ASP.NET INSIDE

ASP.NETをめぐる思考と試行とその冒険

第16回 WebアプリケーションとWindowsアプリケーションのコードを共通化する―その5―

aspx Webサービスを使用したデータの追加、削除、更新

秋月 巌 *AKIZUKI, Iwao*
秋月巌ソリューション事務所

Technology Tools

- ☑ Visual Basic .NET
- ❑ Visual C# .NET
- ☑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ☑ ASP.NET
- ☑ Internet Information Services
- ❑ Other:

Level

★ ★ ★ ★ ★

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、付録CD-ROMの\DOTNET\ASP_NETディレクトリに収録しています。

\SAMPLE
サンプルプログラムとソースコード

.NET 今までの集大成

さて、今回はこのシリーズの最終回として、今まで作成してきたaspx Webサービス[注1]を使用してデータの追加、削除、変更を行なうWebアプリケーションとWindowsアプリケーションの両方を作成する。両方のアプリケーションともグリッド形式でデータベースのテーブルの更新が可能である。更新に必要な処理は共通のaspxファイル（WebApp5b.aspx）に収められているので、このファイルの内容を書き換えることで両方のアプリケーションのデータベースアクセスのロジックを変更することができる。

.NET ストアドプロシージャとの比較

結論として、このシリーズが追求してきた「WebアプリケーションとWindowsアプリケーションのコードを共通化」という目的は達成できたのだろうか。この記事で紹介したサンプルでは共通化したのは単にデータベースアクセスのSQL文だけである。

今回紹介するアプリケーションの機能を必要とするだけならば、WebアプリケーションとWindowsアプリケーションの両方に同じSQL文を埋め込めばいいだけなので、個別で開発したほうが開発効率はずっといいし、保守性だってたいした差はない。差がでるのは、もっと複雑なデータベースアクセスロジックが組み込まれるようなケースである。とはいえ、そのような場

注1）本記事で使用している「aspx Webサービス」とは、通常のASP.NETアプリケーションであるaspxファイルを、Webサービスとして利用する方法のこと。aspx WebサービスはMicrosoftが提供するXML Webサービスと比較して、参照先の変更が簡単だということのほか、ASP.NET以外の知識が必要ないという特徴がある。この記事では、このサービスを呼び出すのにクライアント側でWebClientクラスを使用しているが、Webクライアントであればどのようなものでもサービスを利用することができる。

合でもストアドプロシージャにビジネスロジックを記述しておいて、それに両方のアプリケーションからアクセスするという方法もある。とはいえ、ストアドプロシージャを使う方法は、aspx Webサービスと比較して次のようなデメリットがある。

ストアドプロシージャの問題点

- 専用の開発スキルが必要
- 使用データベースに依存
- データベース専用言語のため、柔軟性、処理速度とも限界が低い
- ひとつの画面で複数の結果セットを扱う場合に複数のやり取りが必要

ただ、実際に今回のサンプルをいろいろと試行していて、プログラミングのコストの多くは画面まわりにあるのだと改めて感じた。Webフォームのような複雑なメカニズムを使ってすら、画面まわりのロジックをWebアプリケーションとWindowsアプリケーションで同一にすることはできない。結局、そのもっとも手間のかかる部分を共通化できないのならば、それ以外のコードを共通化することにどれだけの意味があるのかに疑問を抱いたのも確かである。

今までビジネスロジックを分離させる3階層システムがずいぶんと声高にいわれてきたが、SQLデータベースを使う場合、SQL文自体がかなりのビジネスロジックを含むことができるので、そこからさらにビジネスロジックを分離することにどれだけ意味があるのかという疑問と根は共通である。

.NET メリットとデメリット

結局、結論は「必要な場所で、必要に応じて使うべきだ」ということである。この記事で紹介したサンプルではデータベースの簡単な更新処理にまでaspx Webサービスを使用した。私自身、画面処理以外のすべてをaspx Webサービスを使用してコードを共通化することを企んでいたのだが、そのような使い方はやはり過激すぎるだろう。とはいえ、Windowsアプリケーションからaspx Webサービス化することで、MSDEを多数ユーザーで同時使用した場合のWindowsアプリケーションのパフォーマンス劣化を防ぐという効果を期待するならば、これもひとつの方法である。詳しくは本誌の過去の記事を読んでもらうとして、これでWindowsアプリケーションから多数ユーザーがMSDEに同時アクセスした場合のパフォーマンスは圧倒的に高くなる。

では、どのような場合にaspx Webサービスを使用すべきかというと、これは「どういうときにXML Webサービスを使用するべきか」というのと同じテーマになってしまう。ただ、配置変更が難しいXML Webサービスが、どちらかというとインターネットを経由した企業間の相互サービス提供に向いているのに対して、aspx Webサービスは企業内システムのコンポーネント化に向いていると思う。もともとコンポーネントには“分業”という側面と、“コードの冗長性を排除する”という側面の両方があるわけだが、そのどちらのメリットを望むかによって、使い分ければいいだろう。

私は他の企業にサービスを提供する予定はないので、aspx WebサービスがあればXML Webサービスはいらないと思っている。ところで、私がWebサービスの提供を受けるとしても、XML Webサービスよりもaspx Webサービスのほうがいいな、というのが本音である。aspx Webサービスは、Webアプリケーションであるならば、あらゆる意味で環境を選ばない。もちろん、XML Webサービス対応の環境である必要もない。それはHTMLのクエリ文字列やフォームを使って値を渡せるからである。ただし、プログラム内部から呼び出すには、このサンプルにおけるWebClientクラスのような“Webブラウザのふり”をする仕組みが必要である。

WebアプリケーションとWindowsアプリケーションのコードを共通化することを目的として始めたこのシリーズだったが、画面まわりの扱いの差の大きさを考えると、先月号でも書いたようにDHTMLを使用してWindowsアプリケーション並みの操作性をもつWebアプリケーションを開発したほうがいいのではないかという気がしてい