Strumenti e regole per la serializzazione e deserializzazione di una classe

A cura del docente Giuliano Pellegrini Parisi - © 2009

Le tecniche di serializzazione che .NET mette a disposizione sono divise in due categorie fondamentali:

- serializzazione binaria e serializzazione SOAP
- serializzazione XML

• la serializzazione binaria è molto efficiente in termini di uso della memoria e tempo di esecuzione, ma i dati serializzati sono in formato binario e quindi illeggibili agli esseri umani.
In gergo tecnico si dice che i dati sono “human-unreadable”.

• la serializzazione SOAP è meno efficiente rispetto a quella binaria e lavora in modo quasi identico, con l’unica grande differenza che i dati serializzati vengono trasformati in un formato “human-readable”, più precisamente un formato chiamato XML SOAP (XML Simple Object Access Protocol).

```
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xml
<SOAP-ENV:Body>
<a1:Person id="ref-1" xmlns:a1="http://schemas.microsoft.com/clr/nsassem/Wir
<FirstName id="ref-3">Oscar</FirstName>
<LastName id="ref-4">L</LastName>
<BirthDay>1972-06-16T00:00:00.0000000-05:00</BirthDay>
<Alias id="ref-5"></Alias>
</a1:Person>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

• la serializzazione XML offre una grandissima flessibilità perché permette di memorizzare tutte le informazioni in file con formato XML e quindi “human-readable”, dando la possibilità di trasferire tutti i dati sulla rete tra sistemi completamente diversi.
Segue un esempio generico di file XML relativo alla rubrica di email.

```
<EmailAddresses>

  <EmailAddress Address="joebloggs@zenicom.com" Destination="Business" />
  <EmailAddress Address="joebloggs@athome.com" Destination="Home" />
  <EmailAddress Address="joebloggs@hotmail.com" Destination="Other" />
</EmailAddresses>
```

Serializzazione BINARIA

Quando si vuole utilizzare il meccanismo di serializzazione e deserializzazione su una classe, è necessario prima applicare una regola che permetta l'autorizzazione di questi meccanismi sulla classe indicata.
Senza l'autorizzazione, sarà impossibile serializzare e deserializzare.

Regola fondamentale

```
using System;

[Serializable]
public class Ambascita : Estero,IVisto,IIncorporato
{
  …
}
```

La regola impone di autorizzare la classe ai meccanismi di serializzazione e deserializzazione tramite l'inserimento dell'attributo [Serializable] sopra il nome della classe. E' necessario specificare la libreria System.

Strumenti fondamentali

```
using System.Runtime;
using System.Runtime.Serialization;

[Serializable]
public class Ambascita : Estero,IVisto,IIncorporato,ISerializable
{
  …
}
```

Devo ereditare l'interfaccia ISerializable che contiene gli strumenti pubblici per la realizzazione della serializzazione e deserializzazione. Questa interfaccia si trova nelle librerie sopra indicate.

Strumenti fondamentali

```
using System.Runtime;
using System.Runtime.Serialization;

[Serializable]
public class Ambascita : Estero,IVisto,IIncorporato,ISerializable
{
   public void GetObjectData(SerializationInfo info,StreamingContext context)
   {
   }

   public Ambasciata(SerializationInfo info,StreamingContext context)
   {
   }
}
```

Implemento i due strumenti che permettono di realizzare la serializzazione e deserializzazione.

Strumenti di serializzazione

```
public void GetObjectData(SerializationInfo info,StreamingContext context)
{
    info.AddValue(“NomeEnte”,this._NomeEnte,typeof(string));
    …
}
```

Questo metodo viene utilizzato per realizzare la serializzazione.
La classe “SerializationInfo” stabilisce quale dato membro verrà utilizzato nel processo di serializzazione, in questo caso “this._NomeEnte” mentre con la clausola “typeof(string)” indichiamo il tipo di dato, che in questo caso è una stringa.
Il primo parametro “NomeEnte” è proprio il nome con cui il dato membro verrà serializzato, quindi nel processo di deserializzazione sarà obbligatorio conoscere questo nome.

Strumenti di serializzazione

info.AddValue(“NomeEnte”,this._NomeEnte,typeof(string));

Vorrei ricordarvi che siete all’interno della classe “Ambasciata”, quindi posso accedere in modo diretto al dato membro “this._NomeEnte” che è un dato membro private della classe astratta “Ente”?

In base al lavoro svolto fino ad ora la risposta è NO!

Esiste però un altro modificatore di visibilità (scope) sui dati membro che è chiamato “protected”, il quale permette l’accesso diretto al dato membro direttamente dalle classi figlie.

Strumenti di serializzazione

Modificatori di visibilità:

- private

Il dato membro è visibile solo all'interno della classe

- public

Il dato membro è visibile a tutto il mondo

- protected

Il dato membro è visibile a tutte le classi figlie

Strumenti di serializzazione

```
abstract public class Ente
{
   protected string _NomeEnte;

   …
}
```

Dovremo quindi modificare la visibilità dei dati membri "_NomeEnte","_TipoEnte" e "_Nazione" se voglio serializzare in modo corretto, permettendo quindi all'interno del metodo GetObjectData l'accesso diretto ai dati membro delle classi astratte padre.

Strumenti di serializzazione

```
public void GetObjectData(SerializationInfo info,StreamingContext context)
{
    info.AddValue("NomeEnte",this._NomeEnte,typeof(string));
    info.AddValue("TipoEnte",this._TipoEnte,typeof(string));
    info.AddValue("Nazione",this._Nazione,typeof(string));
    info.AddValue("Presidiato",this._Presidiato,typeof(bool));
    info.AddValue("Capitale",this._Capitale,typeof(string));
    info.AddValue("Incorporato",this._Incorporato,typeof(bool));
    info.AddValue("Nome",this._Nome,typeof(string));
    info.AddValue("Cognome",this._Cognome,typeof(string));
    info.AddValue("Sesso",this._Sesso,typeof(string));
    info.AddValue("Eta",this._Eta,typeof(string));
    info.AddValue("Indirizzo",this._Indirizzo,typeof(string));
    info.AddValue("Citta",this._Citta,typeof(string));
    info.AddValue("TipoVisto",this._TipoVisto,typeof(string));
    info.AddValue("DurataVisto",this._DurataVisto,typeof(string));
}
```

Indico in rosso i dati membro di tipo protected.

Strumenti di deserializzazione

```
public Ambasciata(SerializationInfo info,StreamingContext context)
{
   this._Presidiato=info.GetBoolean(“Presidiato”);
   …
}
```

Questo particolare costruttore di classe viene utilizzato per realizzare la deserializzazione.
La classe “SerializationInfo” contiene il dato che deve essere deserializzato, quindi tramite un apposito metodo relativo al tipo di dato indicare il nome del dato membro che precedentemente è stato assegnato in fase di serializzazione. Il dato deserializzato viene poi memorizzato nel dato membro private della classe.
Come osservate, è necessario conoscere il nome del dato precedentemente utilizzato nella fase di serializzazione ed il suo tipo.

Strumenti di deserializzazione

```
public Ambasciata(SerializationInfo info,StreamingContext context)
{
   this._Presidiato=info.GetBoolean(“Presidiato”);
   this._Capitale=info.GetString(“Capitale”);
   this._Incorporato=info.GetBoolean(“Incorporato”);
   this._Nome=info.GetString(“Nome”);
   this._Cognome=info.GetString(“Cognome”);
   this._Sesso=info.GetString(“Sesso”);
   this._Eta=info.GetString(“Eta”);
   this._Indirizzo=info.GetString(“Indirizzo”);
   this._Citta=info.GetString(“Citta”);
   this._TipoVisto=info.GetString (“TipoVisto”);
   this._DurataVisto=info.GetString (“DurataVisto”);
}
```

Come notate non sono specificati tutti i dati che precedentemente sono stati sottoposti al meccanismo di serializzazione. Entriamo nel dettaglio.

Strumenti di deserializzazione

Perché non sono presenti tutti i dati definiti in fase di serializzazione?

```
public void GetObjectData(SerializationInfo info,StreamingContext context)
{
   info.AddValue(“NomeEnte”,this._NomeEnte,typeof(string));
   info.AddValue(“TipoEnte”,this._TipoEnte,typeof(string));
   info.AddValue(“Nazione”,this._Nazione,typeof(string));
   Info.AddValue(“Presidiato”,this._Presidiato,typeof(bool));
   info.AddValue(“Capitale”,this._Capitale,typeof(string));
   info.AddValue(“Incorporato”,this._Incorporato,typeof(bool));
   info.AddValue(“Nome”,this._Nome,typeof(string));
   info.AddValue(“Cognome”,this._Cognome,typeof(string));
   info.AddValue(“Sesso”,this._Sesso,typeof(string));
   info.AddValue(“Eta”,this._Eta,typeof(string));
   info.AddValue(“Indirizzo”,this._Indirizzo,typeof(string));
   info.AddValue(“Citta”,this._Citta,typeof(string));
   info.AddValue(“TipoVisto”,this._TipoVisto,typeof(string));
   info.AddValue(“DurataVisto”,this._DurataVisto,typeof(string));
}
```

```
public Ambasciata(SerializationInfo info,StreamingContext context)
{
  this._Presidiato=info.GetBoolean(“Presidiato”);
  this._Capitale=info.GetString(“Capitale”);
  this._Incorporato=info.GetBoolean(“Incorporato”);
  this._Nome=info.GetString(“Nome”);
  this._Cognome=info.GetString(“Cognome”);
  this._Sesso=info.GetString(“Sesso”);
  this._Eta=info.GetString(“Eta”);
  this._Indirizzo=info.GetString(“Indirizzo”);
  this._Citta=info.GetString(“Citta”);
  this._TipoVisto=info.GetString (“TipoVisto”);
  this._DurataVisto=info.GetString (“DurataVisto”);
}
```

La tecnica di deserializzazione si applica sul costruttore di classe,quindi i dati che mancano in fase di deserializzazione “NomeEnte”,”TipoEnte” e “Nazione” li troverete implementati nei rispettivi costruttori delle classi astratte padre “Ente,”Funzione” ed “Estero”.

Struttura gerarchica

Regole e strumenti per la serializzazione e deserializzazione in una struttura gerarchica complessa.

Struttura gerarchica classe astratta Ente

```
using System.Runtime;
using System.Runtime.Serialization;

abstract public class Ente : ISerializable
{
  protected string _NomeEnte;

  public Ente(SerializationInfo info,StreamingContext context)
  {
      this._NomeEnte=info.GetString("NomeEnte");
  }

  abstract public void GetObjectData
                                    (SerializationInfo info,
                                     StreamingContext context);
}
```

Struttura gerarchica classe astratta Funzione

```
using System.Runtime;
using System.Runtime.Serialization;

abstract public class Funzione : Ente,IPresidiato
{
   protected string _TipoEnte;

  public Funzione(SerializationInfo info,StreamingContext context)
            : base(info,context)

  {
       this._TipoEnte=info.GetString(“TipoEnte”);
  }


}
```

Struttura gerarchica classe astratta Estero

```
using System.Runtime;
using System.Runtime.Serialization;

abstract public class Estero : Funzione
{
   protected string _Nazione;

  public Estero(SerializationInfo info,StreamingContext context)
            : base(info,context)
  {
      this._Nazione=info.GetString("Nazione");
  }


}
```

Struttura gerarchica classe astratta Ambasciata

```
using System;
using System.Runtime;
using System.Runtime.Serialization;

[Serializable]
public class Ambasciata : Estero,IVisto,IIncorporato
{
    private string _Capitale;
    private bool _Presidiato;
    private string _Nome;
    private string _Cognome;
    private string _Sesso;
    private string _Eta;
    private string _Indirizzo;
    private string _Citta;
    private string _TipoVisto;
    private string _DurataVisto;
    private string _NumeroUomini;
    private bool _Incorporato ;
```

Struttura gerarchica classe astratta Ambasciata

```
public override void GetObjectData(SerializationInfo info,StreamingContext context)
{
    info.AddValue("NomeEnte",this._NomeEnte,typeof(string));
    info.AddValue("TipoEnte",this._TipoEnte,typeof(string));
    info.AddValue("Nazione",this._Nazione,typeof(string));
    info.AddValue("Presidiato",this._Presidiato,typeof(bool));
    info.AddValue("Capitale",this._Capitale,typeof(string));
    info.AddValue("Incorporato",this._Incorporato,typeof(bool));
    info.AddValue("Nome",this._Nome,typeof(string));
    info.AddValue("Cognome",this._Cognome,typeof(string));
    info.AddValue("Sesso",this._Sesso,typeof(string));
    info.AddValue("Eta",this._Eta,typeof(string));
    info.AddValue("Indirizzo",this._Indirizzo,typeof(string));
    info.AddValue("Citta",this._Citta,typeof(string));
    info.AddValue("TipoVisto",this._TipoVisto,typeof(string));
    info.AddValue("DurataVisto",this._DurataVisto,typeof(string));
}
```

Struttura gerarchica classe astratta Ambasciata

```
public Ambasciata(SerializationInfo info,StreamingContext context) : base(info,context)
{
   this._Presidiato=info.GetBoolean(“Presidiato”);
   this._Capitale=info.GetString(“Capitale”);
   this._Incorporato=info.GetBoolean(“Incorporato”);
   this._Nome=info.GetString(“Nome”);
   this._Cognome=info.GetString(“Cognome”);
   this._Sesso=info.GetString(“Sesso”);
   this._Eta=info.GetString(“Eta”);
   this._Indirizzo=info.GetString(“Indirizzo”);
   this._Citta=info.GetString(“Citta”);
   this._TipoVisto=info.GetString (“TipoVisto”);
   this._DurataVisto=info.GetString (“DurataVisto”);
}
```

Chiamate in fase di serializzazione

FormattazioneBinaria.Serialize(…);

```
public override void GetObjectData(…)
{
  …
}
```

ITI Serale
La prima scuola online

Chiamate in fase di deserializzazione

… FormattazioneBinaria.Deserialize(…);