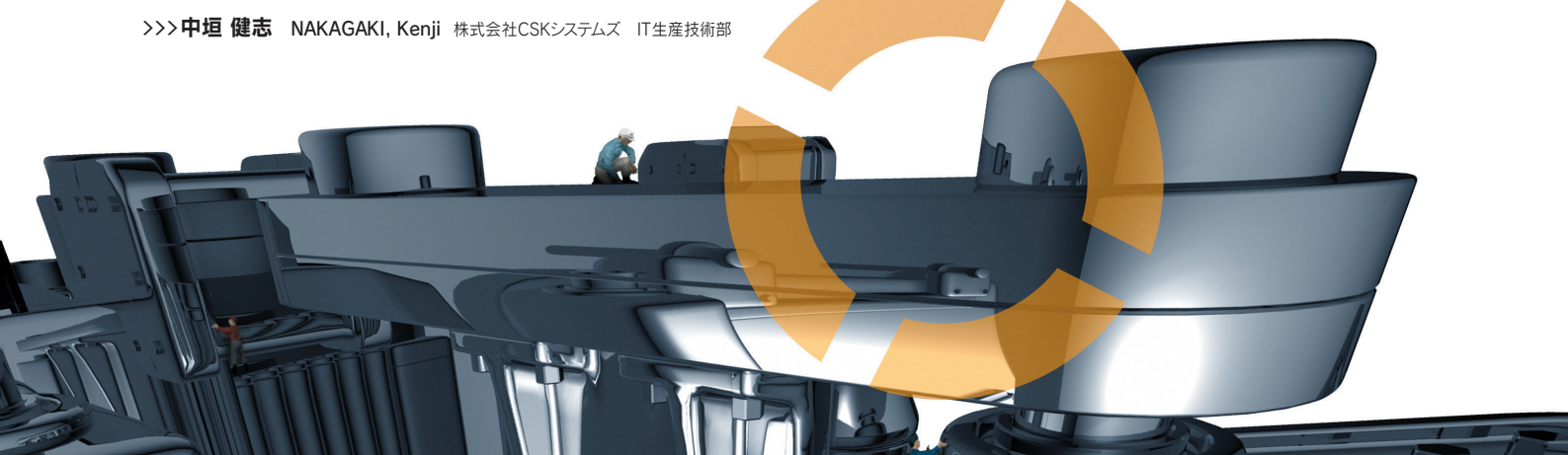
単体テストの応用テクニック

モックを利用してデータベース／UIロジックをテストする

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



さらなるテストカバレッジの向上を求めて

前稿『サンプルで理解する単体テストのコツ』では、アプリケーションのソースコードから環境に依存する部分を取り除くことで、単体テストを実施しやすくする方法について説明しました。しかし環境に依存しない部分の単体テストがしっかりできるようになると、今度は取り除いてしまった部分の単体テストもしたくなりますよね。

そのような要求を満たすべく、環境に依存する部分のソースコードをテストできるようにNUnitを拡張するフレームワークがいくつか提供されています。第2特集で紹介している「NUnit Forms」もそのひとつです。しかしここでは、このようなツールを使わずに、環境に依存したソースコードをテストする方法について検討していきたいと思います。

なお前稿と同様、単体テストの作成／実行には、以下のツールを利用します。

・Visual Studio 2005 Professional
・NCover
・TestDriven.NET

以降では、これらのツールがセットアップ済みであるという前提で説明を進めます。

レベル >>> Level
1 2 3 4 5

言語 >>> Language
- Visual Basic

サンプル >>> Sample
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。

ツール >>> Tool
- Visual Studio 2005 Professional
- NCover v1.5.4 Beta
- TestDriven.NET-2.0.1761 RC1

※記事内ではVisual Basicでコードを紹介していますが、サンプルプログラムはVisual Basic版、C#版の2種類を用意しています。

環境に依存する部分

アプリケーションにおいて、環境に依存した実装を行なわなければならないのはどの部分でしょうか？　主要な部分として、

・データベースに関するロジック
・ユーザーインターフェイスに関するロジック

が考えられます。完全網羅とはいえませんが、この2つをサポートできれば、かなり単体テストのテストカバレッジを上げることができます。

そこで以降では、これらについて単体テストを実施する方法について紹介していきます。

データベースに関するロジック

前稿『サンプルで理解する単体テストのコツ』では、データベースに関するロジックを、

①SQL文を生成するロジック
②SQL文を実行するロジック

の2つに分割し、①について単体テストを作成しました。これは、「正しいSQL文の生成」はプログラムが行なうため"テストで確認"し、「正しいSQL文から正しい結果の取得」はADO.NETあるいはデータベースエンジンが行なうため"テストしなくてよい"という考え方です。

しかし実はもうひとつ、単体テストを作成して確認しなければならないことがあります。それは、

作成したSQL文を実行するための処理が行なわれたかどうか

です。より具体的に言えば、ソースコードの中でSqlCommand.ExecuteメソッドやDataAdapter.Fillメソッドが呼び出されたどうかの確認です。

まず、前稿『サンプルで理解する単体テストのコツ』において、環境に依存する部分としてテストの対象外としたコードをリスト1に示します。

リスト1のSearchItemメソッドは、SqlConnectionクラスやSqlCommandクラスが散見されることからもわかるように、このソースコードを実行するとSQL Serverに実際に接続してしまいます。そこでこのソースコードを、SQL Serverに依存する部分と依存しない部分に分割することで、ADO.NETを使ってデータベースを呼び出す部分について単体テストができるかどうか検討してみましょう。

モックの利用

ここでは、データベースに依存する部分をテストするために、.NET Framework 2.0で新しく導入された「(ADO.NETの)プロバイダに依存しないAPI」[注1]を利用します。これは、データベースの違いを吸収した、統一した基本クラスを使ってADO.NETを利用できるようにする機能です。

データベースへの接続を表わすConnectionクラスを例にとって説明しましょう。ADO.NETではSQL Server用とOLE DB用にそれぞれSqlConnectionクラスとOleDbConnectionクラスが用意されています。ADO.NET 1.1では、これらの基本クラスはSystem.ComponentModel.Componentというクラスになっているので、データベースに関する共通的な処理はありません。したがって、各データベースごとに異なる実装をする必要があります。これに対してADO.NET 2.0では、これらの基本クラスとしてSystem.Data.Common.DbConnectionというクラスが用意されています。このクラスを使うことによって、データベースの種類に関係なく同じ実装を行なうことができます(図1)。

注1）http://msdn2.microsoft.com/ja-jp/library/ex6y04yf.aspx

リスト1：データベースに依存したソースコード

```
Public Overridable Function SearchItem(ByVal itemName As String, _
  ByVal makerName As String, ByVal category As Integer) _
  As DataTable
    Dim selectCommand As SqlCommand = CreateSqlCommand( _
      itemName, makerName, category, My.User.IsInRole("VIP"))

    ' データベースに問い合わせを行なう
    Using connection As New SqlConnection()
      connection.ConnectionString = "Data Source=(local);" & _
        "Initial Catalog=SampleDatabase;Integrated Security=true"
      Try
        selectCommand.Connection = connection
        Dim adapter As New SqlDataAdapter
        adapter.SelectCommand = selectCommand
        connection.Open()
        Dim table As New DataTable("ItemMaster")
        adapter.Fill(table)
        Return table
      Finally
        connection.Close()
      End Try
    End Using
End Function
```