ざっくりわかるインターネットプログラミング

ダイアモンドアプリコット電話研究所
山崎 はるか
YAMAZAKI, Haruka
http://www.nda.co.jp/

第2回 Web（HTTP）接続に挑戦

level

1 2 **3** 4 5

Technology Tools

- ☑ Visual Basic
- ☐ Visual C#
- ☐ Visual C++
- ☐ SQL Server
- ☐ Oracle
- ☐ Access 2002
- ☐ ASP.NET
- ☐ Other:

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

功罪

いま「ファイルの読み込み」を「ハードウェアの物理的動作」としてイメージしている人は、あまりいないんじゃなかろうか。

原因のひとつは“音”だろうな。

以前は、フロッピーやら磁気テープを読みにいくと「カチャ」と音をたてたものだが、いまはHDDを読みにいってもランプが点滅する程度。メモリへのアクセスと大差ない。

そのため、多くの人が「ファイルアクセス＝OS上の論理的な動作」と捉えてしまい、デバイスを物理的に制御している感覚は乏しいと思う。

だからかな。今のプログラマは、メモリの移動もファイルの移動も、それを感覚として実感できるポイントを作ろうとは思わないらしい。

.NET Frameworkのもたらした恩恵は「クラスプログラミング」であることはいうまでもない。だが、その功罪として「宣言してるのか、実行してるのかあいまい」になった。

オブジェクトの相手がメモリなのか外部記憶デバイスなのか、さらにはインターネットサーバーなのか、ダムの放水ゲートなのか。それが「隠蔽」された状態になって、よくわからない。でも実際、動作している。こっちに音や振動が体感できないところでね。

インターネットを含む「通信プログラミング」を学ぶときも、この「隠蔽」は障害になりかねない。

もちろん、クラスはとても便利だし、通信プログラムでも多用するけれども、それを有効に使える人は、それぞれの通信デバイスがどこにあるか、具体的にイメージできている人に限られる。

一方、何がなんだかワケがわかってない初心者に、向こう側が隠蔽されたクラスメンバの羅列を見せたところで、さっぱり何の役にも立たないはずである。

人の脳というのは、どこで何が起きるかイメージできて、はじめて理解で

きる。そういう構造なのだから、簡潔に書きゃいいってもんじゃない。

前回から言ってるように、「設定／接続／交信／切断」に、それぞれの手順を分けることがいかに大切か、“実物”をもってお見せすることにしよう。

設定／接続／交信／切断

ちょいと、サンプル1を見ていただきたい（リスト1）。

これはWebRequestクラスとWebResponseクラスを用いて、Webページにアクセスし、ヘッダーをとってくるサンプル。

うむ。一般的なサンプルの書き方だとこうだな。クラスプログラミングの作法としては正しい。

が、こんなことするから、動作の実態がわからん。

サンプル2を見てみよう（リスト2）。

同じことをしてるサンプルなのに、「設定／接続／交信／切断」にそれぞれの手順を分けるだけで、見ちがえるようにわかりやすくなる。

ブレークポイントをつけて実行するとわかるが、WebRequest.Createでは、まだWebを見にいってはいない。

GetResponseで、はじめてWebにアクセスを行なう。こういう事実はサンプル1ではわからない現象だな。

そしてWebResponse.Closeで切断する。

切るのがWebRequest.Closeでないのは、なぜか。それはWebRequest、WebResponseそれぞれのリファレンスを見ればわかることだが、じつは「どっちで切ってもいい」ってことが書かれていたりする。

つまり、メソッドの受け側（WebResponse）でも接続をブった切れるのだから、このインスタンスの取り扱いは慎重でなければならず、WebRequestを維持したい／WebResponseの内容を使っていたいのであれば、用がなくなるまで解放してはいけないことがわかる。

なら明示的には、WebResponse.Closeで管理したほうが、この場合はよいことになる。好みもあるだろうけどね。

さて。すると、実用的な発展はサンプル3になる（リスト3）。

エラートラップのしやすさに注目していただきたい。

「設定／接続／交信／切断」を明確に分けてプログラミングすれば、トラップしても、その全体構造は変化しないことに気がつくだろう。

これは、通信という原理そのものが「設定／接続／交信／切断」で、互いに連携しつつも、その動作は独立しているからである。

古い言い方が許されるなら「セグメント化されている」とでも言おうか。

その基本構造にのっとってプログラムを組み上げれば、最後まで美しく大きくすることができるというわけだ。雪の結晶ができる原理と同じようにね。

リスト1：サンプル1（一般的なサンプル）

```
Dim myreq As WebRequest = WebRequest.Create("http://www.shoeisha.co.jp")
Dim resp As WebResponse = myreq.GetResponse()

MessageBox.Show(resp.Headers.ToString)
resp.Close()
```

リスト2：山崎はるか流

```
Dim myreq As WebRequest  ' 送信処理系
Dim resp As WebResponse  ' 受け取り解析

' [設定]
' WebRequestプロパティ＆メソッド群
myreq = WebRequest.Create("http://www.shoeisha.co.jp")

' [接続]&[交信]
' サーバーにリクエストしてその結果をWebResponseで調べる
resp = myreq.GetResponse()
MessageBox.Show(resp.Headers.ToString)  ' 例：ヘッダーを表示

' [切断]
resp.Close()
```