.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14回 .NET Framework 2.0で作るプロキシサーバー －その2－ ～Socketクラスをクライアントに使う

クライアント部にもSocketクラスを使う

前回作成したプロキシサーバーでは、他のWebサーバーにアクセスするためにWebClientクラスを使用した。WebClientクラスはHTTPプロトコルが実装されているのでプログラムの記述が簡単になる。とはいえ、プロキシサーバーのクライアント部として使うと、一部、適切に表示できないページがあった。そこでSocketクラスをクライアント部にも用いたのが今回紹介するプロキシサーバーである(図1)。Socketクラスを使用する場合、Webブラウザから受け取った要求をそのまま別のWebサーバーへの要求にまわすので、Webブラウザの偽装の役割は果たせなくなるが、より多くのWebページに対応できるようになる。

これで、前回のプロキシサーバーでは表示不可能だったYahoo!の会員サイトも表示できるようになるのだが、たとえばHotmailは使用することはできない。Hotmailを使用するにはHTTPのCONNECTコマンドを実装する必要があるので、これも対応したのだが、Hotmailが使用するMicrosoftのオンライン認証サービス「パスポート（Passport）」を通らないようである。また、いくつかのページでは表示にやけに時間がかかる場合もあるなど、実用に使う上でいくつかの問題点がある。

ところで、プロキシサーバーを作ることに興味を持つ読者がいるのか少々疑問なのだが、プログラマ

図1：プロキシサーバー

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。

ブルなプロキシサーバーはあまりないので、案外、要望はあるのではないかと考えている。たとえばWebサーバーの場合、IISならばWebアプリケーションやフィルタアプリケーションを開発できるので大概の要望はIISの機能で実現できてしまう。自作のWebサーバーだからできるという点が、あまり思いつかない。

社内からのインターネットアクセスを、複雑な規則を用いて制限したい企業ユーザーや、あるいは子どものインターネットアクセスを制限したい親がいることなどを考えると、.NET Framework用のソースコードが公開されたプロキシサーバーの存在には意味があると考えたのである。プロキシサーバーはWebブラウザからの要求とWebサーバーからの応答を完全に管理できるので、プログラムを書かないと実装できないような複雑な規則を使ってインターネットへのアクセスを制限することができる。

複数のWebブラウザからの同時要求に対応する

今までSocketサーバープログラミングのテンプレートであるAST（Asynchronous Single Threading）サーバーを使ってサーバープログラムを作ってきた。また、ASTサーバーを紹介した2006年4月号では、非同期通信クライアントプログラムテンプレートであるAsyncクライアントを同時に紹介した。しかし、これをプロキシサーバーのクライアント部に使うには、かなり工夫する必要がある。というのはAsyncクライアントはあくまで、1アプリケーション1接続として設計されているからである。プロキシサーバーの場合、複数のWebブラウザから同時に要求があった場合、同時に複数のクライアントで目的のWebサーバーに接続しないといけない。

今回のプロキシサーバーでは「Form1」フォームとは別に「Module1」モジュール（Module1.vb）に、クライアント部の機能を集中させている。「Module1」モジュールは、Asyncクライアントの「Form1」フォーム（Form1.vb）から、フォームのためのプログラムコードを削ったものがベースとなっている。これでひとつのアプリケーションで複数のクライアントSocketを使用することが可能になる。

非同期Socketを使ってプロキシサーバーを実行する

Socketクラスを同期通信で使う場合に比べて、非同期通信で使う場合はプログラムの実装がやっかいである。接続、データの受信、データの送信といった各処理の開始と実行を別のプロシージャに実装しなければならないということがその最大の理由である。そして、データの受信処理時にプロシージャの引数経由で渡されていく一時保存したデータとSocketオブジェクトを格納しているオブジェクトの扱いがまた複雑なのである。このオブジェクトはASTサーバーでは「StateObject」クラスとして実装している。クライアントでひとつの接続しかない場合、たいした問題はない。データの受信時に呼び出されるイベントプロシージャで、受信したデータを「StateObject」オブジェクトに投げ込んでいくだけだからである。対象となる「StateObject」は、この場合別に引数で渡さなくていい。しかし、サーバー側で複数のクライアント接続を受け付ける場合や、複数の接続を持つ場合「StateObject」オブジェクトは、それぞれの接続に対してひとつずつ用意する必要がある。話が複雑になるのは複数の接続をした場合に、それらに対応した「StateObject」オブジェクトを識別するために、通常では引数で渡し続けることが必要だからである。

プロキシサーバーの場合、どのように「StateObject」オブジェクトが使われるかを考えてみよう。

プロキシサーバーはWebブラウザからのリクエストを受け取る。この時点で、受信したリクエストを格納するための「StateObject」オブジェクトが必要である。各接続ごとに「StateObject」オブジェクトはいくつも生成されるので、コールバックプロシージャで正しい「StateObject」オブジェクトを引き続き利用できるように、通常はコールバックプロシージャの引数に指定される。

データ受信時に呼び出されるコールバックプロシージャの中で、プロキシサーバーはターゲットとなるWebサーバーにWebブラウザからの要求を転送し応答を待つ。このときにWebブラウザからの接続に対応した別の「StateObject」オブジェクトが必要となる。

さて、Webサーバーからの応答を受け取るのは、クラ