Delphi OplossingsCourant

Vol 4. No. 3. Een gratis kwartaalpublicatie van **Bob Swart Training & Consultancy (eBob42)** - http://www.eBob42.com

Helmond, 11 september 2002,

Dit nummer van de Delphi OplossingsCourant staat in het teken van Delphi 7 Studio die vandaag officieel in Nederland wordt gelanceerd. De datum 11 september is dan wat gevoelig (laten we hopen dat er vandaag verder niks gebeurt) de lancering van Delphi 7 Studio wil ik niet missen en ik was dan ook vanochtend te vinden achter de stand van Bob Swart Training & Consultancy (eBob42) in Capelle a/d IJssel.

De vorige keer in de Delphi OplossingsCourant had ik het al over het feit dat Delphi 7 nog dit jaar beschikbaar zal komen en een preview versie van Delphi voor .NET zou bevatten (bestaande uit de DCCIL command-line compiler en de VCL for .NET Framework). Wat dat laatste betreft moet ik de lezers helaas een beetje teleurstellen: de Delphi for .NET preview command-line compiler zit wel bij Delphi 7 Studio (op een extra CD-ROM) maar er is helaas nog geen sprake van een VCL for .NET Framework. Gelukkig kunnen we wel genoeg andere nuttige zaken doen, zoals het importeren van COM objecten in .NET (zie pagina 6) en het gebruiken van Delphi for .NET als scripting taal in een ASP.NET omgeving. Daarnaast is het ook mogelijk om .NET Assemblies te exporteren en te gebruiken als COM Objecten in Delphi 7. Op die manier kunnen de .NET wereld en de "oude" Win32 wereld nog steeds met elkaar praten.

De uiteindelijke Delphi voor .NET zal pas later in 2003 beschikbaar komen, maar in ieder geval hebben we vanaf vandaag de kans om managed en safe code voor .NET te bouwen en te gebruiken met de Delphi for .NET preview command-line compiler.

Wie na lezen van de Delphi OplossingsCourant nog vragen of opmerkingen heeft, kan me gerust een mailtje sturen - ik stel alle feedback op prijs (en dat geldt ook voor "verzoeknummers").

Inhoudsopgave

De Delphi OplossingsCourant (DOC) is een gratis productie van Bob Swart Training & Consultancy (eBob42).

Eindredactie: Bob Swart e-mail: doc@eBob42.com

Het volgende nummer van de Delphi Oplossings-Courant zal in december 2002 verschijnen, met nog meer aandacht voor Delphi 7 Studio en de Delphi for .NET preview command-line compiler.

Delphi Clinics in 2002

In de 2e helft van 2002 zal er eens in de 2 weken (op de donderdag) een Delphi 7 Clinic plaatsvinden in Eindhoven. Data en onderwerpen zijn als volgt:

12 sept - **Delphi COM Development**
26 sept - **dbExpress & DataSnap**
10 okt - **WebBroker & InternetExpress**
24 okt - **WebSnap Development**
7 nov - **BizSnap: XML/SOAP & Web Services**
21 nov - **e-Commerce Development**
5 dec - **.NET Framework Development**

voor meer info: http://www.eBob42.com/training

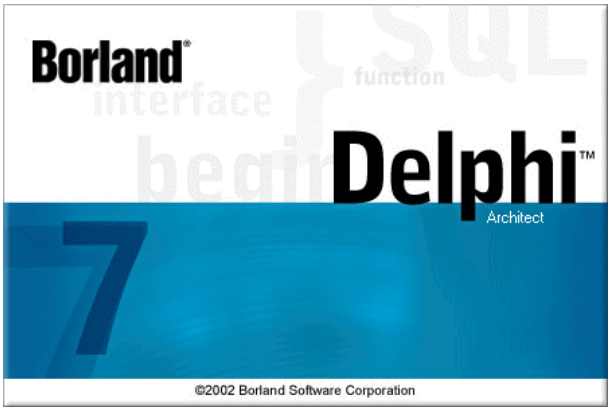
Conferenties 2002

Behalve de Delphi Clinics die weer losbarsten, zal ik ook nog als spreker bij de volgende evenementen aanwezig zijn:

22-27 sept - **Entwickler 2002**, Frankfurt (D)
28-29 okt - **BorCon UK 2002**, London (UK)

Delphi 7 Studio

Op dinsdag 6 augustus werd Delphi 7 Studio aangekondigd, in vier verschillende edities (die nog niet allemaal beschikbaar zijn). Op dit moment zijn Delphi 7 Professional, Enterprise en Architect al te koop en de gratis Delphi 7 Personal editie wordt binnenkort verwacht. Ook kunnen we een 30-dagen proefversie downloaden van Delphi 7 Enterprise.



Nieuwe Features

Er zijn een aantal leuke nieuwe features in Delphi 7 Studio, alhoewel ik er wel eens meer heb gezien in het verleden. De meeste uitbreidingen zijn terug te vinden in BizSnap (vooral SOAP en Web Services) en in de vele third-party tools die nu samen met Delphi 7 Studio in dezelfde doos zitten. Daarnaast staat Delphi 7 dichter bij .NET dan zijn voorganger, doordat we nu .NET assemblies kunnen importeren als COM Objecten, alsmede speciale nieuwe compiler warnings kunnen krijgen voor source code die anders niet zonder meer met de Delphi for .NET preview command-line compiler zal werken.

ModelMaker 6.20

Zo zit er een versie van ModelMaker bij Delphi 7; een demo versie bij Delphi 7 Professional en een echte licentie bij de Delphi 7 Enterprise en Architect edities.

Bold for Delphi

Het verschil tussen Delphi 7 Architect en Delphi 7 Enterprise zit hem in Bold for Delphi, die alleen in de Delphi 7 Architect editie zit. Bold for Delphi is een ontwerptool die daarbij gebruik kan maken van ModelMaker als grafische UML designer (dus dat komt mooi van pas).

Rave Reports (Borland Edition)

QuickReports is kennelijk niet meer de eerste keus, en vanaf Delphi 7 vinden we Rave Reports in de doos. Wie nog steeds QuickReports gebruikt kan het natuurlijk altijd zelf aanschaffen, of de dclqrt70.bpl package met de hand installeren in Delphi 7.

Apache 2

Op internet gebied zijn nog wel enkele leuke nieuwe uitbreidingen te melden. Zo ondersteunt Delphi 7 nu ook Apache 2 als target, alhoewel dat versie 2.0.39 is, die wegens een beveiligingsprobleem reeds vervangen is door versie 2.0.40. Deze laatste versie kent enkele interface wijzigingen die problemen opleveren met Delphi 7 - gebruik dus ofwel de oude versie 2.0.39 met de workaround van Apache op http://httpd.apache.org, of wacht op een update voor Delphi 7 om met Apache 2.0.40 te werken.

Web App Debugger

De Web App Debugger is niet langer op COM gebasseerd, maar gebruikt nu sockets. Dat was wel nodig ook, want nu draait hij tenminste ook onder Linux (met Kylix 3). Als extra feature blijft de Web App Debugger client nu ook "in de lucht" als deze automatisch wordt opgestart - hierdoor blijft de login en sessie informatie beter bewaard tijdens het debuggen. Erg fijn!

IntraWeb 5

Alhoewel WebSnap is uitgebreid met enkele nieuwe scripting mogelijkheden, vinden veel mensen toch dat WebSnap niet echt een RAD manier is om web server toepassingen te ontwikkelen. Met name de leercurve wordt als hoog ervaren.
IntraWeb bevat componenten en wizards om op een visuele wijze (op de RAD-manier) web server toepassingen te bouwen met Delphi. In Delphi 7 Professional zit de IntraWeb 5 Page Mode, terwijl IntraWeb 5 Application Mode in Delphi 7 Enterprise en Architect zit.

Delphi for .NET Preview

En natuurlijk zit ook de Delphi for .NET Preview command-line compiler bij Delphi 7 Studio in de doos (en dus niet bij de 30-dagen proefversie die je kan downloaden).
Helaas nog zonder de zogenaamde VCL for .NET Framework, maar met de mogelijkheden om safe managed .NET code en assemblies te maken en gebruiken, alsmede de eigenschap om als scripting taal in een ASP.NET omgeving te opereren.

Kylix 3

Behalve Delphi 7 is ook Kylix 3 uitgekomen, waarbij zowel een Delphi als een C++ IDE in de Kylix 3 doos zit. Wie echter Delphi 7 aanschaft krijgt een gratis exemplaar van Kylix 3 (voor Delphi) erbij, zodat we nu in één keer voor Windows en Linux kunnen ontwikkelen (zelf gebruik ik Kylix 3 alleen maar als command-line compiler voor mijn Delphi 7 web server toepassingen en web services).

Google Web Services

Er is weer gesleuteld aan de SOAP ondersteuning in Delphi 7 (en Kylix) en er zijn ook heel wat nuttige(re) web services beschikbaar gekomen op het internet. Waaronder de Google Web APIs die je in staat stellen om het web af te zoeken, net als de search engine Google zelf. Je geeft hiermee één of meerdere zoekwoorden en als resultaat krijg je een verzameling web pagina's waar deze zoekwoorden in voorkomen. Omdat je soms nogal veel resultaten kunt krijgen - gesorteerd op basis van de mate waarin de zoekwoorden voorkomen in de pagina's - worden die over het algemeen in groepjes van 10 gepresenteerd (en wie veel bezoekers wil, zou dan ook met name in die eerste 10 vertoond willen worden, omdat er weinig mensen zijn die de volgende 10 willen bekijken).

Google Web APIs

Informatie over de officiële Google Web APIs (beta 2) is te vinden op de Google website zelf te http://www.google.com/apis/. Om de een of andere reden noemt Google de API zelf geen web service (alhoewel daar juist veel hype omheen is de laatste tijd) maar het is er wel degelijk eentje.
Om de Google Web APIs te gebruiken moeten we drie dingen doen:

1. Developer's kit downloaden (658.031 bytes)
2. Google Search key aanvragen
3. Programma bouwen met de Google Search key

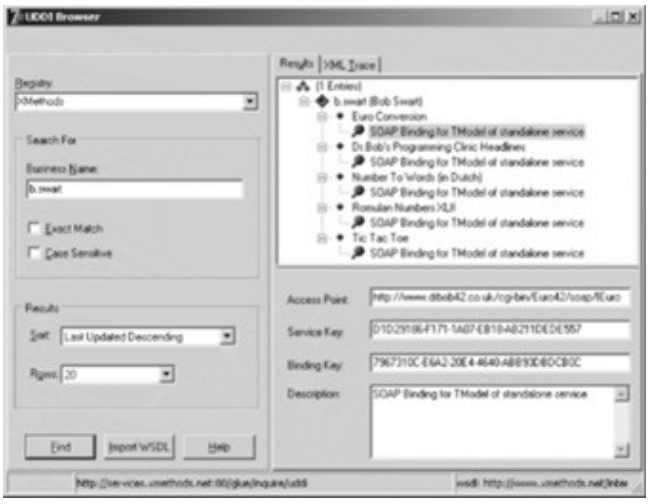
De developer's kit voor beta 2 bevat voorbeeldcode voor .NET en Java, een API reference en een GoogleSearch.wsdl bestand waarin de WSDL (Web Service Description Language) definitie van de Google Search web service in staat. Dit laatste is eigenlijk het enige wat we nodig hebben om aan de slag te kunnen. Later zal blijken dat ook de API reference erg handig is, om te verklaren wat de verschillende methoden en argumenten daad-werkelijk doen.

Google Search Key

Het aanmaken van een Google Account om de Google Search key aan te vragen is niet zo moeilijk. Je moet o.a. een e-mail adres opgeven waar je de key ontvangt. Het gebruik van de Google Web APIs is dan gratis, mits het beperkt blijft tot niet-commercieel gebruik, en tot maximaal 1000 zoekopdrachten per dag. Na die 1000 moet je wachten tot de volgende dag om weer resultaten te kunnen krijgen (ik ben er nog niet precies achter wanneer de volgende dag begint - dat zal wel in een amerikaanse tijdzone zijn). Voor het gemak van de lezer heb ik vast een key aangevraagd die we in dit artikel kunnen gebruiken.
Let er wel op dat als iedereen die ene key blijft gebruiken zal het aantal zoekopdrachten iedere dag natuurlijk snel op zijn, dus wie zelf serieus aan de slag wil met de Google Search web service kan het beste een eigen Google Search key aanvragen. De key voor ons is 1WpiIaxr+k+hbyYbRLZOJfg7X9NgI837 en is ook opgenomen in de source code behorende bij dit artikel, die tevens te downloaden is van mijn website te http://www.eBob42.com/SOAP.

Google Search

Met behulp van het GoogleSearch.wsdl bestand en de Google Search key kunnen we dan aan de slag. De professional versie van Delphi 6.02 of later is voldoende (ik gebruik Delphi 7 Enterprise). Start eerst een nieuw project en bewaar het formulier als MainForm.pas en het project zelf als Google42.dpr. We gaan straks het form opmaken, maar eerst gaan we de GoogleSearch.wsdl gebruiken om een Delphi import unit te genereren, zodat we kunnen zien hoe de Google Search APIs intern zijn opgebouw en hoe we daarmee eigenlijk kunnen zoeken met en in Google.
Doe *File | New - Other*, ga naar de WebServices tab van de Object Repository en kies de WSDL Importer wizard. Normaal gesproken kun je hier de vindplaats URL invoeren van de WSDL, maar je kan hier ook een lokaal bestand opgeven, zoals de GoogleSearch.wsdl uit de Google Search developer's kit, die we nu kunnen gebruiken
Nieuw in Delphi 7 is hier overigens de Search UDDI knop, die ons de mogelijkheid biedt om via de UDDI (Universal Description, Discovery and Integration) op zoek te gaan naar beschikbare web services. UDDI kan gezien worden als de gouden gids van de web services. Een onderwerp voor een artikel op zich en iets wat we op dit moment nog niet nodig hebben, omdat we al hebben wat we zoeken: de WSDL van de Google Search APIs.



Klik op de Next knop zodat we op de volgende (en laatste) pagina van de WSDL Import Wizard komen. Hier zien we links een boomstructuur met daarin de verschillende types en interfaces die gedefinieerd zijn in de Google Search WSDL. De types worden op hun beurt weer opgesplitst in velden, terwijl we bij de interfaces (er is er maar eentje: GoogleSearchPort) de individuele methoden te zien krijgen. Rechts zien we dan meteen de code die gegenereerd zal worden als we op de Finish knop drukken.

Er zijn vijf types: DirectoryCategory, ResultElement, ResultElementArray, DirectoryCategoryArray en GoogleSearchResult. De twee "Array" types zijn uiteraard een array van het type zelf (dus zuiver gesproken zijn er maar drie verschillende types en twee *array of* die types). Ik ga me niet bezighouden met de DirectoryCategory, dus voor ons zijn er eigenlijk maar twee types interessant op dit moment: GoogleSearchResult en ResultElement.
Het interface GoogleSearchPort bestaat uit de methoden: doGetCachedPage, doSpellingSuggestion en doGoogleSearch. De eerste twee zijn niet zinvol voor ons voorbeeld, maar het ziet er naaruit dat doGoogleSearch precies doet wat we willen.

GetGoogleSearchPort

Tot slot hebben we nog een hele nuttig functie genaamd GetGoogleSearchPort, die een interface van type GoogleSearchPort terug geeft. Hierdoor hoeven we niet zelf meer met HTTPRIO component te werken, maar kunnen we volstaan met het aanroepen van GetGoogleSearchPort zonder (default) argumenten. De implementatie van GetGoogleSearchPort bevat alle code die we "vroeger" (lees: met Delphi 6 en Delphi 6.01) nog "met de hand" moesten doen, maar die nu met Delphi 6.02 of Delphi 7 geheel automatisch gaat. Een kind kan de was doen (met SOAP)!
GetGoogleSearchPort levert het GoogleSearchPort interface op, de aanroep van de methode doGoogleSearch levert een GoogleSearchResult op, die op zijn beurt een resultElements array bevat met daarin de individuele resultaat elementen. Dan is de enige vraag nog: hoe vullen we de niet minder dan tien (10!) argumenten van doGoogleSearch met een zinvolle waarde. En op dat punt greep ik even terug naar de Google Search API developers kit die een APIs_Reference.html (van 100,417 bytes) bevat met daarin de benodigde informatie.

doGoogleSearch

Zoals gezegd, de doGoogleSearch method van het GoogleSearchPort interface heeft een hele hoop argumenten en is als volgt gedefinieerd:

```
function  doGoogleSearch(
  const key: String; // de Google Search Key
  const q: String; // de query string
  const start: Integer; // start URLs
  const maxResults: Integer;
  const filter: Boolean; // filter?
  const restrict: String; // beperkingen
  const safeSearch: Boolean; // "adult" filter?
  const lr: String; // taal
  const ie: String; // encodering
  const oe: String):
    GoogleSearchResult; stdcall;
```

Het key argument moet uiteraard onze key 1WpiIaxr+k+hbyYbRLZOJfg7X9NgI837 zijn. Het q argument is de zoekopdracht die we willen geven. Het start argument geeft aan vanaf welk punt we de resultaten willen zien (tellend vanaf 0) dus voor de volgende 10 resultaten moet je start op 10 zetten. MaxResults geeft aan hoeveel resultaten je wil zien (met een maximum van 10, dus die laat ik altijd op 10 staan). Filter geeft aan of we vergelijkbare resultaten eruit willen filteren (vaak zijn dat resultaten van dezelfde website). Restrict kan worden gebruikt om aan te geven dat we binnen Google naar een bepaald subonderwerp aan het zoeken zijn (dat gebruik ik zelf nooit). Het SafeSearch argument kan gebruikt worden om te zorgen dat we geen "adult" resultaten krijgen, zodat ook mijn kinderen zonder gevaar naar een website over poesjes kunnen zoeken. Het lr argument geeft de taal aan die we willen gebruiken zoals nederlands of engels, en de ie en oe argumenten tenslotte geven de input en output codering aan, die ik altijd op latin1 heb staan. Samengevat is mijn aanroep naar doGoogleSearch (met een Query string waarin de zoekwoorden staan) als volgt:

```
SearchResult :=
  doGoogleSearch(
   '1WpiIaxr+k+hbyYbRLZOJfg7X9NgI837',
     Query, 0, 10, True, '', True,
    'lang_en', 'latin1', 'latin1');
```

en dit levert iets op van type GoogleSearchResult, wat we dan alleen nog hoeven uit te pakken.

GoogleSearchResult

GoogleSearchResult heeft een aantal handige properties, zoals de estimatedTotalResultsCount (kennelijk krijgen we alleen maar een benadering van het aantal gevonden resultaten) searchTime en resultElements. Deze laatste is een array waarbij de elementen van type ResultElement zijn. De individuele ResultElementen hebben ook enkele properties die het vermelden waard zijn, zoals title, URL en cachedSize. Omdat we er iedere keer maximaal 10 krijgen, is het eenvoudig om een StringGrid te gebruiken en daarmee de resultaten netjes onder elkaar te krijgen.

```
procedure TForm1.SearchClick(Sender: TObject);
var
  Results: GoogleSearchResult;
  i: Integer;
begin
  Results :=
    GetGoogleSearchPort.doGoogleSearch(
    '1WpiIaxr+k+hbyYbRLZOJfg7X9NgI837',
    edtQuery.Text, 0, 10, True, '',
    True, 'lang_en', 'latin1', 'latin1');
  Caption :=
    Format('%d results in %.2n seconds',
    [Results.estimatedTotalResultsCount,
     Results.searchTime]);
  for i:=Low(Results.resultElements)
    to High(Results.resultElements) do
  begin
    StringGrid1.Cells[0,Succ(i)] :=
      IntToStr(Succ(i));
    StringGrid1.Cells[1,Succ(i)] :=
      Results.resultElements[i].title;
    StringGrid1.Cells[2,Succ(i)] :=
      Results.resultElements[i].URL;
    StringGrid1.Cells[3,Succ(i)] :=
      Results.resultElements[i].cachedSize
  end
end;
```

Als toetje moeten we alleen nog de OnDblClick event handler van de StringGrid invullen, zodat we direct naar de betreffende URL springen als we er op dubbelklikken. Dat gaat als volgt:

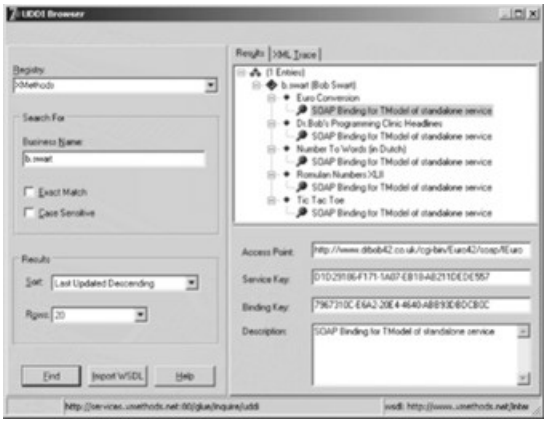
```
procedure TForm1.SGDblClick(Sender: TObject);
begin
  with (Sender as TStringGrid) do
    ShellExecute(Handle, 'open',
    PChar(Cells[2,Row]),nil,nil,SW_NORMAL)
end;
```

Conclusie

In dit artikel heb ik hopelijk laten zien dat het gebruiken van web services in Delphi 6 en hoger een eenvoudige zaak is geworden. Na het importeren van de WSDL is het resultaat een import unit, waar een functie in zit die als resultaat een interface oplevert naar de (remote) web service, waar we vervolgens de methoden van kunnen aanroepen. Haast net zo makkelijk als het neerzetten van een component op een form en zonder nog langer te hoeven weten wat er achter de schermen gebeurt (dit is natuurlijk voor de liefhebber allemaal nog in detail na te lezen in de gegenereerde import unit).
Wie meer wil weten over SOAP en Web Services met Delphi, Kylix, C++Builder of JBuilder (of C#) nodig ik uit om regelmatig het SOAP Bubbles deel van mijn site te http://www.drbob42.com/SOAP te bezoeken. Daar zal ook de source code voor Google42 te downloaden zijn (met de eventuele uitbreidingen en aanpassingen die ik sinds het schrijven van dit artikel heb aangebracht).

Delphi 7 UDDI Browser

Delphi 7 Enterprise heeft nu een UDDI Browser waarmee we op zoek kunnen gaan naar Web Services, mits je tenminste weet waar je moet zoeken en naar wat je moet zoeken. Beide zaken kunnen voor problemen zorgen, wat tot ongewenst (lees: geen) resultaat kan leiden.
Gelukkig zijn er al een aantal mogelijke UDDI registries (de vindplaatsen) gedefinieerd in Delphi 7, en deze lijst kunnen we zelf uitbreiden door het UDDIBrow.ini bestand uit te breiden met nieuwe UDDI registries (wie er nog een paar weet: ik hou me van harte aanbevolen voor extra URLs). De lijst van Registries uit UDDIBrow.ini is terug te vinden in de Registry drop-down combobox. Je kan hier o.a. voor XMethods kiezen. Vul vervolgens in de Business Name editbox de naam van de eigenaar van de Web Service in, zoals xmethods (voor alle web services van XMethods zelf) of b.swart voor de vijf web services van mij die bij XMethods te vinden zijn.



Delphi COM in .NET

Wie de lancering van Delphi 7 op 11 september 2002 ook heeft bijgewoond heeft waarschijnlijk al mijn artikel over het importeren van .NET assemblies als COM objecten hebben ontvangen. Dat was erg leuk om te doen, maar in dit stukje gaan we de andere kant op: bestaande COM objecten gebruiken met de Delphi for .NET preview command-line compiler.

Delphi 7 COM Object

Als we nog geen COM Object hebben om voor dit voorbeeld te gebruiken, kunnen we er met Delphi zelf eentje maken. Start Delphi en doe *File | New - Other*, ga nu naar de ActiveX tab van de Object Repository, kies voor het ActiveX Library icon en klik op OK. Bewaar het project in D7Win32COM.dpr Doe nu weer *File | New - Other*, ga weer naar de ActiveX tab, maar kies nu voor een Com Object. In de COM Object Wizard vul je "eBob42" in als Class Name en klik je vervolgens weer op OK.
Vervolgens komen we in de Type Library Editor terecht. Klik hier op het IeBob42 interface en vervolgens op de groene pijl om een nieuwe method toe te voegen. Geef deze nieuwe method de naam Unmanaged (om te illustreren dat het hier om een unmanaged methode gaat die in "gewoon" Delphi is geïmplementeerd).
Ga naar de Parameters tab van de Type Library Editor, maak het result type leeg (dat maakt er een procedure van) en geef de methode Unmanaged een argument genaamd Message, van type BSTR en [in]. Klik nu op de Refresh Implementation button en schrijf in de code editor de volgende code voor de Unmanaged methode:

```
unit eBob42;
{$WARN SYMBOL_PLATFORM OFF}
interface
uses
  Windows, ActiveX, Classes, ComObj,
  D7Win32COM_TLB, StdVcl;

type
  TeBob42 = class(TTypedComObject, IeBob42)
  protected
    procedure Unmanaged(
      const Message: WideString); stdcall;
  end;

implementation
uses
  ComServ, Dialogs;

procedure TeBob42.Unmanaged(
  const Message: WideString);
begin
  ShowMessage(Message);
end;

initialization
  TTypedComObjectFactory.Create(ComServer,
    TeBob42, Class_eBob42,
    ciMultiInstance, tmApartment);
end.
```

Bewaar deze unit nu in eBob42.pas en compileer het D7Win32COM project. Als dit lukt, kunnen we het COM Object registreren vanuit de Delphi IDE met *Run | Register ActiveX Server*, of vanaf de command-line met regsvr32 of TRegSvr.

Delphi for .NET

Nu we eenmaal een D7Win32COM.dll hebben met daarin een IeBob42 interface en een TeBob42 class, kunnen we een poging wagen om deze vanuit een .NET managed omgeving te gebruiken. De Delphi for .NET preview command-line compiler dus.
Allereerst moeten we de Win32 DLL importeren met tlbimp (de Microsoft .NET Framework Type Library to Assembly Converter). Omdat ik echter de resulterende assembly straks in de GAC (Global Assembly Cache) wil zetten moet ik eerst een strong key aanmaken met behulp van sn (de Microsoft .NET Framework Strong Name Utility):

```
sn -k eBob42.snk
```

Hierna kunnen we tlbimp aanroepen op onze unsafe D7Win32COM.dll en daarbij aangeven dat we de keyfile eBob42.snk gebruiken voor de output file eBob42.dll. Let op dat de volgende aanroep op één lange regel moet gebeuren:

```
tlbimp D7Win32COM.dll /keyfile:eBob42.snk
    /out:eBob42.dll
```

De eBob42.dll is nu een .NET assembly die gebruik maakt van de D7Win32COM.dll. Omdat we een Strong Key gebruikt hebben met eBob42.snk, kunnen we de assembly in de Global Assembly Cache (GAC) plaatsen. Dat kan in twee stappen. Eerst registreren met regasm en daarna in de GAC stoppen met gacutil:

```
regasm eBob42.dll
gacutil -i eBob42.dll
```

Het voordeel van de Global Assembly Cache (GAC) is dat de eBob42.dll nu niet langer in dezelfde directory hoeft te zitten als de .NET toepassing die we nu gaan maken.
De Delphi for .NET preview command-line compiler kan gebruik maken van assemblies door gebruik te maken van de -LU compiler optie. Dat zou dus als volgt gaan:

```
dccil -LUeBob42
```

Helaas klaagt dccil dan dat de eBob42 package niet gevonden kan worden. Een manier op dit op te lossen (het is een preview command-line compiler!) houdt in de eBob42.dll assembly te kopiëren naar de C:\WinNT\Microsoft.NET\Framework\v1.0.3705\ directory. Nu kan de -LU vlag hem wel vinden.
Om gebruik te maken van het COM Object zullen we nog een stukje code moeten schrijven. Het kortst mogelijke gebruik is als volgt:

```
program DotNet;
uses
  eBob42;

var
  Q: eBob42.eBob42Class;

begin
  Q := eBob42.eBob42Class.Create;
  try
    Q.Unmanaged('Hello from Delphi for .NET')
  finally
    Q.Free
  end
end.
```

Merk op dat ik de namespace eBob42 gebruik om aan te geven dat we informaties (types van classes en interfaces) uit de eBob42 assembly gebruiken. Merk tevens op dat er plotseling een class type eBob42Class staat en niet de TeBob42 die ik aan de Delphi 7 kant had aangemaakt.
Daarnaast is het interface type eBob42.IeBob42 ook bekend, alhoewel dat niet compatible is met het eBob42.eBob42Class type. De class type TeBob42 krijg ik niet meer te zien (maar ik zal vast wel iets fout doen - daar kom ik later nog eens op terug).

Het compileren van de DotNet.dpr toepassing met de Delphi for .NET preview command-line compiler gaat nu als volgt:

```
dllil -LUeBob42 DotNet.dpr
```

En daarna kunnen we DotNet.exe uitvoeren, om hiermee vanuit een safe managed .NET toepassing (DotNet.exe) via een import assembly (eBob42.dll) naar een stukje unsafe en unmanaged code te gaan (D7Win32COM.dll).

Met alle risico's van dien, maar ook de voordelen zoals hergebruik van bestaande COM objecten (totdat er tijd en geld is om alles weer helemaal opnieuw in .NET op te bouwen).

Delphi for .NET Assemblies

Wat in de .NET wereld een assembly heet kan - een beetje oneerbiedig - nog het beste vergeleken worden met een DLL zoals we die kennen uit de "oude" Windows wereld. Alleen dan een stuk makkelijker zowel om te maken als om te gebruiken, zoals ik nu zal laten zien.

Assembly

Waar een DLL alleen platte functies kan exporteren, kunnen we in een .NET assembly ook types en classes opnemen, die daarna gebruikt kunnen worden. Als voorbeeld declareer ik hier de class LifeTheUniverseAndEverything in de library genaamd MyASM:

```
library MyASM;
type
  LifeTheUniverseAndEverything = class
    constructor Create;
    function TheAnswer: Integer;
  end;

constructor LifeTheUniverseAndEverything.Create;
begin
end;

function LifeTheUniverseAndEverything.
  TheAnswer: Integer;
begin
  Result := 42
end;

end.
```

Merk op dat ik wel het library keyword gebruik, maar helemaal niks hoef te exporteren. Ik hoef de MyASM.dpr alleen maar te compileren om als resultaat een MyASM.dll te krijgen. Het enige bijzondere wat ik we nog een keer handmatig moet doen, is het kopiëren van de MyASM.dll naar de C:\WinNT\Microsoft.NET\Framework\v1.0.3705\ directory, zodat dccil hem met de -LU vlag kan vinden.

Gebruik van de Assembly

Het gebruik van de Assembly MyASM gaat nu als volgt:

```
program UseASM;
{$APPTYPE CONSOLE}
uses
  MyASM;
var
  X: LifeTheUniverseAndEverything;
begin
  X := LifeTheUniverseAndEverything.Create;
  writeln('The Answer is: ', X.TheAnswer)
end.
```

Compileren gaat met dccil -LUMyASM UseASM.dpr

Bob Swart Training & Consultancy (www.eBob42.com) presenteert in Eindhoven:

Dr.Bob's Delphi 7 Clinics

12 september 2002

Delphi COM Development

Keywords: **Interfaces, COM, Automation, ActiveX, ActiveForms, ADO, MTS/COM+, Active Server Objects**

Tijdens deze Delphi Clinic zullen we kennismaken met de wereld van COM en ActiveX. Via interfaces, IUnknown, IDispatch en TInterfacedObject naar COM Objecten (COM Servers en Clients), Automation (Servers en drie soorten clients: Variants, Dispatch en Interfaces), ActiveX/ActiveForms en ADO (ActiveX Data Objects). Vervolgens gaan we stateless objecten voor Microsoft Transaction Server (MTS) en COM+ ontwikkelen en deployen, en web toepassingen bouwen met Active Server Pages en Objects.

26 september 2002

dbExpress & DataSnap (MIDAS)

Keywords: dbExpress, DataSnap, ClientDataSet, DataSetProvider, Connection components, Deployment

In deze Delphi Clinic behandelen we de cross-platform DataCLX en dbExpress data access laag en de multi-tier architectuur DataSnap (nieuwe naam van MIDAS). De ClientDataSet, DataSetProvider en xxxConnection componenten zullen hierbij uitgebreid aan de orde komen (met aandacht voor TCP/IP Sockets, DCOM, HTTP, SOAP en MTS/COM+ connections).

10 oktober 2002

WebBroker & InternetExpress

Keywords: WebBroker, InternetExpress, Debuggen, Deployment

In deze Delphi Clinic starten we met WebBroker als architectuur om CGI of ISAPI/NSAPI web server toepassingen te ontwikkelen, en breiden deze ervaring uit met InternetExpress. Alle WebBroker en InternetExpress componenten (en enkele custom componenten) zullen behandeld worden. Ook het debuggen van web server toepassingen (met de Web App Debugger) komt uitgebreid aan de orde.

24 oktober 2002

WebSnap Development

Keywords: WebSnap, Adapters, WebSnap Custom Adapters, Login/Access, Sessions, Deployment

In deze Delphi Clinic bekijken we WebSnap van alle kanten. De WebSnap Architectuur onderkent een centrale rol voor Adapters. We zullen zien hoe de Adapters werken en alle componenten van de WebSnap tab van het component palette aan de orde laten komen. Daarnaast besteden we aandacht aan login/logout, user access en sessiemanagement, en ontwikkelen een techniek die ook werkt voor CGI toepassingen (en Kylix). Tot slot bekijken we hoe WebSnap achter de schermen in elkaar zit en bouwen WebSnap custom componenten (zoals een TCreditCardAdapter).

7 november 2002

BizSnap: XML, SOAP & Web Services

Keywords: BizSnap, XML Documents, XML Data Binding, XML Mapper, WSDL, SOAP, Web Services, RIO, Clients, Interoperability

Deze Delphi Clinic begint met een overzicht van de XML ondersteuning in Delphi Enterprise, met XML Document Programming, de XML Data Binding Wizard en tenslotte de XML Mapper (een tool waarmee XML documenten naar data packets te transformeren zijn - en terug).
Daarna verschuift de aandacht naar SOAP en Web Services. Na een korte introduktie in zowel SOAP en WSDL als de toepassing van Web Services, zullen we zien hoe we in Delphi bestaande Web Services kunnen gebruiken en nieuwe cross-platform Web Services kunnen maken en deployen.

21 november 2002

Web Solutions: WebSnap & IntraWeb

Keywords: e-commerce solutions met WebSnap en IntraWeb

Tijdens deze Delphi Clinic bouwen we een "live" multi-tier e-business toepassing met behulp van technieken uit WebSnap en IntraWeb.
We zullen een e-commerce toepassing bouwen met alle toeters en bellen zoals login/logout, registratie nieuwe gebruikers, database access, zoeken, bestellen, winkelwagentje, e-mail support, etc.

5 december 2002

Delphi .NET Framework Development

Keywords: Delphi .NET Framework Development

Tijdens deze Delphi Clinic zullen we zien hoe we Delphi kunnen gebruiken voor .NET Framework Development met de Delphi for .NET preview command-line compiler (zie ook elders in dit blad voor meer informatie).

De lokatie van de Delphi Clinics is de cursusruimte bij Centric PSS in Eindhoven, waar de trainingen worden gegeven op donderdag van 9.00 tot 17.00 uur.
De prijs per clinic is **€420** per persoon (excl 19% btw). Bij twee deelnemers van hetzelfde bedrijf geldt een korting van **5%** - bij drie of meer deelnemers zelfs **10%**.

LET OP: Speciale Aanbiedingen!

* Bij inschrijving voor drie of meer trainingsdagen krijgt u een door mij meegeschreven en gesigneerd boek kado (keuze uit Delphi 6 Developer's Guide en Kylix Developer's Guide).
* Bij inschrijving voor zes trainingsdagen krijgt u de zevende dag kado (dus 7 dagen voor de prijs van zes).

Extra korting is mogelijk bij meer dan vier personen van hetzelfde bedrijf, evenals maatwerk training op lokatie.
Zie http://www.eBob42.com voor meer informatie.