Visual Studioで構築するエンタープライズシステム

Application Architecture for .NETの利用例

第10回 プレゼンテーションレイヤーのフレームワーク実装

株式会社CSKシステムズ
IT生産技術部
中垣 健志
NAKAGAKI, Kenji

はじめに

「新年明けましておめでとうございます！」

Level

1 2 3 4 5

Technology Tools

- ✓ Visual Basic
- ✓ Visual C#
- ○ Visual C++
- ✓ SQL Server
- ○ Oracle
- ○ Access
- ✓ ASP.NET
- ✓ Other:
  ↓
  Visual Studio .NET 2003

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。

今日はN君の会社の仕事始めです。休み中は地元に帰ってのんびりしていたN君ですが、心機一転して仕事に取り組んでいこうと思っています。さて、いきなりですが、今年のなるべく早いうちに、フレームワークを完成させる必要が出てきました。なぜならば、.NET Frameworkを使った大きなプロジェクトが、また始まりそうだからです。そのプロジェクトにタイムリーにフレームワークを提供できれば、今後受注する他のプロジェクトにも、どんどんフレームワークを採用してもらえそうです。

去年は、Soarアプリケーションの成果のうちの半分くらいをフレームワーク化することができました。もうひとがんばりすれば、フレームワークとしての基本機能を全部提供できそうです。N君と一緒にがんばりましょう！

フレームワークを設計する

前回に引き続き、Soarアプリケーションで実現した機能をフレームワークに取り込んでいきましょう。今回は、プレゼンテーションレイヤーで提供する機能を設計していきます。

プレゼンテーションレイヤーが果たす役割は、ユーザーとアプリケーションとの対話を実現することです。ユーザーからの問い合わせを受け付けてビジネスロジック以下の処理を呼び出し、処理結果を再びユーザーへ返すということの繰り返しです。プレゼンテーションレイヤーは、内部的に以下の2つのコンポーネントにより実現されます。

①ユーザープロセスコンポーネント

ユーザーインターフェイスが提供する、複数のインターフェイスの処理順序を管理するコンポーネント。たとえば、「本を検索する画面」「購入者の情報を入力する画面」「確認画面」という3つの画面を経て書籍を購入する機能の場合、3つの画面（≒ユーザーインターフェイス）が正しい順序で遷移することを、ユーザープロセスが実

現することになる。

②ユーザーインターフェイスコンポーネント
ユーザーに対して、目に見えるインターフェイスを提供するコンポーネント。画面や帳票のデザインといった出力要素と、キーボードからの直接入力やファイルのアップロードによる入力要素を担当する。エンタープライズシステムを構成するコンポーネントのうち、もっともユーザーに近い場所に配置される。

◆ユーザープロセスコンポーネント

ユーザープロセスコンポーネントでは、複数のユーザーインターフェイスの関連を調整する役割を持ちます。より具体的にいえば、画面間で受け渡される情報の管理、および画面の遷移そのものの管理を行ないます。

画面間で情報を受け渡すためにSoarアプリケーションでは、複数のインターフェイスでユーザープロセスを共有する方法を採用しました。さらに共有場所として、hiddenタグ、ViewState、Cookie、セッションの4つについて検討し、その結果「セッション」を用いることを決めました[注1]。そこで、

- セッションでユーザープロセスを共有する仕組み
- 画面でユーザープロセスを取得する仕組み

をSoarフレームワークとして提供することにします。

まず、セッションの中にユーザープロセスを保存する方法を考えます。ユーザープロセスクラスは、機能の数だけアプリケーションで定義されます。そしてすべてのユーザープロセスクラスは、それぞれひとつのインスタンスだけ作成されるようにします。ここでデザインパターンをご存知の方は「ひとつだけのインスタンス」と聞いて、シングルトンパターンを使えるのではと考えるかもしれませんが、ここでは誤りです。

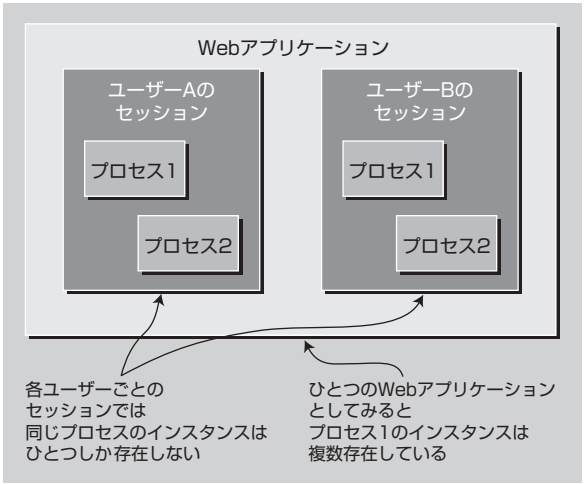
シングルトンパターンは、アプリケーション単位でインスタンスがひとつであることを保障するための仕組みです。しかし、Webアプリケーションの場合には、稼動しているひとつのアプリケーションの中で、複数のユーザーが同時にアクセスします。そのため、これらのユーザーごとにひとつのインスタンスを用意しなければなりません（図1）。

Webアプリケーションにおけるインスタンスの数の関係は、開発中にひとりでしかアプリケーションを動かしていないと発見しづらいものです。Webアプリケーションは常に複数のユーザーが利用することを前提にして、設計と実装を行なう必要があるのです。

では、ユーザーのセッション単位でプロセスを管理するためにはどうすればよいでしょうか？　このような場合の常套策のひとつとして、ユーザー単位で作成される「コンテキストクラス」を用意する方法があります。

コンテキストとは一般的に「文脈」という意味で使われますが、オブジェクト指向でクラスを設計するときには「特定の単位で常にひとつしか存在しないクラス」という意味合いで使われることが多いです。たとえば.NET FrameworkにはSystem.Web.HttpContextというクラスがあります。このクラスは、ユーザーからのHTTPによるリクエスト単位で常にひとつしか存在しません。ユーザープロセスの場合にはセッション単位で常にひとつしかないユーザープロセスコンテキストクラスを用意して、このクラスからプロセスを取得するようにします（図2）。

図1：ユーザープロセスはアプリケーション単位でシングルトンではない



注1）本誌2005年9月号の本連載（ユーザーインターフェイスを実装する 後編）を参照。