.NET Frameworkで作る Windowsサーバー

作ればわかる 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16回 SQL Serverとクライアントプログラムを中継するSQLプロキシサーバーを作る－その1－

物語の前に物語はある

数日前まで、私はこの記事を「ロスト・イン・ラマンチャ」というドキュメンタリービデオの話で書き始めるつもりだった。「ロスト・イン・ラマンチャ」は、ドンキホーテを題材にした映画の撮影が、度重なる不運のもとについに中断に至る過程を撮影したドキュメンタリービデオである。というのは、今回の記事のためにSQL Server用のプロキシサーバーを作り始めたが、多くの問題が発生し、とても締め切りまでに間に合いそうもなかった。締め切りに間に合うような代替の案もなかったので「ロスト・イン・ラマンチャ」を引き合いに出して、壮大なる言い訳を展開する予定だった。しかし、なんとか、サンプルプログラムとして人に見せられるようなものにはなったので言い訳はしないですんだのである。

さて今度は……

今までWebサーバー、メールサーバー、プロキシサーバーと作ってきた。次に代表的なサーバーといえばデータベースサーバーである。これは魅力的なテーマである。しかし、データベースサーバーは他のサーバーと違って、単独で完結するわけではなく、そのデータベースサーバーを利用するプログラムがあって初めて用を足すものである。となると当然、プログラミングインターフェイスが必要である。とはいえ、独自インターフェイスを作ったとしても、私が設計したインターフェイスを新たに学習してプログラミングをしてくれる人もいないだろう。誰もプログラムを作らないデータベースサーバーなんて何の意味もない。OLE DBドライバを書く

レベル >>> Level
1 2 3 4 5

ツール >>> Tool
- Visual Studio 2005 Professional
- SQL Server 2005

言語 >>> Language
- Visual Basic

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。

という手もあるわけだが、それは本連載の手にあまる。

そこで考えた別のアイデアがBizTalk Serverのようなものを作るというものである。実は私はBizTalk Serverが、どういうものなのかよくわかっていない。XMLを使ってデータ交換をするサーバーだという簡単な説明を発売当時に読んだだけである。そこで、BizTalk Serverについて調べ始めた。できそうな機能だけ、実装したっていいわけである。ところが、やはり、BizTalk Serverがどういうものなのかがよくわからない。社内にある複数のシステムをXMLを使って統合するというのが、その目的であるらしい。たとえば、発注システムと在庫管理システムが別々に存在しているとする。発注すれば在庫が増えるわけだから、この2つのシステムが関連して動作するべきである。しかし、統合されていないシステムの場合、ユーザーは発注システムでデータを確認したら、それを在庫管理システムに再入力しなければならない。再入力の方法はファイルのインポートかもしれないし、モニターで確認して手で行なうのかもしれないし、あるいは印刷されたものを見て入力するのかもしれない。しかし、両システムが統合されていれば発注から在庫の管理までが一貫したシステムとして動作することができる。これを実現するのが、BizTalk Serverである。なるほど、どうやら、すばらしい製品らしい。BizTalk Serverをインストールすれば自動的に統合されるわけではなく、プログラミングが必要なので、当然、BizTalk Serverはプログラミングを支援する強力な機能を提供するはずである。しかし、具体的に、どのような機能を提供するのかが、なかなか見えてこない。結局、最後まで理解できなかったのだが、資料を見ているときにBizTalk ServerはSQL Serverプロトコルを実装しているという記述が目にとまった。なるほど、社内プログラムを統合するには、当然、データベースサーバーであるSQL Serverのプロトコルも理解すれば便利だろう。ユーザープログラムがSQL Serverにアクセスするのをキャプチャできれば、システムの統合に役立つはずである。

SQLプロキシサーバー

それまであまり具体的に考えたことがなかったのだが、SQL Serverも当然、プロトコルを使ってクライアントとのやり取りを行なっているわけである。ならば、その仕組みを理解できればSQL Serverのふりをするプログラムができるはずである。たとえば、SQL Serverのふりをする自作データベースサーバーならばADO.NETを用いてユーザープログラムを開発することができる。これならば、新しいプログラミングインターフェイスの学習をすることなくプログラムを作ることができる。

今まで、この記事ではプロトコルが公開されているサーバープログラムだけを作ってきた。プロトコルが公表されていないものを作ったことはない。そこで、まず、SQL Serverのプロトコルを調べるために作ったのが、今回のSQL Server用プロキシサーバー（以下SQLプロキシサーバー）である。このサーバーはまだ開発中なので、かなりの問題があるが、それでもSQL ServerとADO.NETがどのようなやり取りをしているかを目で確認することができる。SQL Server用のプロキシサーバーとはいっても、TCP/IPでやり取りされるデータをそのまま転送しているだけなので、汎用のWinSockプロキシである。とはいえ、現在のところ、SQL Serverで使うことだけを前提として開発しているので、今後、SQL Serverに対応した固有の機能を追加する可能性もある。

プロトコル調査用に作り始めたが、意外と実用性もある

プロトコルの調査用に作ったものだが、案外実用性もあるのではないかと考えている。たとえば、グローバルIPを持つホストで稼動する社外のSQL Serverに、直接はインターネットに繋がっていない社内LAN上のコンピュータからアクセスする場合などにも使用することができる。これは、Web用のプロキシサーバーの効用に似ている。もっとも、Web用のプロキシサーバーは、接続先のWebサーバーが不特定であるのに対して、SQLプロキシサーバーは特定のサーバーにのみアクセスするのが、動作上の大きな違いである。プロトコルには通常接続する相手に関する情報は含まれない。なぜなら、現在、接続している相手が接続相手だからである。

Web用のプロキシサーバーの場合、Webブラウザがプロキシサーバー用の指定を行なう必要がある。そうする