### Modulul 18. Serializarea obiectelor în C#.

#### Ce este serializarea și deserializarea?

Serializarea este procesul de stocare a proprietăților unui obiect și transmiterea acestora în format binar către o sursă externă(fișier, bază de date etc.) cu scopul de a retrage aceste proprietăți și de a recrea și readuce obiectul la starea precedentă.

Librăriile necesare pentru serializare:

**using System.IO;**

**using System.Runtime.Serialization;**

**using System.Runtime.Serialization.Formatters.Binary;**

Pentru a demonstra cum lucrează serializarea unui obiect vom elabora un exemplu elementar.

**Exemplu:**

|  |
| --- |
| namespace Serializare  {  [Serializable]  class Program1  {  public int ID;  public string Name;  static void Main()  {  //Cream obiectul si ii dam valori  Program1 obj = new Program1();  obj.ID = 1573;  obj.Name = ".Serializare";  IFormatter formatter = new BinaryFormatter();  Stream stream = new FileStream("Exemplu.txt", FileMode.Create, FileAccess.Write);  formatter.Serialize(stream, obj);  stream.Close();  }  }  } |

În urma derulării programului vom obține un fișier .txt care va conține informația despre obiectul Program care l-am creat, inclusiv despre variabilele ID și Name care le-am inclus în acesta.

Următorul pas este să deserializăm fișierul și să extragem variabilele din acesta.

**Exemplu:**

|  |
| --- |
| namespace Deserializare  {  class Program  {  static void Main()  {  IFormatter formatter = new BinaryFormatter();  Stream stream = new FileStream("Exemplu.txt", FileMode.Open, FileAccess.Read);  Serializare.Program1 objnew = (Serializare.Program1)formatter.Deserialize(stream);  stream.Close();  Console.WriteLine("ID:" + objnew.ID);  Console.WriteLine("Name:" + objnew.Name);  }  }  } |

|  |
| --- |
| namespace Rezolvare\_Probleme  {  [Serializable]  class Eveniment  {  public DateTime eventDate;  public string eventDescription;  static Random gen = new Random();  public static string[] evName = { "Pastele", "Anul Nou", "Craciun", "Primul Sunet", "Ultimul Sunet" };  static DateTime RandomDay()  {  DateTime start = new DateTime(DateTime.Today.Year, 1, 1);  int range = (DateTime.Today - start).Days;  return start.AddDays(gen.Next(range));  }  public Eveniment()  {  eventDate = RandomDay();  eventDescription = evName[gen.Next(0, evName.Length)];  }  static void Main()  {  Eveniment[] eventList = new Eveniment[5];  for (int i = 0; i < eventList.Length; i++)  eventList[i] = new Eveniment();  List<Eveniment> newList = eventList.ToList();  newList = newList.OrderBy(x => x.eventDate).ToList();  foreach (var item in newList)  Console.WriteLine(item.eventDescription + " " + item.eventDate);  }  }  } |

Pentru a deserializa fișierul parcurgem aceeași procedură dar de data aceasta setăm tipul FileAccess.Read. Următorul pas este să recreăm un obiect nou objnew cu ajutorul **formatter.Deserialize**, vom avea nevoie să folosim conversia forțată pentru a indica ce tip va avea obiectul.

În continuare putem accesa direct variabilele ID și Name din obiectul nou creat.

Probleme pentru rezolvare.

### **Program 1:**

1. Creați o clasă care va stoca 5 evenimente(DateTime) aleatorii într-un tablou. Serializați clasa dată.
2. Deserializați clasa într-o clasă externă.
3. Folosiți LINQ pentru a ordona datele descrescător.
4. Afișați cea mai recentă dată.

#### Proprietățile de serializare [Serializable] și [NonSerializable]

Precum am indicat mai sus obiectele care sunt disponibile pentru serializare au atributul **[Serializable].**

Atributul dat indică, de fapt că toate câmpurile unei clase vor fi serializate(în caz că nu schimbăm acest lucru).

Opus acestuia, atributul [NonSerializable] indică faptul invers, ceea că clasa dată nu poate fi serializată.

Dar aceasta este doar utilitatea de bază a acestor proprietăți, cea avansată este faptul că cu ajutorul lor putem să delimităm fiecare variabilă și să alegem dacă aceasta va fi serializată sau nu.

**Exemplu:**

|  |
| --- |
| namespace Atribute  {  [Serializable]  class ProgramAtribute  {  public int ID;  public string Name;  [NonSerialized] public DateTime data;  static void Main()  {  //Cream obiectul si ii dam valori  ProgramAtribute obj = new ProgramAtribute();  obj.ID = 1573;  obj.Name = ".Serializare";  obj.data = DateTime.Now;  IFormatter formatter = new BinaryFormatter();  Stream stream = new FileStream("Exemplu2.txt", FileMode.Create, FileAccess.Write);  formatter.Serialize(stream, obj);  stream.Close();  stream = new FileStream("Exemplu2.txt", FileMode.Open, FileAccess.Read);  ProgramAtribute objnew = (ProgramAtribute)formatter.Deserialize(stream);  Console.WriteLine("ID:" + objnew.ID);  Console.WriteLine("Name:" + objnew.Name);  Console.WriteLine("Data:" + objnew.data);  }  }  } |

În exemplul de mai sus variabila data, care are valoarea datei curente nu va fi serializată din cauza că am utilizat atributul **[NonSerialized].** Atunci când deserializăm fișierul și creăm un obiect nou din el variabila data va avea valoarea default 01/01/0000 etc.

Probleme pentru rezolvare.

### **Program 1:**

1. Creați o clasă User care va avea următoarele variabile userName, userPassword, ultimaLogare, userIP.
2. În metoda main atribuiți valori la fiecare variabilă.
3. userIP îl puteți determina cu ajutorul metodelor:

**IPV6**

|  |
| --- |
| using System.Net;  string hostName = Dns.GetHostName();  string myIP = Dns.GetHostEntry(hostName).AddressList[0].ToString(); |

**IPV4**

|  |
| --- |
| namespace IP  {  class Extern  {  static void Main()  {  IPAddress[] ipv4Addresses = Array.FindAll(Dns.GetHostEntry(string.Empty).AddressList, a => a.AddressFamily == AddressFamily.InterNetwork);  Console.WriteLine(ipv4Addresses[0]);  }  }  } |

1. Atribuiți clasei proprietatea [Serializable] dar variabila userIP protejați-o de serializare.
2. Serializați obiectul.
3. Dintr-o clasă externă executați deserializarea și afișați informațiile despre utilizator.

#### Tipuri de serializare.

Serializarea care am realizat-o mai sus este de tip binar, dar există mai multe tipuri fiecare cu beneficii și dezavantaje.

1. **Serializarea binară.**

Avantajul acestui tip de serializare este că acesta este foarte ușor de realizat. În esență, serializarea binară ia toate valorile dintr-un obiect și le transformă în formă binară și le stochează, din acest motiv putem oricând să recreăm un obiect la starea acestuia exactă când a fost serializat.

Dezavantajul de bază este că acest tip de serializarea nu poate fi folosit pentru a transmite date către programe scrise în alt limbaj de programare.

1. **Serializarea XML.**

O altă opțiune pentru serializare este **XML.** Serializarea în XML este mai greu de executat, dar aceasta permite mult mai simplu transmiterea datelor către programe externe, scrise în alte limbaje etc.

**Exemplu:**

|  |
| --- |
| namespace SerializareXML  {  [Serializable]  public class Cat  {  public string Name { get; set; }  public int Age { get; set; }  public Cat(){}  public Cat(string name, int age)  {  Name = name;  Age = age;  }  }  class Program  {  static void Main(string[] args)  {  //Crearea obiectului  Cat person = new Cat("Tom", 7);  Console.WriteLine("Obiect creat");  //Serializare  XmlSerializer formatter = new XmlSerializer(typeof(Cat));  using (FileStream fs = new FileStream("cat.xml", FileMode.OpenOrCreate))  formatter.Serialize(fs, person);  //Deserializare  using (FileStream fs = new FileStream("cat.xml", FileMode.OpenOrCreate))  {  Cat newPerson = (Cat)formatter.Deserialize(fs);  Console.WriteLine("Obiect deserializat");  Console.WriteLine($"Name: {newPerson.Name}\nAge: {newPerson.Age}");  }  }  }  } |

Serializarea tablourilor de obiecte

|  |
| --- |
| namespace SerializareXML  {  [Serializable]  public class Cat  {  public string Name { get; set; }  public int Age { get; set; }  public int Greutate { get; set; }  public Cat() { }  public Cat(string name, int age, int greutate)  {  Name = name;  Age = age;  Greutate = greutate;  }  }  class Program  {  static void Main()  {  //Crearea obiectului  Cat[] cat1 = new Cat[] { new Cat("Tom", 7, 7), new Cat("Murca", 5, 4) };  //Serializare  XmlSerializer formatter = new XmlSerializer(typeof(Cat[]));  using (FileStream fs = new FileStream("cat.xml", FileMode.OpenOrCreate))  formatter.Serialize(fs, cat1);  //Deserializare  using (FileStream fs = new FileStream("cat.xml", FileMode.OpenOrCreate))  {  Cat[] newCat = (Cat[])formatter.Deserialize(fs);  foreach (var item in newCat)  {  Console.WriteLine($"Name: {item.Name}\nAge: {item.Age}\nWeight: {item.Greutate}\n");  }  }  }  }  } |

Probleme pentru rezolvare.

### **Program 1:**

1. Creați o clasă care va stoca informațiile despre o țără: string nume, string limba, int populatie.
2. În metoda Main creați 10 obiecte de tip Țară și atribuiți-le valori.
3. Serializați obiectele ca fișier XML.
4. Afișați conținutul fișierului XML cu ajutorul requesturilor LINQ.
5. Găsiți țările cu populația mai mare de 5 milioane.
6. Găsiți țara cu numele Moldova și afișați toate informațiile.