けん太のプログラミング修行記

第15回「けん太、シリアル化を試みる」の巻 ―その1―

オブジェクトデータをファイルへ保存する方法について

碗仔 けん太（Pochi Company）
WANKO, Kenta

Technology Tools

- ☑ Visual Basic .NET
- ☑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ❑ ASP.NET
- ❑ Internet Information Services
- ❑ Other:

Level

★ ★ ☆ ☆ ☆

Samples

・この記事で取り上げたソースコードおよびサンプルプログラムは、付録CD-ROMの\DOTNET\KENTAディレクトリに収録しています。

\LINESVB
複数の直線を使って線画を描くサンプル（VB.NET版）

\LINESCS
複数の直線を使って線画を描くサンプル（C#版）

Visual Basic 6.0のときにはなかった「シリアル化」という便利な機能が、.NETになって追加されました。シリアル化とは、オブジェクトの状態をバイト列の形式にしてファイルやメモリなどに保存したり転送できるようにするためのオブジェクト指向の方法だそうですが、とにかく便利らしい、という噂です。そこで、今回と次回は、シリアル化について考えてみましょう。

シリアル化と逆シリアル化

以前、Visual C++でMFCを使ってプログラムを作ったときに、簡単な方法でさまざまなデータを保存したり取り出すことができるシリアライズの便利さに、とても驚いたことがあります。また、Javaでは直列化と呼ばれている同様の機能がさまざまなクラスでサポートされているようです。

呼び方はさまざまですが、オブジェクト指向になった.NETでは、シリアライズを「シリアル化」、デシリアライズを「逆シリアル化」と呼んでいます。そこで、ここではシリアル化と逆シリアル化という言葉を使うことにしましょう。

それでは、オブジェクト指向プログラミングでも使われているこのシリアル化と逆シリアル化とは、いったいどういうものなのでしょうか。さっそく、ポチの用語集やMSDNライブラリでシリアル化と逆シリアル化について調べてみましょう。

その結果、わかったことは、

> **シリアライズ（serialize）：**
> それぞれのオブジェクトの固有の状態、つまり各インスタンスの状態を、バイト列の形式にしてファイルやメモリなどに保存したり転送できるようにすること
>
> **デシリアライズ（deserialize）：**
> 保存してある情報を取り出したり受け取ったりして各インスタンスを再生成すること

というようです。

そして、これらの機能は本格的なオブジェクト指向プログラミング言語には必ず備わっているようです。

シリアル化とファイルへの保存

シリアル化と逆シリアル化はファイルへオブジェクトデータを保存／復元することだけに使われるわけではありませんが、ここでは話を単純にするために、まず、オブジェクトをファイルに保存する場合について考えてみましょう。

第一に、シリアル化では、各インスタンスの状態を連続したバイト列の形式（serialなバイトデータ）にしてファイルに保存します。

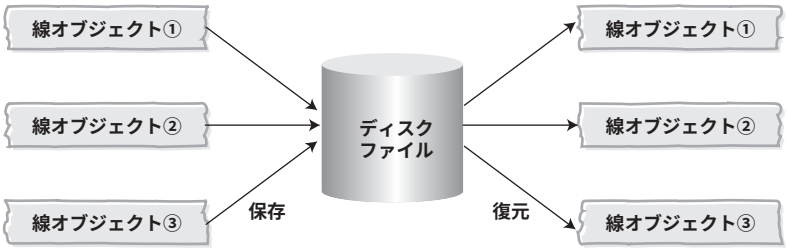
たとえば、「線」というオブジェクトを、両端の座標と線の色で表現するものとしましょう[注1]。どんなに複雑な線画であっても、それぞれの線の、両端の座標と色を保存しておけば、あとでそれらの情報を使って線を再現できます。

そこで、それぞれの線オブジェクト（線のクラスのあるインスタンス）の情報をバイト列の形式でファイルに順番に保存することにします。そして、あとでそれらの情報をファイルから読み出して各線を描けば、もとの複雑な線画を再現できるはずです（図1）。

しかし、これはシリアル化や逆シリアル化を使わなくてもできそうですね？

従来の方法で、ファイルを開いて線のデータを順番にファイルに保存し、あとでファイルを開いてその情報を読み込めば、線画を表示できます。それで何も問題はないわけです。

図1：データの保存と復元



従来の方法でもできることを、わざわざほかの方法で行なう必要が、本当にあるのでしょうか？　シリアル化がとても便利だという噂は単なるウワサだったのでしょうか？

頭をひねっていると、ポチさんが通りかかりました。


どしたの？


シリアル化なんですけど。


シリアライズのこと？　それがどうかした？


シリアル化しなくても、たいていのことはできそうですけど。


確かに、できるよ。


それじゃあ、シリアル化する意味がないんじゃないですか？


うーん、キミ、重要なことを何か忘れていない？


は？


クラスのことさ。

シリアル化とクラス

確かに、図1に示したオブジェクト指向でない従来の方法では、クラスは登場しません。オブジェクト指向プログラミングでは、オブジェクト（インスタンス）はクラス定義から作成します。クラス定義に各インスタンス固有の情報が適用されるとそれぞれの固有の情報をもったオブジェクトが作成されるわけです。

そこで、クラスのインスタンスを作るときだけでなく、ファイルからオブジェクトを復元するときにも、この方法を使うことを考えてみましょう。おそらく、図2のようになるでしょう[注2]。

つまり、最初にオブジェクトを作成するときにクラス定義を使ってインスタンスを作成し、ファイルからオブジェクトを復元するときにも、最初と同じクラス定義を使ってインスタンスを作成することにします。すると、オブジェクトを復元する際に必要になるの

注1）ここでは話を単純にするために「線」オブジェクトを両端の座標と線の色で表現することにしますが、線の太さ、種類などの属性や、図形の種類を増やしたとしても、プログラムが多少複雑になるだけで、話の流れは同じです。また、「線」の代わりにテキストや住所録データなどに置き換えても、本稿の説明は同じように当てはまります。

注2）図2はさまざまなクラスに当てはまるように一般化してあります。図1と比較したいときには、「オブジェクトA」を「線オブジェクト」と置き換えて眺めるとわかりやすいかもしれません。