クラスライブラリ活用法
このクラスはココに効く～文字列基礎編～

大澤 文孝
OSAWA, Fumitaka

Technology Tools

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

Level

★★★☆☆

Samples

はじめに

.NET Framework環境でアプリケーションを構築する場合、さまざまな場面でクラスライブラリを利用します。

.NET Frameworkには多数のクラスライブラリが備わっており、多種多様な処理ができます。しかし逆に、クラスライブラリの数が多すぎるために、慣れない人にとっては、どのクラスライブラリをどのように使えば目的の処理を実現できるのかを理解するのが難しい側面もあります。

そこで本稿では、文字列処理を中心に、アプリケーションを構築するうえで、比較的よく使われそうな処理を、クラスライブラリを使ってどのように組み立てるのか、サンプルプログラムを通して紹介します。

入力された文字のチェック

ユーザーインターフェイスを備えるアプリケーションを構築するときに、必須とも言えるのが「ユーザーが入力した文字のチェック」です。

ユーザーが、数値しか入れられないテキストボックスに英字や漢字を入力したり、英数字しか入れられないテキストボックスに漢字を入力したりした場合には、エラーとしてはじく処理が必要です。

入力された文字をチェックする方法はいくつかありますが、比較的簡単で応用が利くのは「正規表現を使った方法」です。

.NET Frameworkでは、System.Text.RegularExpressions名前空間に備わるクラスを使うと、正規表現を使った処理ができます。

ここでは実例を見ながら、正規表現を使って、文字種をチェックする方法を見てゆきます。

数字かどうかを調べる

まずは、文字列が数字のみで構成されるかどうかを調べるプログラムを示します。そのようなプログラムは、リスト1のようになります。

リスト1：数字かどうかを調べるプログラム

```
Imports System.Text.RegularExpressions

Private Function CheckNumber( _
 ByVal strCheck As String) As Boolean
  Dim m As Match = Regex.Match(strCheck, "^\d+$")  ①
  If m.Success Then                                ②
    ' 合致したので数字
    Return True
  Else
    ' 合致しないので数字ではない
    Return False
  End If
End Function
```

リスト1のCheckNumber関数は、引数strCheckに与えた文字列が数字のみで構成されている場合にはTrue、そうでない場合にはFalseを返すというものです。

たとえば、次のように数字を与えると、Trueを返します。

```
Debug.Write(CheckNumber("1234"))
```

それに対して、次のようにひとつでも数字以外の文字が含まれていれば、Falseを返します。

```
Debug.Write(CheckNumber("1a23"))
```

正規表現に合致するかどうかを調べるには、リスト1-①の部分にあるようにRegexクラス[注1]のMatchメソッドを使います。

```
Dim m As Match = _
 Regex.Match(strCheck, "^\d+$")
```

Matchメソッドの第1引数には調べたい文字列を、第2引数には正規表現を渡します。すぐあとに説明しますが、第2引数に与えている「^\d+$」は、数字のみで構成されていることを示す正規表現です。

Matchメソッドの戻り値は、Matchオブジェクトとなります。

Matchオブジェクトには、Successというプロパティがあり、正規表現に合致した場合にはTrue、そうでない場合にはFalseを保持します。

そこでリスト1-②にあるように、Successプロパティを調べれば、調査対象の文字が正規表現に合致するかどうかを調べることができます。

```
If m.Success Then
  Return True
Else
  Return False
End If
```

正規表現の基本

リスト1に示したように、Regex.Matchメソッドを使えば、正規表現による調査ができます。つまり正規表現の記述方法さえ理解できれば、文字列が正しい文字種で構成されているかどうかを調べることができることになります。

Matchメソッドの第2引数に与える、正規表現で書かれた文字列のことを正規表現の「パターン」と呼びます。パターンの記述方法を理解することが、正規表現を使って文字列をチェックするときの肝となります。

とはいえ正規表現のパターンは、少し複雑で、本稿では全部を記すことはできません。

そこで本稿では、主なパターンの意味と、よく使う正規表現のパターン例を示すに留めます。

詳細については、本稿末に示した参考文献などを参照してください。

パターンとは

パターンとは、対象となる文字列を先頭から調べていったときに、合致するかしないかを示す文字種のことです。

パターンは、いくつかの文字の集まりであり、調査対象となる文字列中の文字の並びを示します。

たとえば、「abc」というパターンは、調査対象となる文字列に含まれる、『a』と『b』と『c』が順に並んだ部分」ということを示します。つまり、Regex.Matchメソッドのパターンとしてabcを与えたときには、調査対象となる文字列中に、『a』と『b』と『c』が順に並んだ部分（abc）」があれば、Successプロパティ はTrueとなります。

```
Dim str As String = "123abcefg"
' 下記の結果はTrueになる
' なぜなら変数strに『abc』と
```

注1）より正確には、System.Text.RegularExpressions名前空間のRegexクラスです。ただし、説明を簡潔にするため、名前空間は省略します。