Delphi OplossingsCourant

Vol. 5. No. 1. Een gratis kwartaalpublicatie van **Bob Swart Training & Consultancy** (**eBob42**) - http://www.eBob42.com

Helmond, 6 januari 2003,

Een paar dagen later dan gepland, maar hier is dan toch eindelijk het nieuwe nummer van de Delphi OplossingsCourant; de eerste van 2003. Dit nummer staat in het teken van de nieuwe ontwikkelingen die ons dit nieuwe jaar te wachten staan. Niet alleen Delphi, maar ook .NET zal voor mij een kernwoord zijn in 2003, en vandaar ook de nieuwe .NET versie van m'n website te http://www.DrBob42.net

Het afgelopen jaar hebben we met Delphi 7 en de Delphi for .NET preview command-line compiler al kunnen proeven welke ondersteuning Borland wil bieden aan Microsoft's .NET Framework. Een eerste update voor de preview command-line compiler is inmiddels ook beschikbaar, en ik verwacht er nog wel een paar voor we de uiteindelijke visuele IDE onder .NET zullen zien (voorlopig alleen nog maar bekend onder de codenaam "Galileo").

In dit nummer van de Delphi OplossingsCourant sta ik nog even stil bij de artikelen die ik voor de IBM DB2 en Borland portal heb geschreven over het gebruik van DB2 met Delphi. De meeste aandacht gaat deze keer echter uit naar de Delphi for .NET preview command-line compiler als scripting taal binnen ASP.NET web forms en web services.

Tot slot volgt nog het overzicht van de zes Delphi 7 Clinics die ik in de eerste helft van 2003 zal verzorgen in Eindhoven. Naast Delphi 7 Clinics ga ik dit jaar ook Delphi Workshops organiseren. Deze zullen - op aanvraag - voor groepjes vanaf twee deelnemers (in Helmond) of meer (op locatie) georganiseerd worden, waarbij iedereen zijn eigen computer en versie van Delphi kan gebruiken. Zie http://www.eBob42.com/workshop voor details.

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 het voorjaar van 2003 verschijnen. Abonnees krijgen automatisch bericht per e-mail

Delphi Clinics in 2003

In de 1e helft van 2003 zal er elke maand weer een Delphi 7 Clinic plaatsvinden in Eindhoven. De data en onderwerpen zijn als volgt:

30 jan - **dbExpress & DataSnap (MIDAS)**
27 feb - **Web Development met IntraWeb**
27 mar - **BizSnap: XML/SOAP & Web Services**
24 apr - **WebSnap e-Commerce Development**
29 mei - **Delphi en .NET Framework**
26 juni - **.NET (Delphi) and J2EE met SOAP**

Zie pagina 10 of http://www.eBob42.com/training voor meer informatie over de Delphi 7 Clinics, of http://www.eBob42.com/workshop voor workshops

Evenementen 2003

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

18 maart - **UK-BUG Meeting**, Londen (UK)
8-10 juni - **DCon 2003**, Londen (UK)
7 okt - **UK-BUG MasterClass**, Londen (UK)
1-5 nov - **BorCon 2003**, San Jose (USA)

Ook bij de **CttM** in mei 2003 hoop ik te spreken.

Delphi en IBM's DB2

Borland heeft een overeenkomst gesloten met IBM om de Borland RAD Tools samen met DB2 te gaan promoten (dat wil zeggen dat Borland RAD Tools een trial versie van DB2 gaan bevatten, en DB2 een trial versie van Delphi, C++Builder of Kylix). Als bijkomend fenomeen is er een speciale IBM DB2 en Borland web portal gestart, waarbij ik eind vorig jaar gevraagd ben om een zestal artikelen te schrijven over het gebruik van IBM's DB2 in Delphi en Kylix (met dbExpress, WebSnap, DataSnap en zelfs BizSnap).
De onderstaande artikelen zijn tussen november en december 2002 op de portal van IBM gepubliceerd:

- Connect Delphi 7 to DB2 with dbExpress
In dit eerste artikel laat ik zien hoe we vanuit Delphi met de dbExpress componenten kunnen praten met IBM's DB2 database tables.

- Display and Modify DB2 master-detail data
In dit artikel laat ik zien hoe we met dbExpress een master-detail relatie met DB2 tables kunnen maken en bewerken.

- Moving from Windows to Linux with Kylix 3
In dit artikel laat ik zien hoe we de Delphi 7 toepassing (onder Windows) kunnen migreren naar Linux en Kylix 3, waarbij we gebruik maken van DB2 onder Linux uiteraard.

- Build DB2 web applications with D7 WebSnap
In dit artikel laat ik zien hoe we de Windows toepassing kunnen nabouwen als web toepassing in Delphi 7 met WebSnap (en dbExpress nog steeds als data-access laag).

- Create Distributed DB2 apps with DataSnap
In dit artikel splits ik de Windows toepassing in twee verschillende "tiers" door gebruik te maken van Delphi 7 en DataSnap. Het server deel maakt nog steeds gebruik van dbExpress om met DB2 te praten, het client deel is nu database onafhankelijk.

- DB2 and Delphi: SOAP & database Web Services
In dit artikel herschrijf ik de multi-tier oplossing van het vorige artikel tot een cross-platform (met Delphi 7 onder Windows en Kylix 3 onder Linux) toepassing door gebruik te maken van SOAP als communicatie protocol en een "Web Service" als database server.

Zie http://www.drbob42.com/examines voor meer informatie en directe links naar de artikelen, en zie www7b.software.ibm.com/dmdd/zones/borland/ voor de web portal van IBM DB2 en Borland.

Delphi & dbExpress fixes

Behalve voor IBM's DB2, kun je met dbExpress nog andere databases benaderen zoals MySQL, InterBase, Informix en zelfs MS SQL Server. Toch loopt niet alles altijd even goed, want ik ben al verschillende probleempjes tegengekomen in de praktijk (waar some wel mee te leven is - mits je op de hoogte bent van de oorzaak en workaround).

dbExpress voor MS SQL Server

De dbExpress driver voor MS SQL Server die bij Delphi 7 zat had de vervelende eigenschap dat het verplicht werd om een username/password mee te geven, zelfs als de SQL Server database er geen een nodig had. Inmiddels is met Delphi 7 Update 1 een nieuwe versie van deze driver beschikbaar.

dbExpress voor DB2

De dbExpress drivers zijn beschikbaar in twee edities: de externe die de LibraryName als DLL gebruiken, en de interne, die de LibraryName als unit meelinken met de executable. Voor DB2 is de LibraryName dbExpDB2.dll en de unit naam dus dbExpDB2. Helaas zijn er problemen met quotes rondom veldnamen als je de dbExpDB2 unit gebruikt (iedere veldnaam en tabelnaam wordt nu plots voorafgegaan door een dubbele quote, maar de laatste letter van de naam vervalt, waardoor er een onzin query wordt opgeleverd). Gebruik van de externe dbExpDB2.dll gaat wel goed, maar de unit dbExpDB2 is dus krot en onbruikbaar!

dbExpress voor MySQL

Wie de dbExpress driver voor MySQL: gebruikt zal als default LibraryName van Delphi de waarde dbexpmys.dll krijgen, die geldig is voor MySQL tot versie 3.23.49. Vanaf die versie moeten we dbExpMySQL.dll gebruiken als LibraryName (de meeste mensen gebruiken 3.23.52 of hoger). Een van de symptomen (van de oude dll) is het vertonen van slechts de eerste 10 velden van een tabel of query.
Een artikel over Delphi, dbExpress en MySQL is in december 2002 gepubliceerd in nummer 88 van The Delphi Magazine, en inmiddels beschikbaar als "free sample" artikel op de website van The Delphi Magazine te http://ww.TheDelphiMagazine.com/Samples/1540/1540.htm

Meer Informatie...

Voor meer informatie over Delphi en dbExpress verwijs ik graag naar de Delphi 7 Clinic over dbExpress en DataSnap op donderdag 30 januari in Eindhoven. Maar het is ook mogelijk om een Delphi workshop over dit onderwerp "te bestellen", zie http://www.eBob42.com/workshop voor details.

ASP.NET Web Forms

In het vorige nummer van de Delphi Oplossings-Courant schreef ik al iets over de Delphi for .NET preview command-line compiler, en deze keer wil ik twee artikelen schrijven over de samenwerking met ASP.NET voor Web Forms en Web Services. In dit eerste artikel wil ik de Delphi for .NET preview command-line compiler gebruiken als scripting taal achter ASP.NET Web Forms.

Configuratie

Er zijn een aantal zaken die we moeten instellen alvorens Delphi for .NET als scripting taal gebruikt kan worden binnen ASP.NET. Op mijn ontwikkel-machines draai ik Windows 2000 Professional SP3, met de .NET Framework SP2. Ik gebruik IIS (Internet Information Server) als web server, en gebruik de C:\Inetpub directory met een (virtual) directory Scripts om mijn ASP.NET web forms (met .aspx) en straks ook web services (met .asmx) neer te zetten. In de virtual directory heb ik een web.config file met de volgende inhoud:

```
<configuration>
<system.web>
 <compilation debug="true">
  <assemblies>
   <add assembly="DelphiProvider"/>
  </assemblies>
  <compilers>
   <compiler language="Delphi" extension=".pas"
    type="Borland.Delphi.DelphiCodeProvider,
    DelphiProvider"/>
  </compilers>
 </compilation>
</system.web>
</configuration>
```

Elke ASP.NET pagina (met .aspx of asmx extentie) in de Scripts directory zal nu in de web.config kijken en zien dat Delphi een geldige taal is, met extensie .pas, en een Borland.Delphi.DelphiCodeProvider in DelphiProvider. Deze moeten we nog wel even toevoegen. Maar daartoe een bin sub-directory in de Scripts virtual directory aan, en kopieer daar de DelphiProvider.dll naartoe. Dat werkt op al mijn .NET machines, met uitzondering van eentje, waar ik in de root van mijn web server (dus niet in de Scripts virtual directory) de bin directory moest maken met de DelphiProvider.dll erin. Het is me nog niet duidelijk wát deze ene machine anders is of doet dan de andere .NET machines die ik heb.

Als derde en laatste stap moet je de drie files uit Delphi for .NET Preview\aspx\framework kopiëren naar WinNT\Microsoft.NET\Framework\v1.0.3705 (dat zijn dccasp.exe, dccil.cfg en rlink32.dll - deze drie files zijn nodig om de door DelphiProvider.dll gegenereerde .pas code te compileren).

Delphi en .NET Updates

Er is zowel voor het .NET Framework als voor de Delphi for .NET preview command-line compiler een update uitgekomen. Ik gebruik bijvoorbeeld .NET Framework met SP2. De Delphi for .NET preview command-line compiler is ingesteld op de default (eerste) versie van het .NET Framework. Dat wil zeggen dat de meegeleverde .dcuil files alleen onder deze versie van het .NET Framework geschikt zijn. Voor latere versies van het .NET Framework, zoals SP2, moeten we expliciet de nieuwe .dcuil files genereren door de rebuild.bat uit de Delphi for .NET Preview\source directory uit te voeren.

Celsius naar Fahrenheit

Als voorbeeld web form wil ik vandaag (nog) niet met de euro werken, maar een andere converter bouwen, namelijk van Celsius naar Fahrenheit. Op een normaal Delphi form kan ik met twee TEdits (genaamd edtCelsius en edtFahrenheit) en een tweetal TButtons (genaamd CelsiusToFahrenheit en FahrenheitToCelsius) de conversie in beide OnClick event handlers als volgt implementeren:

```
procedure TForm.CelsiusToFahrenheit
  (Sender: TObject);
begin
  edtFahrenheit.Text := FloatToStr(9 / 5 *
    StrToFloat(edtCelsius.Text) + 32)
end;

procedure TForm.FahrenheitToCelsius
  (Sender: TObject);
begin
  edtCelsius.Text := FloatToStr(5 / 9 *
    (StrToFloat(edtFahrenheit.Text) - 32))
end;
```

Dat zal weinig wereldschokkend zijn, vermoed ik. Het leuke is echter dat we straks precies dezelfde Delphi code in een ASP.NET pagina zullen gebruiken. Maar dan zonder Delphi form. En toch de twee editboxen en buttons.

Delphi en ASP.NET

Een web pagina gemaakt met ASP.NET eindigt op de .aspx extentie en kan zowel scripting code als HTML bevatten. In het voorbeeld gebruik ik twee edit controls (in ASP.NET ook wel textbox genoemd) en twee buttons met elk een OnClick event handler:

```
<form runat="server">
<asp:textbox id="edtCelsius" runat="server"/>
<asp:textbox id="edtFahrenheit"runat="server"/>
<p>
<asp:button text="Celsius to Fahrenheit"
OnClick="CelsiusToFahrenheit" runat="server"/>
<asp:button text="Fahrenheit to Celsius"
OnClick="FahrenheitToCelsius" runat="server"/>
</form>
```

Het resultaat van deze HTML code is twee editboxes en twee buttons (om het mooier te maken kun je nog wat HTML labels plaatsen zodat je in ieder geval weet welke editbox voor Celsius is en welke voor Fahrenheit). De controls zijn speciale ASP controls wat je kunt zien aan de asp: prefix.
De pagina is nog niet compleet overigens, want de beide event handlers (de CelsiusToFahrenheit en de FahrenheitToCelsius) moeten ook nog geschreven worden. En dat doe ik in Delphi, als volgt:

```
<%@Import Namespace="Borland.Delphi.SysUtils"%>

<script language="Delphi" runat="server">
  procedure CelsiusToFahrenheitClick(
    Sender: System.Object; E: EventArgs);
  begin
    edtFahrenheit.Text := FloatToStr(9 / 5 *
      StrToFloat(edtCelsius.Text) + 32)
  end;

  procedure FahrenheittoCelsiusClick(
    Sender: System.Object; E: EventArgs);
  begin
    edtCelsius.Text := FloatToStr(5 / 9 *
      (StrToFloat(edtFahrenheit.Text) - 32))
  end;

</script>
```

Deze code lijkt heel erg op wat we in de linker kolom zagen. Het verschil is dat de event handlers onder .NET een extra argument E van type EventArgs hebben. Verder staat er een speciaal import regel voor het begin van het script om de SysUtils unit - voluit Borland.Delphi.SysUtils - te importeren aangezien we daar de FloatToStr en StrToFloat uit gebruiken.
Wat toch wel het meest tot de verbeelding moet spreken is het feit dat de beide event handlers gewoon gebruik kunnen maken van edtCelsius en edtFahrenheit alsof het normale edit controls zijn. De twee asp:textbox declaraties hebben daar voor gezorgd, en zijn door de DelphiProvider.dll vertaald naar variabele declaraties voor de twee ASP.NET controls die dus gewoon gebruikt kunnen worden.

Komt Allen Tezamen...

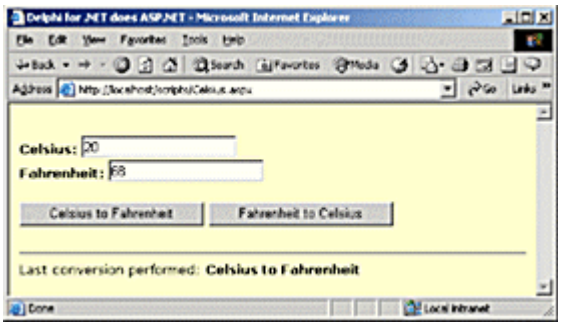
Voor ik de HTML code uit de linker kolom en de Delphi code hierboven samenvoeg tot een werkend voorbeeld, wil ik eerst nog even benadrukken dat we met de Delphi for .NET preview command-line compiler al gebruik kunnen maken van alle basis zaken in het .NET Framework.
En waar een Delphi ontwikkelaar haast zonder na te denken de StrToFloat en FloatToStr functies zal gebruiken, daar zal een C# .NET ontwikkelaar de Convert class uit de System assembly gebruiken om Convert.ToDouble en Convert.ToString te gebruiken om strings naar doules (en terug) te converteren.
Aangezien de System assembly ook beschikbaar is voor Delphi for .NET toepassingen, zouden we dus ervoor kunnen kiezen om de twee FloatToStr en StrToFloat aanroepen te vervangen door de reeds beschikbare Convert.ToString en Convert.ToDouble methods. Scheelt weer een import.
Behalve deze kleine aanpassing in de Delphi event handlers, heb ik ook een extra control toegevoegd van type asp:label genaamd Message, en die gebruik ik om aan het eind van iedere event handler even aan te geven wat de laatst gekozen conversie is geweest (kan handig zijn bij het testen van dit web form). De uiteindelijke code is als volgt:

```
<html>
<head>
<title>Delphi for .NET does ASP.NET</title>

<script language="Delphi" runat="server">

 procedure CelsiusToFahrenheitClick
   (Sender: System.Object; E: EventArgs);
 begin
  edtFahrenheit.Text := Convert.ToString(9/5 *
    Convert.ToDouble(edtCelsius.Text)+32);
  Message.Text := 'Celsius to Fahrenheit'
 end;

 procedure FahrenheittoCelsiusClick
   (Sender: System.Object; E: EventArgs);
 begin
  edtCelsius.Text := Convert.ToString(5/9 *
   (Convert.ToDouble(edtFahrenheit.Text)-32));
  Message.Text := 'Fahrenheit to Celsius'
 end;

</script>
</head>

<body bgcolor="ffffcc">
<font face="verdana" size="2">
<form runat="server">
 <br><b>Celsius:</b>
     <asp:textbox id="edtCelsius"
      runat="server"/>
 <br><b>Fahrenheit:</b>
    <asp:textbox id="edtFahrenheit"
     runat="server"/>
 <p>
   <asp:button text="Celsius to Fahrenheit"
    OnClick="CelsiusToFahrenheitClick"
    runat="server"/>
   <asp:button text="Fahrenheit to Celsius"
    OnClick="FahrenheittoCelsiusClick"
    runat="server"/>
</form>
<hr>
Last conversion performed: <b>
<asp:label id="Message" runat="server"/></b>
</body>
</html>
```

Bovenstaande inhoud staat in file Celsius.aspx en heb ik in de Scripts virtual directory geplaatst op mijn web server. Uiteraard moeten daar het .NET Framework alsmede de Delphi for .NET preview command-line compiler op geconfigureerd zijn.

Run!

De virtual directory die ik gebruikt heb heet Scripts, dus kan ik URL http://localhost/Scripts/Celsius.aspx gebruiken om de web form te vertonen. Na het invullen van 20 in the Celsius editbox en een klik op de "Celsius to Fahrenheit" knop krijg ik het volgende resultaat (68 graden Fahrenheit):



Merk op dat het de eerste keer enige tijd (een paar seconden) kan duren voor de Celsius.aspx pagina antwoord geeft. Dat komt omdat - achter de schermen - de DelphiProvider.dll de benodigde Delphi source code genereert, die vervolgens door dccasp.exe wordt gecompileerd tot een native .NET assembly die dan door ASP.NET wordt gebruikt om het antwoord te geven. De tweede keer zal het veel sneller gaan, omdat de source code dan niet meer gegenereerd en gecompileerd hoeft te worden (de assembly wordt gecached). Dit gaat net zolang tot je de Celsius.aspx pagina wijzigt, waarna het code genereren en compileren weer opnieuw moet plaatsvinden.

Temporary ASP.NET Files

Wie interesse heeft om de gegenereerde Delphi source code voor de Celsius.aspx pagina te bekijken kan dat doen door <%@ Page Debug="true" %> op te nemen in de Celsius.aspx file, of door een regel met <compilation debug="true"> in de web.config file op te nemen. Dat laatste is al het geval - in ieder geval in mijn configuratie.

Na deze eventuele aanpassing, kun je in de C:\WinNT\Microsoft.NET\Framework\v1.0.3705\ Temporary ASP.NET Files directory kijken en moet je letten op de .pas files. Je kan ze lezen, maar uiteraard heeft wijzigen geen zin, want de resulterende assemblies (als de compilatie is gelukt) zullen hergebruikt worden, en niet de (temporary) ASP.NET files. Maar het is wel leerzaam om te zien wat het resultaat van dit voorbeeld is.

Celsius.aspx

Wie dit voorbeeld wil uitproberen, of de Celsius naar Fahrenheit omzetter "live" wil zien kan naar http://www.eBob42.com/cgi-bin/Celsius.aspx gaan.

ASP.NET Web Services

Na de ASP.NET web forms is het dan nu tijd om een ASP.NET web service te bouwen (en te importeren) met de Delphi for .NET preview command-line compiler.

.NET Web Services

Om een web service voor .NET te bouwen moeten we de System.Web.Services assembly gebruiken, die al zo'n beetje alles bevat wat we nodig hebben. Voor dit voorbeeld keer ik terug naar de Euro conversies, en ik leidt onze nieuwe class Euro42 af van de WebService class. Vervolgens moet ik iedere methode die ik toevoeg en aan de buitenwereld als "web service methode" bekend wil maken laten voorafgaan door het WebMethod attribute. Het gebruik van dit attribuut is een taaluitbreiding in Delphi for .NET (ten opzichte van Delphi voor Windows), maar is ook aanwezig in C#. Het maakt de bouw van web services uitermate eenvoudig, en het interface deel van onze Euro42 class kan er bijvoorbeeld als volgt uit komen te zien:

```
<%@ WebService Language="Delphi"
    Class="eBob42.Euro42" %>

unit eBob42;
interface
uses
  System.Web.Services;

type
  Euro42 = class(WebService)
  public
    [WebMethod]
    function About: String;
    [WebMethod]
    function GuldenToEuro
      (Gulden: double): double;
    [WebMethod]
    function EuroToGulden
     (Euro: double): double;
   end;
```

De implementatie is niet veel moeilijker eigenlijk, en bevat de volgende code:

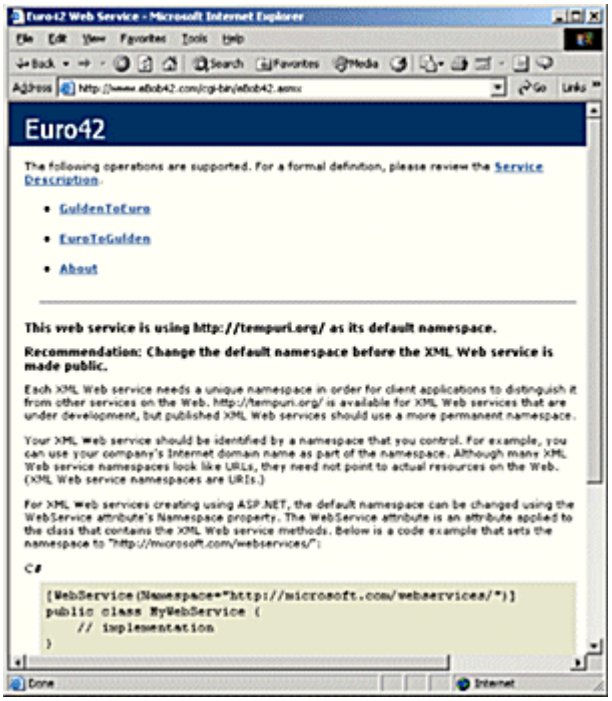
```
implementation

function Euro42.About: String;
begin
  Result := 'Dr.Bob''s Euro42 Web Service' +
   ' written in Delphi for .NET (preview)'
end;

const
  GuldenPerEuro = 2.20371;

function Euro42.GuldenToEuro
  (Gulden: double): double;
begin
  Result := Gulden / GuldenPerEuro
end;
```

```
function Euro42.EuroToGulden
  (Euro: double): double;
begin
  Result := Euro * GuldenPerEuro
end;

end.
```

Om deze Euro42 web service daadwerkelijk te kunnen gebruiken hoeven we de eBob42.asmx file (waar de code in staat) alleen nog maar in de Scripts virtual directory neer te zetten waar we eerder in dit nummer ook al de file Celsius.asmpx hadden geplaatst.

Delphi for .NET Web Services

Teneinde de Delphi for .NET web service te testen moeten we de URL in een browser intikken, die in ons geval dus http://localhost/Scripts/eBob42.asmx is. Merk op dat we dan meteen een waarschuwing krijgen dat we een zgn. *default namespace* gebruiken:

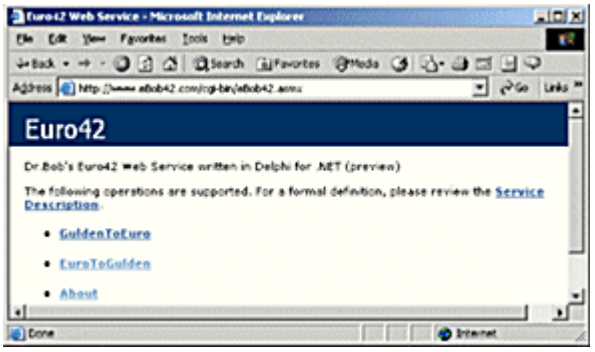


Er worden suggesties gedaan voor C# en VB.NET om een namespace toe te voegen aan de web service. Wij gebruiken echter geen C# of VB.NET maar de Delphi for .NET preview command-line compiler. Gelukkig verschilt de code voor Delphi for .NET niet veel van die van C#, we moeten namelijk een attribute genaamd WebService toevoegen aan de class definitie, met daarin een property genaamd Namespace die een waarde krijgt. Let op dat de naam Namespace case-sensitive is, dus NameSpace is fout (en wordt niet herkend). Been there, done that, etc.
Behalve de Namespace property kun je ook een eigen Description toevoegen aan het WebService attribute.

Het gevolg is dat de Euro42 class een extra lange attribute regel krijgt voor de declaratie zelf start:

```
type
[WebService(Namespace='http://www.eBob42.com',
  Description='Euro42 Web Service ' +
 'written in Delphi for .NET (preview)')]
  Euro42 = class(WebService)
  public
    [WebMethod]
      ...
  end;
```

Als we de gewijzigde eBob42.asmx web service nu in de Scripts directory neerzetten en de URL in de browser verversen (met Shift+Refresh), dan is de default namespace warning verdwenen, en in plaats daarvan krijgen we de description alsmede de drie methoden die met het [WebMethod] attribute zijn gemaakt:



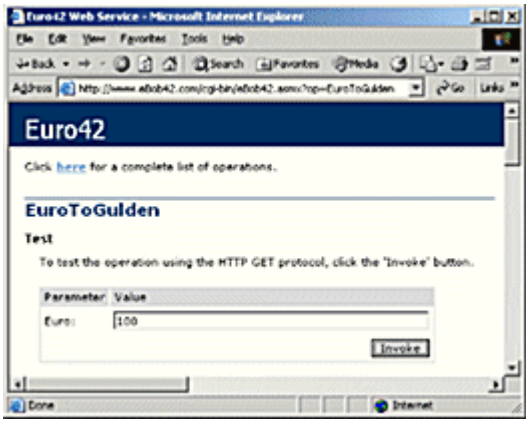
WSDL voor Delphi for .NET Web Services

In het screenshot hierboven zie je al de link naar de "Service Description" van de web service, die naar http://localhost/scripts/eBob42.asmx?WSDL wijst wat uiteraard de WSDL (Web Service Description Language) beschrijving van de web service en zijn drie methoden oplevert. Misschien leuk om te zien, maar zelfs voor de drie simpele methoden die ik heb gemaakt is het toch al enkele pagina's die we te zien krijgen. Gelukkig hoeven we er zelf (nog) helemaal niks mee te doen, en het is alleen maar fijn dat (ook) een .NET web service in staat is om zijn eigen WSDL te produceren, net als we al eerder bij Delphi 6 en 7 Enterprise konden zien (toen nog met behulp van de speciale WSDLHTMLPublish component overigens). Deze keer zit het produceren van WSDL in de base class WebService "verborgen", en dat krijgen we dus kado als we zelf een web service Euro42 van deze base class afleiden.
We komen hier straks nog op terug als we de Euro42 web service gaan gebruiken in een andere tool. Laten we eerst echter eens kijken hoe we de nieuwe .NET web service kunnen testen zónder hem te hoeven importeren in een andere omgeving.

Delphi for .NET Web Services Testen

Onder .NET is het vrij eenvoudig om een .NET web service te testen. In het vorige screenshot waren de drie methoden van Euro42 te zien: GuldenToEuro, EuroToGulden en About. Als we op één van deze methoden klikken - bijvoorbeeld About - komen we in een pagina die met een button die de About methode voor je zal testen. Als je op die button klikt wordt de About methode aangeroepen, en krijgen we het SOAP resultaat te zien in een nieuw browser window. Voor de About method is dat als volgt (let ook op de namespace http://www.eBob42.com):

Zoals te zien is, bestaat het resultaat uit een SOAP envelope met de XML string "Dr.Bob's Euro42 Web Service written in Delphi for .NET (preview)", precies zoals we hadden gemaakt.
De About methode is niet echt spannend, dus laten we hetzelfde eens doen met de GuldenToEuro en EuroToGulden methodes. In die gevallen zul je zien dat we behalve een button ook een editveld hebben om de input parameters mee te geven (de Gulden of Euro die geconverteerd moet worden).



In bovenstaande screenshot heb ik 100 ingevuld als waarde voor Euro, en wil ik de EuroToGulden methode aanroepen. Het resultaat is als volgt:

En inderdaad, 100 Euro is 220.371 guldens.

Delphi for .NET Web Services gebruiken

Als we eenmaal tevreden zijn na het (lokaal) testen van de Delphi for .NET web service wordt het tijd om deze in praktijk te gaan gebruiken. De huidige versie van de Delphi for .NET preview command-line compiler bevat helaas nog geen mogelijkheden om WSDL te importeren en er Delphi for .NET import units van te maken.
Om de Euro42 web service onder .NET te gebruiken zullen we dus even een uitstapje naar C# moeten maken. De .NET SDK bevat een tool genaamd WSDL.exe die te vinden is in de Bin directory van de FrameworkSDK directory. We kunnen WSDL.exe aanroepen met een aantal argumenten, waaronder /language die de waarde "cs" (voor C#) kan krijgen, maar uiteraard geen "d" voor Delphi. Met de /n switch kunnen we de namespace aangeven die gegenereerd moet worden in de import unit. Dit is nodig omdat we straks met de Delphi for .NET preview command-line compiler de gegenereerde C# code weer willen gebruiken. De uiteindelijke aanroep naar WSDL.exe is als volgt (let op dat dit één lange regel moet zijn):

```
wsdl.exe /language:cs /n:Euro42
 http://localhost/scripts/eBob42.asmx?WSDL
```

We krijgen als resultaat een C# import file genaamd Euro42.cs. In deze source file vinden we de namespace Euro42, alsmede de class Euro42 met de methodes About, GuldenToEuro en EuroToGulden. Er zit nog veel meer in de gegenereerde C# import unit, maar dat is voornamelijk om de communicatie met de SOAP server op te zetten, de argumenten door te geven, etc. gebruikmakend van een zgn. proxy class. Voor we zelf deze proxy class kunnen gebruiken moeten we de gegenereerde Euro42.cs file eerst compileren met de C# command-line compiler (ook die zit gratis bij het .NET Framework zelf - je hebt dus geen Visual Studio .NET nodig om mee te doen). De aanroep van de C# compiler csc.exe is als volgt, waarbij we /t:library moeten meegeven:

```
csc.exe /t:library /out:Euro42.dll Euro42.cs
```

Het resultaat is de Euro42 proxy DLL, die we kunnen gebruiken om de web service te gebruiken. Een klein C# programma om dit te doen - in file UseEuro42.cs - is als volgt:

```
using System;
namespace Euro42
{
 class UseEuro42
 {
  static void Main(string[] args)
  {
    Euro42 euro = new Euro42();
```

```
    Console.WriteLine(euro.About());
    double gulden = euro.EuroToGulden(100);
    Console.WriteLine("100 euro = " +
            gulden.ToString() + " guldens.");
    Console.WriteLine("100 gulden = " +
            euro.GuldenToEuro(100).ToString()
                    + " euros.");
  }
 }
}
```

Het compileren van dit programma gaat als volgt (deze keer wordt het een executable, dus hebben we de /t:library niet meer nodig, maar moeten we wel aangeven dat we de Euro42.dll willen gebruiken):

```
csc.exe /r:Euro42.dll UseEuro42.cs
```

Het resultaat is een UseEuro42.exe native .NET executable die de Euro42 web service gebruikt en de volgende output oplevert:

```
Dr.Bob's Euro42 Web Service written in
  Delphi for .NET (preview)
100 euro = 220.371 guldens.
100 gulden = 45.3780216090139 euros.
```

Het zou natuurlijk nog leuker zijn als we de Euro42 web service met de Delphi for .NET preview command-line compiler zouden kunnen gebruiken. Omdat we de WSDL niet direct kunnen importeren tot een Delphi import unit, zullen we dit op een andere manier moeten aanpakken. Gelukkig is .NET een taalonafhankelijk framework, en dus kunnen we vanuit Delphi for .NET gewoon gebruik maken van de gegenereerde C# import unit Euro42.cs, of liever gezegd van de gecompileerde Euro42.dll assembly.
Het volgende Delphi for .NET console programma doet ongeveer hetzelfde als het C# programma dat we eerder zagen, en staat in file UseEuro42.dpr:

```
program UseEuro42;
{$APPTYPE CONSOLE}
uses
  Euro42;

var
  E: Euro42.Euro42;
begin
  E := Euro42.Euro42.Create;
  writeln(E.About);
  writeln('100 euro = ',
    E.EuroToGulden(100), ' guldens.');
  writeln('100 gulden = ',
    E.GuldenToEuro(100), ' euros.')
end.
```

Let vooral op dat ik de Euro42 unit gebruik, terwijl we daar niet de Delphi source code voor hebben. Dit is een verwijzing naar de Euro42 assembly (oftewel Euro42.dll die het resultaat is van de C# compilatie van de gegenereerde C# import unit). Merk op dat de variabele E van type Euro42.Euro42 is.

De eerste Euro42 slaat op de naam van de assembly (de namespace binnen deze assembly), terwijl de tweede Euro42 de naam van de web service class is binnen deze assembly. Op deze manier kunnen we met de -LU vlag van de dccil Delphi for .NET preview command-line compiler aangeven dat we de Euro42 assembly willen gebruiken, en dan daaruit de Euro42 class. Oftewel:

```
dccil -LUEuro42 UseEuro42.dpr
```

Dit levert nog wel een foutmelding op (alhoewel het geen compiler error is maar meer een CLR meta data error bij het genereren van de executable, die vervolgens gewoon genegeerd wordt door de dccil compiler). In een latere update zal dit probleem waarschijnlijk uit de wereld zijn. Daar kom ik nog wel een keertje op terug.
Het resultaat is in ieder geval eveneens een UseEuro42 executable die dezelfde output oplevert als de C# versie.

Importeren met Delphi 7

Als laatste wil ik nog even aangeven hoe we de .NET web service kunnen importeren met Delphi 7. Daar kunnen we de WSDL Importer voor gebruiken uit de Object Repository, maar ook een command-line tool (om in de stemming te blijven), die bij Delphi de naam WSDLImp heeft en in de Delphi\Bin directory te vinden is. Aanroepen gaat als volgt:

```
WSDLImp http://localhost/scripts/eBob42.asmx?WSDL
```

Dit levert een file eBob42.pas op (en niet Euro42.pas), die o.a. de volgende code bevat voor het interface type Euro42Soap en de GetEuro42Soap functie om dit interface op te leveren:

```
type.
  Euro42Soap = interface(IInvokable)
  ['{A6AE3E42-89EC-46DE-4455-E72DB3AB4E75}']
   function  About: WideString; stdcall;
   function  GuldenToEuro(const Gulden: Double):
             Double; stdcall;
   function  EuroToGulden(const Euro: Double):
             Double; stdcall;
  end;

  function GetEuro42Soap
    (UseWSDL: Boolean=System.False;
    Addr: string='';
    HTTPRIO: THTTPRIO = nil): Euro42Soap;
```

Vanuit een Delphi toepassing kunnen we nu de GetEuro42Soap functie aanroepen om het Euro42Soap interface op te leveren, waarvan we dan de methodes About, GuldenToEuro en EuroToGulden kunnen aanroepen.

Zie http://www.eBob42.com/cgi-bin/eBob42.asmx om de Euro42 web service zelf uit te proberen.

O'Reilly Book Review: Programming .NET Web Services

This book was published in September 2002 (so it's brand new), and is very pleasant to read - it took me just over a week to read it from cover to cover. The book shows how to create web services in the .NET Framework with ASP.NET, but does not spend time on the (ASP).NET fundamentals or C# language syntax, as some other books on .NET web services sometimes do.

The book consists of 11 chapters and two appendices. Not all of them equally interesting, but pretty complete. The first chapter is more a little history overview that puts web services into perspective. Interesting to read (even for your boss). It also discusses related techniques such as SOAP, WSDL and UDDI, and how they work in .NET.

The second chapter is the place where we start to build our first web services in C# and ASP.NET. For the first example, you don't even need Visual Studio .NET for this, only the .NET Framework (and a text editor, but even notepad will do). Later examples show how we can use Visual Studio .NET to build as well as deploy the web service engines. The next chapter (3) is focused on consuming web services, explaining the publish/find/bind model, and then continuing in detail on the binding process. Again, apart from using Visual Studio .NET, the chapter also shows how we can consume web services without it (using wsdl.exe and csc.exe).

After these basics, chapter 4 continues with data types in web services, from simple .NET primitive data types to more complex data types, including .NET DataSets.

Chapter 5 explains how we can manage state in web services, starting with the great state debate (the three options developers face when state is required), and a discussion on stateless vs. stateful design. The chapter then describes ASP.NET's session state facility for use in web services, and finally the use of SOAP Headers.

Chapter 6 is all about asynchronous services - where a client can make a call or request to a web service engine, and returns to pick it up at a later time. This ensures that the client (consumer) does not have to wait (or is blocked) while the call is executes. The chapter shows how to implement this in different ways.

The next two chapters (7 and 8) are about caching, profiling and debugging web services. These chapters can be used to learn how to optimise and debug your web services using ASP.NET and Visual Studio .NET. For logging purposes, we'll even see how to implement SOAP extensions.

Chapter 9 is about security and authentication. It starts by listing a number of potential vulnerabilities, such as the network, operating system, web server, ASP.NET platform and finally the web service application itself. The focus is then turned to the last three: from IIS to ASP.NET and our own application (using a ticket system).

Chapter 10 is an interesting one, about publishing and discovery. This is where UDDI (universal description, discovery and integration) and tModels come in. In my view UDDI is best used in an intranet environment, but this book also believes in UDDI for the internet.

Chapter 11, finally, is about interoperability of web services implemented in different environments, and using different SOAP message formats (like document style versus RPC style). The SOAP interoperability test - consisting of 11 echo methods - is also covered, but it doesn't go into some of the more complex issues such as interoperability of .NET DataSets. Unfortunately, only a few SOAP toolkits are actually mentioned, so we have to use the references and URLs mentioned in the chapter to get more information about the SOAP interoperability between Visual Studio .NET and - for example - Borland SOAP tools such as Delphi or JBuilder (or plain Java for that matter).

Appendix A contains a quick reference for the System.Web.Services namespace, and appendix B contains a list of descriptions of some of the more commonly used web service technologies. Finally, there's a 13 page index, which I did not have to use (yet), since the chapters themselves are very clear about their content.

A strong point of the book is that we do not always have to use the full-blown Visual Studio .NET environment to follow along, although some more advanced topics don't hesitate to show the detailed screenshots and steps to do it in Visual Studio .NET. C# is used as programming language throughout the book, which is also positive in my view.

For readers who have been introduced to web services using Delphi, this book will contain some familiar information, but also a lot of additional techniques and ways to build web services for .NET specifically (I found the detailed configuration instructions for ASP.NET very helpful). If you're serious about web services for .NET, and already know a little bit of C# or ASP.NET, then I can recommend this book to you without hesitation.

Auteurs: Alex Ferrara & Matthew MacDonald
Uitgever: O'Reilly
ISBN: 0-596-00250-5
Prijs: $ 39.95, 396 pages

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

Dr.Bob's Delphi 7 Clinics

30 januari 2003

dbExpress & DataSnap (MIDAS)

Keywords: dbExpress, DataSnap, ClientDataSet, DataSetProvider, Connection components, Performance, 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).

27 februari 2003

Web Development met IntraWeb

Keywords: IntraWeb, Page Mode, Application Mode, Login, Sessions, State Management, IntraWeb Components, Deployment

In deze Delphi Clinic bekijken we IntraWeb van alle kanten. Nadat WebBroker door InternetExpress (Delphi 5) en daarna met WebSnap (Delphi 6) is uitgebreid, bleven vele ontwikkelaars toch zitten met een gevoel van "kan het niet wat meer op de Delphi manier?" Met IntraWeb - een third-party tool van AtoZedSoftware - die bij Delphi 7 zit, kunnen we dit nu eindelijk: web server toepassing bouwen zoals je normale Delphi programma's bouwt. Tijdens deze Delphi Clinic laat ik zien hoe we met IntraWeb nieuwe web server toepassingen kunnen maken, hoe we die kunnen deployen op een web server (welke vormen wel, welke vormen niet), en ook hoe we met IntraWeb onze bestaande WebBroker of WebSnap toepassingen kunnen uitbreiden met IntraWeb "Page Mode" functionaliteiten (zodat we niet nog een keer bestaande investeringen hoeven weg te gooien of opnieuw hoeven te beginnen).
De meeste aandacht gaat echter uit naar IntraWeb in "Application Mode", de IntraWeb componenten (en hoe we die zelf kunnen bouwen) en State Management binnen IntraWeb wat geheel transparant is (maar ik laat zien hoe en wat er achter de schermen gebeurt).

27 maart 2003

BizSnap: XML, SOAP & Web Services

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

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 (en beheren via UDDI).

24 april 2003

WebSnap e-Commerce Development

Keywords: WebSnap (Custom) Adapters, Login/Access, Sessions, Deployment, Intro ExpressWeb Framework

Tijdens deze Delphi Clinic bouwen we een e-business toepassing met behulp van technieken uit WebSnap. Het resultaat is een e-commerce toepassing met alle toeters en bellen zoals login/logout, registratie van nieuwe gebruikers, database access, zoeken, bestellen, het winkelwagentje, e-mail support, etc.
Tot slot volgt een korte intro **ExpressWeb Framework** van Developer Express aan het eind van deze Clinic.

29 mei 2003

Delphi .NET Framework Development

Keywords: Delphi .NET Framework Development, ASP.NET, VCL for .NET

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. Onderwerpen zijn het gebruik van COM in .NET (en vice versa), .NET assemblies, VCL for .NET en Delphi als scripting taal bij ASP.NET voor web forms en web services (zie ook elders in dit blad voor meer informatie).

26 juni 2003

.NET (Delphi) en J2EE (JBuilder) via SOAP

Keywords: Delphi .NET Framework Development, JBuilder J2EE, SOAP

Deze Delphi Clinic zal tevens over Borland JBuilder gaan, met speciale gastspreker Dave Henneman. We bouwen SOAP web services in zowel JBuilder (J2EE) als Delphi for .NET, waarna deze onderling geïmporteerd en gebruikt worden. Het resultaat is een bioscoopkaartjes-reserveringssysteem. met alle leuke toeters en bellen.

De locatie 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 (Delphi 6 Developer's Guide, Kylix Developer's Guide of C++Builder Developer's Guide).
* Bij inschrijving voor vijf trainingsdagen krijgt u de zesde dag kado (dus zes dagen voor de prijs van vijf).

Extra korting is mogelijk bij meer dan vier personen van hetzelfde bedrijf, evenals maatwerk training of speciale Delphi workshops bij u op locatie.