Visual Studio を知る
- 製品概要
- エディション間の機能比較
- 最適な開発環境の構築 (MSDN Subscription とは)

Visual Studio を買う
- 評価版
- お得な買い方
- 参考価格
- ライセンス情報
- オンライン販売
- 販売パートナー

「使って覚える Visual Studio 2008」では、さまざまな課題に対して Visual Studio 2008 が提供している機能やソリューションを、ステップ バイ ステップで紹介いたします。ステップに沿って Visual Studio 2008 の機能を使ってみることで、開発の現場ですぐに活用できる実践的な知識を身につけていただけます。

第 3 回 特殊なテストやテスト管理をやってみよう

Step 1

タイムアウトのテストをしてみましょう。まずは、テスト対象のメソッドを作成していきます。

Visual Studio から新しいクラスライブラリ プロジェクトを作成します。
メニューバー [ファイル] - [新規作成] - [プロジェクト] をクリックします。[新しいプロジェクト] ダイアログが表示されるので、[テンプレート] に [クラス ライブラリ] を選択、[プロジェクト名] に 「その他のテスト」 と入力して、[OK] ボタンをクリックします。

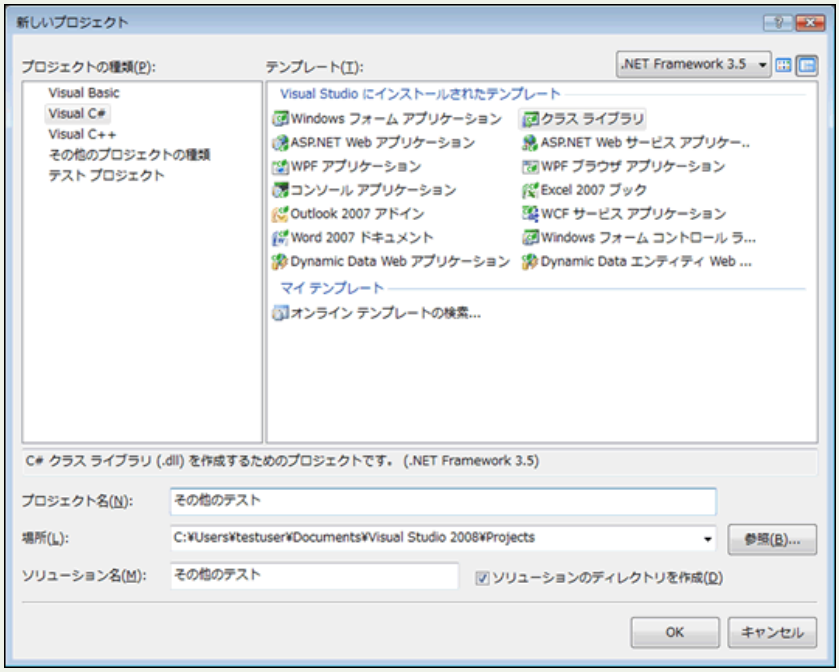

図 1. [新しいプロジェクト] ダイアログからクラス ライブラリ プロジェクトを作成

処理時間がかかる処理を作成します。ここでは簡単にパラメーターで渡された時間 Thread.Sleep メソッドで待機するコードを記述します。

C#

```
public void 時間がかかる処理(int 待ち時間)
{
    System.Threading.Thread.Sleep(待ち時間);
}
```

Visual Basic

```
Public Sub 時間がかかる処理(ByVal 待ち時間 As Integer)
    System.Threading.Thread.Sleep(待ち時間)
End Sub
```

Step 2

テスト対象メソッドの実装が終わったら、次に単体テストを実装していきましょう。

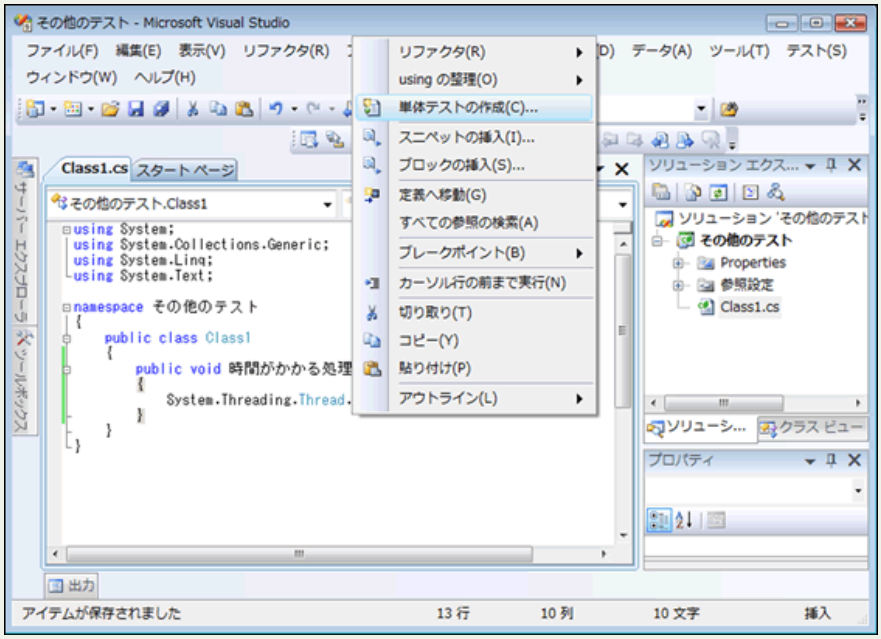

図 2. 右クリック メニューから [単体テストの作成] をクリック

テスト対象のメソッド上で右クリックして、[単体テストの作成] をクリックします。

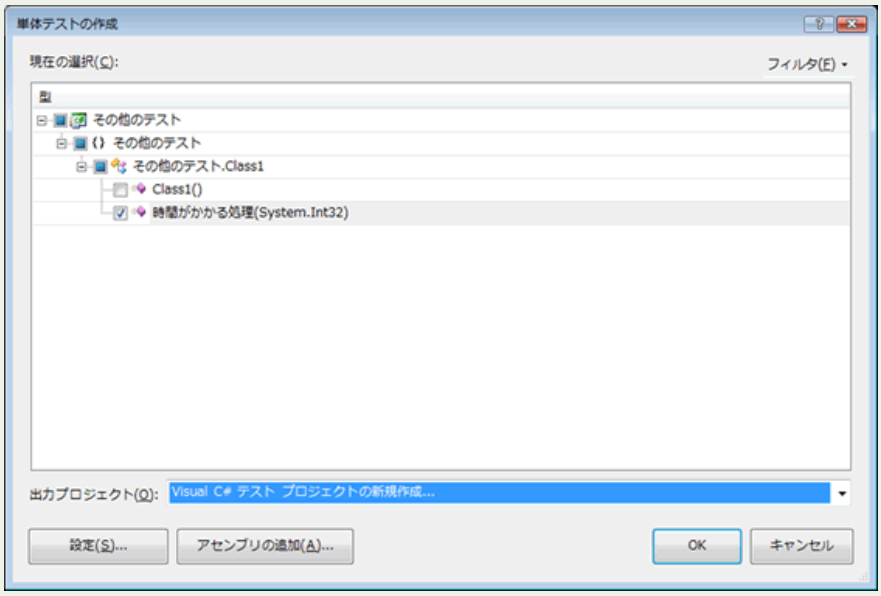

図 3. [単体テストの作成] でテスト作成対象のメソッドを選択

[単体テストの作成] ダイアログが表示されるので、対象のメソッドがチェックされていることを確認して、[OK] ボタンをクリックします。

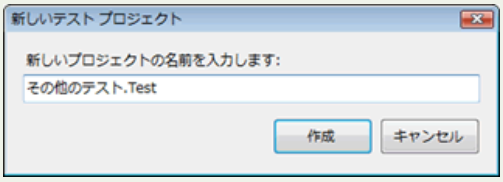

図 4. テスト プロジェクトの名前を入力

[新しいテスト プロジェクト] ダイアログが表示されるので、「その他のテスト.Test」 と入力して、[作成] ボタンをクリックします。

Step 3

それでは、タイムアウトをチェックするテスト コードを記述して行きましょう。

テスト メソッドにタイムアウト値を設定します。[テスト ビュー] (表示されていない場合は、メニューバーの [テスト] -

[ウィンドウ] - [テスト ビュー] をクリックします)で先ほど作成した「時間がかかる処理Test」メソッドを選択します。

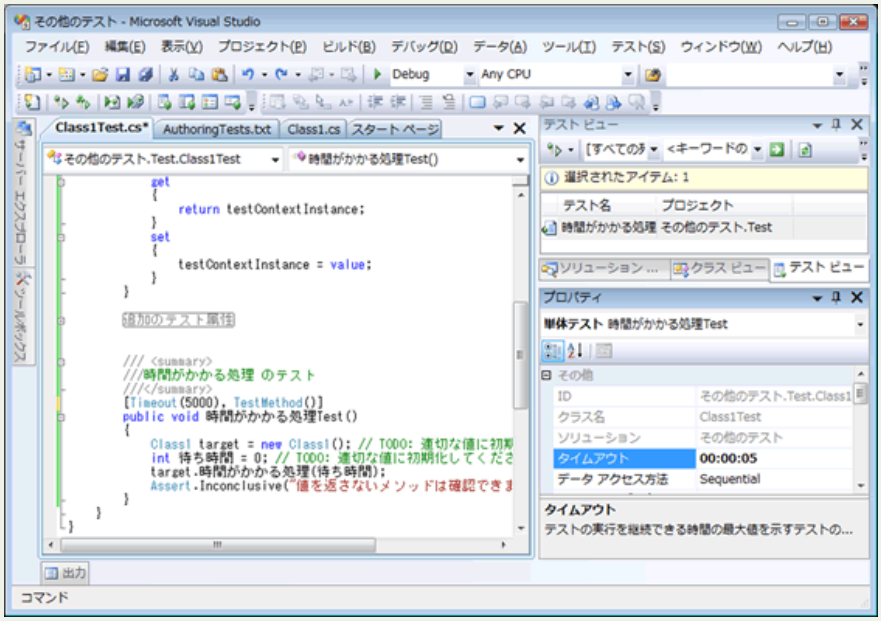

図 5. メソッドのタイムアウト時間を設定

[プロパティ] の [タイムアウト] にタイムアウト値を設定します。ここでは、5 秒を設定します。

テスト コードを確認すると Timeout 属性が追加されて、ミリ秒で 5000 が指定されていることが確認できます。では、次にテスト コードを実装していきます。

C#

```
/// <summary>
///時間がかかる処理 のテスト
///</summary>
[Timeout(5000)] [TestMethod()]
public void 時間がかかる処理Test()
{
    Class1 target = new Class1();
    int 待ち時間 = 6000;
    target.時間がかかる処理(待ち時間);
}
```

Visual Basic

```
< Timeout (5000)> < TestMethod ()> _
Public Sub 時間がかかる処理Test()
    Dim target As Class1 = New Class1
    Dim 待ち時間 As Integer = 6000
    target.時間がかかる処理(待ち時間)
End Sub
```

Step 4

それでは、テストを実行してみましょう。
[テスト ビュー] のツールバーから、[選択範囲の実行] ボタンをクリックします。

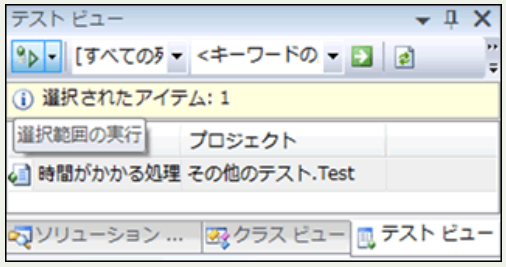

図 6. [選択範囲の実行] ボタンをクリックして、テストを実行

数秒待つと、テスト結果に [タイムアウト] と表示されるのが確認できます。

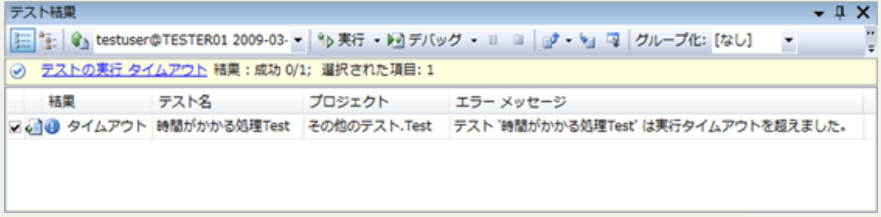

図 7. テスト結果にタイムアウトが表示される

タイムアウトは、処理に時間がかかりすぎているという異常な状態なのでテスト失敗として表示されます。

Step 5

処理によっては、例外をスローする場合もあります。Visual Studio 2008 の単体テストでは、連外がスローされたことを確認するテストコードを記述することができます。それでは、例外をスローするメソッドを実装して、テストを行っていきましょう。

テスト対象のプロジェクトに戻って、以下のメソッドを追加します。

C#

```
public void 例外処理()
{
    throw new InvalidOperationException();
}
```

Visual Basic

```
Public Sub 例外処理()
    Throw New InvalidOperationException
End Sub
```

ここでは、InvalidOperationException をスローするように作成します。

Step 6

テスト メソッドを作成して、メソッドを呼び出すと例外が発生することを確認して行きましょう。
「例外処理」メソッドを右クリックして、[単体テストの作成] をクリックします。

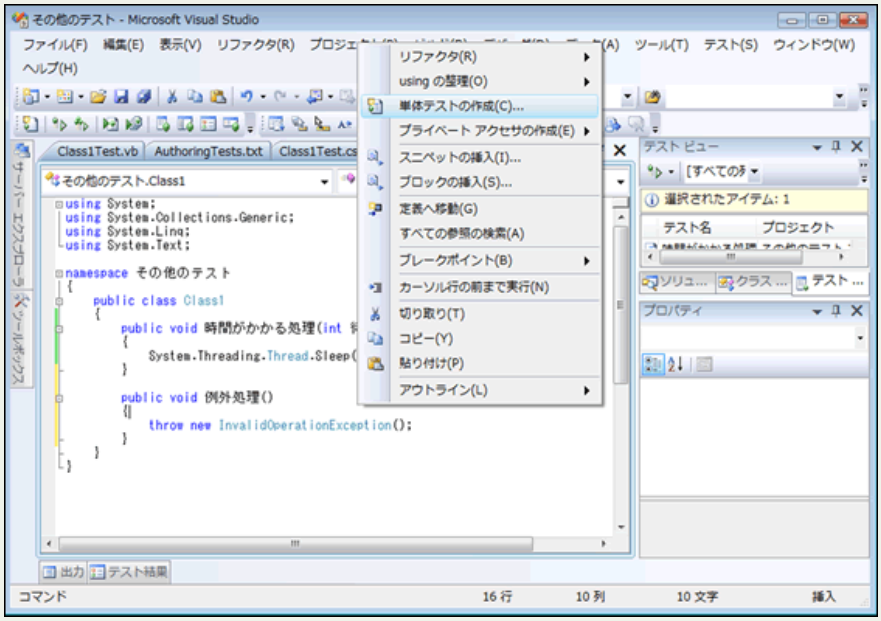

図 9. 右クリック メニューから [単体テストの作成] をクリック

[単体テストの作成] ダイアログが表示されるので、[現在の選択] に [例外処理] にチェックが入っていて、[出力プロジェクト] に [その他のテスト.Test] が選択されていることを確認して、[OK] ボタンをクリックします。

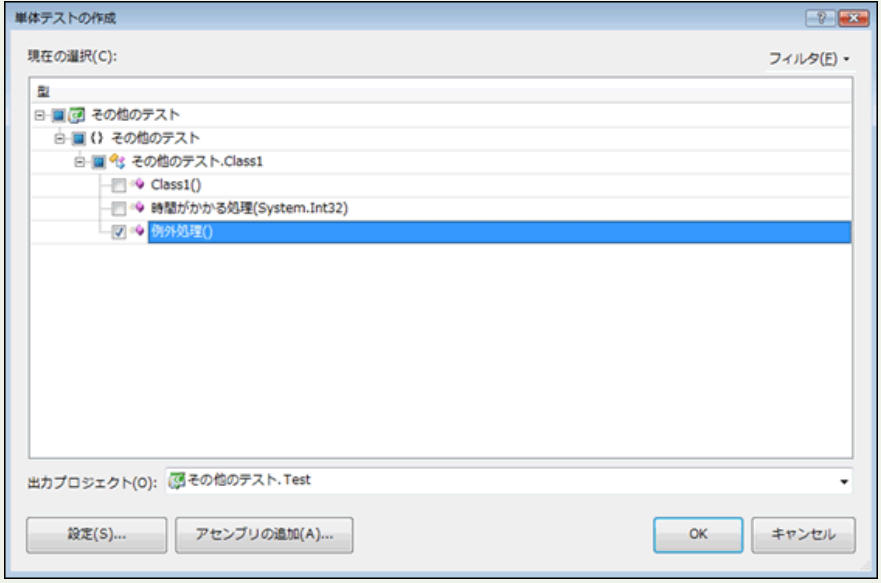

図 10. [単体テストの作成] でテスト作成対象のメソッドを選択

Step 7

正しく例外がスローされることを確認するために、テスト コードを記述します。例外の確認をするためには、テスト メソッドに ExpectedException 属性を追加します。

C#

```
/// <summary>
///例外処理 のテスト
///</summary>
[TestMethod()]
[ExpectedException(typeof(System.InvalidOperationException))]
public void 例外処理Test()
{
    Class1 target = new Class1();
    target.例外処理();
}
```

Visual Basic

```
'''<summary>
'''例外処理 のテスト
'''</summary>
<TestMethod()> <ExpectedException(GetType(System.InvalidOperationException))> _
    Public Sub 例外処理Test()
    Dim target As Class1 = New Class1
target.例外処理()
End Sub
```

このコードでは、メソッドの呼び出しで InvalidOperationException が発生することを期待していることを表しています。

Step 8

それでは、テストを実行してみましょう。
[例外処理Test] を選択した状態で [テスト ビュー] のツール バーの [選択範囲の実行] をクリックします。

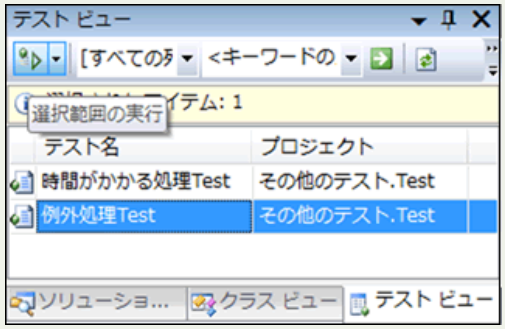

図 11. テスト メソッドを選択して、[選択範囲の実行] をクリック

しばらく待つと、[テスト結果] に [成功] が表示されます。

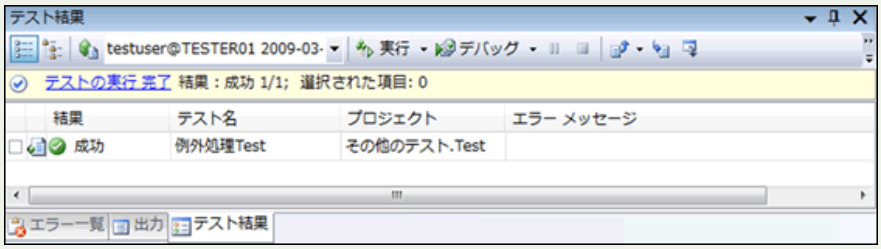

図 12. テストの実施結果が表示される

このようにメソッドの中で InvalidOperationException がスローされたことで、テストが成功していることがわかります。

Step 9

単体テストを作成していくとテスト メソッドがだんだんと増えていきます。このうち、一部のテストのみを選択して実施するケースが出てきます。ここでは、[テスト リスト エディタ] と呼ばれる機能を使用して、テストを管理する方法について解説します。

単体テストが増えている状態を擬似的に作成するために、次のように実装のないテスト メソッドを作成します。

C#

```
[TestMethod()]
public void ブランクテスト1()
{ }
[TestMethod()]
public void ブランクテスト2()
{ }
[TestMethod()]
public void ブランクテスト3()
{ }
[TestMethod()]
public void ブランクテスト4()
{ }
[TestMethod()]
public void ブランクテスト5()
{ }
```

Visual Basic

```
<TestMethod()> _
Public Sub ブランクテスト1()
End Sub
<TestMethod()> _
Public Sub ブランクテスト2()
End Sub
<TestMethod()> _
Public Sub ブランクテスト3()
End Sub
<TestMethod()> _
Public Sub ブランクテスト4()
End Sub


<TestMethod()> _
Public Sub ブランクテスト5()
End Sub
```

メニュー バーの [テスト] - [ウィンドウ] - [テスト リスト エディタ] をクリックして、[テスト リスト エディタ] を表示します。

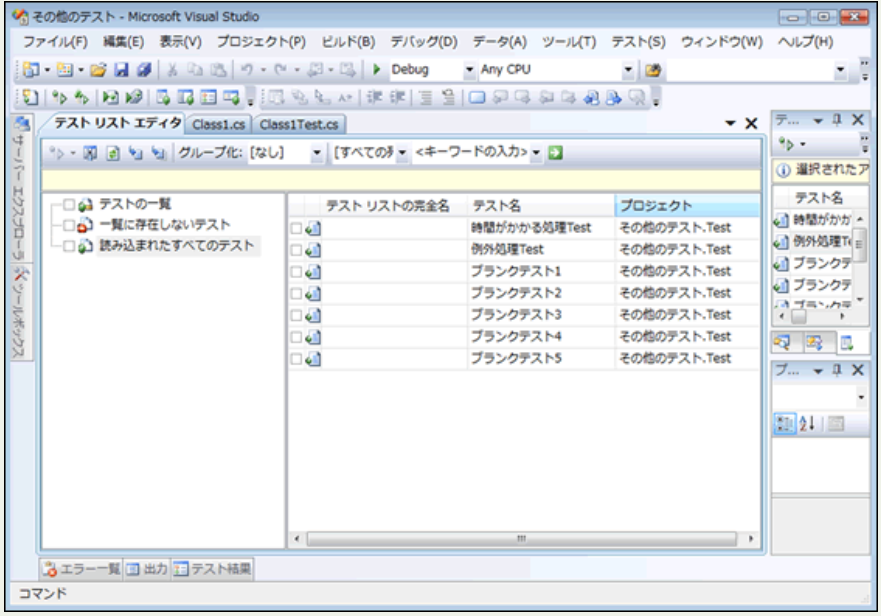

図 13. 作成したすべてのテストが表示される

[読み込まれたすべてのテスト] が選択されていて、ソリューションに含まれるすべてのテスト メソッドが一覧に表示されていることが確認できます。では、「ブランク テスト」だけをまとめたグループを作成してみましょう。「ブランク テスト 1」～「ブランク テスト 5」 までを選択して、右クリックして、[新しいテスト リスト] をクリックします。

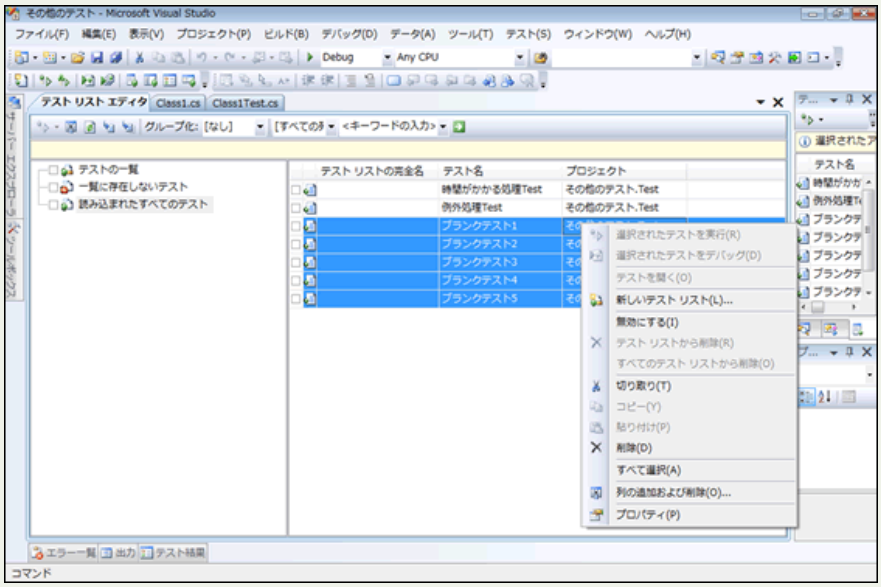

図 14. テストを選択して、右クリック メニューの [新しいテスト リスト] をクリック

[新しいテスト リストの作成] ダイアログが表示されます。

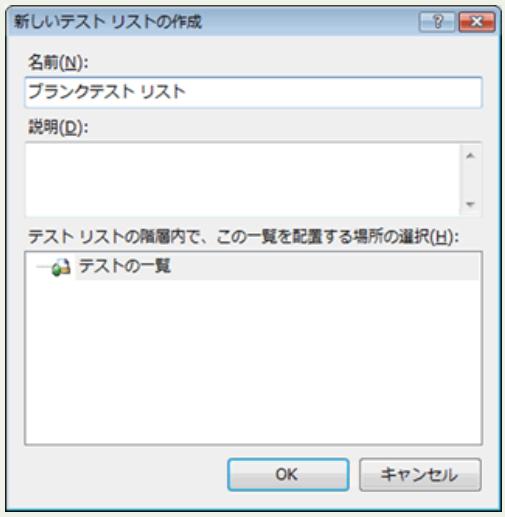

図 15. 追加するテスト リストの名前を入力する

[名前] に 「ブランク テスト リスト」 と入力して、[OK] ボタンをクリックします。ダイアログが閉じると、[テストの一覧] が展開できるようになっているので、[+] をクリックして展開すると、作成した [ブランク テスト リスト] が追加さ

れていることが確認できます。

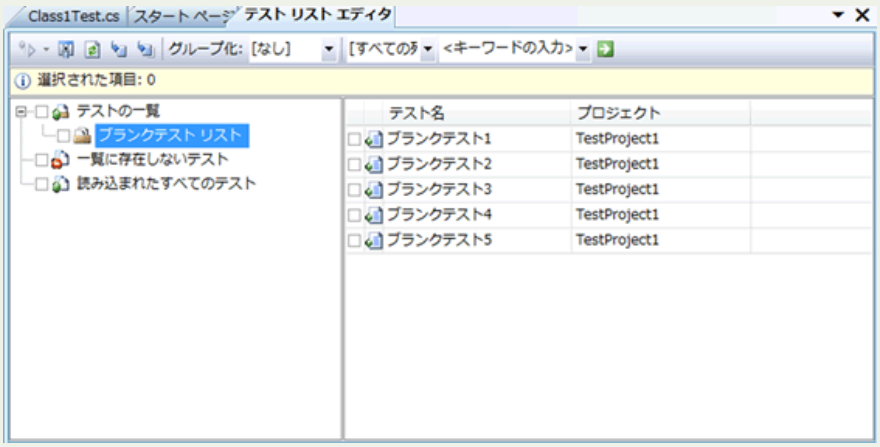

図 16. 作成したテスト リストに追加したテストが表示される

Step 10

テスト リスト エディタで作成したテストを実行してみましょう。テスト リスト エディタでは、論理的に分けたグループ単位でテストを行うことができますが、さらにグループの中から選択したテストのみを行うこともできるようになっています。

先ほど作成した 「例外処理テスト リスト」の中から奇数のものだけを選択して、[テスト リスト エディタ] のツールバーの [選択されたテストの実行] ボタンをクリックします。

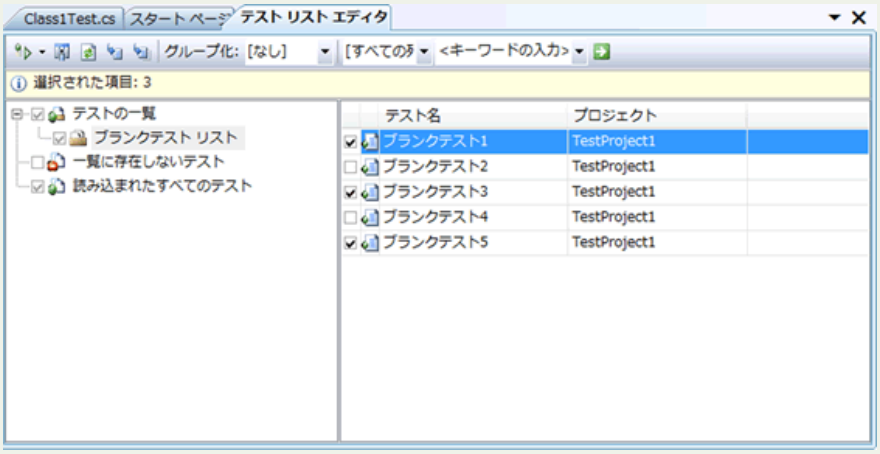

図 17. 奇数番目のテストをチェックしてテストを実行する

[テスト結果] ウィンドウに選択したテストのみが実行されて結果が表示されることが確認できます。

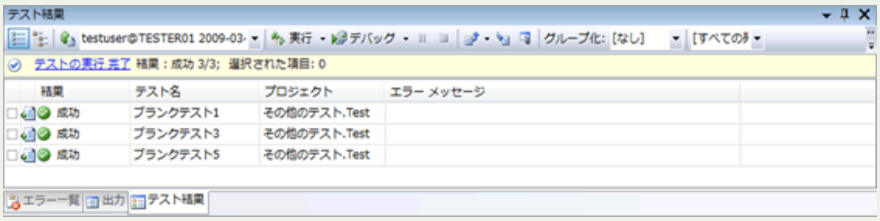

図 18. テストの実施結果が表示される

Step 11

さて、実施対象のテストが選択できると説明してきましたが、機能が未実装などで一時的にテストを実行しない場合、毎回チェックを外すのではなく、テストを無効にすることができます。

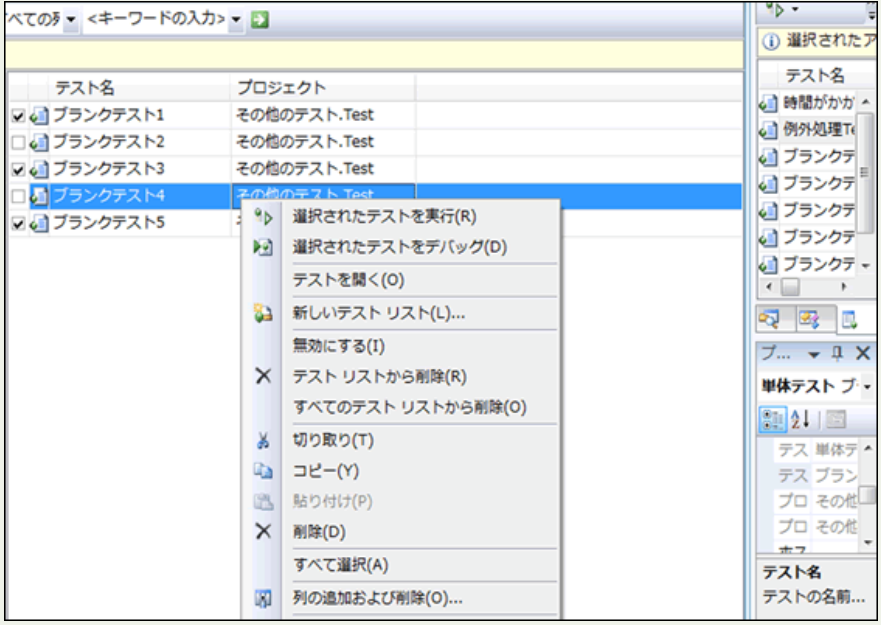

図 19.テストを右クリックして [無効にする] をクリックする

無効にしたいテストを右クリックして、[無効にする] をクリックします。

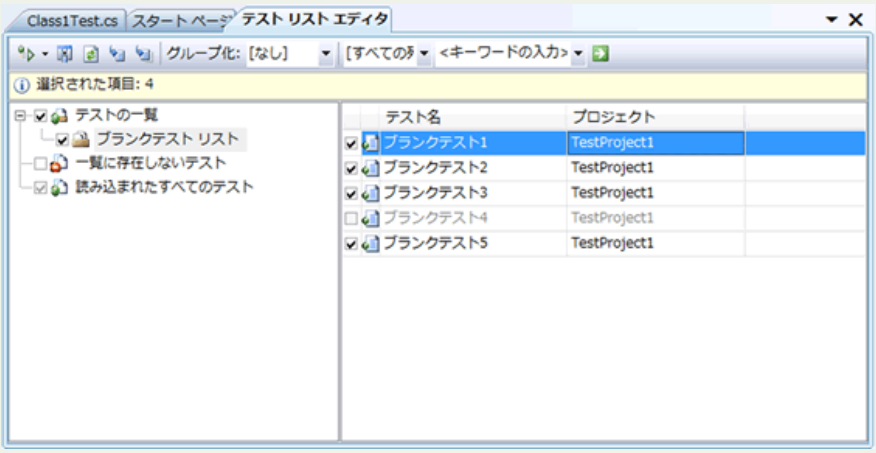

図 20.無効にしたテストが灰色で表示される

テストの表示が灰色に変わってチェックが外れていることが確認できます。この状態でテストを実行すると 「ブランク テスト 4」 以外が実行されることがわかります。

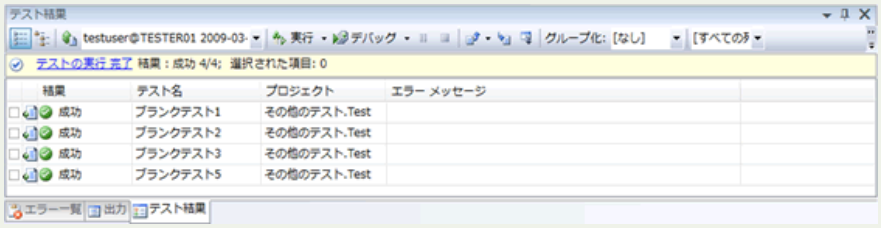

図 21.無効にしたテスト以外のテストが実行される

- 1. ツールを使って単体テストの効率を向上 »
- 2. データ駆動テストで境界値テストもらくらく！ テスト パターンの読み込み »
- 3. 特殊なテストやテストの管理にも対応！ 単体テストの便利な機能 »
- NEXT 4. 【上級編】一歩先行くテスト管理 テストの抜け漏れを可視化 »
- 評価版のダウンロード »

▲ Top of Page

Microsoft