.NET Frameworkで作るWindowsサーバー

作れればわかるアプリケーションの動作とメカニズム

秋月 巌 AKIZUKI, Iwao〈秋月巌ソリューション事務所〉▶http://www.akizuki.co.jp/

第15回 .NET Framework 2.0で作るプロキシサーバー―その3―～カスタマイズ可能なプロキシサーバーの作成、そして文字コードとストリーム

カスタマイズ可能なプロキシサーバー

前々回はクライアント部にWeb Clientクラスを使ったプロキシサーバーを紹介し、前回はサーバー部、クライアント部ともにSocketクラスを使用したプロキシサーバーを紹介した。今回は、前回のサンプルプロキシサーバーをカスタマイズし、次のような機能を追加してみた。

- 特定の拡張子（EXE）を持つファイルのダウンロードの禁止
- 特定のドメインのサイト（xxx.com）へのアクセスの禁止
- 特定の文字列を含むWebページの取得の禁止
- 特定の文字列を含むWebページの文字列を置き換え
- すべてのページに広告を表示

カスタマイズのためのプロシージャは分離されているので、読者はプロキシサーバーの本体のプログラムを理解していなくてもプログラムを追加できる。

さて、前回のプロキシサーバーはターゲットのWebサーバーが送り返したデータを、バイナリデータとしてそのまま転送していたので、文字コードの問題は発生しなかった。しかし、今回のプロキシサーバーは内容を文字列として分析する必要があるので、適切な文字コードでエンコード／デコードをする必要がある。また、前回からはバイナリデータの転送に対応するために、受信データバッファにStringBuilderクラスではなくBinaryWriterクラスを使用するように変更した。BinaryWriterクラスは、Streamオブジェクトを使用してデータを一時的に保持することができる。

文字列と文字コードの関係を表示するサンプルプログラム

というわけで、新しいプロキシ

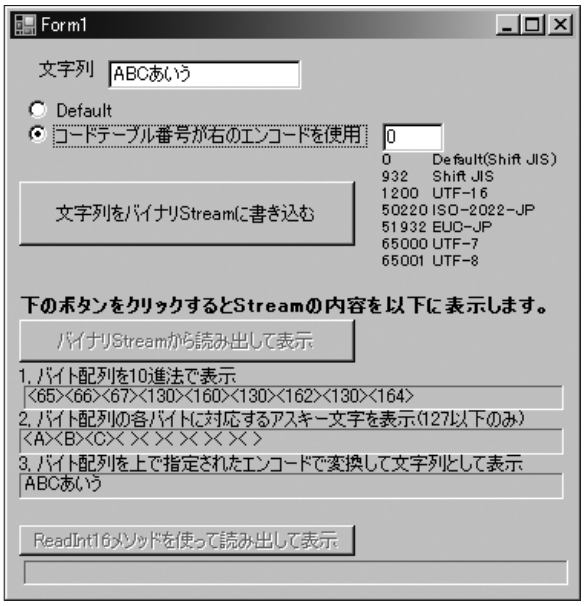
レベル ››› Level
1 2 3 4 5

言語 ››› Language
- Visual Basic

ツール ››› Tool
- Visual Studio 2005 Professional

サンプル ››› Sample
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、
http://www.shoeisha.com/mag/windev/
からダウンロード可能です。

図1：サンプル1「文字列エンコードプログラム」



サーバーの説明をする前に、これら文字コードとStreamオブジェクトの扱いをまとめて解説する。そのためのサンプルとして用意したプログラムがサンプル1の文字列エンコードプログラム（図1）である。このサンプルプログラムは文字列を、ユーザーが指定したエンコードで変換し、バイナリデータとしてStreamオブジェクトに格納し、それをバイナリの十進表記で表示したり、任意のエンコードでデコードして文字列として表示する機能がある。

文字を使ってコミニュケーションをするには、コンテンツ製作者が書いた文章がそのまま、読む人の手元に表示されればいいわけである。もっとも妥当な方法は、文字を映像として記録し、それを読み手に渡す方法である。これならば文字コードは無関係である（画像フォーマットの問題はある）。本というのは、この方法を採用した媒体だということができる。文字で内容を伝えるすべての本がそうだと書こうと思ったが、点字本のような特殊なものもあるので一概には言えない。

文字コードを使う理由

コンピュータが文字を映像として扱わない最大の理由が、データ容量にあることはほとんどの人が知っているだろう。画面いっぱいの文字を表示する画像ファイルは白黒で圧縮しても数十キロバイトあるが、テキストファイルならば数キロバイトで済む。たとえば、あるテキストを白黒イメージでGIFファイル化したものは49KBのファイルサイズだが、テキストファイルで保存すると4.9KBなので10分の1のサイズである。もっとも、GIFファイルとして圧縮している時点でエンコード／デコードをしているとも考えられるし、また、読み取れる限界まで解像度を落とすことで画像ファイルのサイズを小さくすることもできるので、あまり有効な比較とはいえない。それでも文字の記録された画像データを、読み取れる状態のまま、GIFファイルよりさらに10分の1にするのは難しいのではないだろうか。

もちろん、文字を画像ではなくコードとして扱う理由はデータサイズの問題だけではない。とはいえ、現在のコンピュータが文字を文字コードとして扱うのは、あまりにも基本的な前提となっているので、なぜ画像ではなく文字コードなのかと考えると、なにもかもゼロから考えないといけないように思えてくる。人間にとって文字とは、図形の発音への変換だと考えることができる。われわれは、たとえば「あ」という形を見ると、それに対応した発音を思い出すような変換コード表を頭の中にもっている。もっとも、これはひらがな、カタカナ、アルファベットといった表音文字の場合だけである。漢字のような表意文字の場合、話は複雑になる。「机」という表意文字を見た場合、私達は机という具体的なイメージや機能を思い浮かべると同時に「つくえ」という発音も思い浮かべる。しかし、実は同様のことが表音文字でも起こり得る。「コンピュータ」という文字列を見たとき、実は私達はコンピュータという発音を思い浮かべてコンピュータをイメージするのではない。「コンピュータ」という文字列の形が、直接コンピュータという概念に結びつく。だから「こんぴゅーた」と平仮名で書かれると、それがコンピュータであることに思い至るのに一瞬遅れるのである。この場合、一度、音声化してから意味に変換しているというよりは、われわれが、ひらがな⇔カタカナ、という直接の変換能力を持っていて、それが実行されるためであるように思う。ところで、たとえば「帰る」という意味を表わすのに「カエル」と表記したとしよう。私達は、それを「帰る」と判断するよりは、両生類のカエルと判断するだろう。しかし「イエニカエ