Test med NUnit

Denne artikel introducerer NUnit.

Den forklarer ideen med NUnit. Og den viser hvordan man konkret bruger det.

Den forudsætter kendskab til C# / VB.NET og noget generel udviklings erfaring.

Skrevet den **09. Feb 2009** af **arne_v** I kategorien **Programmering / .NET** | ★★★☆☆

Historie:
V1.0 - 30/01/2005 - original

Indledning

Artiklen ligner meget min artikel om JUnit, men da NUnit er en portering af JUnit til .NET platformen, så kan det jo ikke overraske.

Alle kode eksempler vil blive vist både som C# og VB.NET, men konceptet i NUnit ligger over det sprog specifikke.

Unit test

Unit test er ikke en ny opfindelse. Det har man brugt i mange år. Developere skulle teste deres lille stump inden hele projektet blev afleveret til test.

Men der er kommet rigtig meget fokus på den i de sidste 5-10 år. Det er ikke længere bare noget som man principielt burde huske at gøre men en meget vigtig del af hele udviklings processen.

Det hænger lidt sammen med begreber som XP (eXtreme Programming), Refactoring, Agile Software Development, Test Driven Development etc.. Men er også accepteret udenfor disse metodikker.

Man er beyndt at skrive unit test kode til at teste systematisk og automatisk.

Det med automatisk er uhyre vigtigt i. Hvis man har en stor suite af unit tests som tester en given kode, så kan man nemmere tillade sig at rette i den, fordi man kan teste store dele af funktionaliteten på få sekunder/minutter.

Samtidigt har unit test også flyttet sig tidsmæssigt.

Engang skrev man unit test kode til sidst efter at man havde skrevet

selve koden. Det er nu almindeligt accepteret, at det ikke er så godt. Folk skriver nemlig så unit test koden efter hvordan de ved koden virker d.v.s. at de finder færre fejl.

Så nu er det alment accepteret at man bør skrive unit test koden først udfra hvad koden bør kunne og ikke hvad den kan.

Nogen er endda begyndt at flytte det at skrive unit teste kode op som en del af design. Det er jo en præcis måde at beskrive hvad koden skal gøre og dermed dets interface udadtil.

NUnit

Der er en familie af unit test frameworks som populært kaldes xUnit.

JUnit til Java
NUnit til .NET
CppUnit til C++
etc.

Vi vil her beskæftige os med NUnit.

NUnit version 1.x var en direkte portering af JUnit og havde efter sigende en del java'ismer, mens NUnit version 2.x er blevet rigtig .NET'sk.

NUnit kan hentes her:
http://www.nunit.org/

Og man downloader en MSI installer som man bare installerer helt normalt.

Eksempel

Lad os tage et simpelt eksempel for at illustrere hvordan det ser ud i praksis.

Vi har følgende kode som skal testes:

MathVector.cs

```
public class MathVector
{
    private int[] v;
    public MathVector(int n) {
        v = new int[n];
    }
    public MathVector(int[] v) {
       this.v = v;
    }
    public MathVector Add(int k)
    {
        int[] res = new int[v.Length];
        for(int i = 0; i < v.Length; i++) res[i] = v[i] + k;
```

```
        return new MathVector(res);
    }
    public MathVector Sub(int k) {
        int[] res = new int[v.Length];
        for(int i = 0; i < v.Length; i++) res[i] = v[i] - k;
        return new MathVector(res);
    }
    public MathVector Mul(int k) {
        int[] res = new int[v.Length];
        for(int i = 0; i < v.Length; i++) res[i] = v[i] * k;
        return new MathVector(res);
    }
    public MathVector Div(int k) {
        int[] res = new int[v.Length];
        for(int i = 0; i < v.Length; i++) res[i] = v[i] / k;
        return new MathVector(res);
    }
    public MathVector Mod(int k) {
        int[] res = new int[v.Length];
        for(int i = 0; i < v.Length; i++) res[i] = v[i] - (v[i]/k);
        return new MathVector(res);
    }
    public int Count
    {
        get
        {
            return v.Length;
        }
    }
    public int[] V
    {
        get
        {
            return v;
        }
    }
}
```

MathVector.vb

```
Public Class MathVector
    Private _v As Integer()

    Public Sub New(ByVal n As Integer)
        _v = New Integer(n) {}
    End Sub

    Public Sub New(ByVal _v As Integer())
        Me._v = _v
    End Sub

    Public Function OpAdd(ByVal k As Integer) As MathVector
```

```
        Dim res(_v.Length - 1) As Integer
        Dim i As Integer
        For i = 0 To (_v.Length - 1)
            res(i) = _v(i) + k
        Next
        Return New MathVector(res)
    End Function

    Public Function OpSub(ByVal k As Integer) As MathVector
        Dim res(_v.Length - 1) As Integer
        Dim i As Integer
        For i = 0 To (_v.Length - 1)
            res(i) = _v(i) - k
        Next
        Return New MathVector(res)
    End Function

    Public Function OpMul(ByVal k As Integer) As MathVector
        Dim res(_v.Length - 1) As Integer
        Dim i As Integer
        For i = 0 To (_v.Length - 1)
            res(i) = _v(i) * k
        Next
        Return New MathVector(res)
    End Function

    Public Function OpDiv(ByVal k As Integer) As MathVector
        Dim res(_v.Length - 1) As Integer
        Dim i As Integer
        For i = 0 To (_v.Length - 1)
            res(i) = _v(i) \ k
        Next
        Return New MathVector(res)
    End Function

    Public Function OpMod(ByVal k As Integer) As MathVector
        Dim res(_v.Length - 1) As Integer
        Dim i As Integer
        For i = 0 To (_v.Length - 1)
            res(i) = _v(i) - (_v(i) \ k)
        Next
        Return New MathVector(res)
    End Function

    Public ReadOnly Property Count() As Integer
        Get
            Return _v.Length
        End Get
    End Property

    Public ReadOnly Property V() As Integer()
        Get
            Return _v
        End Get
    End Property
```

```
End Class
```

(der er en simpel fejl !)

Vi laver nu følgende test kode:

MathVectorTest.cs

```
using NUnit.Framework;

[TestFixture]
public class MathVectorTest
{
    [SetUp]
    protected void Init() {
    }
    [TearDown]
    protected void Dispose() {
    }
    [Test]
    public void Add() {
        int[] v = { -1000000, -1, 0, 1, 1000000 };
        MathVector mv = new MathVector(v);
        MathVector mv2 = mv.Add(777);
        Assert.AreEqual(mv.Count, mv2.Count, "Add Count");
        for(int i = 0; i < mv2.Count; i++) {
            Assert.AreEqual(mv.V[i] + 777, mv2.V[i], "Add " + i);
        }
    }
    [Test]
    public void Sub() {
        int[] v = { -1000000, -1, 0, 1, 1000000 };
        MathVector mv = new MathVector(v);
        MathVector mv2 = mv.Sub(777);
        Assert.AreEqual(mv.Count, mv2.Count, "Sub Count");
        for(int i = 0; i < mv2.Count; i++) {
            Assert.AreEqual(mv.V[i] - 777, mv2.V[i], "Sub " + i);
        }
    }
    [Test]
    public void Mul() {
        int[] v = { -1000000, -1, 0, 1, 1000000 };
        MathVector mv = new MathVector(v);
        MathVector mv2 = mv.Mul(777);
        Assert.AreEqual(mv.Count, mv2.Count, "Mul Count");
        for(int i = 0; i < mv2.Count; i++) {
            Assert.AreEqual(mv.V[i] * 777, mv2.V[i], "Mul " + i);
        }
    }
    [Test]
    public void Div() {
        int[] v = { -1000000, -1, 0, 1, 1000000 };
```

```
        MathVector mv = new MathVector(v);
        MathVector mv2 = mv.Div(777);
        Assert.AreEqual(mv.Count, mv2.Count, "Div Count");
        for(int i = 0; i < mv2.Count; i++) {
            Assert.AreEqual(mv.V[i] / 777, mv2.V[i], "Div " + i);
        }
    }
    [Test]
    public void Mod() {
        int[] v = { -1000000, -1, 0, 1, 1000000 };
        MathVector mv = new MathVector(v);
        MathVector mv2 = mv.Mod(777);
        Assert.AreEqual(mv.Count, mv2.Count, "Mod Count");
        for(int i = 0; i < mv2.Count; i++) {
            Assert.AreEqual(mv.V[i] % 777, mv2.V[i], "Mod " + i);
        }
    }
}
```

MathVectorTest.vb

```
Imports NUnit.Framework

<TestFixture()> _
Public Class MathVectorTest

    <SetUp()> _
    Protected Sub Init()
    End Sub

    <TearDown()> _
    Protected Sub Dispose()
    End Sub

    <Test()> _
    Public Sub OpAdd()
        Dim v As Integer() =  {-1000000, -1, 0, 1, 1000000}
        Dim mv As MathVector = New MathVector(v)
        Dim mv2 As MathVector = mv.OpAdd(777)
        Assert.AreEqual(mv.Count, mv2.Count, "OpAdd Count")
        Dim i As Integer
        For i = 0 To (mv2.Count - 1)
            Assert.AreEqual(mv.V(i) + 777, mv2.V(i), "OpAdd " & i)
        Next
    End Sub

    <Test()> _
    Public Sub OpSub()
        Dim v As Integer() =  {-1000000, -1, 0, 1, 1000000}
        Dim mv As MathVector = New MathVector(v)
        Dim mv2 As MathVector = mv.OpSub(777)
        Assert.AreEqual(mv.Count, mv2.Count, "OpSub Count")
```

```
        Dim i As Integer
        For i = 0 To (mv2.Count - 1)
            Assert.AreEqual(mv.V(i) - 777, mv2.V(i), "OpSub " & i)
        Next
    End Sub

    <Test()> _
    Public Sub OpMul()
        Dim v As Integer() =  {-1000000, -1, 0, 1, 1000000}
        Dim mv As MathVector = New MathVector(v)
        Dim mv2 As MathVector = mv.OpMul(777)
        Assert.AreEqual(mv.Count, mv2.Count, "OpMul Count")
        Dim i As Integer
        For i = 0 To (mv2.Count - 1)
            Assert.AreEqual(mv.V(i) * 777, mv2.V(i), "OpMul " & i)
        Next
    End Sub

    <Test()> _
    Public Sub OpDiv()
        Dim v As Integer() =  {-1000000, -1, 0, 1, 1000000}
        Dim mv As MathVector = New MathVector(v)
        Dim mv2 As MathVector = mv.OpDiv(777)
        Assert.AreEqual(mv.Count, mv2.Count, "OpDiv Count")
        Dim i As Integer
        For i = 0 To (mv2.Count - 1)
            Assert.AreEqual(mv.V(i) \ 777, mv2.V(i), "OpDiv " & i)
        Next
    End Sub

    <Test()> _
    Public Sub OpMod()
        Dim v As Integer() =  {-1000000, -1, 0, 1, 1000000}
        Dim mv As MathVector = New MathVector(v)
        Dim mv2 As MathVector = mv.OpMod(777)
        Assert.AreEqual(mv.Count, mv2.Count, "OpMod Count")
        Dim i As Integer
        For i = 0 To (mv2.Count - 1)
            Assert.AreEqual(mv.V(i) Mod 777, mv2.V(i), "OpMod " & i)
        Next
    End Sub
End Class
```

Bemærk at vi bruger attributter til at markere hvad der er test klasse
oghvad der er test metoder.

Jeg kalder Assert AreEqual med 3 argumenter:
- forventet værdi
- faktisk værdi
- tekst som identificerer stedet i koden

(jeg anbefaler klart at man bruger versionen med 3 argumenter, da den identificerende tekst ofte er meget nyttig)

Der findes også IsTrue IsNull etc. metoder.

Og når vi kører med:

```
csc /t:library MathVector.cs
csc /r:MathVector.dll /r:"\program files\nunit 2.2\bin\nunit.framework.dll"
/t:library MathVectorTest.cs
nunit-console MathVectorTest.dll
```

```
vbc /t:library MathVector.vb
vbc /r:MathVector.dll /r:"\program files\nunit 2.2\bin\nunit.framework.dll"
/t:library MathVectorTest.vb
nunit-console MathVectorTest.dll
```

Får vi følgende fejl:

```
NUnit version 2.2.0
Copyright (C) 2002-2003 James W. Newkirk, Michael C. Two, Alexei A. Vorontsov,
Charlie Poole.
Copyright (C) 2000-2003 Philip Craig.
All Rights Reserved.

OS Version: Microsoft Windows NT 5.0.2195.0    .NET Version: 1.1.4322.573

.....F
Tests run: 5, Failures: 1, Not run: 0, Time: 0,203125 seconds

Failures:
1) MathVectorTest.Mod : Mod 0
        expected:<-1>
         but was:<-998713>
   at MathVectorTest.Mod()
```

Vi kigger lidt i koden og opdager at der mangler en multiplikation,
så vi retter koden til:

```
    public MathVector Mod(int k) {
        int[] res = new int[v.Length];
        for(int i = 0; i < v.Length; i++) res[i] = v[i] - (v[i]/k)*k;
        return new MathVector(res);
    }
```

```
    Public Function OpMod(ByVal k As Integer) As MathVector
```

```
        Dim res(_v.Length - 1) As Integer
        Dim i As Integer
        For i = 0 To (_v.Length - 1)
            res(i) = _v(i) - (_v(i) \ k)*k
        Next
        Return New MathVector(res)
    End Function
```

Og nu får vi:

```
NUnit version 2.2.0
Copyright (C) 2002-2003 James W. Newkirk, Michael C. Two, Alexei A. Vorontsov, C
harlie Poole.
Copyright (C) 2000-2003 Philip Craig.
All Rights Reserved.

OS Version: Microsoft Windows NT 5.0.2195.0    .NET Version: 1.1.4322.573

.....
Tests run: 5, Failures: 0, Not run: 0, Time: 0,078125 seconds
```

Videre med NUnit

Udover at skrive fejl til consollen så skriver NUnit også en rapport til en fil, som kan gemmes, emailes elle rprocesses af et andet program (det sidste er nemt fordi den er i XML).

TestResult.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!--This file represents the results of running a test suite-->
<test-results name="C:\e5\MathVectorTest.dll" total="5" failures="1" not-run="0"
date="30-01-2005" time="18:21">
  <test-suite name="C:\e5\MathVectorTest.dll" success="False" time="0,109375"
asserts="0">
    <results>
      <test-suite name="MathVectorTest" success="False" time="0,09375"
asserts="0">
        <results>
          <test-case name="MathVectorTest.Add" executed="True" success="True"
time="0.047" asserts="6" />
          <test-case name="MathVectorTest.Sub" executed="True" success="True"
time="0.000" asserts="6" />
          <test-case name="MathVectorTest.Mul" executed="True" success="True"
time="0.000" asserts="6" />
          <test-case name="MathVectorTest.Div" executed="True" success="True"
time="0.000" asserts="6" />
          <test-case name="MathVectorTest.Mod" executed="True" success="False"
time="0.016" asserts="2">
            <failure>
```

```
              <message><![CDATA[Mod 0
    expected:<-1>
     but was:<-998713>]]></message>
              <stack-trace><![CDATA[   at MathVectorTest.Mod()
]]></stack-trace>
            </failure>
          </test-case>
        </results>
      </test-suite>
    </results>
  </test-suite>
</test-results>
```

Man kan køre flere test cases sammen ved at lave et NUnit project.

AllTests.junit

```
<NUnitProject>
    <Settings activeconfig="Debug"/>
    <Config name="Debug">
        <assembly path="bin\debug\MathVectorTest.dll"/>
        <assembly path="bin\debug\AbcTest.dll"/>
        <assembly path="bin\debug\XyzTest.dll"/>
    </Config>
    <Config name="Release">
        <assembly path="bin\release\MathVectorTest.dll"/>
        <assembly path="bin\release\AbcTest.dll"/>
        <assembly path="bin\release\XyzTest.dll"/>
    </Config>
</NUnitProject>
```

Og teste den med:

```
nunit-console AllTests.nunit
```

Udover nunit-console programmet er der også et nunit-gui program, hvor man kan loade sit NUnit projekt og, køre det og se små grønne og røde cirkler.

Og husk at fordelene ved NUnit bliver større jo større koden er. Den er nyttig ved et enkelt Visual Studio projekt som man builder og kører unittest på. Men den er uvurderlig, hvor et build job starter hver nat kl. 24 som builder i 2 timer og kører unit test i 4 timer, så der ligger en test rapport til programmørerne når de møder om morgenen med alle de små hovsaer.

God test lyst.

Kommentar af skwat d. 21. Jul 2005 | 1

Hjæp, mine arme, jeg kan ikke få dem ned!

Kommentar af karsten_larsen d. 27. Apr 2008 | 2

Rigtig god artikel

Kommentar af mysitesolution d. 07. Apr 2008 | 3

Rigtig god artikel ;) vil helt sikkert få glæde af den. Man kan dog undre sig hvorfor det ellers fantastiske (delvist 2008) ikke har sådan nogle test indbygget? eller har de store versioner det?

Kommentar af nielle d. 19. Apr 2005 | 4

Ok artikel, men jeg synes måske nok at du kunne have gået bare lidt i dybden med betydningen af attributterne. I øvrigt vil jeg gerne anbefale NUnit2Report (http://nunit2report.sourceforge.net/) som et godt værktøj til at forvandle NUnit xml-rapporter til nogle meget læsevenlige html-rapporter.

Kommentar af the_ghost d. 30. Jan 2005 | 5

Rigtig god artikel. Jeg har selv brugt JUnit en del, og har kigget lidt efter noget lign. til .NET -> C#

Kommentar af talrinys d. 30. Jan 2005 | 6

Wow, så vidt jeg kan se en kanon artikel, men skal lige få kigget eksemplerne lidt bedre igennem