コーディングテクニックだけでは生きてゆけない？

特集2

遥かなるアーキテクトへの道

株式会社CSK
中垣 健志 NAKAGAKI, Kenji

経験を積むことによって誰だってアーキテクトになれるかもしれない!?
プログラミングとはちょっと異なる“経験値”をあげるための指針や.NET Frameworkにおけるアーキテクトへの初めの一歩を紹介する。
アプリケーションのプログラミングという視点ではなく、問題解決の手段として.NET Frameworkを理解する術を読み取って欲しい。

Technology Tools

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

Level

★★★☆☆

Samples

.NET Frameworkを有効活用するために

「.NET Frameworkを採用すれば、品質を確保したまま納期を短縮してシステムが開発できる」

このようなキャッチコピーを信じてプロジェクトを始めてみたが、最終的に思うような効果が出せなかった。このような経験はないだろうか？

もちろんこれは.NET Frameworkそのものに非があるというよりは、それを使う人に非があって失敗することがほとんどだ。.NET Frameworkといえども、システム開発における銀の弾や万能薬にはなりえないのだ。しかし、だからといって.NET Frameworkが役に立たないかといえば決してそんなことはない。要は.NET Frameworkの長所と短所を知り尽くし、うまく効果が出せるようなアーキテクチャの設計をすることが重要なのだ、それができれば冒頭に示したキャッチコピーはきっと現実のものとなるだろう。

このような説明をすると大体次のような反論を受けることになる。

「じゃあ、いったい.NET Frameworkの長所と短所を知り尽くしている技術者はどこにいるんだい？」

画家を志す人のうち、ピカソやルノワールのようなすばらしい芸術を生み出す能力を持つ人が限られているように、エレガントなアルゴリズムでソースを書ける技術者はほんの一握りのエキスパートに限られてしまう。しかしアーキテクトにとって必要なのは独創力ではなく“問題解決能力”である。過去に成功したアーキテクチャが存在すればもう一度その事例を真似すればいいのであり、むしろ積極的に実績ある過去の事例を再利用できるほうがアーキテクトとしては優れているといえる。

ゆえにアーキテクトにとっては天性の才能よりも過去の経験のほうが重要であり、それは努力次第で誰もが身につけられるものなのだ。

最近のMicrosoftはスマートクライアントに力を入れているように見受けられるが、実際のシステム開発の現場ではまだまだWebアプリケーションの開発も多く存在している。そこで今回の記事では、ASP.NETでの開発におい

て.NETアーキテクトとして知っておいてもらいたい.NET Frameworkの基礎知識について述べていく。

.NET Frameworkで用意されている機能

まずは.NET Frameworkで用意されている機能のなかで、ASP.NET開発のために知っておいてもらいたいものを以下に列挙する。

・ポストバック
・DataAdapter、DataSet
・データ連結
・トランザクション
・認証、承認
・セキュリティ
・コンフィギュレーション
・バリデーション

では、それぞれについて見ていくことにしよう。

ポストバック

Webの画面はHTMLで作成されている。JavaScriptなどの簡易なスクリプト言語を除けば画面上にロジックは存在しえない。したがってサーバー側のプログラムは「あるリクエストに対して内部処理を行なった後、HTMLイメージをブラウザに返す」というのが、基本的な構造になっている。

この当たり前な考え方をASP.NETでは一新した（表1）。過去にVisual BasicやVC++で開発者に広く受け入れられたイベントドリブン開発手法をWeb開発に持ち込んだのだ。

イベントドリブン開発手法では画面にさまざまなコントロールが配置される。それぞれのコントロールや画面自身にはクリックや内容が変更されたときなどに発生するさまざまなイベントが用意されている。そのイベントの中で必要なものだけにイベントプロシージャを実装して処理を実現する。

ASP.NETでも同様の考え方で画面を作成することができる。すなわち、Webページ上にさまざまなWebコントロールを配置していき、ボタンが押されたときに実行される処理を同ページ内かコードビハインドに、イベントプロシージャという形で実装できるのだ。また、このイベントプロシージャ内には画面のレイアウト（≒HTML文の生成）に関する処理は一切含まれない。HTMLの生成はASPXファイルが担う役割だからだ。

この考え方を実現するために、ASP.NETでは「ポストバック」という概念を取り入れた。ASP.NETで作られたページではすべてのコントロールがひとつの大きなFORMタグによってくくられている。FORMタグでは自分自身のページへポストするようにACTION属性を定義する。発生したイベントはASP.NET内で自動的に判別され、適切なイベントプロシージャが呼ばれる仕組みになっている。

ASP.NETで発生するイベントを大別すると以下の3種類となる

・ページ読み込みイベント
・変更系イベント
・実行系イベント

ページ読み込みイベントは、ページが読み込まれたときに発生する。通常このイベント内では、画面の初期化に関する処理が記述される。例としては、コントロールへの初期値代入や、リストボックスの項目要素の設定などが挙げられる。

注意点としては、ASP.NETの仕様により、ページ読み込みイベントは初めてページを開いたとき（ポスト時）だけでなく、実行系イベントプロシージャによるポストバック時にも発生するということだ。つまり、画面上のボタンが押されるたびにページ読み込みイベントが発生することになる。このような仕様により、System.Web.UI.Pageオブジェクトに用意されているIsPostBackプロパティを用いてポストとポストバックのイベントを判断するのが一般的なプログラミング方法となる（リスト1）。

変更系イベントは、テキストボックスやリストボックスなどの内容が変更されたときに発生する。ただし、このイベントは対象となるコントロールのAutoPostBackプロパティの値によって発生するタイミングが変わってくる。その結果、イベントプロシージャ内にどのような処理を書くのかが変わ

表1：考え方の違い

	通常	ASP.NET
レイアウト	HTMLそのもの	WebコントロールがHTMLを隠蔽
ロジック	画面と独立	画面に属する