Citire-Scriere fisiere

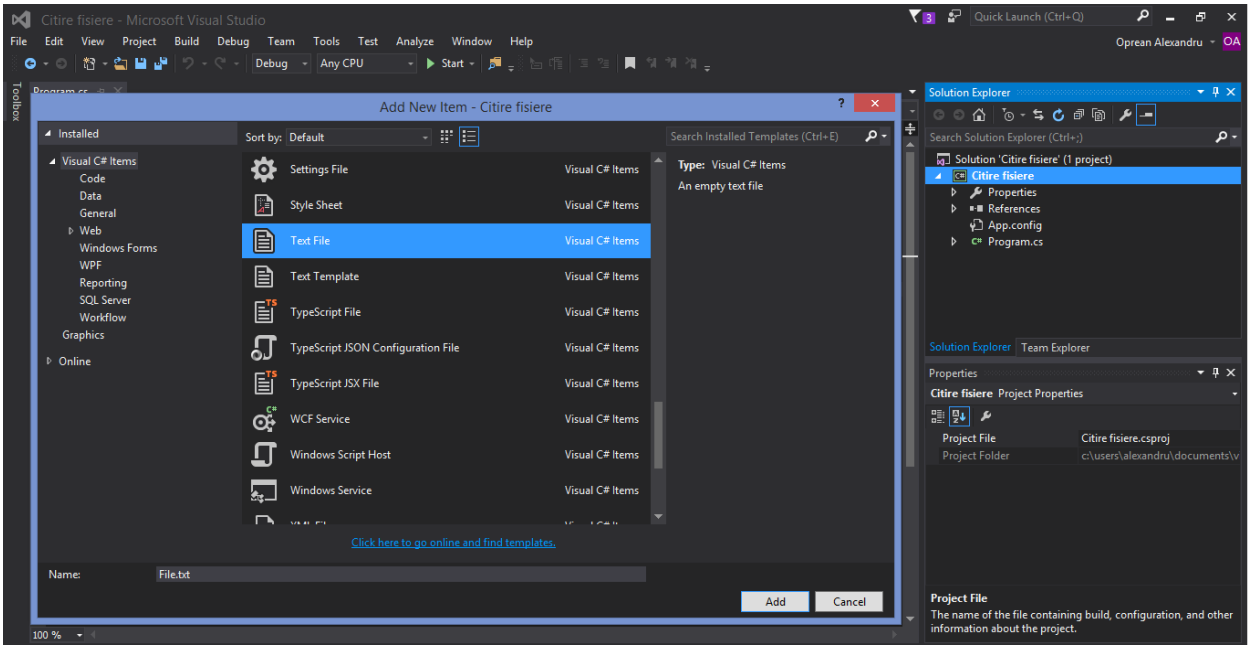
Ca in orice limbaj de programare o componenta foarte importanta este citirea si scrierea din si in diverse formate de text. In cadrul acestui laborator vom invata sa citim si sa scriem in fisiere de tip text, csv, json si xml.

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**

Citirea se realizeaza astfel:

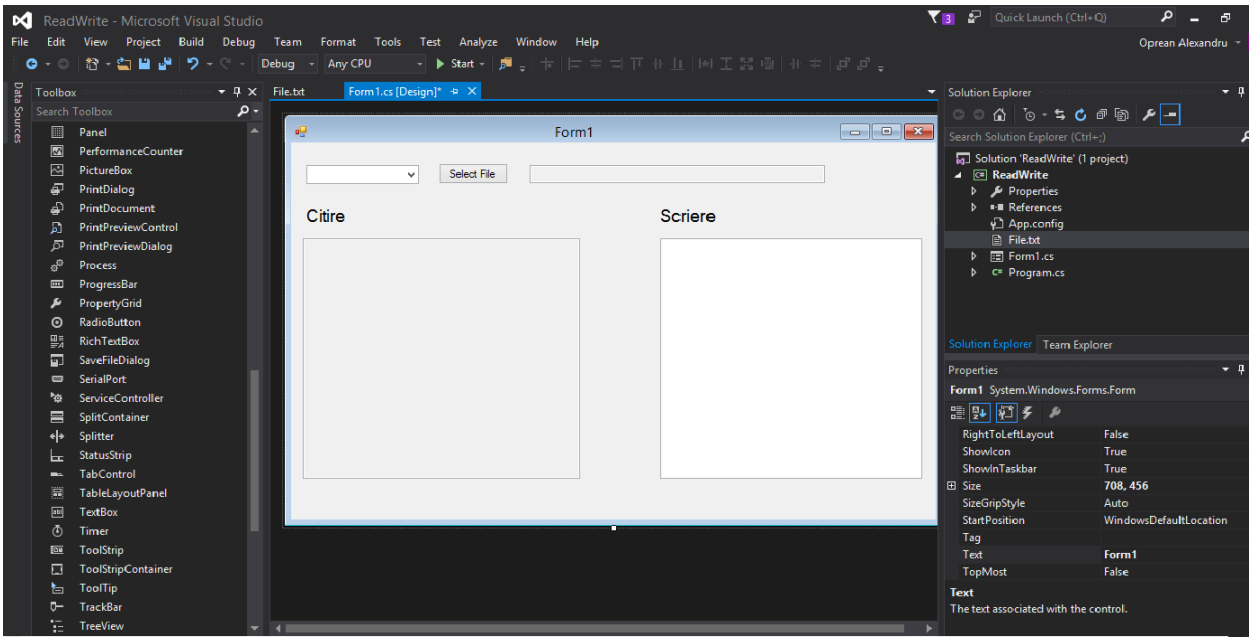
1. Se creaza in cadrul proiectului un fisier text (Fig. 1) pe care il vom numi File.

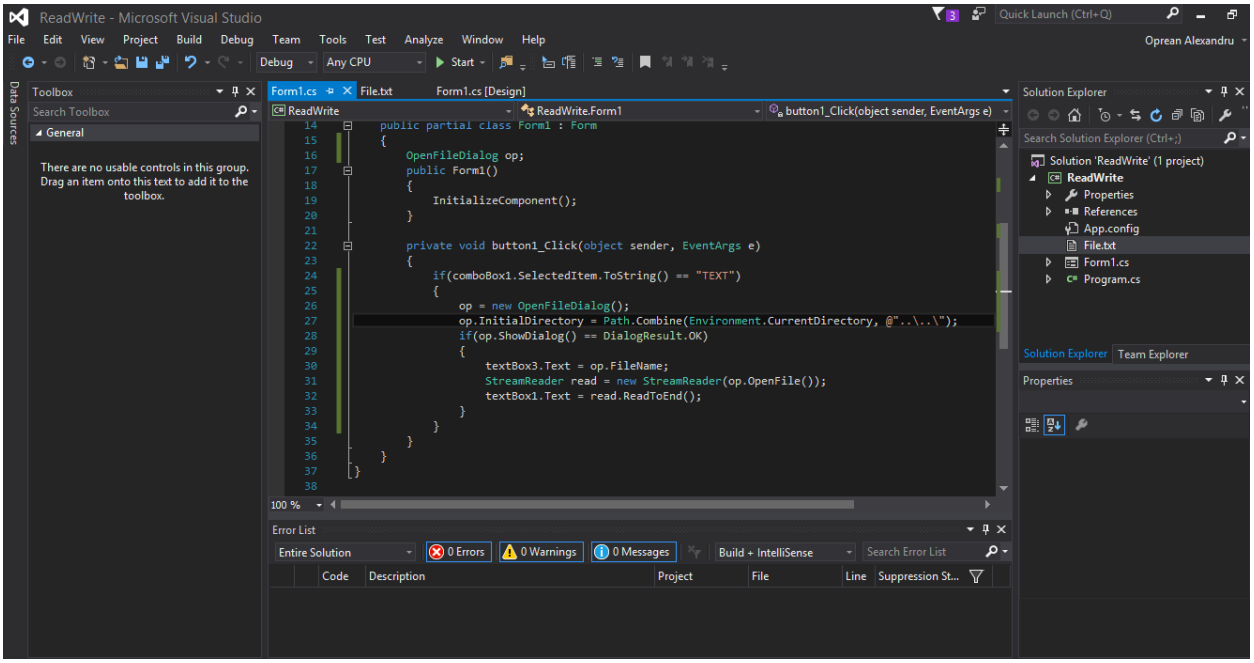


2. In cadrul Fisierului vom scrie pe 3 linii separate “Text fisier”, “Alt text” si “Ultimul text” apoi Salvam.

3. Apoi vom realiza Form-ul ca in fig. urmatoare cu setarile urmatoare:

- Textbox-ul de la citire si cel de langa buton vor fi ReadOnly
- In combobox vom introduce valorile “TEXT, CSV, JSON si XML”





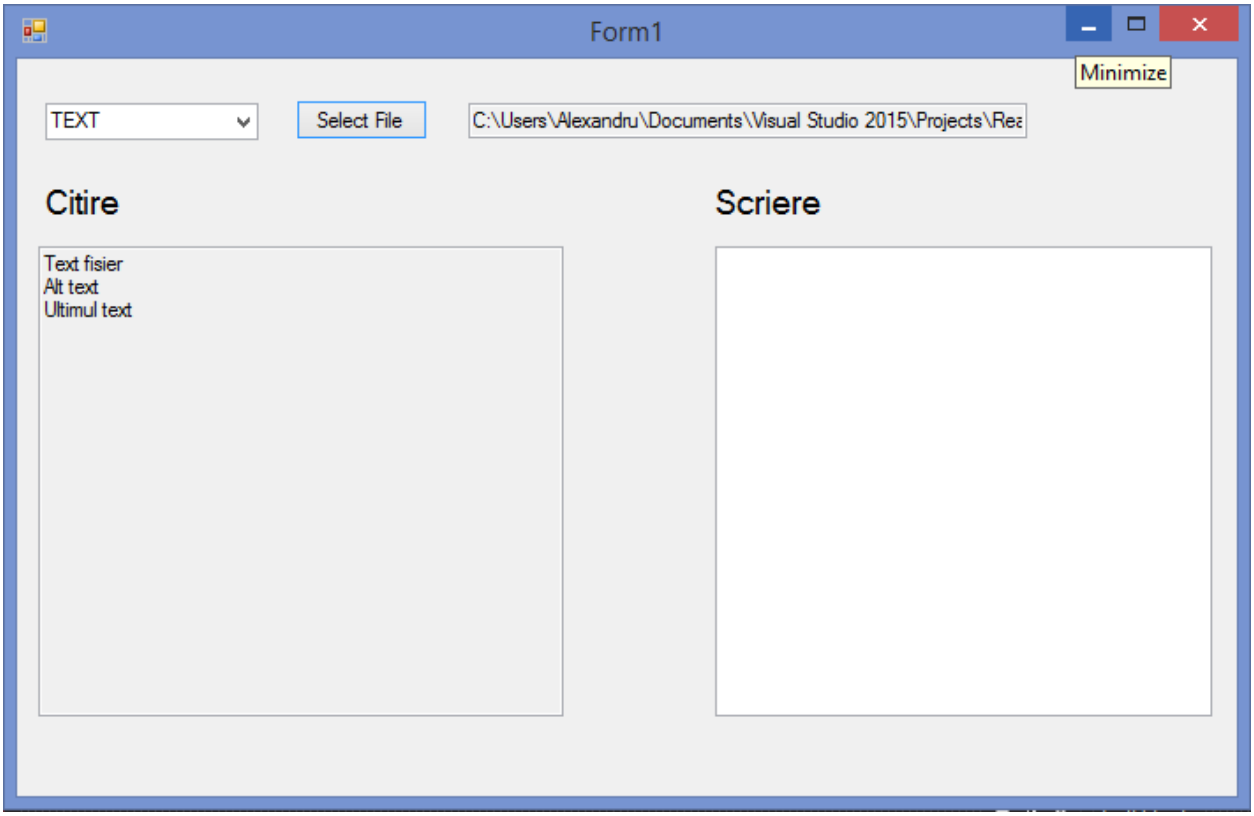
In codul de mai sus am realizat citirea dintr-un fisier folosind Un Open File Dialog. Un Open File Dialog ne lasa sa cautam path-ul si fisierul pe care noi dorim sa il folosim de pe harddisk. In primul rand am facut clasa OpenFileDialog Accesibila intregii clase (vom folosi aceasta clasa si pentru citire si pentru scriere).

In al 2-lea rand am verificat daca combobox-ul nostru are selectat optiunea pentru text si daca o are am initializat clasa de open file dialog. Dupa initializare i-am dat Directorul in care va cauta prima data folosind proprietatea InitialDirectory. **Path.Combine** dupa cum ii spune si numele combina 2 path-uri, in primul rand combina folder-ul in care noi lucram `Environment.CurrentDirectory` si `@"..\..\"` care ii spune compiler-ului sa se duca mai sus cu 2 foldere. Acest lucru este necesar deoarece folder-ul in care se compileaza aplicatia este cel de debug si nu cel al proiectului efectiv.

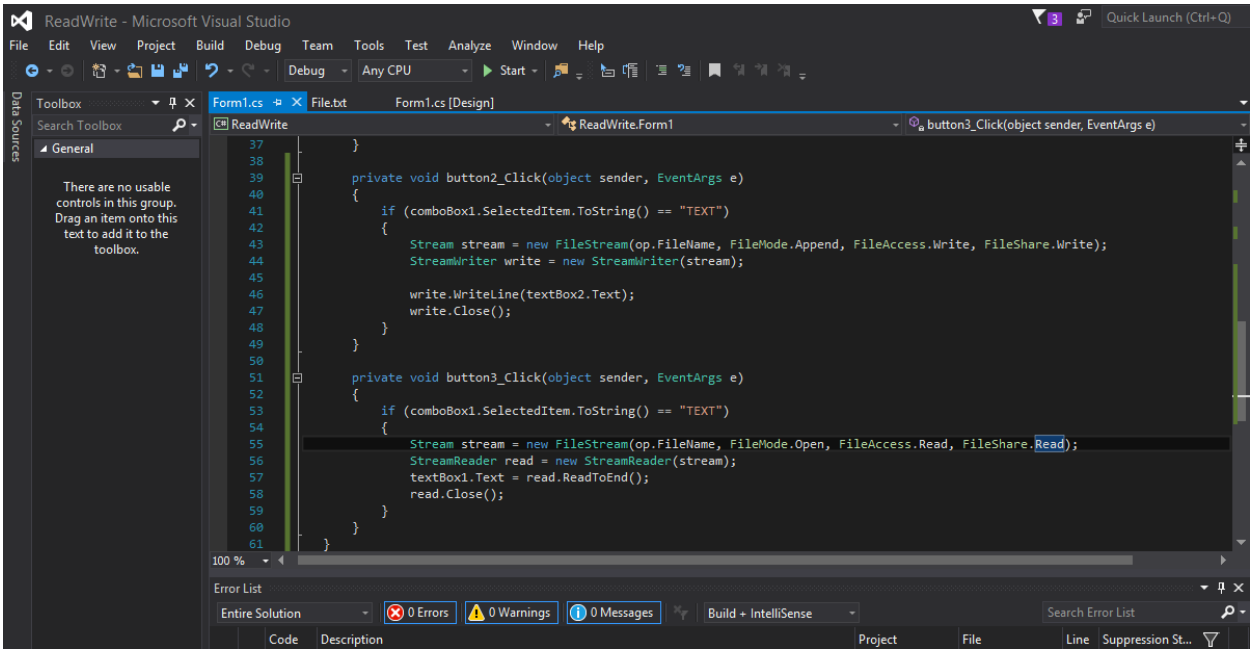
`if(op.ShowDialog() == DialogResult.OK)` verifica daca in momentul in care inchidem fereastra am selectat un fisier si acesta este valid.

Ultimul pas este acela de a crea un StreamReader (aceasta este o clasa folosita pentru a citi fisiere text), a-i atribui fisierul nostru si a apela metoda ReadToEnd() si afisam rezultatul in textbox.

Iar rezultatul ar trebui sa fie urmatorul:



Pasul urmator va fi acela de a crea inca 2 butoane (“Write” si “Read”).

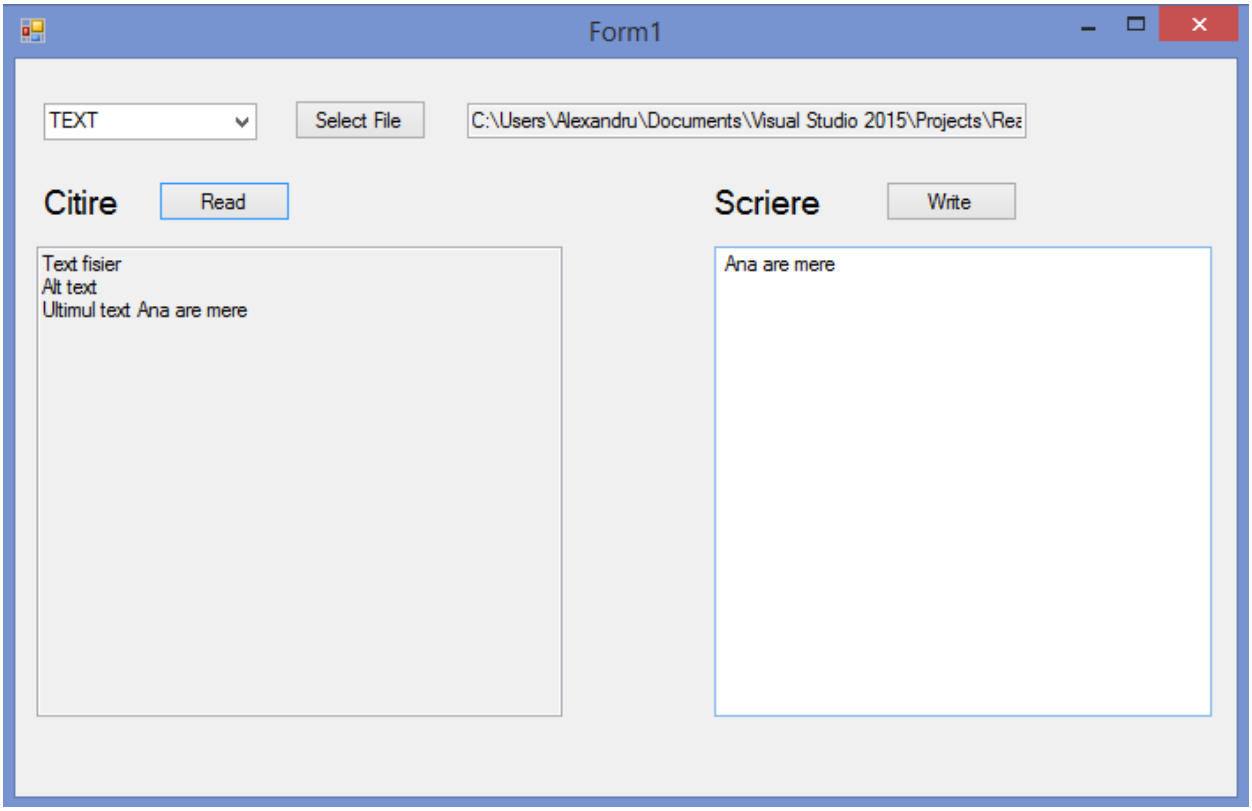


In cadrul butonului 2 si butonului 3 vom implementa scrierea si citirea (atunci cand deja avem fisierul selectat si dorim sa facem o recitire din text)

In cadrul ambelor metode cream un nou Stream caruia ii atribuim urmatorii parametri:

- op.FileName: Deja avem selectat fisierul nu mai trebuie sa il deschidem, astfel doar trimitem numele fisierului;
- FileMode.Append/Open: Pentru scriere vom alege append pentru ca dorim sa adaugam la fisier si nu sa ii face override (ceea ce s-ar intampla in cazul folosirii Open).
- FileAccess.ReadWrite: accesul initial este ReadOnly si daca nu selectam ReadWrite nu vom putea scrie in fisier.

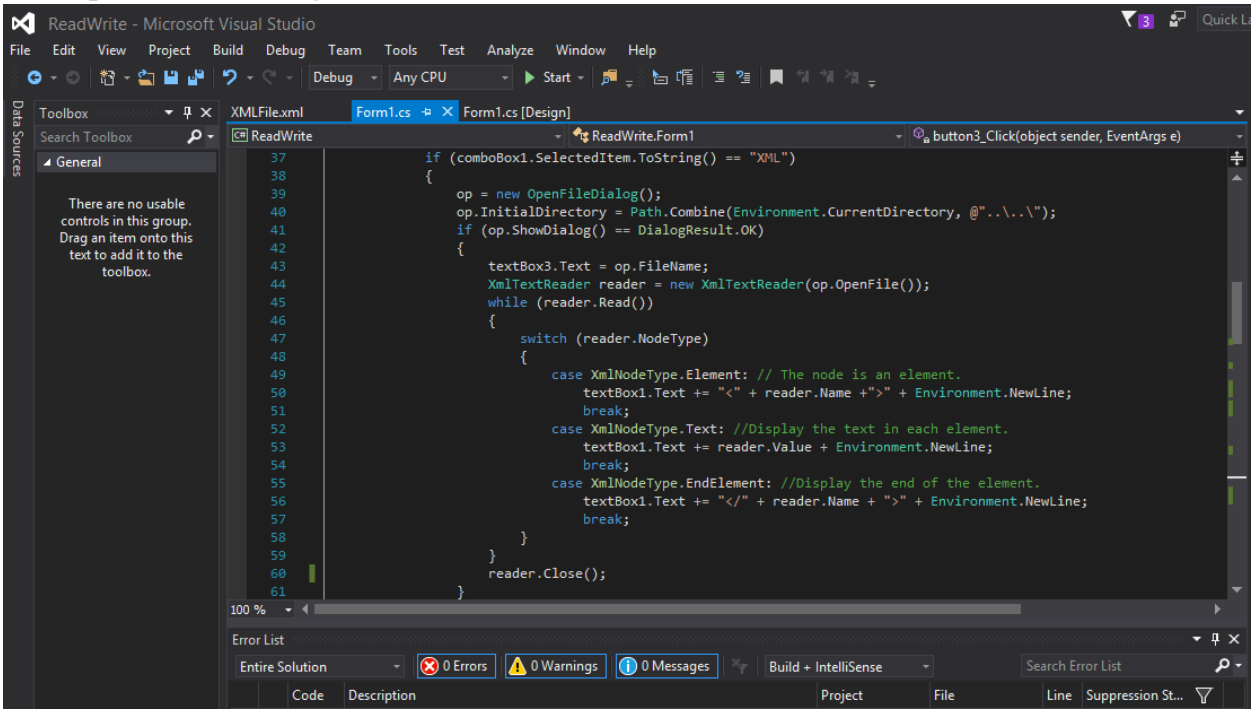
Iar rezultatul ar trebui sa fie acesta:



Scriere si citire din XML.

1. Se creaza un nou fisier XML (vezi Fig. 1 la text) si se copiaza in el continutul de la aceasta pagina https://msdn.microsoft.com/en-us/library/ms762271(v=vs.85).aspx

2. Se importa libraria System.Xml



Pentru a citi un XML se foloseste acelasi principiu de la citirea textelor. Se verifica daca este selectat combobox-ul pe XML, se creaza o clasa XmlTextReader care primeste stream-ul nostru din open file dialog, apoi se parseaza intreg documentul folosind functia Read(). Apoi de fiecare data cand intalneste un tag, valoarea unui tag sau inchiderea unui tag va afisa valorile sau numele acestora.

Exemplu:

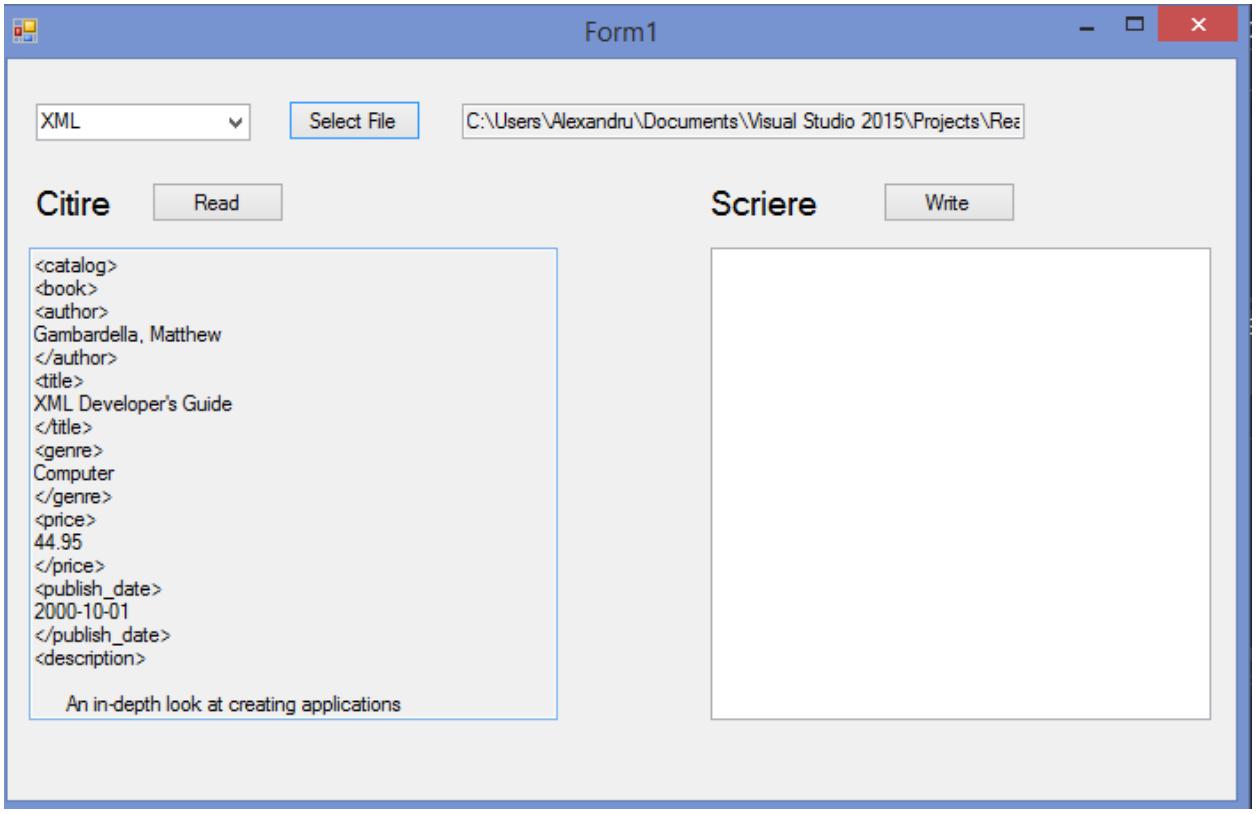
```
XmlNodeType.Element: - <Book>

XmlNodeType.Text: - “Head First C#”

XmlNodeType.EndElement: - </Book>
```

O alta comanda noua este Environment.NewLine care este echivalentul “\n”.

Iar la rulare ar trebui sa avem rezultatul din figura urmatoare:



Scriere:

```
    }
    if (comboBox1.SelectedItem.ToString() == "XML")
    {
        Stream stream = new FileStream(op.FileName, FileMode.Truncate, FileAccess.Write, FileShare.Write);
        stream.Flush();
        XmlWriter writer = XmlWriter.Create(stream);
        writer.WriteStartDocument();
        writer.WriteStartElement("Book");
        writer.WriteElementString("Title", textBox2.Text);
        writer.WriteEndElement();
        writer.WriteEndDocument();

        writer.Close();
        stream.Close();
    }
}
```

Pentru Scrierea intr-un fisier XML lucrurile stau putin altfel:

In primul rand am ales FileMode.Truncate pentru a sterge textul precedent si a crea crea un continut nou. Pentru scriere se foloseste libraria XmlWriter care la fel primeste stream-ul precedent.

Cu ajutorul XmlWriter putem sa controlam foarte usor forma pe care o va avea documentul nostru.

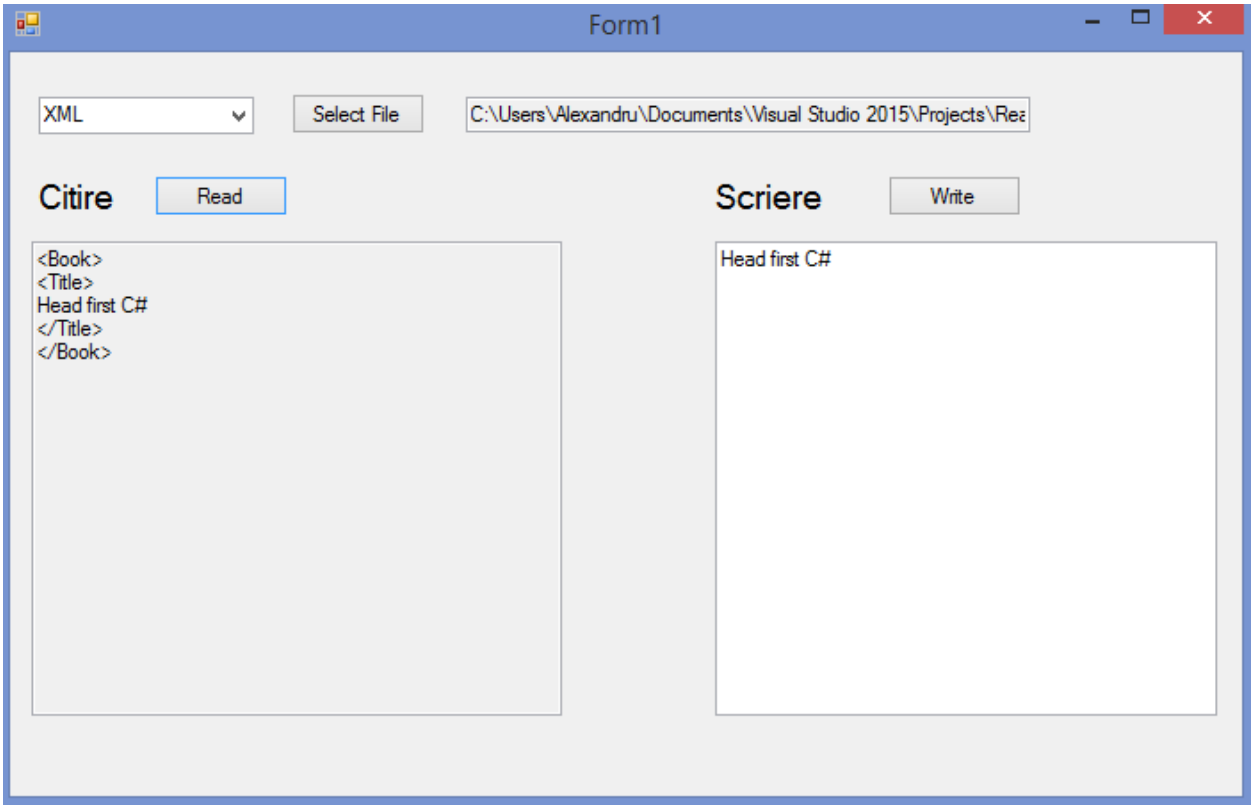
`writer.WriteStartDocument(); -` va crea definitia unui document xml nou (ex: <?xml version="1.0"?>);

`writer.WriteStartElement("Book"); -` va crea un nou tag care va avea numele “Book”;

`writer.WriteElementString("Title", textBox2.Text); -` va crea un nou tag numit “Title” si ii va atribui valoarea pe care noi o vom pune in textbox si va inchide automat tag-ul;

`writer.WriteEndElement(); -` va inchide tag-ul “Book”;

`writer.WriteEndDocument(); -` va declara sfarsitul documentului Xml.



Putem observa ca C# ne ajuta foarte mult in crearea unui document Xml si ca nu este necesara introducerea manuala a tag-urilor.