実際に作りこんでわかるWebアプリケーション

.NETによるビジネスアプリケーション開発講座

第2回 コンポーネントシステムの開発（その2）とASP.NET

日本ユニシス株式会社
.NETビジネスディベロプメント
尾島 良司 *OJIMA, Ryoji*

Technology Tools

- ☑ Visual Basic .NET
- ☑ Visual C# .NET
- ☑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ☑ ASP.NET
- ☑ Internet Information Services
- ☑ Other:
  - Visio for Enterprise Architect
  - Visual SourceSafe
  - NUnit

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\NETARCHディレクトリに収録しています。

\DNSTORE
サンプルプログラムとソースコード
・DNSTORE.SSA：

はじめに

本連載では、サンプルアプリケーションの開発を通じて、ビジネスアプリケーションをどのように開発してゆけばよいのかを考察してゆきます。

前回は「開発方法論の概略」「開発環境の準備」「データベースの構築」、そして「コンポーネントシステムの開発（その1）」としてNUnitやCOM+、ADO.NETの適用方針を考察しました。

今回は「コンポーネントシステムの開発（その2）」と「ASP.NET」です。コンポーネントシステムのあるべき姿について考え、ASP.NETの適用方針を決定します。サンプルアプリケーションのソースコードは付録CD-ROMに収録してあります。ぜひ実際に動かしてみてください。

コンポーネントシステムの開発（その2）

ここでは、以下の項目について順に考察してゆきます。

- **コンポーネントシステム再考**
- **コンポーネントシステムの粒度**
- **コンポーネントシステムのインターフェイス**
- **コンポーネントシステムの実装**

前回は要素技術を選択し、適用方法を検討して、コンポーネントシステムを完成させました。

ただし、前回は背景となる理論を明確にしていませんでしたから、コンポーネントシステムがどうしてあのような形をしているのかはナゾだったかもしれません。

そこで（作った後に書くのはアレなのですが）、「その2」として背景となる理論を考察しましょう。

コンポーネントシステム再考

まずは、コンポーネントシステムのおさらいを。

コンポーネントシステムは日本ユニシスの用語で、企業システム（“アプリケーション”ではないことに注意）をコンポーネント指向で構築するための

ビルディングブロックです。

「そんな独自の考え方をもち込まないで、普通のコンポーネントを組み合わせて企業システムを作ればよいのでは？」

いいえ。普通のコンポーネントでは企業システムをうまく構築できません。企業システムはリレーショナルデータベースを必要とし、リレーショナルデータベースはいわゆるコンポーネント指向を阻害しますから。

だから、企業システムをコンポーネント指向で作成する場合は、リレーショナルデータベースを隠蔽する大きなコンポーネント（これをコンポーネントシステムと呼びます）をもち出す必要があるのです。

もう少し考えてみましょう。ビルディングブロックは、組み合わせて初めて価値をもちます。そう、コンポーネント指向でコンポーネントを貼り付けるためのグルー（糊）が必要だったように、コンポーネントシステムにもグルーが必要なのです。

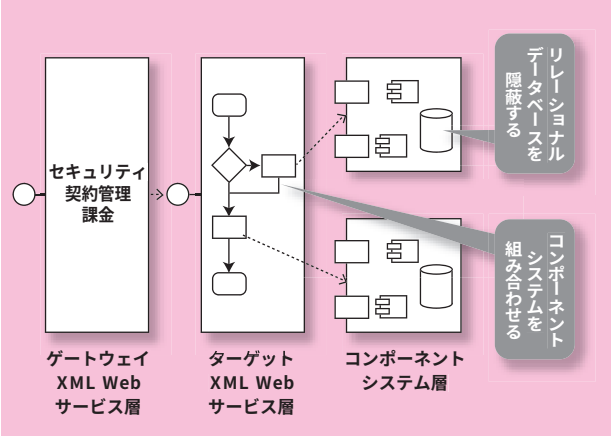
このグルー機能を、コンポーネントシステムの上位層である“XML Webサービス3層モデル”の“ターゲットXML Webサービス層”にもたせることにしましょう。

と、ここまでが前回までの話でした（図1）。

コンポーネントシステムの粒度

コンポーネントに関する議論でもっとも紛糾する議題は“粒度”です。再考が済んだところで、粒度を決定してしまいましょう。

図1：構築中のコンポーネントシステム



コンポーネントシステムには包含関係はありません。「コンポーネントシステム再考」で述べたように、コンポーネントシステムは上位層である“ターゲットWebサービス層”によって関係付けられるのですから[注1]。

そう、包含関係がないので、すべてのコンポーネントシステムは同じ粒度になるのです。

では、その粒度を具体的に。

コンポーネントシステムの目的はリレーショナルデータベースの隠蔽です。よって、スキーマの切れ目がコンポーネントシステムの粒度になります。

コンポーネントシステムを導く作業を具体的に説明しましょう。まず、DOAでシステムを分析する。次にER図を描く。でも、普通に分析を進めてゆくとすべてのエンティティが関係付けられてしまって“切れ目”なんかできません。どうするのか？　切っても支障のない場所、具体的にはデータが完全に同期しなくてもよい（もしくは同期のコストが安い）場所を見つけて切ってゆくのです。切ることが可能な場所がなくなったら、そこがコンポーネントシステムの境界。

たとえば、DnStoreのデータベースを普通に分析すると、結果は図2で示すER図になります。さて、コンポーネントシステム分割のために正規化を崩しましょう。ProductとPurchaseItemの部分が同期している必要は少ないですよね。よって、ここで切る。崩した結果は図3で示すER図になります。

これ以上の切れ目はありませんから、コンポーネントシステム分割はこれで終わり。この結果が前回の実装になっています。

コンポーネントシステムのインターフェイス

SAMPLE 1

次に、コンポーネントシステムのインターフェイスを考えてみましょう。どのような要素技術をどのような形で用

注1）密接に結びついたコンポーネントシステム群が互いを直接呼び出す場合もあるとは思いますが、その方式は裏口だと考えます。

SAMPLE 1　\DNSTORE\Catalog\CatalogComponentSystem.cs
\DNSTORE\Purchase\PurchaseComponentSystem.cs