Bob Swart is auteur, trainer en consultant bij Bob Swart Training & Consultancy.

Delphi 8 for .NET

HOE WERKT DELPHI 8 FOR .NET EN WAT ZIJN DE VERSCHILLEN MET VISUAL STUDIO.NET?

Dit artikel introduceert Delphi 8 for .NET, en laat zien hoe we .NET-toepassingen kunnen ontwikkelen met de nieuwste IDE voor het .NET Framework. Omdat de meeste lezers op de hoogte zullen zijn van de mogelijkheden van Visual Studio.NET, gaat dit artikel met name in op de verschillen, zowel in positieve als wat minder positieve zin.

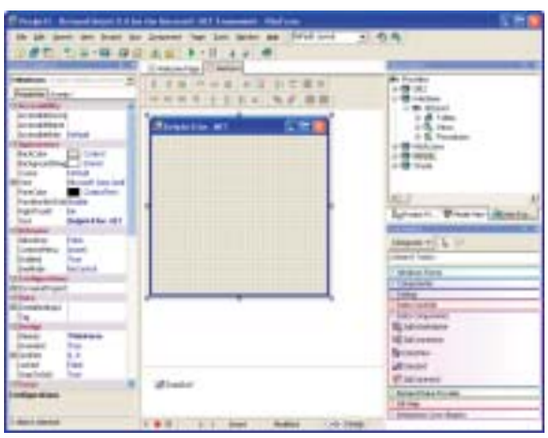
Delphi 8 for the Microsoft .NET Framework is de officiële naam, maar de meeste ontwikkelaars noemen het gewoon Delphi 8 of Delphi 8 for .NET (alleen Delphi for .NET is niet volledig, want bij Delphi 7 zat eind 2002 al een Delphi for .NET preview command-line compiler, die echter niet te vergelijken is met wat nu als Delphi 8 for .NET beschikbaar is). Alhoewel Delphi 8 for .NET een relatieve nieuwkomer is op het .NET Framework, geldt dat niet voor de taal Delphi zelf. Delphi 1.0 wordt op Valentijns dag in 1995 gelanceerd, en was in feite de 8ste generatie van Turbo Pascal compiler, die het eerste daglicht ziet in het begin van de 80-er jaren. Met Delphi 8 is de compiler zelf toe aan versie 16 (ook de Delphi compiler in C++Builder en Kylix hebben gezorgd voor aanpassingen aan het versienummer). Vanaf Turbo Pascal versie 1.0 t/m Delphi versie 2.0 is trouwens niemand minder dan Anders Hejlsberg de architect geweest voordat hij naar Microsoft vertrekt om daar een belangrijke rol te spelen in de ontwikkeling van de taal C# en het .NET Framework zelf.

Migratie en Portabiliteit

Alhoewel het .NET Framework erg belangrijk is voor Borland, zijn er nog veel ontwikkelaars die Delphi 7 gebruiken om 32-bit Windows-toepassingen te bouwen en te onderhouden, en hier lijkt voorlopig nog geen einde aan te komen. De volgende versie van Delphi - versie 9 – stelt ontwikkelaars zelfs in staat om vanuit één omgeving zowel Windows als .NET-toepassingen te bouwen. Zover is het echter nog niet, en op dit moment moeten we Delphi 7 nog gebruiken om Win32-toepassingen te bouwen (of te onderhouden), want Delphi 8 (for .NET) kan "alleen" maar .NET-toepassingen produceren. Met nog wel een extraatje - zie daarvoor het onderdeel over Assemblies en Libraries.

WinForms

Delphi 8 for .NET biedt net als Visual Studio de mogelijkheid om WinForms-toepassingen te bouwen. De WinForms Designer lijkt niet alleen als twee druppels water op die van Visual Studio, maar is daadwerkelijk de designer van Microsoft. Dat heeft als voordeel dat gebruikers van Visual Studio zonder al teveel problemen de proefversie van Delphi 8 for .NET kunnen gebruiken om eens te proeven hoe het werkt.[1]



Afbeelding 1. Delphi 8 for .NET IDE voor WinForms Project

In afbeelding 1 zien we de Object Inspector, WinForms Designer, een Tool Palette met componenten en rechtsboven een venstertje waarin je met de Project Manager, Model View (daarover later) of the Data Explorer kan werken.

ADO.NET en BDP

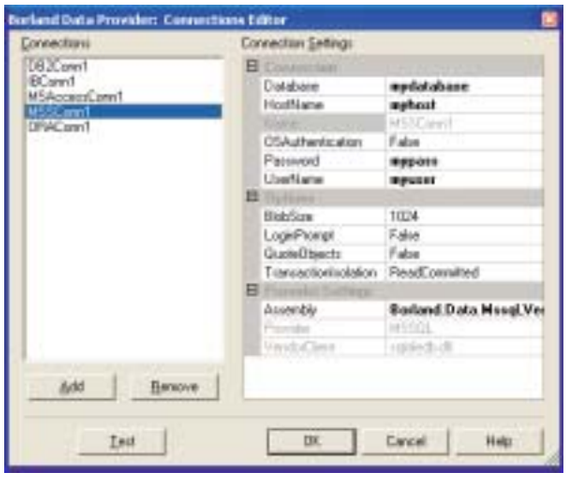
Om met databases te werken in Windows Forms of Web Forms toepassingen kunnen we ADO.NET gebruiken. Ook Delphi 8 for .NET ondersteunt ADO.NET, alhoewel je niet de design-time wizards zult aantreffen die wel in Visual Studio.NET aanwezig zijn. Het gebruik van ADO.NET gaat in Delphi 8 for .NET dan ook wat moeizamer. De reden dat Delphi 8 for .NET beschikt over een eigen laagje over ADO.NET met de naam Borland Data Providers. Het ADO.NET pattern is gevolgd, dus we hebben nu een BdpConnection, BdpDataAdapter, BdpCommand, BdpCommandBuilder, etc. die met de standaard .NET DataSet, en DataView kunnen werken op de manier zoals we die gewend zijn. Het grootste verschil zit hem in het feit dat de BdpConnection door middel van het aanpassen van de ConnectionString ook van database type zelf kan wijzigen. De overige BdpComponenten zullen niet hoeven te wijzigen, en zijn 100% database onafhankelijk (net als de .NET DataSet zelf bijvoorbeeld).

Borland levert met Interbase zelf ook een database, maar het is belangrijker dat Delphi gebruikt kan worden in combinatie met alle grote spelers in de markt. Standaard zitten er al vijf BDP drivers in Delphi 8 (voor DB2, Interbase, Microsoft Access, Microsoft SQL Server, en Oracle), en er zijn al enkele third-party aanbieders die ook hun eigen BDP drivers hebben geschreven en aanbieden. Dit is interessant voor ontwikkelaars die een .NET-toepassing willen schrijven zonder daarbij bij voorbaat al voor een bepaalde database te kiezen.

In afbeelding 2 is de Connections Editor te zien waarmee de ConnectionString property van de BdpConnection component te configureren is. Voor iedere aanwezige DBMS driver (opgesomd in bestand BdpDataSourcex.xml) is al een voorbeeldconnectie opgenomen. De sectie met ProviderSettings in afbeelding 2 geeft aan welke properties op basis van het DBMS zijn ingesteld, en zelf kun je dan nog de Connection properties en eventuele Options instellen.

[1] Lezers kunnen de proefversie van Delphi 8 for .NET downloaden van de Borland website te http://www.borland.com/products/downloads/download_delphi_net.html of op CD bestellen bij nl-inside-sales@borland.com.

net#5_p22-25 1.12 22 18-05-2004, 10:11:41



Afbeelding 2. Connections Editor

Een andere feature van de Borland Data Provider-componenten is de mogelijkheid om "live" data tijdens design-time te vertonen (ook in ASP.NET pagina's met de DB Web controls).

VCL for .NET

Naast ondersteuning voor Windows Forms-toepassingen, kunnen we met Delphi for .NET ook safe en managed VCL for .NET-toepassingen ontwikkelen. VCL staat hier voor de Visual Component Lilbrary die al sinds 1995 onderdeel is van Delphi 1 t/m 7. De controls in de Windows.Forms hierarchy lijken sterk op de controls in de VCL hierarchy - er is een duidelijke overeenkomst. Er is echter ook een aantal verschillen, met name in de properties en sommige events of methoden. De VCL kent bijvoorbeeld aan ieder component zowel een Parent als een Owner toe (de Parent is de visuele vader, de Owner beheert het geheugen). Iets wat niet onder WinForms gebeurt. De verschillen tussen WinForms en VCL zijn niet dramatisch, en voor een ontwikkelaar redelijk snel te leren, maar voldoende om bestaande Delphi-broncode niet zonder meer om te zetten naar een vergelijkbare WinForms-versie.

Dit is de reden waarom er speciaal voor .NET een VCL is gekomen, inclusief een eigen VCL for .NET Designer. VCL for .NET is niet een laag op WinForms, maar een laag op de Win32 API. Het biedt dezelfde "interface" als de VCL voor Windows, maar dan alleen maar gebruikmakend van safe, managed code (even los van de Win32 API aanroepen). Een VCL-toepassing is te migreren naar een VCL for .NET-toepassing door alle stukken "unsafe" code om te schrijven of aan te passen. De weg terug is zelfs eenvoudiger: een VCL for .NET-toepassing kan met Delphi 7 gecompileerd worden tot een Win32-toepassing (mits geen gebruik is gemaakt van de Delphi-taaluitbreidingen die in Delphi 8 zijn geïntroduceerd).

Van de ruim 100 VCL-componenten hebben slechts een paar dozijn de overstap naar .NET en VCL for .NET (nog) niet meegemaakt. De volledige source code van VCL for .NET is meegeleverd met Delphi 8 for .NET, en ook enkele (maar nog zeker niet alle) third-party leveranciers brengen .NET-edities uit van hun Delphi-componenten.

Onder de VCL-componenten die .NET (nog) niet gehaald hebben zit o.a. de ondersteuning om databases via ADO te benaderen. Gelukkig bevat de VCL een flink aantal (alternatieve) manieren om met databases te werken. Zo is er de Borland Database Engine (BDE) met ondersteuning voor dBASE en Paradox-tabellen - nu ook voor .NET. Onder Windows was hier ook nog een SQL Links-laag bij om met SQL Server, DB2, Oracle, Interbase, etc. te werken, maar deze technologie is afgeschreven en niet meer beschikbaar onder .NET (of de volgende versies van Delphi). De gedoodverfde vervanger van BDE/SQL Links heet dbExpress, en is er in de smaken Windows en Linux. En nu ook in een .NET-variant. Via dbExpress kunnen we met SQL Server, DB2, Oracle, Interbase en onder .NET ook Sybase werken. Daarnaast is ook Interbase Express (IBX) in een .NET-variant aanwezig.

VCL versus WinForms

De VCL for .NET en WinForms-componenten leven naast elkaar - ieder in zijn eigen wereld. VCL for .NET-componenten kunnen niet in WinForms-projecten gebruikt worden. "Normale" Delphi classes uit de RunTime Library (RTL) kunnen dat uiteraard wel. WinForms controls kunnen in VCL for .NET-projecten gebruikt worden, doordat Delphi 8 for .NET daar een speciale "wrapper" omheen kan genereren. Net als het importeren van een COM-object in Win32.

ASP.NET 1.1

Delphi 8 for .NET bevat ondersteuning voor ASP.NET, zowel voor het bouwen van ASP.NET Web Services als ASP.NET Web Forms-toepassingen. Wie ASP.NET wil combineren met ADO.NET in Delphi 8 for .NET, komt er wederom achter dat er meer ondersteuning aanwezig is voor de Broland Data Providers. Daarnaast is er een verzameling DB Web voor ASP.NET Web Forms aan toegevoegd. Dit zijn componenten die het werken met databases ASP.NET Web Forms vereenvoudigen; een hoop zaken worden automatisch gedaan of makkelijker gemaakt (zoals de automatische data binding, navigatie van het ene record naar het andere binnen een DataSet, edit/update en paginering binnen een DataGrid, etc.).

De DB Web controls zijn eenvoudig te deployen, en ook bruikbaar binnen andere (ASP).NET-ontwikkelomgevingen (alhoewel ze niet los te koop zijn). De volledige source code van de DB Web controls is ook aanwezig, deze keer in C# geschreven.

Assemblies en Libraries

We kunnen natuurlijk ook .NET assemblies bouwen met Delphi 8 for .NET. Dit kan zelfs op twee manieren: via packages en via het library keyword. Een Delphi package is de aanbevolen manier om .NET assemblies te bouwen, die vervolgens andere .NET-ontwikkelomgevingen (en toepassingen) kunnen gebruiken. Hierbij moeten we dan wel de benodigde afhankelijkheden meeleveren, zoals bijvoorbeeld de Borland.Delphi.dll assembly met de system unit van Delphi. Deze en enkele tientallen andere Borland.Delphi assemblies kan iemand, die een Delphi 8-licentie bezit, vrij distribueren. Optioneel kun je er ook voor kiezen om alle Borland.Delphi assemblies mee te linken in de package, teneinde een grote maar opzichzelfstaande assembly op te leveren. Dit kun je ook met executables zelf doen, waardoor je het aantal te deployen bestanden kunt terugbrengen tot slechts één (die dan wel groot is, omdat alle benodigde Borland.Delphi assemblies statisch meegelinkt zijn).

Het alternatief voor de package is een library. Deze mogelijkheid is een overblijfsel uit de Win32-wereld en biedt de mogelijkheid om een Dynamic Link Library te bouwen. Met Delphi 8 for .NET krijg je een .NET assembly, maar dan eentje waarbij op voorbaat alle afhankelijkheden meegelinkt zijn (dus één grote monolitisch IL-bestand). Dit heeft als voordeel dat iedere ontwikkelaar die geen Delphi 8 bezit alleen maar deze grote assembly hoeft te gebruiken en deze zonder problemen kan deployen. Behalve dan Delphi 8-ontwikkelaars zelf, die tegen het probleem aanlopen dat de Borland.Delphi.System namespace zowel in de grote assembly is opgenomen, als iedere Delphi 8-toepassing die je wilt bouwen. En dat levert foutmeldingen op vanwege o.a. de dan optredende dubbele globale data-definities.

Waarom dan toch een library gebruiken? Eigenlijk alleen vanwege een andere eigenschap, namelijk de mogelijkheid om de code binnen de library als "unsafe" te markeren. Dat levert een assembly op die weliswaar unsafe is, maar te gebruiken is vanuit

net#5_p22-25 1.12 23 18-05-2004, 10:11:45

een .NET-toepassing en een Win32-toepassing. Dezelfde binary .NET assembly dus. Dit is geen source code-compatibiliteit (dezelfde source code die tot een Win32 DLL of een .NET assembly kan leiden), maar binary compatibiliteit, waarbij zowel een .NET executable als een Win32 executable dezelfde binary kunnen gebruiken. Het grote voordeel van deze mogelijkheid is dat je als Delphi-ontwikkelaar op dit moment code kunt ontwikkelen die zowel binnen Delphi 7 (Win32) toepassingen te gebruiken is (als DLL), maar ook binnen .NET-toepassingen te gebruiken is als (unsafe) assembly. Dit laatste kan niet alleen vanuit Delphi 8 for .NET, maar ook vanuit Visual Studio.NET en andere .NET-ontwikkelomgevingen.
Bekijk als voorbeeld eens de code in listing 1 (HashPasswordUnit.pas). Dit is unit van een Delphi 8 library assembly, die met de *{$unsafecode on}* compiler directive gemarkeerd is, waardoor PEVerify aangeeft dat hij niet safe is. De assembly maakt gewoon gebruik van de eigenschappen van het .NET Framework, bijvoorbeeld om een password te encrypten via de FormsAuthentication.HashPasswordForStoringInConfigFile uit System.Web.Security.

De met Delphi 8 for .NET gecompileerde assembly is te gebruiken vanuit een Delphi Win32-toepassing (gecompileerd met Delphi 2 t/m 7). Zie hiervoor listing 2 (Win32Client.dpr). Dezelfde met Delphi 8 for .NET gecompileerde assembly is echter ook vanuit een .NET-toepassing, zoals een C#-toepassing te gebruiken. Merk op dat de code in listing 1 alleen maar functies laat zien, die geëxporteerd worden, en geen classes. Dat is iets wat in een Win32 DLL normaal was, maar de meeste .NET-ontwikkelomgevingen zijn het meer gewend om classes te importeren en te gebruiken, en geen globale functies (of variabelen) meer. Om daarin te voorzien voegt de Delphi 8 compiler een impliciete class genaamd "Unit" toe om vervolgens daarin alle globale functies en eventuele variabelen in op te bergen. Op die manier kunnen ook C#- en andere .NET-ontwikkelaars toch zonder problemen bij de andere globale methoden komen. Overigens zijn

```
unit HashPasswordUnit;
{$UNSAFECODE ON}
interface

function HashPasswordMD5(const Passwd: String): String;

function HashPasswordSHA1(const Passwd: String): String;

exports
  HashPasswordMD5,
  HashPasswordSHA1;

implementation
uses
  System.Web.Security;

function HashPasswordMD5(const Passwd: String): String;
begin
  Result := FormsAuthentication.
   HashPasswordForStoringInConfigFile(Passwd, 'MD5')
end;

function HashPasswordSHA1(const Passwd: String): String;
begin
  Result := FormsAuthentication.
   HashPasswordForStoringInConfigFile(Passwd, 'SHA1')
end;

end.
```

Listing 1.

```
program Win32Client;
{$APPTYPE CONSOLE}

type
  TPasswordFormat = (SHA1, MD5);

function HashPasswordMD5(Password: PChar): PChar; stdcall;
  external 'HashPassword.dll';

function HashPasswordSHA1(Password: PChar): PChar; stdcall;
  external 'HashPassword.dll';

var
  Passwd: String;
begin
  write('Password: ');
  readln(Passwd);
  writeln('['+Passwd+']');
  write('MD5: ');
  writeln(HashPasswordMD5(PChar(Passwd)));
  write('SHA1: ');
  writeln(HashPasswordSHA1(PChar(Passwd)));
end.
```

Listing 2.

globale routines gewoon mogelijk binnen de CLR. Het is dus niet zo dat Delphi iets zou doen wat niet mag.

Enterprise Core Objects

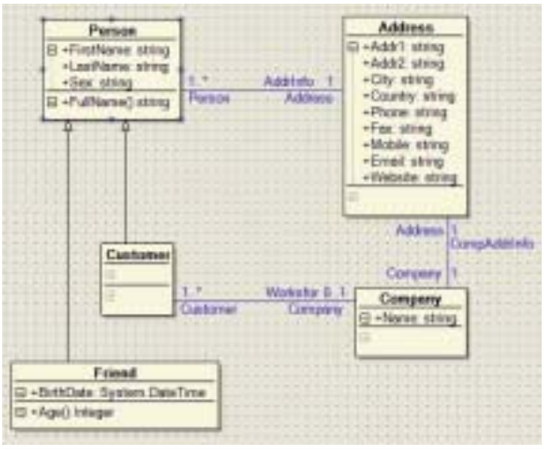
De Architect-editie van Delphi 8 for .NET bevat met Enterprise Core Objects (ECO) ondersteuning voor het zogenaamde model driven application development. Dit houdt in dat er op basis van UML-diagrammen (die je zelf kunt ontwerpen in de UML Designer) code gegenereerd wordt, alsmede een objectmodel dat opgeslagen kan worden als XML-bestand of in een database (zoals SQL Server of elke andere database die door de Borland Data Providers kan worden aangesproken). Er hoeft dus geen aparte vertaling met de hand meer plaats te vinden van het ontwerp (model) naar de source code. Het ontwerp en de code lopen altijd synchroon.
In afbeeling 3 is een voorbeeldmodel te zien in de UML Designer. We kunnen daar classes, relaties (associations), overervingen, notes en andere links aan toevoegen. Een class kan zowel attributen als operators hebben. Om de objecten uit de EcoSpace te gebruiken kunnen we Object Constraint Language (OCL) expressies samenstellen. Hiervoor is Delphi 8 for .NET uitgerust met een OCL editor.

ECO bevat elementen van Together en Bold, twee bedrijven die vorig jaar door Borland zijn overgenomen en waarvan de gevolgen dus zichtbaar worden in de Borland ontwikkelomgeving. Een eerste versie van ECO was ook al te zien in C#Builder, en de editie die bij Delphi 8 for .NET aanwezig is, heeft al weer wat nieuwe mogelijkheden. Er blijven wel nog wat wensen over: op dit moment is het namelijk nog niet mogelijk om het objectmodel (de zogeheten EcoSpace) als multi-user database te benaderen. Ook de mogelijkheden om er een multi-tier server van te bouwen zijn daarmee nog niet aanwezig. Dit staat als aandachtspunt op de wensenlijst, maar het is nog niet bekend wanneer deze extra functionaliteit beschikbaar komt.

Conclusie

Delphi 8 for .NET biedt de mogelijkheid om nieuwe .NET-toepassingen te bouwen (WinForms, Web Forms, etc.). Daarnaast stelt het Delphi 7-ontwikkelaars in staat om hun bestaande code geheel of gedeeltelijk naar .NET over te hevelen. Voor visuele zaken geldt dat VCL for .NET het grootste deel van de bestaande VCL (voor Win32) onder .NET beschikbaar maakt. Behalve

net#5_p22-25 1.12 24 18-05-2004, 10:11:48



Afbeelding 3. ECO Model in UML Designer

migratie van code, is de reverse P/Invoke (unsafe) assembly een extra mogelijkheid om nieuwe functionaliteit toe te voegen aan bestaande Win32-toepassingen én nieuwe .NET-toepassingen. De Borland Data Providers bieden enkele voordelen boven ADO.NET, maar wie toch ADO.NET wil gebruiken zal merken dat daar nog weinig design-time ondersteuning voor in Delphi 8 for .NET zit. Het bouwen van .NET Compact Framework (CF) toepassingen is overigens nog niet mogelijk met Delphi 8 for .NET. Hiervoor moeten we nog Visual Studio.NET gebruiken, maar Microsoft en Borland werken hard samen om dit ook mogelijk te maken voor Delphi-ontwikkelaars. Ik verwacht zelf een groeiend belang van het Model Driven Application development model en Enterprise Core Objects. Delphi 8 for .NET kan hier een leuke rol in gaan spelen.

Referenties

- Borland Delphi 8 for .NET: http://www.borland.com/delphi_net
- Borland Data Provider (BDP) for the Microsoft .NET Framework Microsoft .NET Framework: http://www.borland.com/products/white_papers/del_borland_data_provider_for_dotnet.html
- Migrating Borland Delphi apps to the .NET Framework with Delphi 8 - Bob Swart: http://www.borland.com/products/white_papers/del_migrating_delphi_win32_to_dotnet.html
- Why VCL for .NET? - Danny Thorpe: http://bdn.borland.com/article/0,1410,31983,00.html
- Borland Delphi 8 for the Microsoft .NET Framework 1.1: http://www.drbob42.com/examines/examin49.htm

Bob Swart is een onafhankelijk ontwikkelaar, auteur, trainer, webmaster en consultant en heeft een eenmanszaak onder de naam Bob Swart Training & Consultancy (eBob42) in Helmond. Bob is bereikbaar via Bob@eBob42.com (www.eBob42.com).

MICROSOFT VIRTUAL PC 2004

Virtual PC 2004 is een krachtige virtuele omgeving waarmee u meerdere pc-gebaseerde besturingssystemen en toepassingen op één werkstation kunt uitvoeren. Elke virtuele machine bootst een compleet hardwaresysteem na, van processor tot netwerkkaart, in een op zichzelf staande, geïsoleerde software-omgeving. Toepassingen in Windows XP, Windows 2000, Windows NT, Windows 98, Windows 95, Windows 3.1 en zelfs DOS of OS/2, en kunnen tegelijkertijd op dezelfde pc worden uitgevoerd, waarbij opnieuw opstarten niet nodig is. Dankzij Virtual PC kunt u toepassingen uitvoeren op één systeem die normaal gesproken onderling niet compatibel zijn. Virtual PC 2004 zorgt voor een veiligheidsnet tussen de verschillende virtuele omgevingen waardoor u de compatibiliteit kunt handhaven met bestaande applicaties.
Dit biedt mogelijkheden voor de volgende scenario's:

- Ontwikkelaars die op één machine zowel de ontwikkel-, test- en productie-omgeving draaien.
- Testen van applicaties op verschillende besturingssystemen zonder onnodig veel hardware te herconfigureren.
- Het oplossen van problemen bij applicaties die onderling niet compatible zijn.
- Het wegnemen van afhankelijkheid van een toepassing die een bepaalde versie besturingssysteem vereist.
- Mogelijkheid van een gefaseerde migratie. U kunt laat de 'oude' applicatie nog in een virtuele omgeving draaien terwijl u de PC en applicaties upgrade naar een nieuwe Windows-versie.

De volgende besturingssystemen kun u op Virtual PC 2004 installeren

Microsoft besturingssystemen: Windows 95, Windows 98, Windows Millennium Edition (Windows Me), Windows NT®, Windows 2000, Windows XP and MS-DOS®.
Andere besturingssystemen: IBM OS/2, Red Hat Linux, Novell NetWare en vele andere. Alleen de volgende besturingssystemen OS/2 Warp Versie 4 Fix Pack 15, OS/2 Warp Convenience Pack 1 en OS/2 Warp Convenience Pack 2 worden als niet-Microsoft gast besturingssystemen ondersteund.

Microsoft®
Virtual PC

net#5_p22-25 1.12 25 18-05-2004, 10:11:48