FEATURE 2

ADO.NETの初歩からXMLデータベースまで一刀両断
AccessやExcelのデータも自由自在

シリアライズを使えば簡単にできるXMLデータベース

テキスト型データベースよりワンランク上を目指して

PROJECT KySS
http://www.PROJECTKySS.NET/

Technology Tools

- ☑ Visual Basic .NET
- ☐ Visual C# .NET
- ☐ SQL Server 2000
- ☐ Oracle 9i
- ☑ Access 2003
- ☐ ASP.NET
- ☐ Internet Information Services
- ☑ Other:
  XML

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¥F01_02ディレクトリに収録しています。

¥DATA：
サンプルで利用するデータ（XML、MDBファイル）

¥MYXMLSERIALIZE_1
シリアル化の基本サンプル

¥MYXMLSERIALIZE_2
XML文書の作成と復元サンプル

¥MYXMLSERIALIZE_INSERT
データの追加サンプル

¥MYXMLSERIALIZE_DELETE
データの削除サンプル

シリアル化の基本

XML文書を作成するには、いろいろな方法があります。テキストエディタでコツコツ入力したり、Office 2003を使ったり、既存データをXSLTで変換したり。VB.NETで、XmlDocumentクラスのCreateElementやCreateTextNode、XmlTextWriterクラスのWriteStartElementなどのメソッドを使い、生成プログラムを書く方法もあります。でも、どの方法も、ちょっと手間がかかりそう!!　そこでお勧めしたいのが、シリアル化機能です。

.NET Frameworkでは、オブジェクトの状態を保ったり転送しやすいように、直列並びに変換（＝シリアル化）しておき、必要になれば元の形に復元（デシリアル化）する機能がサポートされています[注1]。XMLシリアル化機能を使うと、XmlElementオブジェクトやXmlNodeオブジェクトなどを、XMLストリームに変換することができます。また、逆シリアル化機能により、XMLストリームから元のオブジェクトを復元することができます。

単純にいえば、.NETでは、XMLを生成するだけでなく「保存し、必要になったときに利用できる機能が用意されている」ということです。

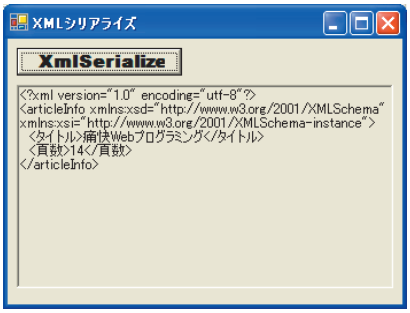
なお、本稿ではポイントとなる処理部分のみ説明してゆきます。ソースコードの全体は、付録CD-ROMに収録しているサンプルプログラムを参照してください

簡単なXML文書を作る

では、さっそくシリアル化機能を使ってみましょう。手始めに、リスト1のような、<タイトル>と<頁数>という2個だけの要素からなる簡単なXML文書を作成してみます（サンプルmyXml

注1）NET Frameworkは、XMLシリアル化だけでなくバイナリシリアル化もサポートしていますが、XMLは標準的なデータ互換フォーマットであり、Webベースでデータの共有や再利用を行なう形式として最適です。XMLシリアル化機能は、機密性の高い情報よりも、公開したり共有したり二次利用するデータに対して使うとよいでしょう。

図1：サンプルmyXmlSerialize_1の実行結果



Serialize_1)。

このサンプルでは、画面上に配置したボタンをクリックすることにより、シリアル化を実行しています。さらに、作成されたXMLファイルを読み込んで、TextBoxコントロール内に表示させています（図1)。

読み込む名前空間は以下の3つです。

- **System.Xml.Serialization名前空間：シリアル化の実現**
- **System.IO名前空間：生成されたファイルの出力**
- **System.Xml名前空間：作成されたXML文書の読み込み／表示**

シリアル化や逆シリアル化を行なうには、XmlSerializerクラスを使います。

シリアル化対象のクラス定義

シリアル化の対象とするarticleInfoクラスを定義し、title（String型）とpage（Integer型）のメンバを定義します。<タイトル><頁数>という日本語の要素名で書き出すため、XmlElement Attribute属性を使って、要素名を変更しています。この部分のコードがなく、たとえば「Public title As String = String.Empty」のように記述すると、変数名の<title><page>という要素名で書き出されます。

リスト1：シリアル化機能を使って作成するXML文書

```
<?xml version=”1.0” encoding=”utf-8”?>
<articleInfo>
  <タイトル>痛快Webプログラミング</タイトル>
  <頁数>14</頁数>
</articleInfo>
```

```
Public Class articleInfo
  <XmlElement("タイトル")> Public _
   title As String = String.Empty
  <XmlElement("頁数")> Public _
   page As Integer = 0
End Class
```

Button1_Clickプロシージャ ～シリアライズ処理

まず、Path.GetDirectoryメソッドを使って、パスを取得します。このサンプルでは、作成されたXMLファイルを¥myXmlSerialize_1¥bin¥dataフォルダに保存しています。そこで、引数にApplication.StartUpPathを指定して、アプリケーションの実行パス（¥myXmlSerialize_1¥bin¥）の親ディレクトリ(myXmlSerialize_1)を取得しています。

```
Dim appStartPath As String = _
  Application.StartupPath
Dim filePath As String = _
  Path.GetDirectoryName(appStartPath)
```

その上で、XMLシリアル化の対象となる新しいarticleInfoオブジェクトを生成し、titleプロパティとpageプロパティに値を指定します。

```
Dim myArticle As articleInfo = _
  New articleInfo
With myArticle
  .title = "痛快Webプログラミング"
  .page = 14
End With
```

次に、シリアル化するクラスの型で初期化されたXmlSerializerオブジェクトを生成し、書き込むXMLファイル名で初期化されたStreamWriterオブジェクトを生成します。StreamWriterの生成時に文字コードを指定しない場合はUTF-8になります。また、第2引数にFalseを指定しているため、このファイルは上書き保存となります。

```
Dim mySerializer As XmlSerializer = _
  New XmlSerializer(GetType(articleInfo)
Dim writer As StreamWriter = _
  New StreamWriter( _
  filePath & "¥data¥articleInfo_1.xml", _
  False)
```

実際に書き込むには、XmlSerializerクラスのSerializeメソッドを使用します。「XmlSerializer.Serialize (Stream, Object)」の書式で、StreamWriterオブジェクトを使って、シリアライズされたオブジェクトをファイルに書き込むことができます。

```
mySerializer.Serialize(writer, myArticle)
```

既存データの利用 ～XML文書の作成と復元

先ほどのサンプルmyXmlSerialize_1を発展させて、今度はAccess 2003で管理されているデータをもとに、XML文書を作成してみましょう（サンプルmyXmlSerialize_2)。実務では、既存のデータを利用してXML文書を作成するケースが少なくないからです。