Serializzazione binaria di oggetti con attributi privati

autore Bandiera Roberto

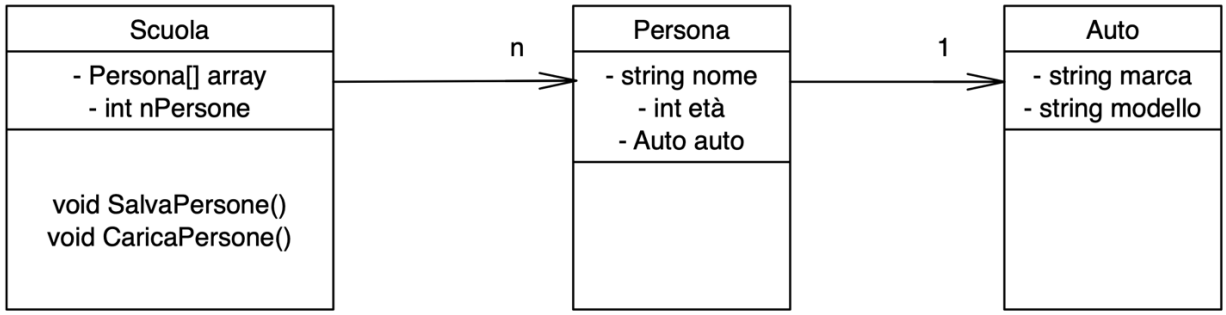
Keywords: C#, .Net7, serializzazione binaria, metodi di estensione, attributi privati, oggetti composti

Premessa:
Poiché la classe BinaryFormatter è ora ritenuta obsoleta, considerata non sicura e non può più essere usata (https://learn.microsoft.com/it-it/dotnet/standard/serialization/binaryformatter-security-guide articolo del 28/11/2022), per effettuare la serializzazione binaria di oggetti con attributi privati si possono utilizzare le classi BinaryWriter e BinaryReader, che però prevedono la programmazione delle singole operazioni per scrivere e leggere i dati su disco.

Obiettivo:
L'obiettivo di questo lavoro è esemplificare la serializzazione e la deserializzazione binaria di un array di oggetti.

Metodo:
Vengono scritti appositi metodi di estensione delle classi BinaryWriter e BinaryReader per consentire una agevole scrittura, e lettura, degli oggetti da archiviare su disco.
Si ricorda che i metodi di estensione consentono di aggiungere nuove funzionalità a classi già esistenti, senza la necessità di ricompilarne il codice.
Per contenere i metodi di estensione delle classi BinaryWriter e BinaryReader, devono essere create delle apposite classi statiche.

Il diagramma delle classi che viene considerato in questo esempio è il seguente:



La Scuola contiene un array di Persone, che a loro volta possono contenere una Auto.
La classe Scuola ha i metodi SalvaPersone() e CaricaPersone() per effettuare, rispettivamente, la serializzazione e la deserializzazione dell'array di persone.
L'array è gestito in modo semidinamico: ha una dimensione fissa molto grande e risulta parzialmente riempito.
L'attributo nPersone indica quante sono le persone effettivamente presenti nell'array e, al contempo, indica la prima posizione libera dell'array. Infatti, le nPersone sono caricate nelle posizioni indicizzate da 0 a nPersone-1.
Tutti gli attributi sono privati, e pertanto occorrono metodi accessori per la lettura ed eventualmente assegnazione di valore agli stessi.

Le classi che effettuano serializzazione XML e JSON agiscono solo su attributi pubblici, pertanto si ricorre ad una serializzazione binaria utilizzando le classi BinaryWriter e BinaryReader.
Il BinaryWriter ha il metodo Write() che consente di scrivere su disco variabili di tipo elementare come stringhe, caratteri, numeri interi, numeri double, valori booleani.
Pertanto si aggiungono i metodi di estensione per scrivere oggetti di tipo Auto e di tipo Persona:

```
// metodi di estensione per la serializzazione

public static void WriteAuto(this BinaryWriter bw, Auto a)
{
   bw.Write(a.DammiMarca());
   bw.Write(a.DammiModello());
}


public static void WritePersona(this BinaryWriter bw, Persona p)
{
       bw.Write(p.DammiNome());
       bw.Write(p.DammiEtà());
       Auto a = p.DammiAuto();
       if (a != null)  // controllo la presenza dell'auto
       {
          bw.Write(true);  // ha un'auto
          bw.WriteAuto(a);
       }
       else
       {
          bw.Write(false); // niente auto
       }
}
```

Notare che, poiché non è garantito che tutte le persone abbiano un'auto, si deve fare un controllo e scrivere su disco un valore booleano che informa sulla presenza o meno dell'auto.

Il BinaryReader ha solo metodi come ReadString(), ReadChar(), ReadInt32(), ReadDouble(), ReadBoolean() per leggere dati di tipo elementare.
I metodi di estensione per leggere oggetti di tipo Auto e di tipo Persona sono i seguenti:

```
public static Auto ReadAuto(this BinaryReader br)
{
   string marca = br.ReadString();
   string modello = br.ReadString();
   return new Auto(marca, modello);
}


public static Persona ReadPersona(this BinaryReader br)
{
   string nome = br.ReadString();
   int età = br.ReadInt32();
   bool haAuto = br.ReadBoolean();
   if (haAuto)
   {
      Auto a = br.ReadAuto();
      return new Persona(nome, età, a);
   }
```

```
   else
   {
      return new Persona(nome, età);
   }
}
```

I suddetti metodi vengono collocati in apposite classi statiche.
Nella classe Scuola ci sono i metodi per serializzare e per deserializzare l'intero array di persone con tutto il loro contenuto:

```
public void SalvaPersone()
{
    // serializzazione
    FileStream file = File.Create("dati.bin");
    BinaryWriter bw = new BinaryWriter(file);
    bw.Write(nPersone);
    for (int i = 0; i < nPersone; i++)
    {
         bw.WritePersona(array[i]);
    }
    file.Close();
}


public void CaricaPersone()
{
    // deserializzazione
    FileStream file = File.OpenRead("dati.bin");
    BinaryReader br = new BinaryReader(file);
    nPersone = br.ReadInt32();
    for (int i = 0; i < nPersone; i++)
    {
       array[i] = br.ReadPersona();
    }
    file.Close();
}
```

Notare che, oltre all'array vero e proprio, è necessario serializzare anche la variabile nPersone che specifica il numero di persone presenti nell'array.

Il metodo Main() della classe Program consente di testare il funzionamento della serializzazione:

```
// oggetti da caricare nell'array
Auto a1 = new Auto("opel", "mokka");
Persona p1 = new Persona("gianni", 20, a1);
Auto a2 = new Auto("mercedes", "gla");
Persona p2 = new Persona("luca", 30, a2);
Persona p3 = new Persona("piero", 40);

// creazione e riempimento dell'oggetto scuola
Scuola scuola = new Scuola();
scuola.AggiungiPersona(p1);
scuola.AggiungiPersona(p2);
```

```
scuola.AggiungiPersona(p3);

// salvataggio
scuola.SalvaPersone();

// creo una nuova scuola e vi carico i dati salvati
scuola = new Scuola();
scuola.CaricaPersone();

// verifica
int n = scuola.DammiNumeroPersone();
for (int i = 0; i < n; i++)
{
   Persona p = scuola.DammiPersona(i);
   Console.WriteLine(p);
}
```

si ottiene il seguente output:

```
gianni 20 opel mokka
luca 30 mercedes gla
piero 40
```

Conclusioni:
In questo lavoro abbiamo visto un modo semplice per ottenere le funzionalità della classe BinaryFormatter, il cui uso ora è proibito.
Si tratta semplicemente di scrivere una coppia di metodi di estensione per ciascuna delle classi che si deve poter serializzare.
Grazie a questi metodi diventa immediato ottenere la serializzazione di oggetti anche complessi.

Il codice completo delle classi dell'applicazione è il seguente:

```
public class Scuola
{
      private Persona[] array;
      private int nPersone;

      public Scuola()
      {
            array = new Persona[100];
            nPersone = 0;
      }

      public void AggiungiPersona(Persona p)
      {
             array[nPersone] = p;
             nPersone++;
      }
```

```
      public int DammiNumeroPersone()
      {
             return nPersone;
      }

      public Persona DammiPersona(int index)
      {
             return array[index];
      }

      public void SalvaPersone()
      {
            // serializzazione
            FileStream file = File.Create("dati.bin");
            BinaryWriter bw = new BinaryWriter(file);
            bw.Write(nPersone);
            for (int i = 0; i < nPersone; i++)
            {
                bw.WritePersona(array[i]);
            }
            file.Close();
       }

       public void CaricaPersone()
       {
            // deserializzazione
            FileStream file = File.OpenRead("dati.bin");
            BinaryReader br = new BinaryReader(file);
            nPersone = br.ReadInt32();
            for (int i = 0; i < nPersone; i++)
            {
                array[i] = br.ReadPersona();
            }
            file.Close();
        }
}
```

```
public class Persona
{
      private string nome;
      private int età;
      private Auto auto;

      public Persona(string unNome, int unaEtà)
      {
           nome = unNome;
           età = unaEtà;
           auto = null;
      }

      public Persona(string unNome, int unaEtà, Auto unAuto)
      {
           nome = unNome;
           età = unaEtà;
           auto = unAuto;
      }
```

```
      public string DammiNome()
      {
           return nome;
      }

      public int DammiEtà()
      {
           return età;
      }

      public Auto DammiAuto()
      {
           return auto;
      }

      public void CompraAuto(Auto unAuto)
      {
           auto = unAuto;
      }

      public override string ToString()
      {
           return nome + " " + età + " " + auto;
      }
}


public static class EstensionePersona
{
      // metodi di estensione per la serializzazione
      public static void WritePersona(this BinaryWriter bw, Persona p)
      {
            bw.Write(p.DammiNome());
            bw.Write(p.DammiEtà());
            Auto a = p.DammiAuto();
            if (a != null)
            {
                  bw.Write(true);  // ha un'auto
                  bw.WriteAuto(a);
            }
            else
            {
                  bw.Write(false); // niente auto
            }
      }

      public static Persona ReadPersona(this BinaryReader br)
      {
            string nome = br.ReadString();
            int età = br.ReadInt32();
            bool haAuto = br.ReadBoolean();
            if (haAuto)
            {
                  Auto a = br.ReadAuto();
                  return new Persona(nome, età, a);
            }
            else
            {
```

```
                    return new Persona(nome, età);

                }
        }
}
```

```
public class Auto
{
      private string marca;
      private string modello;

      public Auto(string unaMarca, string unModello)
      {
            marca = unaMarca;
            modello = unModello;
      }

      public string DammiMarca()
      {
             return marca;
      }

      public string DammiModello()
      {
             return modello;
      }

      public override string ToString()
      {
             return marca + " " + modello;
      }
}


public static class EstensioneAuto
{
    // metodi di estensione per la serializzazione
    public static void WriteAuto(this BinaryWriter bw, Auto a)
    {
        bw.Write(a.DammiMarca());
        bw.Write(a.DammiModello());
    }

    public static Auto ReadAuto(this BinaryReader br)
    {
        string marca = br.ReadString();
        string modello = br.ReadString();
        return new Auto(marca, modello);
    }
}
```