### Modulul 16. Continuare în XML.

#### Conversia tipului datelor la filtrare

Precum am văzut la lecția precedentă putem utiliza filtrele pentru a obține doar înregistrările care coincid cu o anumită condiție. Neajunsul de bază al metodei folosite precedent era faptul că aceasta verifica datele doar în formă string, ceea ce nu este practic pentru cazurile reale.

Soluția pentru această problemă este folosirea metodei CONVERT pentru a converti datele din documentul XML într-un anumit tip de date.

Atrageți atenția că funcția CONVERT se include în interiorul semnelor “” și necesită 2 parametri:

* Câmpul din care vom extrage valoarea
* Tipul datelor în care vom converti valoarare scris în forma dată:

'System.Int32'

Dacă nu sunteți sigur care este clasa exactă a tipului de date de care aveți nevoie consultați: <https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/data-types/>

Neajunsul folosirii acestei metode este evident, aceasta nu conține nici un mecanism de securitate împotriva excepțiilor, o simplă greșeală manuală poate cauza întreg programul să nu funcționeze sau să funcționeze greșit.

**Exemplu:**

|  |
| --- |
| namespace FiltrareXMLNumere  {  class FiltrareaXML  {  static void Main()  {  XmlReader xmlFile;  xmlFile = XmlReader.Create("exemplu2.xml", new XmlReaderSettings());  DataSet ds = new DataSet();  ds.ReadXml(xmlFile);  DataView dv = new DataView();  dv = new DataView(ds.Tables[0], "CONVERT(Produs\_pret, 'System.Int32') >= 6000", "Produs\_pret", DataViewRowState.CurrentRows);  dv.ToTable().WriteXml("Rezultat.xml");  Console.WriteLine("XML creat");  }  }  } |

În exemplul de mai sus am folosit exact aceleași metode ca la lecția precedentă doar că de data aceasta am inclus conversia câmpului Produs\_pret pentru a compara valorile ca integer în loc de string, astfel vom obține rezultatul de care avem nevoie.

Probleme pentru rezolvare.

### **Program 1:**

1. Pe baza exemplului pentru acasă(Tranzacții) afișați doar tranzacțiile care au fost completate cu succes(bool statut==true) și sortați-le după data efectuării.

#### Conversia tipului datelor la sortare

În exemplul precedent am reușit să facem conversia filtrului pe care îl folosim, dar sortarea datelor a rămas neschimbată.

Pentru a elimina această problemă vom folosi o metodă indirectă. Vom crea un DataTable, care în esență este un tabel de date, acesta va copia DOAR structura documentului XML (adică doar numărul și denumirile coloanelor și rândurilor), dar nu va copia datele din interior.

Apoi vom face o conversie explicită a coloanei Produs\_pret din string in integer.

După ce am efectuat acest pas putem importa datele din DataSet în DataTable, ele deja vor prelua formatul necesar și vor deveni integer.

În final vom folosi aceeași metodă folosită anterior pentru filtrarea și sortarea datelor.

**Exemplu:**

|  |
| --- |
| namespace FiltrareXMLNumere  {  class FiltrareaXML  {  static void Main()  {  XmlReader xmlFile;  xmlFile = XmlReader.Create("exemplu2.xml", new XmlReaderSettings());  DataSet ds = new DataSet();  ds.ReadXml(xmlFile);  //Clonati structura  DataTable ds2 = ds.Tables[0].Clone();  //Transformati tipul datelor din coloana Produs\_pret  ds2.Columns["Produs\_pret"].DataType = typeof(Int32);  //Importati din nou continutul  foreach (DataRow row in ds.Tables[0].Rows)  ds2.ImportRow(row);  //Filtrare sortarea  DataView dv = new DataView();  dv = new DataView(ds2, "Produs\_pret >= 3000", "Produs\_pret DESC", DataViewRowState.CurrentRows);  dv.ToTable().WriteXml("Rezultat.xml");  Console.WriteLine("XML creat");  }  }  } |

Probleme pentru rezolvare.

### **Program 1:**

1. Pe baza exemplului de mai sus sortați tranzacțiile din anul 2020 după valoarea acestora în ordine descendentă.

#### Serializarea și deserializarea fișierelor XML

Pentru a face fișierele XML mai ușor de procesat și de adăuga posibilități extinse în citirea acestora putem folosi serializarea acestora. Serializarea unui fișier XML adăuga un set de informații utile în interiorul acestuia precum descrierea structurii documentului, tipul datelor stocate în fiecare coloană etc.

Vom trece mai extins scopurile serializării și deserializării obiectelor și fișierelor în cadrul Lecției 18.

|  |
| --- |
| namespace SerializareXML  {  class SerializareaXML  {  static void Main()  {  //Deschiderea XML  XmlReader xmlFile;  xmlFile = XmlReader.Create("exemplu2.xml", new XmlReaderSettings());  DataSet ds = new DataSet();  ds.ReadXml(xmlFile);  //Serializare  StreamWriter serialWriter;  serialWriter = new StreamWriter("XMLserializat.xml");  XmlSerializer xmlWriter = new XmlSerializer(ds.GetType());  xmlWriter.Serialize(serialWriter, ds);  serialWriter.Close();  ds.Clear();  Console.WriteLine("Serializare finisata cu succes!");  }  }  } |

Probleme pentru rezolvare.

### **Program 1:**

1. Pe baza exemplului de mai sus serializați fișierul XML cu tranzacții și stocați-l într-un fișier XML nou.

|  |
| --- |
| namespace DeserializareXML  {  class DeserializareaXML  {  static void Main()  {  DataSet ds = new DataSet();  XmlSerializer xmlSerializer = new XmlSerializer(typeof(DataSet));  FileStream readStream = new FileStream("XMLserializat.xml", FileMode.Open);  ds = (DataSet)xmlSerializer.Deserialize(readStream);  readStream.Close();  DataView dv = new DataView(ds.Tables[0]);  dv.ToTable().WriteXml("XMLdeserializat.xml");  Console.WriteLine("XML deserializat");  }  }  } |

Probleme pentru rezolvare.

### **Program 1:**

1. Pe baza exemplului de mai sus deserializați fișierul XML creat la problema precedentă.