Leendert Versluijs is managing consultant bij Capgemini

# Mono 1.0: een crossplatform ontwikkelstrategie voor .NET

## .NET-TOEPASSINGEN OP WINDOWS, LINUX EN MAC OS X

**In .NET Magazine nummer 4 staat een interview met Miguel de Icaza waarin hij spreekt over de open source .NET-variant Mono. Ten tijde van het interview was Mono nog in een bètafase, maar inmiddels is versie 1.0 officieel beschikbaar. De grote vraag voor .NET-ontwikkelaars is natuurlijk wat met Mono mogelijk is. In dit artikel kun je lezen hoe je het beste crossplatform kunt ontwikkelen en waar je op moet letten bij het programmeren van je applicaties.**

Natuurlijk blijft Microsoft Windows het belangrijkste platform voor .NET, maar het is nu dankzij Mono ook mogelijk om .NET-programma's op andere platformen te draaien. Hiermee maakt .NET een belofte waar. Maar wat moet je als ontwikkelaar daar nu mee in de praktijk? .NET draait toch immers erg goed op Windows. Waarom zou je dan proberen het ook op Linux of Mac OS X te draaien? Wel, er zijn nu eenmaal bedrijven en instanties die platformonafhankelijkheid belangrijk vinden of voor welke de licentiekosten van Windows een probleem zijn. Tot nu toe was het Java of C/C++ dat de klok sloeg bij deze bedrijven en instanties. Maar nu Mono beschikbaar is, zal ook .NET toegepast kunnen worden in crossplatform-ontwikkeling.

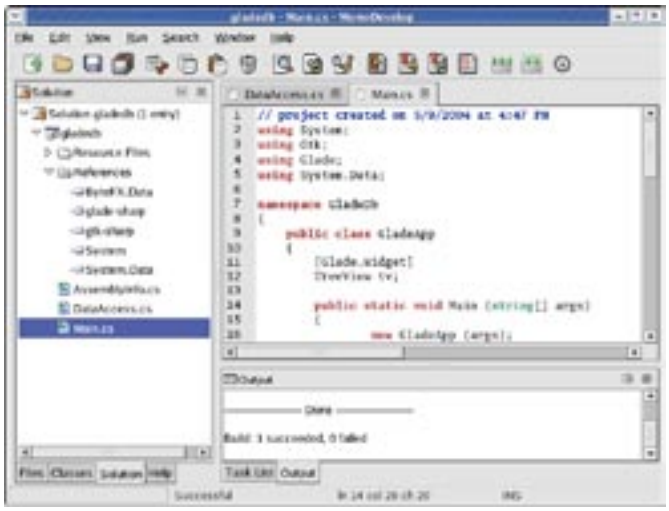
### Ontwikkelplatform

Aangezien Mono nog niet zo lang bestaat, halen de ontwikkeltools niet het niveau van Visual Studio .NET en is het maar de vraag of ze ooit op dat niveau zullen komen. Het beste dat Mono te bieden heeft is MonoDevelop, dat een onderdeel is van SharpDevelop. MonoDevelop is speciaal ontwikkeld om ervoor te zorgen dat ook onder Linux een IDE beschikbaar is. Helaas ontbreken nog de ontwerptools voor Windows Forms en Web Forms. Ook is er nog geen ondersteuning voor versiebeheer waardoor werken in teams met MonoDevelop lastig is. Wel zijn open source tools als NUnit, NDoc en NAnt ook beschikbaar voor Mono op Linux. En de MSDN library van Mono, genaamd MonoDoc, is eveneens aanwezig en zelfs geïntegreerd in MonoDevelop. Helaas zijn veel onderwerpen nog niet voorzien van extra uitleg. Je zult daarom nog vaak gebruik maken van de MSDN library.

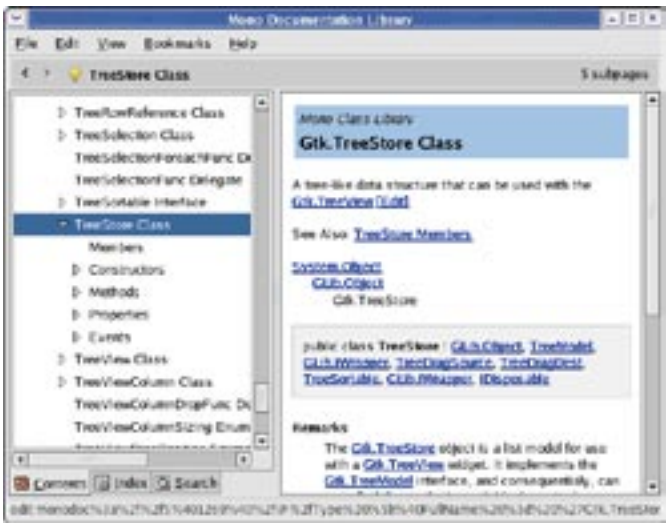
Toch is het aanbod voor een ontwikkelaar nog aan de magere kant vergeleken met wat er op Windows beschikbaar is. Daarom is het toch verstandig, als je voor Mono wilt ontwikkelen, om gebruik te maken van Visual Studio.NET. Zeker gezien het feit dat de assemblies die gemaakt zijn met Visual Studio.NET zonder problemen ook draaien onder Mono. Je kunt dus de assemblies op Windows in Visual Studio.NET programmeren en compileren en vervolgens testen op Linux onder Mono. Hierbij is het van belang dat je tijdens het ontwikkelen in Visual Studio.NET rekening houdt met het feit dat je applicatie ook onder Mono op Linux moet draaien. Niet alles wat Windows te bieden heeft is beschikbaar op andere platformen en ook niet alles van de Framework SDK is beschikbaar in Mono.

### Linux of Windows?

De makers van Mono zijn van start gegaan met een C#-compiler en hebben niet zoveel aandacht besteed aan Visual Basic .NET. Dit blijkt ook wel uit het feit dat Mono 1.0 Visual Basic.NET officieel nog niet ondersteunt. Dit geldt ook voor programma's die met



Afbeelding 1. MonoDevelop: de Mono IDE onder Linux




Afbeelding 2. MonoDoc: de MSDN library van Mono

```
using System;
using System.IO;

namespace MonoCompatible
{
  class Demo
  {
    [STAThread]
    static void Main(string[] args)
    {
      string rootdir = string.Empty;
      // Get root directory of first available logical drive
      foreach ( string s in Environment.GetLogicalDrives() )
      {
        try
        {
          if ( Directory.Exists( s ) )
          {
            rootdir = s;
            break;
          }
        }
        catch
        {
          // Drive currently not available e.g. A:\ without disk
        }
      }
      // Create the directory
      string dirname = rootdir + "MyDir";
      if ( ! Directory.Exists( dirname ) )
      {
        Directory.CreateDirectory( dirname );
      }
      // Determine filename
      string filename = dirname + Path.DirectorySeparatorChar +
        "MyFile.txt";
      // Write file in correct format for each operating system
      StreamWriter sw = new StreamWriter( filename );
      sw.Write( "Line 1" + Environment.NewLine + "Line 2" +
        Environment.NewLine );
      sw.Close();
    }
  }
}
```

**Codevoorbeeld 1.**
**Crossplatform-ondersteuning in de Framework SDK**

Visual Basic.NET in Visual Studio.NET zijn gemaakt, omdat deze programma's gebruik maken van een aantal Visual Basic-specifieke assemblies. Als je dus Mono-compatibel wilt ontwikkelen, kun je maar beter met C# gaan coderen.
In .NET is een goede ondersteuning aanwezig voor interop met COM en WIN32 dll's. Linux ontbeert deze ondersteuning, daarom kun je ze maar beter niet gebruiken. Als het toch niet anders kan, is het raadzaam een wrapper te maken die afhankelijk van het platform andere interop-calls doet.
Als je software gaat maken voor verschillende besturingsystemen moet je ook rekening houden met de kenmerken van deze besturingsystemen. Zo is het filesysteem van Linux anders dan dat van Windows. Je hebt onder Linux geen drives, directories worden gescheiden met een '/' en niet met een '\' en alle directory- en bestandsnamen zijn case-sensitive. Daarnaast is de enter in een tekstbestand in Linux anders dan in Windows. Gelukkig kan het .NET Framework ook op andere besturingsystemen werken, waardoor er constanten en methoden beschikbaar zijn voor gebruik. In codevoorbeeld 1 staat code die een tekstbestand met twee regels wegschrijft in de directory C:\MyDir op Windows en in /MyDir op Linux.

Naast de in codevoorbeeld 1 getoonde zaken kun je nog meer vinden in de Environment en Path-classes. Bijvoorbeeld de Path.InvalidPathCharacters om te controleren of er geen illegale tekens in je bestandnaam voorkomen.

## ASP .NET en webservices

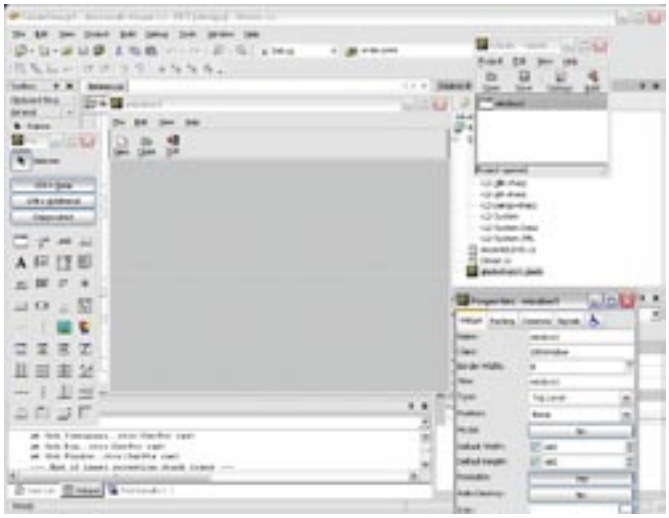
Wat betreft ASP.NET en webservices is het met de compatibiliteit goed geregeld. Alle varianten van ASP.NET-bestanden (ASPX, ASMX, ASCX) worden inclusief de mogelijkheid van code behind ondersteund. Caching, dat erg belangrijk is voor een goede performance in ASP.NET, ondersteunt Mono. Ook werken webreferenties die met Visual Studio.NET gemaakt worden zonder problemen in Mono. Alleen ontbreken de webservice-enhancements 1.0 & 2.0 nog in Mono 1.0. Deze staan gelukkig wel gepland voor de volgende release.

Wel anders is de webserver die door Mono gebruikt wordt. Standaard wordt met Mono XSP meegeleverd waarmee je alleen maar ASP.NET-pagina's in kunt hosten. Als je naast ASP.NET-pagina's ook HTML- of andere pagina's wilt hosten, moet je gebruik maken van de mod_mono-module voor Apache. Daarmee zorg je dat alle ASP.NET-paginaverzoeken aan een Apache Server worden afgehandeld door XSP. Eigenlijk is dit dezelfde werkwijze als waarmee de Internet Information Server (IIS) de ASP.NET-verzoeken in een apart proces afhandelt. Net als ASP.NET in IIS, beschikt XSP ook over de mogelijkheid om een testpagina voor een webservice te genereren. Een aardige extra is de mogelijkheid om op deze testpagina een client-proxy te genereren voor C# en Visual Basic .NET. Om ASP.NET te kunnen draaien op Apache moet je na de installatie wel handmatig de configuratiefile van Apache aanpassen.
Besteed bij het ontwikkelen van een Mono-compatibele ASP.NET-applicatie aandacht aan de directory- en bestandsnamen. Zorg er voor dat links naar images en andere pagina's de juiste casing hebben. Als je met Visual Studio.NET ontwikkelt op Windows maakt het niet zoveel uit, maar op Linux werkt het gewoon niet als de casing verkeerd is.

## Windows-applicaties

De Windows Forms classes van het .NET Framework maken direct gebruik van de WIN32 API. Deze API is niet standaard beschikbaar op Linux waardoor het nu nog niet mogelijk is om Windows Forms te gebruiken in Mono. In de volgende release van Mono zal er wel ondersteuning zijn voor Windows Forms door het gebruik van Wine die de WIN32 API emuleert. Tot die tijd is er een goed alternatief beschikbaar dat op alle platforms werkt. De makers van Mono zijn namelijk ook de ontwikkelaars van Gnome, de grafische desktop voor Linux. Zij hebben een GTK#-schil om de GTK-libraries gecreëerd op dezelfde wijze als


**Afbeelding 3. Testpagina voor webservices in XSP**



Afbeelding 4. Visual Studio.NET met Glade-integratie

Microsoft .NET Windows Forms een schil is om WIN32 API. Nu is GTK# alleen een library met controls; of widgets zoals dat in GTK# heet. Als de ontwikkelaar daarmee een form bouwt, moet hij alles handmatig coderen. Dat dit omslachtig is en handiger kan, hebben de makers van GTK ook gezien. Zij hebben daarom een extra GTK-library gemaakt met de naam Glade. Met deze Glade-library kun je met een XML-bestand een formdefinitie vastleggen, waarna de Glade-library ervoor zorgt dat deze op het scherm verschijnt. Dit is ook de manier van formontwerp waar Microsoft naar toe gaat met XAML. Net als met GTK is ook voor Glade een schil gemaakt in Mono genaamd Glade#. De Glade Designer is een applicatie waarmee de ontwerper forms kan maken en opslaan als Glade XML-bestand. Deze ontwerptool is beschikbaar op Linux en Windows en er is zelfs een integratie voor Visual Studio.NET beschikbaar. Voor .NET-ontwikkelaars is Glade wel even wennen, omdat de manier van werken bij Glade niet is gebaseerd op vaste positionering, zoals in de Designer in Visual Studio.NET. Het lijkt op het ontwerp van een HTML-pagina met tabellen. Door deze manier van ontwerpen is de applicatie beter schaalbaar voor verschillende resoluties. Wel bestaat nog de mogelijkheid om met vaste positionering te werken, maar dat wordt afgeraden. Codevoorbeeld 2 en het bijbehorende Glade-bestand (codevoorbeeld 3) tonen met de kleurmarkeringen aan hoe de code en het Glade-bestand samenwerken.
Jammer genoeg is de integratie met Visual Studio.NET nog niet compleet, wat tot gevolg heeft dat je handmatig de eventhandlers die in het Glade XML-bestand staan, moet toevoegen omdat anders de applicatie niet werkt. Databinding is in GTK# al wel aanwezig, maar helaas nog niet met datasets. Dit betekent dat bij gebruik van databinding de dataset moet worden vertaald naar een TreeStore, het type dat in GTK# gebruikt wordt als datacontainer.
Natuurlijk zijn er verscheidene crossplatform libraries die zowel op Mono als het .NET Framework werken, bijvoorbeeld wxWidgets, maar de ondersteuning voor GTK# is toch het uitgebreidst in Mono.

## Databasetoegang

De ondersteuning voor andere databases dan SQL Server is in Mono uitgebreider dan in het .NET Framework . Zo zijn er onder andere managed code libraries in Mono aanwezig voor toegang tot MySQL, IBM DB2, Oracle, PostgreSQL en natuurlijk Microsoft SQL Server. Deze libraries zijn ook te gebruiken in het .NET Framework en vormen dus een goed alternatief voor de OLEDB-oplossing die nu vaak nodig is. Deze keuzemogelijkheid is erg belangrijk omdat de databasekeuze mede afhangt van het platform waarop het moet draaien. Sommige databases zijn niet beschikbaar op alle platformen en dat kan bepalend zijn voor de keuze. Als je applicatie bijvoorbeeld inclusief database op meer platformen moet draaien, wordt de keuze behoorlijk beperkt. In Mono heeft men hier wel rekening mee gehouden en is de Mono.Data.ProviderFactory-class gemaakt, waarmee de database die gekozen is in het configuratiebestand, kan worden aangepast. In de praktijk moet je je afvragen of dit wenselijk is, omdat alle SQL statements en stored procedures moeten worden gemaakt voor alle ondersteunde databases. Het is mogelijk om een standaard als de SQL-92 aan te houden voor het coderen van SQL-statements, maar dit blijkt toch lastiger dan het lijkt omdat de ontwikkelaar soms handige functies nodig heeft.

```
using System;
using Gtk;
using Glade;

public class WindowMain
{
   [Glade.Widget]
   Entry txtF;
   [Glade.Widget]
   Entry txtC;

   public WindowMain (string[] args)
   {
      Glade.XML gxml = new Glade.XML (null, "gui.glade", "windowMain", null);
      gxml.Autoconnect (this);
   }

   // Connect the Signals defined in Glade
   public void OnWindowDeleteEvent (object o, DeleteEventArgs args)
   {
      Application.Quit ();
      args.RetVal = true;
   }

   //
   public void on_menuFileQuit_activate ( object o, EventArgs args )
   {
      Application.Quit();
   }

   public void on_btnQuit_clicked ( object o, EventArgs args )
   {
      Application.Quit();
   }

   public void on_btnToF_clicked ( object o, EventArgs args )
   {
      double c = Convert.ToDouble( txtC.Text );
      double f = (212.0-32.0)/100.0 * c + 32.0;
      txtF.Text = f.ToString();
   }

   public void on_btnToC_clicked ( object o, EventArgs args )
   {
      double f = Convert.ToDouble( txtF.Text );
      double c = 100.0/(212.0-32.0) * (f - 32.0 );
      txtC.Text = c.ToString();
   }

   public static void Main (string[] args)
   {
      Application.Init();
      new WindowMain (args);
      Application.Run();
   }
}
```

Codevoorbeeld 2.
GladeSharp demo-code

Een zeer belangrijk punt waar op gelet moet worden zijn transacties. In het .NET Framework kun je met custom attributes transacties aangeven op classes die overerven van ServicedComponent. ServicedComponent maakt daarbij gebruik van COM+ dat niet beschikbaar is op andere platformen. Het is dus niet mogelijk om het transactiemechanisme, dat het .NET Framework in EnterpriseServices aanbiedt, te gebruiken wanneer de applicatie ook onder Mono moet draaien. Databasetransacties moeten worden geprogrammeerd met de transaction-class in de gebruikte databaselibrary. Databaseoverstijgende transacties zal de ontwikkelaar zelf moeten uitwerken door de bouw van een eigen DTC.

## Extra libraries

Mono heeft behalve de libraries uit de Framework SDK ook nog een groot aantal extra libraries. Deze libraries zijn te gebruiken met Mono, maar het is ook mogelijk deze libraries toe te passen met het .NET Framework. Aangezien Novell de sponsor en eigenaar is van het Monoproject zijn er Mono-libraries aanwezig voor de ondersteuning van Novell LDAP. Net als Windows Forms gebruikmaakt van de Win32 API maakt de security library van het .NET Framework gebruik van de Windows Crypto API. Deze Crypto API is natuurlijk niet beschikbaar op Linux. Mono beschikt daarom over managed security libraries die de Crypto API niet nodig hebben. De Mono security libraries verzorgen bijvoorbeeld diverse encryptie-algoritmen, SSL-ondersteuning en X509-certificaten. Zoals eerder genoemd is Mono ontwikkeld door de makers van Gnome. Rondom Gnome is een rijke set met libraries ontstaan die nu ook beschikbaar zijn vanuit Mono. Zo kun je ART-libraries gebruiken voor het betere grafische werk. ART is te vergelijken met GDI+. Maar er zijn nog veel meer libraries beschikbaar voor gebruik in applicaties. Voorwaarde is natuurlijk is wel dat de applicatie op GTK# gebaseerd is en niet op Windows Forms.

```
<?xml version="1.0" standalone="no"?> <!--*- mode: xml -*-->
<!DOCTYPE glade-interface SYSTEM "http://glade.gnome.org/glade-2.0.dtd">

<glade-interface>

<widget class="GtkWindow" id="windowMain">
  <property name="visible">True</property>
  <property name="title" translatable="yes">Glade Demo</property>
  <property name="type">GTK_WINDOW_TOPLEVEL</property>
...
  <signal name="delete_event" handler="OnWindowDeleteEvent"/>

  <child>
    <widget class="GtkVBox" id="vbox1">
      <property name="visible">True</property>
      <property name="homogeneous">False</property>
      <property name="spacing">0</property>

  <widget class="GtkMenuBar" id="menubar1">
    <property name="visible">True</property>
...
    <child>
  <widget class="GtkHBox" id="hbox1">
...
    <child>
      <widget class="GtkTable" id="table2_u62 ?
        <property name="visible">True</property>
        <property name="n_rows">3</property>
        <property name="n_columns">2</property>
...
    <widget class="GtkLabel" id="label2">
      <property name="visible">True</property>
      <property name="label" translatable="yes">Fahrenheit</property>
...
    <widget class="GtkEntry" id="txtC">
      <property name="visible">True</property>
      <property name="can_focus">True</property>
      <property name="editable">True</property>
...
    <widget class="GtkButton" id="btnToC">
      <property name="visible">True</property>
      <property name="can_focus">True</property>
      <property name="label" translatable="yes">Celsius?</property>
      <property name="use_underline">True</property>
      <property name="relief">GTK_RELIEF_NORMAL</property>
      <signal name="clicked" handler="on_btnToC_clicked"
      last_modification_time="Sun, 09 May 2004 12:19:58 GMT"/>
    </widget>
...
  </child>
</widget>
```

**Codevoorbeeld 3.**
**Glade-definitie bestand (bij ... is er een sprong het XML-bestand gemaakt)**

## Setup

Applicatieontwikkeling voor verschillende platformen betekent ook installatie voor al die platformen. Elk platform heeft zo zijn eigen manier van installatie. Op Windows worden MSI-packages gebruikt die je makkelijk met Visual Studio.NET kan maken. Maar op Linux is het een heel ander verhaal. Daar zijn de belangrijkste manieren van softwaredistributie een tarball met de source-code of een RPM met de assemblies. Hierbij moeten de RPM's vaak nog specifiek voor een bepaalde Linuxdistributie gemaakt worden. Waarschijnlijk is één RPM voldoende als alleen maar managed code gebruikt wordt. Net als het .NET Framework niet standaard op elke Windowsmachine aanwezig is, is ook Mono niet standaard aanwezig op elke Linuxmachine. Houdt daar dus rekening mee.

## De moeite waard

Gezien het feit dat er nog niet veel bekend is over stabiliteit en schaalbaarheid van Mono in productieomgevingen is het verstandig om .NET-software voorlopig op Windows met het .NET Framework te draaien. Maar je kunt nu al wel de software voorbereiden zodat het ook op andere platformen met Mono kan draaien. Vooral ASP.NET-applicaties en webservices zonder WSE zijn vrij probleemloos voor het .NET Framework en Mono te ontwikkelen. Voor Windows-applicaties is het echter een ander verhaal. In dat geval zal de ontwikkelaar een keuze moeten maken tussen Windows Forms en GTK#. Als de applicatie ook op andere platformen moet draaien, is GTK# op dit moment de enige optie. Ook wat betreft de database moet je vooruitkijken. Op welk platform moet de database nu en in de toekomst draaien en welke eisen worden aan de database gesteld? Vooral transactiemanagement is een onderwerp dat aandacht verdient. Het blijft uiteraard altijd mogelijk om onderdelen uit Mono te gebruiken in de applicatie terwijl die helemaal niet Mono-compatibel is. Hierbij komen de libraries Mono.Security en Novell.LDAP voor databasetoegang om de hoek te kijken.
Mono biedt ontzettend veel voor .NET-ontwikkelaars en is zeker de moeite waard om eens te onderzoeken. Voor meer informatie over Mono of als je het wilt downloaden is http://www.mono-project.com de aangewezen website.

**Leendert Versluijs** is managing consultant bij Capgemini en gespecialiseerd in .NET-ontwikkeling. www.capgemini.nl

**Nuttige internetadressen**

- http://www.mono-project.com
- http://www.monodevelop.com
- http://www.dotgnu.org
- http://www.go-mono.nl
- http://www.go-mono.com/faq.html