ASP.NET

Multi Web UI アプリケーション開発の実践—その4
UIの裏に存在するロジックの実装

最終回

西谷 亮 *NISHIYA, Ryo*
マイクロソフト株式会社
.NETマーケティング部
フィールドテクニカル
エバンジェリスト

はじめに

ついに最終回を迎えた当連載ですが、そんなことも気にせずに粛々と話を進めてゆきましょう。

前回は、ひとつのサンプルアプリケーションをベースとして、どのような設計に基づいて実装を進めるかを検証しました。今回は、その続きとして、ユーザーインターフェイス（以下UI）層の裏に存在するコンポーネント群がどのように構成されているかを見るために、コードのウォークスルーを進めてゆきます。

なお、使用するサンプルは下記のWebサイト、

「ASP.NET Web Solution Guide」
http://www.microsoft.com/japan/msdn/net/aspnetsolution/

からダウンロードすることができます。

コンポーネント群の役割

本稿で使用する「ASP.NET Web Solution Guide」のサンプルアプリケーションにおいて利用されるコンポーネントは、2つのプロジェクトに分かれて構成されています。ソリューションエクスプローラを参照するとすぐにわかると思いますが、“Controller”と“Models”というプロジェクトです（図1）。

各々の役割を検証しながら、実装の様子を見てゆきます。

▶Controllerプロジェクト

Controllerプロジェクトは、前回に解説した実装のモデル（設計）にあてはめて考えると、UIとバックエンドのロジックをつなぐ“仲立ち”になっています。このライブラリの存在によってUIとビジネスロジックの結合度を落と

本稿で前提となるもの

OS	Windows 2000（SP2）以降
開発環境	Visual Studio.NET Professional Edition以上 Internet Information Services 5.0以降 Internet Explorer 6.0 Mobile Internet Toolkit SQL Server 2000またはSQL Server 2000 Desktop Engine（MSDE）

初級　中級　上級

図1：サンプルのプロジェクト構成

し、システムに対しての柔軟性を与えることができるようになっています。

また、画面遷移の情報などもこのコンポーネントの中にもつように実装されています。これによって、UIの変更やビジネスロジックの変更などへ対応する場合の“問題の局所化”を実現しています。

▶Modelsプロジェクト

Modelsプロジェクトの中では、データアクセスなどの詳細なビジネスロジックが実装されています。商品管理から社員情報の取り扱いなど、各々の役割ごとにコンポーネントを用意しています。

このコンポーネントの構成は、アプリケーションの作成時にあらかじめ設計されたモデルに基づいています。このような構成をとっておくことで、UIとビジネスロジックでの問題の局所化に加えて、システム全体における変更に対する耐性を高めることも可能になります。

Controllerプロジェクトの実装を検証する

では、実際のコードを順に追ってゆきます。まずはControllerプロジェクトのコンポーネントから見てみましょう。

このControllerプロジェクトには「Controller.vb」という、ひとつのファイルだけが用意されています。このソースコードが、Controllerというクラスを構成しています。

いくつかの実装上のポイントを追ってゆきましょう。

まず、クラスの宣言部分を見てみましょう。以下のコード部分では、System.Web名前空間の利用がImportsキーワードによって宣言されています。

```
Imports System.Web
Public Class Controller
  Public Sub New()
  End Sub
```

これは、WebアプリケーションやXML Webサービスで利用されるコンポーネントを作成するときのポイントのひとつです。

このSystem.Web名前空間を利用すると、Webアプリケーションにおいてクライアントと通信している情報をコンポーネントの内部から利用できるようになります。この情報を「HttpContext」と呼んでいます。このHttpContextを元にして、コンポーネントがクライアントに対してデータを提供したり、逆にデータを得るなどということが可能となります。

Controllerプロジェクトのコンポーネントではリスト1のようなメソッドの実装の中で、このSystem.Web名前空間の情報を利用しています。

リスト1のGoOrderというメソッドは、商品の発注ページを表示させるために利用されるものです。ここでは、まず、

```
Dim ctx As HttpContext = HttpContext.Current
```

というコードによって、アプリケーションとクライアントとの間で交換されているHttpContextの情報を、変数「ctx」の内部に格納しています。この変数には、セッションの情報なども格納されています。そこで、このユーザーとのセッションにおいて利用されるデータベースの値をSession変数の中に格納しています。

「sm」という名称で宣言されているStockManagerというクラスは、Modelsプロジェクトに含まれているビジネスロジック群のひとつです。このインスタンスに用意されているGetProductsメソッドを利用して、商品情報や発注に使用される社員情報／顧客情報などを順次格納しています。

たとえば、

```
ctx.Session.Add("Products", sm.GetProducts())
```

という行では、ユーザーとのセッションに「Products」というSession変数を用意し、StockManagerクラスのGetProductsメソッドから取得されるデータを格納する、という意味を表わしています。

必要なデータの格納が終了した後は、適切なページへの遷移を実行しています。

```
ctx.Server.Transfer("OrderForm.aspx")
```

というコードは、Serverオブジェクトの Transferメソッドによって OrderForm.aspxへリダイレクトするということを表わしています。

このGoOrderメソッド以外の部分も順次スクロールしながら見てみましょう。どのメソッドも同じように、要求が行なわれるとHttpContextの情報を用いて必要なデータを渡したり、逆に読み取ったりしながら画面遷移を実行している様子を確認することができる