Soubory v C#

David Horák

Úvod a System.IO

- V jazyce C# a prostředí .NET nabízí řadu objektů pro práci jak se soubory tak i adresáři
- Tyto objetky jsou seskupeny do jmenného prostoru System.IO
- Do objektů jsou zahrnuty třídy, ale i výjimky, události atd.
- Ze tříd máme k dispozici mmj. File, StreamReader, StreamWriter a FileStream
- !!! Pro užití těchto objektů je třeba přidat do zdroj. kódu using System.IO

Třída File

- Poskytuje metody pro základní manipulaci se soubory
- To zahrnuje vytváření, kontrolu existence, mazání, přesun, kopírování i čtení a úprava atributů a práci s daty
- Třída je statická

Příklady metod

- File.Create(string)
  - Pokud již soubor existuje tak ho přepíše
- File.Exists(string)
- File.Copy(string, string)
  - První argument je zdroj a následně cíl
- GetLastAccessTime(string)
- Všechny stringy jsou cestami

```
1 //Ukazka kodu
2 string s = "/home/dum/soubor.txt";
3 string x = "/home/kop/soubor2.txt";
4 if(!File.Exists(s))
5 {
6         //Musime uzavrit pred kopirovanim
7         File.Create(s).Close();
8 }
9 File.Copy(s, x);
```

StreamWriter

- Umožňuje zápis textu do souboru
- Utváří objekt
- Výchozí chování při existenci souboru je jeho přepsání, pokud neexistuje utvoří jej
- Konstruktor: StreamWriter(string)
  - Jedno z možných přetížení StreamWriter(string, bool)
  - String je opět cestou, bool určuje zda bude soubor přepsán, či k němu budeme připisovat
- Při vytvoření objektu nelze soubor číst, nebo do něj zapisovat mimo konkrétní objekt

StreamWriter

- Pro zápis využíváme metod
  - Write(string) – zapíše řetězec
  - WriteLine(string) – zapíše řetězec a odřádkuje
  - Při zápisu se nejdříve plní vnitřní buffer StreamWriteru
- Další metody
  - Close() – uzavře stream a uvolní soubor pro jiné instance a procesy
  - Flush() – vyprázdní buffer a zapíše na
    - U malých dat není třeba, flush nastane automaticky s uzavřením streamu
    - Lze zastoupit nastavením vlastnosti AutoFlush na true
- !!! Je třeba vždy uzavřít stream po dokončení zápisu

Ukázka StreamWriteru

```
1 //Ukazka kodu
2 string s = "Soubor.txt";
3 StreamWriter sw = new StreamWriter(s);
4 sw.WriteLine("Hello World");
5 sw.Write("Neodřádkuje ");
6 sw.WriteLine("Konec");
7 sw.Close();
```

```
[Ukazky]$ cat Soubor.txt
Hello World
Neodřádkuje Konec
[Ukazky]$
```

```
//Rozsirime jeste o jeden radek
using(StreamWriter swa = new StreamWriter(s, true))
{
        swa.WriteLine("Bonus");
}
```

```
[Ukazky]$ cat Soubor.txt
Hello World
Neodřádkuje Konec
Bonus
[Ukazky]$
```

StreamReader

- Umožňuje číst text ze souboru
- Utváří objekt
- Při neexistenci souboru vyvolá výjimku
- Konstruktor : StreamReader(string)
  - String je cestou
  - Má přetížení
- Při vytvoření objektu soubor lze číst mimo instanci a proces, ale nelze do něj zapisovat

StreamReader

- Pro čtení využíváme
  - Read() – přečte jeden znak
  - ReadLine() – přečte celý řádek
- Další metody
  - Peek() – vrátí znak jako Read(), ale nepřičte k ukazateli
  - Close() – Uzavře stream a uvolní soubor pro jiné instance a procesy
- Vlastností EndOfStream zjišťujeme, zda jsme přečetli všechno
- !!! Je potřeba vždy uzavřít stream po dokončení čtení

Ukázka StreamReaderu

```
15 //Kod navazuje na ukazku ze StreamWriteru
16 using(StreamReader sr = new StreamReader(s))
17 {
18         while(!sr.EndOfStream)
19         {
20                 Console.WriteLine(sr.ReadLine());
21         }
22 }
```

```
[Ukazky]$ ./UkazkySoubory
Hello World
Neodřádkuje Konec
Bonus
[Ukazky]$ cat Soubor.txt
Hello World
Neodřádkuje Konec
Bonus
[Ukazky]$
```

FileStream

- Umožňuje čtení i zápis do souboru ve formě bytů
- Utváří objekt
- Konstruktor: FileStream(string, FileMode, FileAccess, FileShare)
  - Opět má i přetížení, povinné jsou však jenom první dva argumenty
  - String je cesta, o zbytku později
- Při vytvoření objektu je výchozí chování uzamknout buď úplně (Zápis) nebo pouze pro čtení (Čtení). FileShare však toto dokáže změnit

FileStream

- Argumenty konstrkutoru
  - FileMode – specifikuje režim souboru
    - Hlavními hodnotami jsou Open, Create, OpenOrCreate
    - Pokud specifikujeme pouze FileMode tak se hodnoty zbytku nastaví implicitně vyjma případu OpenOrCreate, zde musíme nastavit FileAccess explicitně vždy
    - Pokud soubor již existuje Create jej přepíše
  - FileAccess – specifikuje přístup k souboru
    - Hodnotami jsou Read, Write, ReadWrite
    - Pokud vynecháme tak při režimu Open je implictiní nastavení Read a při Create Write
    - Při režimu Create nelze použít Read
  - FileShare – specifikuje přístup k souboru pro ostatní instance a procesy
    - Hodnotami jsou None, Read, Write, ReadWrite, Delete
    - Implicitně podle přístupu instance Read-Read, Write/ReadWrite-None

Filestream

- Hlavní metoda pro zápis
  - Write(byte[], int, int)
    - Pole jsou byty k zapsání, první int offset v poli, druhý int velikost bufferu
- Hlavní metoda pro čtení
  - Read(byte[], int, int)
    - Pole se tentokrát plní byty, zbytek argumentů je stejný jako u zápisu
- Další metody
  - Seek(long, SeekOrigin) – nastaví pozici pro čtení/zápis
    - Long je offset od SO, SeekOrigin nabývá hodnot Begin, Current, End
  - Close() – uzavře stream a uvolní soubor
  - Flush() – zapíše data z vnitřního bufferu na disk

Ukázka FileStreamu

```
1  //Ukazka kodu
2  byte[] data = {65,66,67,10,97,98,10};
3  byte[] read = new byte[7];
4  File.Create("FSSoubor.txt").Close();
5
6  using(FileStream fs = new FileStream("FSSoubor.txt", FileMode.OpenOrCreate, FileAccess.ReadWrite))
7  {
8                          fs.Write(data, 0, 4);
9                          fs.Flush();
10
11                         fs.Seek(-4, SeekOrigin.Current);
12                         fs.Read(read, 0, 7);
13                         Console.Write(System.Text.Encoding.UTF8.GetString(read));
14
15                         fs.Write(data, 4, 3);
16                         fs.Flush();
17                         fs.Seek(0, SeekOrigin.Begin);
18                         fs.Read(read, 0, 7);
19                         Console.Write(System.Text.Encoding.UTF8.GetString(read));
20 }
```

```
[Ukazky]$ ./UkazkySoubory
ABC
ABC
ab
[Ukazky]$ cat FSSoubor.txt
ABC
ab
[Ukazky]$
```

Závěr

- Jmenný prostor System.IO nám umožňuje, jednoduchou ale dostatečně podrobnou práci se soubory
- File nám umožňuje základní operace se soubory
- StreamWriter/Reader se hodí na textové soubory a mají nízkou režii, avšak jsou více omezené a nehodí se na jiné typy souborů
- FileStream je více robustní, ale v případě textových souborů může být výrazně náročnější na režii

Děkuji za pozornost