Introduktion til NAnt

Denne artikel beskriver NAnt, som er et værktøj til at builde .NET applikationer med.

Den beskriver nogle af de mest brugte NAnt tasks.

Den forudsætter solidt kendskab til .NET (enten C# eller VB.NET) og lidt erfaring med build af større systemer.

Skrevet den **08. Feb 2009** af **arne_v** I kategorien **Programmering / .NET** | ★★☆☆☆

Historie:
V1.0 - 13/03/2005 - original

Hvad er Nant

NAnt er en portering af et Java værktøj Apache ant til .NET platformen.

Det er et make program som bruger en XML fil til at beskrive hvad der skal gøres.

Hvis du kender make, så kender du princippet.

For dem som ikke kender make, så er det en løsning på problemet, hvor man skal have bygget et lidt større projekt og det kræver 10, 100 eller måske endda 1000 kommandoer. Det er jo håbløst at taste manuelt. Og laver man et script som udfører alle kommandoer hver gang, så kan det tage mange timer at køre. Make er intelligent d.v.s. at den indeholder instruktioner til at kunne udføre alle kommandoer, men den nøjes med at udføre dem som er nødvendige (den checker det ved at sammenligne tids stempler på filer).

NAnt gør det samme. Man definerer hvordan projektet skal buildes i en XML fil. Man kører NAnt. NAnt finder selv ud af hvilke dele af projektet der skal rebuildes og gør det. NAnt understøtter alle de ting man har brug for i .NET udvikling.

Hvorfor NAnt

Visual Studio er jo ganske glimrende til at bygge et projekt med. Hvorfor ikke bruge det i.s.f. at lære et nyt værktøj som NAnt ?

Det er der flere grunde til:

1) NAnt understøtter langt mere end bare simpel build - man kan både teste og deploye koden

2) build med NAnt kan automatiseres

3) XML filen er en fremragende dokumentation af hvordan builden foregår

NAnt har ikke fået nogen voldsom udbredelse i .NET verdenen endnu, men Apache ant bruges nok i 2/3 af alle Java projekter, så det er alligevel en særdeles velafprøvet teknologi.

Brug

NAnt kan hentes her:
http://nant.sourceforge.net/

Og du vil sikkert også have disse extra's:
http://nantcontrib.sourceforge.net/

Du kan trygt tage 0.85RC2

Installationen består i at:
- unzippe
- tilføje bin dirs til PATH

Makefile's i NAnt hedder *.build (selvom de altså indeholder XML).

Jeg vil i det efterfølgende kort beskrive nogle af de mest gængse tasks i NAnt.

Jeg vil kun berøre en ganske lille del af de næsten uendelige muligheder der er i NAnt.

Men når man først er kommet igang med NAnt har man sjældent problemer med at finde det man skal bruge i NAnt's dokumentation.

I de efterfølgende eksempler vil jeg vise både C# og VB.NET - der er naturligvis ikke nogen nævneværdig forskel på dem, men jeg tror at læserne ved at se det i det sprog de kender bedst kan fokusere mere på brugen af NAnt fremfor selve kode stumperne som jo er uinteressante.

Simpelt eksempel

Lad os starte med et meget simpelt eksempel.

hello.cs

```
using System;

public class HelloWorld
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Hello world");
    }
}
```

hellocs.build

```
<project name="hellocs" default="run">
    <target name="build">
        <csc optimize="true" target="exe" output="hello.exe">
            <sources>
                <include name="hello.cs"/>
            </sources>
        </csc>
    </target>
    <target name="run" depends="build">
        <exec program="hello.exe"/>
    </target>
</project>
```

Kommandoen:

nant /f:hellocs.build

vil builde og køre programmet (default target er run og target run kræver target build).

Kommandoen:

nant /f:hellocs.build build

vil kun builde programmet (target build).

hello.vb

```
Imports System

Public Class HelloWorld
    Public Shared Sub Main(ByVal args As String())
        Console.WriteLine("Hello world")
    End Sub
End Class
```

hellovb.build

```
<project name="hellovb" default="run">
    <target name="build">
        <vbc optionoptimize="true" optionexplicit="true" target="exe"
output="hello.exe">
            <sources>
                <include name="hello.vb"/>
            </sources>
        </vbc>
    </target>
    <target name="run" depends="build">
```

```
        <exec program="hello.exe"/>
    </target>
</project>
```

Kommandoen:

nant /f:hellovb.build

vil builde og køre programmet (default target er run og target run kræver target build).

Kommandoen:

nant /f:hellovb.build build

vil kun builde programmet (target build).

Det er faktisk ret simpelt. Jeg vil ikke kommentere XML indholdet i
.build filerne - det må være selvforklarende.

Mere komplekst eksempel

Ovenstående eksempel er naturligvis så simpelt at brug af NAnt er ganske
uinteressant.

Men her kommer et eksempel med lidt mere kød på.

Eksemplet bruger NUnit. Men hvis du interesserer dig for tools som NAnt,
så kender du sikkert også NUnit.

special.cs

```
using System;
using System.Globalization;

namespace E_NAnt
{
    public class SpecialFormats
    {
        public static string FormatDateTime(DateTime dt)
        {
            return dt.ToString("dd-MMM-yyyy HH:mm", new CultureInfo("en-US",
false));
        }
        public static string FormatMoney(decimal m)
        {
            return m.ToString("0.00 kroner", new CultureInfo("en-US", false));
        }
    }
}
```

specialtest.cs

```
using System;

using NUnit.Framework;

using E_NAnt;

[TestFixture]
public class SpecialFormatsTest
{
    [SetUp]
    protected void Init()
    {
    }
    [TearDown]
    protected void Dispose() {
    }
    [Test]
    public void FormatDateTimeNormal()
    {
        Assert.AreEqual("13-Mar-2005 21:15", SpecialFormats.FormatDateTime(new
DateTime(2005, 3, 13, 21, 15, 0)), "2 digit day");
        Assert.AreEqual("03-Mar-2005 21:15", SpecialFormats.FormatDateTime(new
DateTime(2005, 3, 3, 21, 15, 0)), "1 digit day");
    }
    [Test]
    public void FormatDateTimeAbnormal()
    {
        Assert.AreEqual("13-Mar-1965 21:15", SpecialFormats.FormatDateTime(new
DateTime(1965, 3, 13, 21, 15, 0)), "before 1970");
        Assert.AreEqual("13-Mar-1865 21:15", SpecialFormats.FormatDateTime(new
DateTime(1865, 3, 13, 21, 15, 0)), "before 1900");
    }
    [Test]
    public void FormatMoneyVarious()
    {
        Assert.AreEqual("123.45 kroner", SpecialFormats.FormatMoney(123.45m),
"standard");
        Assert.AreEqual("123.00 kroner", SpecialFormats.FormatMoney(123m), "no
decimals");
        Assert.AreEqual("0.45 kroner", SpecialFormats.FormatMoney(0.45m),
"only decimals");
        Assert.AreEqual("1234567890.00 kroner",
SpecialFormats.FormatMoney(1234567890m), "huge");
    }
}
```

specialcs.build

```
<project name="specialcs" default="test">
    <target name="clean">
```

```
        <delete failonerror="false">
            <fileset>
                <include name="special.dll"/>
                <include name="specialtest.dll"/>
            </fileset>
        </delete>
    </target>
    <target name="build" depends="clean">
        <csc optimize="true" target="library" output="special.dll">
            <sources>
                <include name="special.cs"/>
            </sources>
        </csc>
    </target>
    <target name="buildtest" depends="clean">
        <csc optimize="true" target="library" output="specialtest.dll">
            <sources>
                <include name="specialtest.cs"/>
            </sources>
            <references>
                <include name="\program files\nunit
2.2\bin\nunit.framework.dll"/>
                <include name="special.dll"/>
            </references>
        </csc>
    </target>
    <target name="test" depends="build,buildtest">
        <nunit2>
            <formatter type="Plain"/>
            <test>
                <assemblies>
                    <include name="specialtest.dll"/>
                </assemblies>
            </test>
        </nunit2>
    </target>
</project>
```

Kørsel:

```
C:\e4>nant /t:net-1.1 /f:specialcs.build
NAnt 0.85 (Build 0.85.1869.0; rc2; 12-02-2005)
Copyright (C) 2001-2005 Gerry Shaw
http://nant.sourceforge.net

Buildfile: file:///C:/e4/specialcs.build
Target framework: Microsoft .NET Framework 1.1
Target(s) specified: test


clean:
```

```
   [delete] Deleting 2 files.

build:

      [csc] Compiling 1 files to 'C:\e4\special.dll'.

buildtest:

      [csc] Compiling 1 files to 'C:\e4\specialtest.dll'.

test:

   [nunit2] Tests run: 3, Failures: 0, Not run: 0, Time: 0,03125 seconds
   [nunit2]
   [nunit2]
   [nunit2]

BUILD SUCCEEDED

Total time: 0.9 seconds.


C:\e4>nant /t:net-2.0 /f:specialcs.build
NAnt 0.85 (Build 0.85.1869.0; rc2; 12-02-2005)
Copyright (C) 2001-2005 Gerry Shaw
http://nant.sourceforge.net

Buildfile: file:///C:/e4/specialcs.build
Target framework: Microsoft .NET Framework 2.0 Beta 1
Target(s) specified: test


clean:

   [delete] Deleting 2 files.

build:

      [csc] Compiling 1 files to 'C:\e4\special.dll'.

buildtest:

      [csc] Compiling 1 files to 'C:\e4\specialtest.dll'.

test:

   [nunit2] Tests run: 3, Failures: 0, Not run: 0, Time: 0,03125 seconds
   [nunit2]
   [nunit2]
   [nunit2]

BUILD SUCCEEDED

Total time: 1.2 seconds.
```

```
C:\e4>nant /t:mono-1.0 /f:specialcs.build
NAnt 0.85 (Build 0.85.1869.0; rc2; 12-02-2005)
Copyright (C) 2001-2005 Gerry Shaw
http://nant.sourceforge.net

Buildfile: file:///C:/e4/specialcs.build
Target framework: Mono 1.0 Profile
Target(s) specified: test


clean:

   [delete] Deleting 2 files.

build:

      [csc] Compiling 1 files to 'C:\e4\special.dll'.
      [csc] Compilation succeeded

buildtest:

      [csc] Compiling 1 files to 'C:\e4\specialtest.dll'.
      [csc] Compilation succeeded

test:

   [nunit2] Tests run: 3, Failures: 0, Not run: 0, Time: 0,046875 seconds
   [nunit2]
   [nunit2]
   [nunit2]

BUILD SUCCEEDED

Total time: 1.8 seconds.
```

special.vb

```
Imports System
Imports System.Globalization

Namespace E_NAnt
    Public Class SpecialFormats
        Public Shared Function FormatDateTime(ByVal dt As DateTime) As String
            Return dt.ToString("dd-MMM-yyyy HH:mm", New CultureInfo("en-US",
False))
        End Function
        Public Shared Function FormatMoney(ByVal m As Decimal) As String
            Return m.ToString("0.00 kroner", New CultureInfo("en-US", False))
        End Function
    End Class
```

```
End Namespace
```

specialtest.vb

```
Imports System

Imports NUnit.Framework

Imports E_NAnt

<TestFixture()> _
Public Class SpecialFormatsTest
    <SetUp()> _
    Protected Sub Init()
    End Sub

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

    <Test()> _
    Public Sub FormatDateTimeNormal()
        Assert.AreEqual("13-Mar-2005 21:15", SpecialFormats.FormatDateTime(New
DateTime(2005, 3, 13, 21, 15, 0)), "2 digit day")
        Assert.AreEqual("03-Mar-2005 21:15", SpecialFormats.FormatDateTime(New
DateTime(2005, 3, 3, 21, 15, 0)), "1 digit day")
    End Sub

    <Test()> _
    Public Sub FormatDateTimeAbnormal()
        Assert.AreEqual("13-Mar-1965 21:15", SpecialFormats.FormatDateTime(New
DateTime(1965, 3, 13, 21, 15, 0)), "before 1970")
        Assert.AreEqual("13-Mar-1865 21:15", SpecialFormats.FormatDateTime(New
DateTime(1865, 3, 13, 21, 15, 0)), "before 1900")
    End Sub

    <Test()> _
    Public Sub FormatMoneyVarious()
        Assert.AreEqual("123.45 kroner", SpecialFormats.FormatMoney(123.45D),
"standard")
        Assert.AreEqual("123.00 kroner", SpecialFormats.FormatMoney(123D), "no
decimals")
        Assert.AreEqual("0.45 kroner", SpecialFormats.FormatMoney(0.45D),
"only decimals")
        Assert.AreEqual("1234567890.00 kroner",
SpecialFormats.FormatMoney(1234567890D), "huge")
    End Sub
End Class
```

specialvb.build

```
<project name="specialvb" default="test">
    <target name="clean">
        <delete failonerror="false">
            <fileset>
                <include name="special.dll"/>
                <include name="specialtest.dll"/>
            </fileset>
        </delete>
    </target>
    <target name="build" depends="clean">
        <vbc optionoptimize="true" optionexplicit="true" target="library"
output="special.dll">
            <sources>
                <include name="special.vb"/>
            </sources>
        </vbc>
    </target>
    <target name="buildtest" depends="clean">
        <vbc optionoptimize="true" optionexplicit="true" target="library"
output="specialtest.dll">
            <sources>
                <include name="specialtest.vb"/>
            </sources>
            <references>
                <include name="\program files\nunit
2.2\bin\nunit.framework.dll"/>
                <include name="special.dll"/>
            </references>
        </vbc>
    </target>
    <target name="test" depends="build,buildtest">
        <nunit2>
            <formatter type="Plain"/>
            <test>
                <assemblies>
                    <include name="specialtest.dll"/>
                </assemblies>
            </test>
        </nunit2>
    </target>
</project>
```

Kørsel:

```
C:\e4>nant /t:net-1.1 /f:specialvb.build
NAnt 0.85 (Build 0.85.1869.0; rc2; 12-02-2005)
Copyright (C) 2001-2005 Gerry Shaw
http://nant.sourceforge.net

Buildfile: file:///C:/e4/specialvb.build
Target framework: Microsoft .NET Framework 1.1
```

```
Target(s) specified: test


clean:

   [delete] Deleting 2 files.

build:

      [vbc] Compiling 1 files to 'C:\e4\special.dll'.

buildtest:

      [vbc] Compiling 1 files to 'C:\e4\specialtest.dll'.

test:

   [nunit2] Tests run: 3, Failures: 0, Not run: 0, Time: 0,03125 seconds
   [nunit2]
   [nunit2]
   [nunit2]

BUILD SUCCEEDED

Total time: 0.9 seconds.


C:\e4>nant /t:net-2.0 /f:specialvb.build
NAnt 0.85 (Build 0.85.1869.0; rc2; 12-02-2005)
Copyright (C) 2001-2005 Gerry Shaw
http://nant.sourceforge.net

Buildfile: file:///C:/e4/specialvb.build
Target framework: Microsoft .NET Framework 2.0 Beta 1
Target(s) specified: test


clean:

   [delete] Deleting 2 files.

build:

      [vbc] Compiling 1 files to 'C:\e4\special.dll'.

buildtest:

      [vbc] Compiling 1 files to 'C:\e4\specialtest.dll'.

test:

   [nunit2] Tests run: 3, Failures: 0, Not run: 0, Time: 0,046875 seconds
   [nunit2]
   [nunit2]
   [nunit2]
```

```
BUILD SUCCEEDED

Total time: 1.3 seconds.
```

(VB.NET koden virker ikke med Mono 1.0.5 så derfor er den ikke med)

Her ser vi hvordan man kan:
- bygge sine libraries
- bygge sine test cases
- køre sine test cases

Og:
* mod flere targets: .NET 1.1, .NET 2.0 Beta, Mono 1.0 etc..
* fuldt scriptable

Et realistiske scenarie for professionel brug af NAnt kan være:
- windows scheduler starter et script som kører NAnt kl. 0100
- NAnt henter al source code ud af source control til et nyt directory træ
- NAnt builder
- NAnt kører test cases
- om morgen studerer Tech Lead for projektet output for fejl
- på formiddags mødet beder Tech Lead de programmører som har et eller der ikke builder eller ikke passe test cases om at fixe det

Videre med NAnt

Når først du kommer igang med NAnt lærer du hurtigt at slå NAnt tasks op i dokumentationen (som er i HTML).

Mest relevante task:

csc - C# compiler
vbc - VB.NET compiler
cl - C++ compiler
jsc - JScript.NET compiler
vjc - J# compiler
wsdl - generere stub fra WSDL
gac* - manipuler GAC
iisdir - manipuler IIS virtual directories
exec - kør programmer
copy - kopier filer
delete - slet filer
zip - zip filer
unzip - unzip filer
nunit2 - kør NUnit test cases
vss* - VSS source control
cvs* - CVS source control

Jeg vil også lige henlede opmærksomheden på toolet SlingShot som kan konvertere en .sln fil til en .build fil:

slingshot -nant -sln xxxx.sln build.basedir=.. > xxxx.build

Kommentar af nielsbrinch d. 24. May 2005 | 1

Rigtig god lille introduktion til et build-værktøj som tilsyneladende er markeds-dominerende på samme måde som NUnit og log4net er det. Artiklen er ikke til nybegyndere.

Kommentar af optical d. 20. Mar 2005 | 2

fabelagtigt :)

Kommentar af talrinys d. 15. Mar 2005 | 3

Hold da kæft, som newb programmør lyder det da ultroligt lækkert, vil afprøve det i løbet af de næste par dage.

Kommentar af zzzzzzzzzz d. 15. Mar 2005 | 4

God, men forklar også hvordan man installer det, så vil den være helt perfekt!