Programarea Aplicațiilor Windows – curs 6

Conf. dr. Cristian CIUREA
Departamentul de Informatică și Cibernetică Economică
Academia de Studii Economice București
cristian.ciurea@ie.ase.ro

Agenda

1. Lucrul cu fișiere text
2. Serializare/Deserializare
3. Parsare/Generare XML

Citire fișiere text

În C# se folosește clasa **`Stream`** din care a fost derivată clasa **`FileStream`** pentru a-i adăuga clasei **`Stream`** funcțiile necesare citirii și scrierii din/în fisiere text.

Citirea dintr-un fișier text se realizează astfel:

- se include biblioteca corespunzătoare:

```
using System.IO;
```

Citire fișiere text

- se creează o instanță a clasei **FileStream**:

```
FileStream fisier = new FileStream("fisier.txt", FileMode.Open, FileAccess.Read);
```

FileMode specifică ce trebuie să facă FileStream-ul cu fișierul atunci când îl accesează și are următoarele opțiuni: **Append**, **Create**, **CreateNew**, **Open**, **OpenOrCreate**, **Truncate**.

FileAccess specifică privilegiile respectivului fișier și are opțiunile: **Read**, **ReadWrite**, **Write**.

Citire fișiere text

- se creează o instanță a clasei **StreamReader**, derivată din **TextReader**:

```
StreamReader sr = new StreamReader(fisier);
```

Există patru posibilități de citire dintr-un fișier: **Read()**, **ReadBlock()**, **ReadLine()**, **ReadToEnd()**.

```
string s = sr.ReadToEnd();
```

- se închid stream-ul și fișierul:

```
sr.Close();
fisier.Close();
```

Scriere fișiere text

Scrierea într-un fișier text se realizează astfel:

- se include biblioteca corespunzătoare:

```
using System.IO;
```

- se creează o instanță a clasei **FileStream**:

```
FileStream fisier = new FileStream("fisier.txt", FileMode.Create, FileAccess.Write);
```

- se creează o instanță a clasei **StreamWriter**, derivată din **TextWriter**:

```
StreamWriter sw = new StreamWriter(fisier);
```

Scriere fișiere text

Exista două posibilități de scriere într-un fișier text: **`Write()`**, **`WriteLine()`**.

```
sw.Write("Hello");
```

- se închid stream-ul și fișierul:

```
sw.Close();
fisier.Close();
```

Serializare/Deserializare

Salvarea datelor într-un fișier binar se face prin serializare.

Serializarea este o metodă ce permite transformarea unui obiect într-o secvență de octeți din care să poată fi refăcut ulterior obiectul original.

Serializarea permite unui obiect să fie convertit într-un flux de date, care apoi este salvat într-un fișier binar.

Serializare/Deserializare

Operația de serializare presupune parcurgerea etapelor:

- declararea unei clase ca fiind serializabilă, prin adăugarea înaintea definiției clasei a atributului `[Serializable]`
- adăugarea bibliotecilor corespunzătoare:

```
using System.IO;
using System.Runtime.Serialization.
Formatters.Binary;
```

Serializare/Deserializare

- instanțierea unui obiect din clasa **FileStream**:

```
FileStream fileStream = new FileStream("stud.dat", FileMode.Create, FileAccess.Write);
```

- instanțierea unui obiect din clasa **BinaryFormatter**:

```
BinaryFormatter bf = new BinaryFormatter();
```

Serializare/Deserializare

- apelul metodei **`Serialize()`** din clasa **`BinaryFormatter`**:

```
bf.Serialize(fileStream, lista);
```

- închiderea stream-ului:

```
fileStream.Close();
```

Serializare/Deserializare

Procesul invers de citire a unui obiect serializat pentru a-i reface starea originală se numește **deserializare**.

Operația de deserializare presupune parcurgerea etapelor:

- instanțierea unui obiect din clasa **FileStream**:

```
FileStream fileStream = new FileStream("stud.dat", FileMode.Open, FileAccess.Read);
```

Serializare/Deserializare

- instanțierea unui obiect din clasa **BinaryFormatter**:

```
BinaryFormatter bf = new BinaryFormatter();
```

- apelul metodei **Deserialize()** din clasa **BinaryFormatter**:

```
ArrayList lista =
(ArrayList)bf.Deserialize(fileStream);
```

- închiderea stream-ului:

```
fileStream.Close();
```

Parsare/Generare XML

Citirea dintr-un fișier XML se poate face utilizând metoda **`Read()`** din clasa abstractă **`XmlReader`** din namespace-ul **`System.Xml`**.

Pașii pentru citire și parsare fișier XML:

- instanțierea unui obiect din clasa **`StreamReader`**:

```
StreamReader sr = new StreamReader("nbrfxrates.xml");
```

- citirea conținutului fișierului:

```
string str = sr.ReadToEnd();
```

Parsare/Generare XML

- obținerea unui obiect din clasa **`XmlReader`** prin apelul metodei **`Create()`**:

```
XmlReader reader = XmlReader.Create(new
StringReader(str));
```

- prelucrarea obiectului din clasa **`XmlReader`**:

```
while (reader.Read()) {...}
```

Parsare/Generare XML

Pentru generarea unui fișier XML se parcurg următorii pași:

- instanțierea unui obiect din clasa **MemoryStream**:

```
MemoryStream memStream = new MemoryStream();
```

- instanțierea unui obiect din clasa **XmlTextWriter**:

```
XmlTextWriter writer = new XmlTextWriter(memStream, Encoding.UTF8);
```

Parsare/Generare XML

- apelul metodelor **WriteStartElement()**, **WriteEndElement()**, **WriteValue()**, **WriteAttributeString()** din clasa **XmlTextWriter**:

```
writer.WriteStartElement("xml:CursEUR");
writer.WriteAttributeString("valuta",
"EUR");
writer.WriteValue(tbEUR.Text);
writer.WriteEndElement();
```

Parsare/Generare XML

- extragerea într-un **`string`** a conținutului obiectului din clasa **`MemoryStream`** și scrierea într-un fișier:

```
string xmlString = Encoding.UTF8.
GetString(memStream.ToArray());
File.WriteAllText("fis.xml", xmlString);
```

- apelul metodei **`Close()`** pentru obiectele **`XmlTextWriter`** și **`MemoryStream`**:

```
writer.Close();
memStream.Close();
```

Bibliografie

[1] I. Smeureanu, M. Dârdală, A. Reveiu – *Visual C# .NET*, Editura CISON, București, 2004.

[2] C. Petzold – *Programming Microsoft Windows with C#*, Microsoft Press, 2002.

[3] L. O'Brien, B. Eckel – *Thinking in C#*, Prentice Hall.

[4] J. Richter – *Applied Microsoft .NET Framework Programming*, Microsoft Press, 2002.

[5] http://acs.ase.ro/paw