Curso C# Conceitos Básicos

Diogo Cezar Teixeira Batista

xgordo@gmail.com

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
UTFPR-CP

Cornélio Procópio - 2008

Agenda I

Manipulação de Arquivos

Definição

Utilizado para: criar, editar e excluir arquivos ou diretórios.

- Pacote `System.IO`;

As principais classes que estão nesse pacote

Tabela: Principais classes do System.IO

Classe	Uso
Directory, File, DirectoryInfo, e FileInfo	Cria, exclui e move arquivos e diretórios. Ainda retorna informações específicas sobre arquivos ou diretórios
FileStream	Usado para escrever/ler informações em arquivo com ajuda das classes StreamReader e StreamWriter
StreamWriter e StreamReader	Lê e escreve um informação textual
StringReader e StringWriter	Lê e escreve um informação textual a partir de um *buffer* de string

Classes DirectoryInfo e FileInfo

Definição

Utilizadas para recuperar informações de arquivos e diretórios.

- *DirectoryInfo* usado para extrair informações de diretórios;
- *FileInfo* usado para extrair informações de arquivos.

Propriedades e métodos que essas classes oferecem

Tabela: Propriedades e métodos de *DirectoryInfo* e *FileInfo*

Propriedade/Método	Uso
Attributes	Retorna os atributos associados aos arquivos
CreationTime	Retorna a hora de criação do arquivo
Exists	Checa se o arquivo/diretório existe
Extension	Retorna a extensão do arquivo
LastAccessTime	Retorna a hora do último acesso
FullName	Retorna o nome completo do arquivo/diretório
LastWriteTime	Retorna a hora da ultima escrita no arquivo/diretório
Name	Retorna o nome do arquivo/diretório
Delete()	Exclui o arquivo/diretório

Criando diretórios e subdiretórios

Para criar um diretório utiliza-se a seguinte notação:

Código 1: Criação de diretório

```
1  DirectoryInfo dir1 = new DirectoryInfo(@"F:\WINNT");
```

Para criar um subdiretório:

Código 2: Criação de subdiretórios

```
1  DirectoryInfo dir = new DirectoryInfo(@"F:\WINNT");
2  try{
3    dir.CreateSubdirectory("Sub");
4    dir.CreateSubdirectory(@"Sub\MySub");
5  }
6  catch(IOException e){
7    Console.WriteLine(e.Message);
8  }
```

Acessando as propriedades

Para acessar as propriedades de um diretório utiliza-se a seguinte notação:

Código 3: Propriedades de um diretório

```
1  Console.WriteLine("Full Name is : {0}", dir1.FullName);
2  Console.WriteLine("Attributes are : {0}", dir1.Attributes.
       ToString());
```

Criando diretórios e subdiretórios

Exemplo de acesso às propriedades de arquivos:

Código 4: Propriedades de arquivos

```
1  DirectoryInfo dir = new DirectoryInfo(@"F:\WINNT");
2  FileInfo[] bmpfiles = dir.GetFiles("*.bmp");
3  Console.WriteLine("Total number of bmp files", bmpfiles.
       Length);
4  Foreach( FileInfo f in bmpfiles)
5  {
6    Console.WriteLine("Name is : {0}", f.Name);
7    Console.WriteLine("Length of the file is : {0}", f.Length)
         ;
8    Console.WriteLine("Creation time is : {0}", f.CreationTime
         );
9    Console.WriteLine("Attributes of the file are : {0}",
10                     f.Attributes.ToString());
11 }
```

Criando arquivos usando a classe FileInfo

Definição

Com a classe *FileInfo*, é possível criar novos arquivos, acessar suas informações, exclui-los e move-los.

- Oferece métodos para abrir, ler e escrever um arquivo.

Criando arquivos com a classe FileInfo

O seguinte exemplo mostra como é possível criar um arquivo texto e acessar suas informações.

Código 5: Criando arquivos com a classe FileInfo

```
1  FileInfo fi = new FileInfo(@"F:\Myprogram.txt");
2  FileStream fstr = fi.Create();
3  Console.WriteLine("Creation Time: {0}",fi.CreationTime);
4  Console.WriteLine("Full Name: {0}",fi.FullName);
5  Console.WriteLine("FileAttributes: {0}",fi.Attributes.
       ToString());
6  //Way to delete Myprogram.txt file.
7  Console.WriteLine("Press any key to delete the file");
8  Console.Read();
9  fstr.Close();
10 fi.Delete();
```

Método Open()

Definição

Abre um arquivo

- Deve-se passar no construtor, o modo de abertura e acesso ao arquivo.

Exemplo Método Open()

O seguinte exemplo ilustra essa situação:

Código 6: Abrindo arquivos com a classe FileInfo

```
1  FileInfo f = new FileInfo("c:\myfile.txt");
2  FileStream s = f.Open(FileMode.OpenorWrite, FileAccess.Read)
       ;
```

Entendendo a classe FileStream I

Definição

Ao abrir ou criar arquivos, o atribuímos para a classe *FileStream*. Ela pode

- Escreve ou lê arquivos, com a ajuda das classes *StreamWriter* e *StreamReader*.

Exemplo FileStream

Código 7: Escrevendo/Lendo com FileStream

```
1  FileStream fs = new FileStream(@"c:\mcb.txt", FileMode.
       OpenOrCreate, FileAccess.Write);
2  StreamWriter sw = new StreamWriter(fs);
3  sw.write("teste");
4  sw.writeLine("teste");
5  sw.Close();
6
7  FileStream fs = new FileStream(@"c:\mcb.txt", FileMode.
       OpenOrCreate, FileAccess.Write);
8  StreamReader sr = new StreamReader(fs);
9  string texto;
10 texto = sr.ReadToEnd();
11 sr.Close();
```

Métodos CreateText() e OpenText()

- O método *CreateText()* retorna um *StreamWriter* que escreve um arquivo.
- O método *OpenText()* retorna um *StreamReader* que lê um arquivo.
- Esses métodos são utilizados quando trabalha-se com arquivos de texto puro.

Exemplo CreateText() e OpenText()

Código 8: CreateText e OpenText

```
1  FileInfo fi = new FileInfo("c:\myfile.txt");
2  StreamReader txtR;
3  txtR = fi.OpenText();
4  string read = null;
5  while ((read = txtR.ReadLine()) != null){
6    Console.WriteLine(read);
7  }
8  s.Close();
9  // Método ReadToEnd();
10 Console.WriteLine(txtR.ReadToEnd());
11
12 FileInfo fi = new FileInfo("c:\myfile.txt");
13 StreamWriter txtW;
14 txtW = fi.CreateText();
15 txtW.Write("teste");
16 txtW.Close();
```

Exemplos Codificados

- Para fixar os conceitos, vamos estudar os seguintes exemplos:
  1. ManipulacaoArquivos: Utiliza as classes *StreamWriter* e *StreamReader* puras para criar um mini editor de textos.
  2. BlocoDeNotas: Utiliza a classe FileInfo em conjunto com as classes *StreamWriter* e *StreamReader* para criar um editor de texto completo.

Exercícios para praticar

1. Crie um aplicativo que receba um nome e um telefone e o armazene em linhas diferentes em um arquivo texto.
2. Crie um aplicativo que salve o conteúdo de uma *listBox* em um arquivo.
   - A localização do arquivo deve ser feita pelos *Dialogs* vistos nos exemplos.

Exercícios para entregar

- Exercício para entregar:
  1. Recupere o seu programa de cadastro de carros e implemente persistência de dados em um arquivo texto. O programa deve automaticamente salvar e recuperar os carros ao ser iniciado (recuperar) ou fechado (salvar);
  2. A partir do modelo de agenda (moodle) implemente as seguintes funcionalidades:
     - Pesquisar: deve pesquisar por um nome cadastrado, caso ache o elemento, o programa deve seleciona-lo e mostrar as suas informações;
     - Excluir: deve-se excluir o nome selecionado. Esse nome também deve ser excluído do arquivo texto.
     - Novo: deve-se limpar os dados, e preparar o programa para um novo cadastro;
     - Ordenar (**Desafio**): Faça um método que ordene os nomes e os salve de forma ordenada. Não é permitido utilizar o método Sort(); da lista de objetos.