Manipulação de Arquivos Aplicativos / Utilitários

Linguagem de Programação / Técnicas de Programação

Prof. Antonio Marcos Alvarez

Projetos para Manipulação - Arquivos

Usaremos “.NET” (Incluiremos em nossas soluções):

```
using System.IO;
```

System.IO (Namespace)

Faz parte do “.NET”, contendo diversos recursos desenvolvidos para manipulação, operações e suporte básico aos diretórios (pastas) e principalmente arquivos e fluxo de dados, com classes (atributos e métodos) que podem ser usadas para READ / WRITE dados.

System.IO (Namespace) – Classes (Algumas)

File: Fornece métodos estáticos para a criação, cópia, exclusão, deslocamento e abertura de um arquivo, além de ajudar na criação de objetos “FileStream”.

Disponibiliza operações com um determinado arquivo sem a necessidade de instancia-lo, porém, realiza analise de segurança a cada ação (operação) que está efetuando em seu objeto (Arquivo), seu uso é adequado em arquivos simples, menos complexos e dinâmicos, com baixo nível de operações a serem efetuadas, não sendo tão prejudicial a perca de performance.

Link: https://docs.microsoft.com/pt-br/dotnet/api/system.io.file?view=net-5.0

FileInfo: Fornece propriedades e métodos de instância para a criação, cópia, exclusão, deslocamento e abertura de arquivos, além de ajudar na criação de objetos “FileStream”. Essa classe não pode ser herdada.

Disponibiliza operações com um determinado arquivo com a necessidade de instancia-lo, porém, não haverá perca de performance ao utilizar diversos recursos (operações), já que este método não avalia constantemente o arquivo (objeto), somente quando for determinado.

Link: https://docs.microsoft.com/pt-br/dotnet/api/system.io.fileinfo?view=net-5.0

FileStream: Fornece um Stream (fluxo de dados) para um arquivo, dando suporte a operações de leitura e gravação síncronas e assíncronas.

Encapsula recursos de IO (Input/Output) “read”, “write”, “update” e “delete” de um determinado arquivo.

Link: https://docs.microsoft.com/pt-br/dotnet/api/system.io.filestream?view=net-5.0

“Stream” no pé da letra significa “córrego” ou “riacho”, por este motivo damos ênfase ao “streaming” no processo de FLUXO, no caso em TI, de DADOS/INFORMAÇÃO. Podemos associar também, a outros serviços, como exemplo o “Microsoft Stream” é um serviço corporativo de compartilhamento de vídeo.

System.IO (Namespace) – Classes (Algumas)

IOException: Classe “base” para exceções (erros e falhas) geradas ao acessar informações usando fluxos de dados, arquivos e diretórios. A exceção é gerada quando ocorre um erro de IO. A biblioteca de classes inclui os seguintes “tipos”, cada um dos quais é uma classe derivada de ***IOException*** (praticamente são autoexplicativas):

DirectoryNotFoundException: quando um diretório (Pasta) não pode ser encontrado;

EndOfStreamException: tentativa de leitura após o término de um fluxo;

FileNotFoundException: “falha” na tentativa de acessar um arquivo que não existe no disco;

FileLoadException: quando um “assembly” é encontrado, mas não pode ser carregado;

PathTooLongException: quando um “PATH” (caminho) é maior que o tamanho máximo definido pelo sistema;

PipeException: quando ocorre um erro em um “PIPE” (canal, interface de comunicação) nomeado.

Link: https://docs.microsoft.com/pt-br/dotnet/api/system.io.ioexception?view=net-5.0

Exemplificar - Criando uma Solução em C# (C Sharp)

Vamos exemplificar essa Classe, criando uma Solução em C# (C Sharp), para isso vamos utilizar:

Os blocos **TRY** (Experimentar), **CATCH** (Captura) e **FINALLY** (Finalmente) , usado para manipulação de exceções (Possíveis Falhas e Erros).

“try”: Um bloco usado em “C#” para particionar o código que pode ser afetado por uma exceção (falha ou Erro).

“catch”: Os blocos associados são usados para tratar qualquer exceção resultante do bloco “try”, portanto este bloco “finally” contém o código que é executado se uma exceção (falha/erro) é lançada ou não no bloco. Podemos usar um bloco “try” que exigirá um ou mais blocos “catch” associados ou um bloco “finally” ou ambos.

“finally”: Um bloco que permite a limpeza das ações que são realizadas em um bloco “try”. Se estiver presente, o bloco “finally” será executado por último, depois do bloco “try” e de qualquer bloco “catch” de correspondência. Este bloco sempre é executado, se uma exceção é lançada ou um bloco que corresponde ao tipo de catch exceção é encontrado.

Link: https://docs.microsoft.com/pt-br/dotnet/csharp/fundamentals/exceptions/

Código: Ex_File_01

Program.cs

Ex_File_01 | Ex_File_01.Program | Main(string[] args)

```
1  using System;
2  using System.IO;
3  namespace Ex_File_01
4  {
       0 referências
5      class Program
6      {
           0 referências
7          static void Main(string[] args) {
8              //Declarando as Variáveis do Local e Destino do Arquivo
9              //Usaremos o "@" para não ter que que duplicar os "\"
10             string localFile = @"C:\Users\User\Desktop\teste\Banco\Arquivo_01.txt";
11             string destinoFile = @"C:\Users\User\Desktop\teste\Banco\Arquivo_02.txt";
12
13             // Try: Manipulação de Exceções (Tratando Falhas / Erros)
14             try {
15                 //Usando a Classe FileInfo temos que declarar e Instanciar
16                 FileInfo fi = new FileInfo(localFile);
17                 //Utilizando um dos vários métodos disponíveis da Classe
18                 fi.CopyTo(destinoFile);
19
20                 //Usando a Classe File, não precisamos instanciar para operações mais simples
21                 //Delcaramos uma variavel do tipo vetor string[], e instanciamos o resultado do método da Classe File
22                 String[] LinhasArquivo = File.ReadAllLines(localFile);
23                 //Exibir todas as linhas do método (ReadAllLines) da Classe File que estão no Vetor LinhasAqruivo
24                 foreach(string linha in LinhasArquivo) {
25                     Console.WriteLine(linha);
26                 }
27             }
28             catch (IOException Erro) {
29                 Console.WriteLine("Um erro aconteceu:" + Erro.Message);
30             }
31         }
32     }
33 }
```

* Usar o path (caminho) correto de onde está e nome do arquivo completo.