NUnitを使った

ハイクオリティプログラミングのススメ

テスティングフレームワークによる.NET開発

第3回

テストの書き方と考え方

浅井 斉 *ASAI, Hitoshi*
株式会社テクノロジックアート
テクニカルデプトシステム
デベロップメントグループ

Technology Tools

- ☑ Visual Basic .NET
- ❑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ❑ ASP.NET
- ❑ Internet Information Services
- ☑ Other:
  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\HIQUALITYディレクトリに収録しています。

\SHOPPINGMEMO
買い物メモプログラム

過ちを犯さないために

ほんの小さな過ちでした。ただ空腹に耐えかねて、たった一切れのパンを盗んだだけのことです。でも、19年間の監獄生活を経て、ジャン・バルジャンの心は大きくゆがんでしまったのです。あたたかい食事とやわらかなベッドを与えてくれた司教から銀の食器を奪い、煙突掃除屋の貧しい少年からわずかばかりのコインさえも奪ってしまいました。もしも19年前に彼の心を正してくれる人がいてくれたならば、彼の心がこのようにゆがむことはなかったでしょう……。

連載第3回目となる今回は、ここまで学んできたテスティングフレームワークを利用した便利なテスト&デバッグと、より効率と品質を高めるための開発手法であるテスト駆動開発を実践に活かすためにも、もっとも重要となるテストの書き方／考え方を学んでゆこうと思います。テストの書き方と考え方を学ぶことで、あなたがいつか犯してしまうかもしれない間違いを、きっと正しながら進んでゆけることでしょう。ほんの小さな出来心を、人生さえ左右してしまう大過に発展させないためにも、本稿で詳しいテストの書き方／考え方を学んでゆくことをお勧めします。

本稿では、テストの書き方／考え方を学ぶため、まずはテストコードの作り方から解説してゆきます。これは、主に開発ツールの使い方と、テスティングフレームワーク内のクラスライブラリの使い方になります。その後、テストコードの考え方について解説してゆきます。これは、テスト駆動開発においてテストを作成するタイミング、そして、必要に応じたテストコードを作るための考え方の解説になります。前回、前々回の連載の中で、テスティングフレームワークとテスト駆

動開発については、少しでも理解されていると思いますので、今回は実践的なノウハウも含めて、場面に応じた解説をしてゆこうと思います。

テストコードの作り方

前回、前々回と、解説の中でサンプルのテストコードを扱ってきましたが、それらについては詳しく解説しませんでした。本稿では、まずテストコードの作り方として、テスティングフレームワークの詳しい使用方法とテストコードの書き方を解説してゆきたいと思います。

早速テストコードの書き方を解説したいところですが、その前に準備段階として、Visual Studio .NETのソリューションの中に、テストプロジェクトを組み込むところから解説してゆきます。これまでの連載の中でも簡単な流れは紹介しましたが、ここでは作成するシステムの形態に応じて、プロジェクト構成を考えながら解説します。

少し復習になりますが、テストプロジェクトという概念を説明します。Visual Studio .NETでは、ソリューションがひとつのソフトウェアを表わし、そこに含まれる各プロジェクトがソフトウェアを構成するDLLファイルやexeファイルとしての役割を担います。テストプロジェクトも、プロジェクトのひとつとしてソリューション内の各プロジェクトをテストする役割を担います。つまり、ソフトウェアの一部として、ソフトウェアのテストを行なう役割を果たすプロジェクトが、テストプロジェクトです。

◆Visual Studio＋NUnit Addinによるテストプロジェクトの作り方

では、Visual Studio .NETのソリューション構成の中にNUnit Addinのテストプロジェクトを作成する方法を見てゆきましょう。

小規模なソフトウェアの場合

画面や機能が、それぞれ単一のプロジェクトで済むような規模であれば、テストプロジェクトも単一で構いません。以下のような名前付けになります。

画面　：[ApplicationName].Forms
機能　：[ApplicationName].Libraries
テスト：[ApplicationName].Test

大規模なソフトウェアの場合

複数の種類の複数のプロジェクトが混在するような規模であれば、テストプロジェクトはプロジェクト種類ごとにひとつずつ用意します。以下のような名前付けになります。

画面　　：[ApplicationName].Forms.[PartsName]
機能　　：[ApplicationName].Libraries.[PartsName]
画面テスト：[ApplicationName].Forms.Test
機能テスト：[ApplicationName].Libraries.Test

さらに、テスト対象プロジェクトの構成に合わせて、いくつかのパターンを紹介します。ここでは、以下の3つのパターンを想定した上で、テストプロジェクトの最適な構成を見てゆきましょう。

Windows Formを利用する場合

Windows Formを利用したアプリケーションプロジェクトはexeファイルとして出力されますが、テストプロジェクトからexeファイルを出力するプロジェクトをテストすることはできません。ですから、画面のテストを行なうことを考慮して、画面を格納するプロジェクトと、exeを出力するローダプロジェクトを別々に作成します。また、テストを行ないやすくするため、Windows Formは画面表示に専念させ、機能の実装を別のプロジェクトにして、図1のような構成にします。

Web Formを利用する場合

Webアプリケーションの場合、テストプロジェクトから直接テストを行なうことができます。なので、Windows Formを利用する場合のようにローダを用意する必