1 – Botão "Browse"

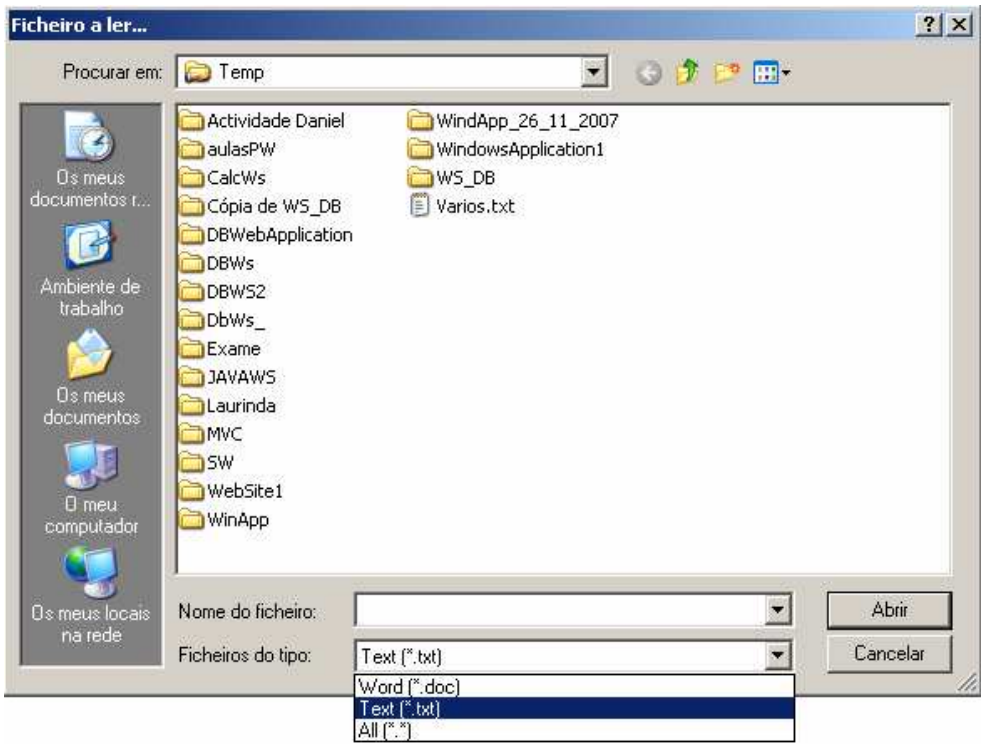

Figura 2 – Controlo OpenFileDialog

Para manipular directorias utiliza-se o controlo *FolderBrowserDialog*. Veja-se o exemplo seguinte:

```
    private void button2_Click(object sender, EventArgs e)
      {
          FolderBrowserDialog fb = new FolderBrowserDialog();
          if (fb.ShowDialog() == DialogResult.OK)
          {
```

```
                MessageBox.Show(fb.SelectedPath);
            }
        }
```

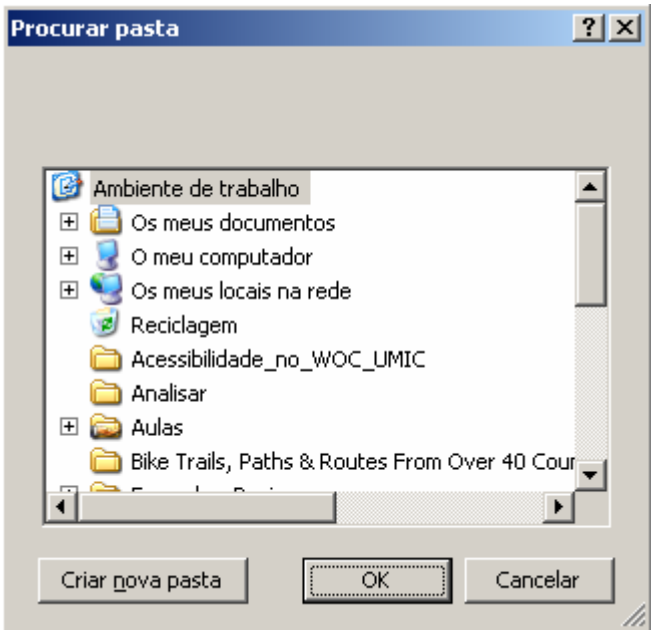

Figura 3 – Controlo FolderBrowserDialog

Exemplos

Método que lê um ficheiro em modo binário

```
/// <summary>
/// Ler de um ficheiro em modo binário
/// </summary>
/// <param name="filename">Nome do Ficheiro</param>
/// <returns>Conteúdo do ficheiro</returns>
public static string ReadFromFileB(string fileName)
{
   string s = "";
   try
   {
     FileStream fs = new FileStream(fileName, FileMode.Open);
     using (BinaryReader br = new BinaryReader(fs,Encoding.Unicode))
     {
        s=br.ReadString();
        br.Close();
     }
   }
   catch (Exception e)
   {
     //return erro!
     throw new FileNotFoundException("Problema..." + e.Message);
```

```
    }
    return s;
  }
```

Método que grava em modo binário:

```
/// <summary>
/// Grava num ficheiro em modo Binário
/// </summary>
/// <param name="fileName">Nome do ficheiro</param>
public static void SaveToFileB(string fileName, string t)
{
   if (File.Exists(fileName))          //se existe, append
   {
       FileStream fs = new FileStream(fileName, FileMode.Create);
       using (BinaryWriter sw = new BinaryWriter(fs,Encoding.Unicode))
       {
       sw.Write(t);
       sw.Close();
       }
   }
   else                                //senão, cria e grava
  {
   try
   {
       FileStream fs = new FileStream(fileName, FileMode.Append);
       using (BinaryWriter sw = new BinaryWriter(fs,Encoding.Unicode))
       {
            sw.Write(t);
            sw.Close();
       }
   }
  catch (FileNotFoundException e)
   {
       //return erro!
       throw new FileNotFoundException("Problema..." + e.Message);
    }
  }
}
```

Guardar um conjunto de valores para ficheiro

A classe *Exercicio* possui um vector de inteiros que se propõe guardar em ficheiro. O primeiro registo do ficheiro tem o número total de elementos guardados.

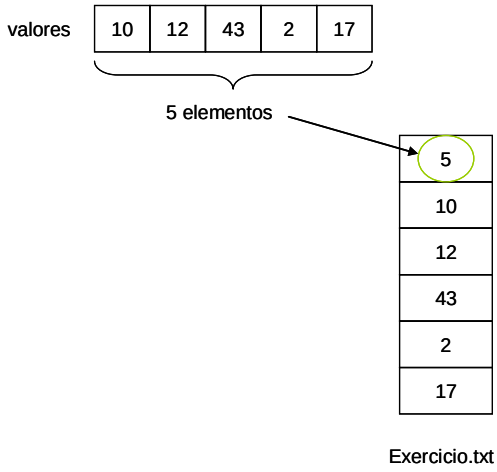

Figura 4 – Exercício

```
//IPCA-EST
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace Aula_06_05_2008
{
    /// <summary>
    /// Guardar um conjunto de inteiros para ficheiro
    /// </summary>
    class Exercicio
    {
        //atributos
        public int[] valores;
        private int tot;

        /// <summary>
        /// construtor
        /// </summary>
        /// <param name="n"></param>
        public Exercicio(int n)
        {
            valores = new int[n];
            tot = n;
        }

        /// <summary>
        /// Carrega vector com valores
        /// </summary>
        public void AddVals()
        {
            for (int i = 0; i < tot; i++)
                valores[i] = i;
        }

        /// <summary>
        /// Guarda todos os valores para filcheiro
        /// Na 1ª posição do ficheiro guarda o total de elementos a gravar
        /// </summary>
        /// <param name="fileName">Nome do ficheiro</param>
        public void SaveValores(string fileName)
        {
```

```
            FileStream fs = new FileStream(fileName, FileMode.Create);
            BinaryWriter fb = new BinaryWriter(fs);

            if (valores.Length == 0)
                fb.Write(0);
            else
            {
                //primeira linha tem a dimensão do vector
                fb.Write(valores.Length);
                foreach (int v in valores)
                    fb.Write(v);
            }
            fb.Close();
        }

        /// <summary>
        /// Carrega todos os valores existente en ficheiro
        /// O primeiro valor corresponde ao número total de elementos
        /// existentes no ficheiro
        /// </summary>
        /// <param name="fileName"></param>
        public void LoadVals(string fileName)
        {
            valores = null;
            FileStream fs = new FileStream(fileName, FileMode.Open);
            BinaryReader fb = new BinaryReader(fs);

            //ler a primeira linha=tamanho do vector
            tot=fb.ReadInt32();
            //tot = fb.Read();
            valores = new int[tot];

            for (int i = 0; i < tot; i++)
            {
                valores[i] = fb.ReadInt32();
                //valores[i] = fb.Read();
            }

            fb.Close();
        }

        /// <summary>
        /// Método Auxiliar para mostrar o conteúdo do vector
        /// </summary>
        public void ShowVals()
        {
            foreach (int v in valores)
                Console.WriteLine(v);
        }
    }
}
```

```
private void button1_Click(object sender, EventArgs e)
        {
            Exercicio ex = new Exercicio(10);
            ex.AddVals();
            ex.SaveValores("Exercicio.txt");
            ex.LoadVals("Exercicio.txt");
            ex.ShowVals();
        }
```

O exemplo seguinte mostra a aplicação de vários métodos da classe File.IO. (adaptado de http://msdn.microsoft.com/en-us/library/system.io.file.aspx)

```
    class Test
    {
    public static void Main()
    {
        string path = @"c:\temp\MyTest.txt";
        if (!File.Exists(path))
        {
            // cria ficheiro
            using (StreamWriter sw = File.CreateText(path))
            {
                sw.WriteLine("Viva");
                sw.WriteLine("o");
                sw.WriteLine("Benfica");
            }
        }

        // Abre o ficheiro para leitura
        using (StreamReader sr = File.OpenText(path))
        {
            string s = "";
            while ((s = sr.ReadLine()) != null)
            {
                Console.WriteLine(s);
            }
        }

        try
        {
            string path2 = path + "temp";
            // garantir que o destino não existe
            File.Delete(path2);

            // copia o ficheiro
            File.Copy(path, path2);
            Console.WriteLine("{0} copiado para {1}.", path, path2);

            // apaga o ficheiro novo criado
            File.Delete(path2);
            Console.WriteLine("{0} foi apagado com sucesso.", path2);
        }
        catch (Exception e)
        {
            Console.WriteLine("Algo correu mal: {0}", e.ToString());
        }
    }
}
```

Referências

- C# School – Programmers Heaven (cf. Site da Disciplina)

- Microsoft® Visual C#® .NET 2003 Developer's Cookbook, ISBN : 0-672-32580-2
- http://www.functionx.com/vcsharp/fileprocessing/Lesson04.htm

continua

lufer, jcsilva, ajtavares, marco