.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

第3回 Webサーバーを自作する－その3－

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

Windows Server

level

1 2 3 **4** 5

Technology Tools

- ☑ Visual Basic
- ☐ Visual C#
- ☐ Visual C++
- ☐ SQL Server
- ☐ Oracle
- ☑ Access
- ☐ 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。

サーバーアプリケーションをWindowsサービス化する

前回までで、自作のWebサーバーはバイナリファイルも操作することができるようになったほか、Webアプリケーションの開発も可能になった。基本的な機能は満たされたので、今回はこのWebサーバーをWindowsサービス化することにした。

WebサーバーはTCP/IPサーバーアプリケーションであるが、それとWindowsサービスかどうかとはまったく無関係である。Windowsサービスはアプリケーションの“実行形態”であり、TCP/IPサーバーはアプリケーションの“動作形態”である。ただ、TCP/IPサーバーは常時起動が要求されることが多いため、Windowsサービスとして実行されることが望ましいかもしれない。とはいえ、私はあまりWindowsサービスが好きではない。なぜなら、Windowsサービス用のプログラムはインストールしないといけないし、そのインストールやアンインストールも私には不明な部分があるからである。だから、今回はWindowsサービス化に合わせてWebサーバーの機能も強化したが、通常のWindowsアプリケーション版も同時に提供する。

これならばインストールしなくても動作するから、簡単にアプリケーションの動作を確認することができる。そして今後の機能強化もWindowsアプリケーション版に対してのみ行なう。現在のサンプルWebサーバーは実行ファイルをダブルクリックするだけで実行できるので、その簡便性を失いたくないからである。ただし、このWindowsアプリケーション版も、いつでも簡単にWindowsサービスに変更できるように実装を進める。つまり、ユーザーインターフェイス部とサービス部を分離して開発するようにする。Windowsサービスはユーザーインターフェイスを持つことができない。そのため、サービスを設定する

マネージャプログラムは別アプリケーションとして実装する必要がある。IISやSQL Serverを思いだしてほしい。どちらも、Windowsサービスとして実装されているが、設定を実施するためのプログラムであるインターネットサービスマネージャやSQL Enterprise Managaerは、別のWindowsアプリケーションとして実装されている。

今までのサンプルでは、ユーザーインターフェイス部とサービス部も同じVisual Basic .NETのフォームに実装していたが、今回はこれを切り離す。また、Windowsサービス版用に、サービスの起動マネージャも作成する。

Windowsサービス化するメリット

一方でWindowsサービス化するメリットも考えるべきだろう。まず、ログインしなくてもサーバープログラムを実行できることが挙げられる。しかし、ログインしないことにどれくらいのメリットがあるのかも疑問である。たしかに、ログインするとユーザーインターフェイスが操作可能なので、管理者以外の誰かが勝手に操作してしまうというようなことがあるかもしれない。しかし、それならば「コンピュータのロック」機能を使って他の人が触れなくすることも可能である。

また、起動時に自動的に実行されるというのもメリットといえるだろうか。しかし、これもスタートアップフォルダに入れるとか、アプリケーションの自動起動を指定すれば他の方法でも実現可能である。ただし、Socketアプリケーションをスタートアップフォルダで指定して自動起動すると、正常に起動できない場合もあるので注意が必要である。

常時起動しているサーバーアプリケーションがいつもタスクバーにあると邪魔なので、起動時に画面に何も表示されないWindowsサービスのほうがいいという人もいるかもしれない。とはいえ、最小化時にタスクトレイにアイコンを表示するタイプのWindowsアプリケーションならば、それほど邪魔にはならないのではないだろうか。あるいはWindowsアプリケーションでも、どこにも表示しないという手もあるが、あまり一般的な方法とはいえない。今回、Windowsアプリケーション版では、タスクトレイにアイコンを表示する機能も実装した。

Windows NT系のサーバーOSでは、多くのWindowsサービスが同時に稼動している。これらをサービスマネージャで一括して管理できるというのは、やはり、メリットだといえるだろう。普段、ユーザーが操作する必要のないアプリケーションが、タスクバーやタスクトレイに並ぶのは見苦しいうえに使いづらい。

また、サービスは各サービスごとに独立したプロセスを使用しない。これこそがWindowsサービス化する最大のメリットといえるかもしれない。数多くのサービスが、それぞれひとつのプロセスを使用したら、リソースの消費もばかにはできないだろう。しかし、Webサーバーというのは、そのサーバーマシンにおいて重要な役割を持つ場合が多いと思うので、個人的にはWebサーバーのためにひとつのプロセスを使用してもいいのではないかと思う。

結局のところ、Windowsサービス化する決定的な理由もないのだが、しない決定的な理由もない。Windowsサーバーにおけるサーバーアプリケーションの作法としては、多分、Windowsサービスとして実行するのが正しいのだと思う。

サンプルプログラムの概要

サンプルプログラムは、次のような5つのプログラムによって構成される。ただし、今回解説するのはサンプル2までである。

サンプル1 **Windowsアプリケーション版Webサーバー**
（図1、¥Sample1_Webサーバー¥bin¥web_server.exe）

サンプル2 **cookie用サンプルWebアプリケーション**
（図2a・b、¥Sample1_Webサーバー¥bin¥webapp3.shtml、webapp4.shtml他）

サンプル3 **Webサーバー設定マネージャ**
（図3、¥Sample3_Webサーバー設定マネージャ¥bin¥web_server_manager.exe）

サンプル4 **Windowsサービス版Webサーバー**
（図4、¥Windowsサービス版¥Sample4_Webサーバー¥bin¥web_server_service.exe）

サンプル5 **Windowsサービス起動ツール**
（図5、¥Windowsサービス版¥Sample5_サービス起動ツール¥bin¥web_service_tool.exe）