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8回 フレームワークの基本構造を理解する

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

朝晩の風の冷たさに、秋が感じられるようになってきました。秋用に新調したニットのネクタイをきりっと締めなおして、N君は仕事に取り掛かります。

「前のプロジェクトの成果をフレームワーク化する」というテーマで難しいのは、バラバラに作成した便利な機能をひとつにまとめることです。ひとつひとつは魅力的な機能でも、すべてが協調して動いてこそ良いフレームワークになるのです。いくつかの機能を組み合わせてはバラしていく作業は、なんとなく野球やサッカーの監督業を思い出させます。1番から9番まですべてホームランバッターを並べるよりは、それぞれの打順に応じた役割をこなせるスペシャリストで構成するのが望ましい——小学生のときに読んだ「野球入門」にこう書いてありました。クラスの役割を考慮した、一番良いクラス構成にするにはどうしたらいいのでしょうか。N君の今日の仕事は、「クラスの役割分担についてじっくり考えること」になりそうです。

Level

1 2 3 4 5

Technology Tools

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

フレームワークの作成

Application Architecture for .NET（以下AAfN）は、Microsoftが提供するWebアプリケーションの構築方法です。AAfNの特徴としては網羅的である一方、概念的でもあり即座に実装に結びつけることが難しいということがあります。これまでの連載では、「概念的」な部分をいくつか取り上げて「実践的」な実装方法の説明を行なってきました。

しかしこのままではAAfNという考え方こそ再利用しているものの、実装作業に関してはすべてゼロからの開発になってしまいます。AAfNを使って複数のプロジェクトで生産性や品質に寄与するためには、実践的な実装方法のうち普遍的なものを再利用可能なプログラムとしてあらかじめ用意する必要があります。このようなプログラムは、かつて「共通ライブラリ」という形で提供されていました。現在ではオブジェクト指向やWebアプリケーションという最近の流れを受けて「フレームワーク」という形で提供するのが主流になっています。

Webアプリケーションそのものを作るときもフレームワークを作ると

きも、行なわれる作業そのものに大きな違いはありません。すなわち、「要件定義→設計→実装→テスト」というフェーズを踏んでシステムを完成させていきます。

要件定義

要件定義では、フレームワークに実現させたい要件を定義していきます。しかし通常のWebアプリケーションにおける要件定義に比べて、フレームワークの要件定義には難しい点があります。それはフレームワークの場合、要件定義を行なうときにその内容を評価する「利用者」がいないということです。したがって、フレームワークの設計者自身が利用者の役割も果たす必要があります。まずは過去に作成したWebアプリケーション（ここではSoarアプリケーション）から、他のプロジェクトでも使えそうな機能を列挙することから始めると良いでしょう。

ここでは例として、これまでの本連載の内容を踏まえ、あれば便利だと思われる機能を挙げてみましょう。

- 統一されたデータ操作
- 複雑なSQL文の自動生成
- データの排他制御コントロール
- トランザクションの制御
- ビジネスロジックの「部品化」
- 画面遷移のコントロール
- 画面に配置されたコントロールの制御
- 入力値の妥当性確認
- 認証情報の管理
- 統一された承認処理
- ログ出力インターフェイス
- 統一化された例外構造と例外補足処理
- etc.

もちろんこれは一例であり、これらをすべてフレームワークで実現すべきかどうかはまだわかりません。どれだけの機能を持ち込めばよいかは、フレームワークの特色を決める重要なポイントです。そこで、以降で機能の選別を行なう際の指針についてみていきます。

指針1 「あれば便利な機能」ではなく「必ず使わなければならない」機能であること

よく、フレームワークとライブラリを混同してしまうことがあります。どちらも、あらかじめ用意されたプログラムを用いて実装時の手間を省くことができます。しかし両者には決定的な違いがあります。それは、ライブラリは「あれば便利な機能」であるのに対して、フレームワークは「必ず使わなければならない機能」であることです。別の観点から両者の違いを見てみると、ライブラリは呼び出されるものであるのに対して、フレームワークは呼び出すものである、といえます（図1）。

ASP.NETを例に挙げて、考えてみましょう。

ASP.NETでライブラリに相当するものとしては、ボタンやテキストボックスなどのコントロールが挙げられます。Webページを作成するときには、必要となる任意のコントロールを使うことができますが、すべてのコントロールを必ず使わなければならないというわけではありません。あらかじめ完成されているコントロールを実装者が呼び出して画面上に配置していきます。

これに対してフレームワークに相当するものとしては、Webフォームを呼び出す仕組みが挙げられます。Web

図1：ライブラリとフレームワークの違い