Delphi OplossingsCourant

Vol. 8. No. 2, Een gratis electronische publicatie van **Bob Swart Training & Consultancy** (**eBob42**) - http://www.eBob42.com

Helmond, 5 september 2006,

Welkom bij het nieuwe nummer van de Delphi OplossingsCourant. Precies acht maanden na het vorige nummer, is het vandaag weer een bijzondere dag, maar daarover straks meer.

Eerst iets anders: in februari 2006 maakte Borland opmerkelijk nieuws bekend: de onderneming heeft namelijk besloten om zich volledig op zgn. ALM (Application Lifecycle Management) tools te richten, en de bestaande IDE (Integrated Development Environment) tools zoals Delphi, C++Builder, C#Builder en JBuilder passen niet in dat plaatje. Met als gevolg dat Borland op zoek is gegaan naar een overname kandidaat voor de IDE divisie.
Als Delphi ontwikkelaar kun je hier in eerste instantie natuurlijk flink van schrikken. Maar in tweede instantie, als je ook de reacties van de leden van het Delphi R&D Team hebt gehoord, lijkt het allemaal wel mee te vallen, en misschien wel beter voor Delphi dat het op deze manier een eigen onderdak vindt. Een van de gevolgen is namelijk dat niet langer Borland (met tientallen verschillende producten), maar een kleinere onderneming nu zijn aandacht op Delphi kan richten. Behalve Delphi (voor Win32 en .NET) vallen overigens ook C++Builder, C#Builder, JBuilder, InterBase en zelfs Kylix onder de IDE divisie en dus ook het nieuwe bedrijf. Er was al een product dat een aantal persoonlijkheden bij elkaar stopt onder de naam Borland Developer Studio (BDS). Dat product krijgt niet alleen een nieuwe naam (het "Borland" deel zal waarschijnlijk komen te vervallen), er zijn ook sinds vandaag (5 september 2006) vier speciale *"single-personality"* Turbo versies uitgekomen. Turbo Delphi, Turbo Delphi for .NET, Turbo C++ en Turbo C# zijn zowel in betaalbare Professional als gratis Explorer versie te verkrijgen. In dit nummer van de Delphi OplossingsCourant meer informatie over deze tools en de eerste ervaringen in de praktijk.

Delphi 2006 Clinics

In de tweede helft van dit jaar geef ik weer een drietal Delphi 2006 trainingen:

19 oktober
ASP.NET 1.1 Web Development
http://www.eBob42.com/training/2006asp.htm

9 november
Enterprise Core Objects III Development
http://www.eBob42.com/training/2006eco.htm

30 november
Advanced ASP.NET Web Development
http://www.eBob42.com/training/2006asp2.htm

Wie niet bij een van deze dagen aanwezig kan zijn, maar toch interesse heeft in deze of andere Delphi onderwerpen kan altijd (geheel vrijblijvend) contact opnemen om te kijken of we een Delphi Workshop in Helmond of een Custom Clinic op locatie (in-house) kunnen organiseren. Daarbij vullen we de gewenste onderwerpen en agenda helemaal op maat in.
Zie http://www.eBob42.com/workshop voor alle mogelijkheden, of stuur vrijblijvend een mailtje.

Daarnaast is het mogelijk om cursusboeken over verschillende Delphi onderwerpen aan te schaffen voor zelfstudie (plus e-mail ondersteuning indien nodig). Zie http://www.eBob42.com/courseware.

Deze cursusboeken worden ook door andere Delphi trainingsbedrijven gebruikt, en de nieuwe Borland Developer Tools Divisie (ook wel "DevCo" genaamd) verkoopt vanaf deze week mijn ingebonden Delphi 2006 ASP.NET 1.1 Web Development cursusboek via http://shop.borland.com.

Veel leesplezier, en schroom niet om op- of aanmerkingen te mailen of in mijn weblog achter te laten te http://www.eBob42.com/weblog.

Groetjes,

Turbo Delphi 2006

Turbo versies van Delphi voor Win32, Delphi voor .NET, C++ (voor Win32) en C# zijn nu verkrijgbaar in betaalbare Professional en gratis Explorer edities.

Op dinsdag 5 september zijn vier nieuwe producten in twee verschillende edities toegevoegd aan het aanbod van Borland. Eigenlijk zijn het geen nieuwe producten, maar slechts nieuwe "verpakkingen" om het bestaande aangepaste product Borland Developer Studio 2006 dat ook onder de naam Delphi 2006, C++Builder 2006 en C#Builder 2006 verkrijgbaar is (in alle gevallen echter met dezelfde inhoud: de BDS 4.0 IDE met daarin persoonlijkheden voor Delphi Win32, Delphi for .NET, C++Builder en C#). In dit artikel ga ik dieper in op de (on)mogelijkheden en de voor- en nadelen van de nieuwe Turbo edities.

Allereerst even een paar belangrijke feiten: de Turbo edities van Delphi, C++ en C# zijn niet nieuwere versies of upgrades van BDS 2006. Integendeel, het is exact dezelfde IDE, alleen waar BDS 4.0 de keuze biedt uit vier verschillende persoonlijkheden (Delphi voor Win32 of .NET, C++ of C#), daar biedt een Turbo versie slechts één persoonlijkheid. Je kan echter maar één Turbo versie tegelijkertijd op je machine hebben staan: dus of Turbo Delphi voor Win32, of Turbo Delphi voor .NET maar niet allebei. Dat is jammer, want wie alleen Delphi doet wil wellicht graag zowel de Win32 als .NET editie van Turbo Delphi op zijn machine hebben. En wie alleen aan Win32 doet wil wellicht Turbo Delphi en C++ voor Win32, en zo zouden .NET ontwikkelaars wellicht zowel Turbo Delphi and C# for .NET willen hebben. De technische reden voor deze beperking zit hem in het feit dat de Turbo versies in feite dezelfde IDE gebruiken als de gehele Borland Delphi Studio 2006. In alle gevallen gaat het om BDS 4.0, en die gebruikt een vaste directory- en registry-opmaak voor zijn informatie. Wie bereid is te experimenteren slaagt er wellicht in om meerdere Turbo versies naast elkaar te installeren, maar dat is wel op eigen risico (officieel wordt het niet ondersteund). Een consequentie is tevens dat je de Turbo versies niet op een machine kunt zetten waar de volledige BDS 2006 zoals Delphi 2006 al staat. Uiteraard is het wel mogelijk om een Turbo versie te installeren op een machine waar Delphi 2005 of een andere versie van Delphi, C++Builder of C#Builder staat. Zolang het maar geen Delphi 2006 is.
Van iedere Turbo versie zijn twee edities verkrijgbaar: de betaalbare Professional en de gratis Explorer editie, die overigens allebei ook met Borland InterBase 7.5 Developer Edition worden geleverd.

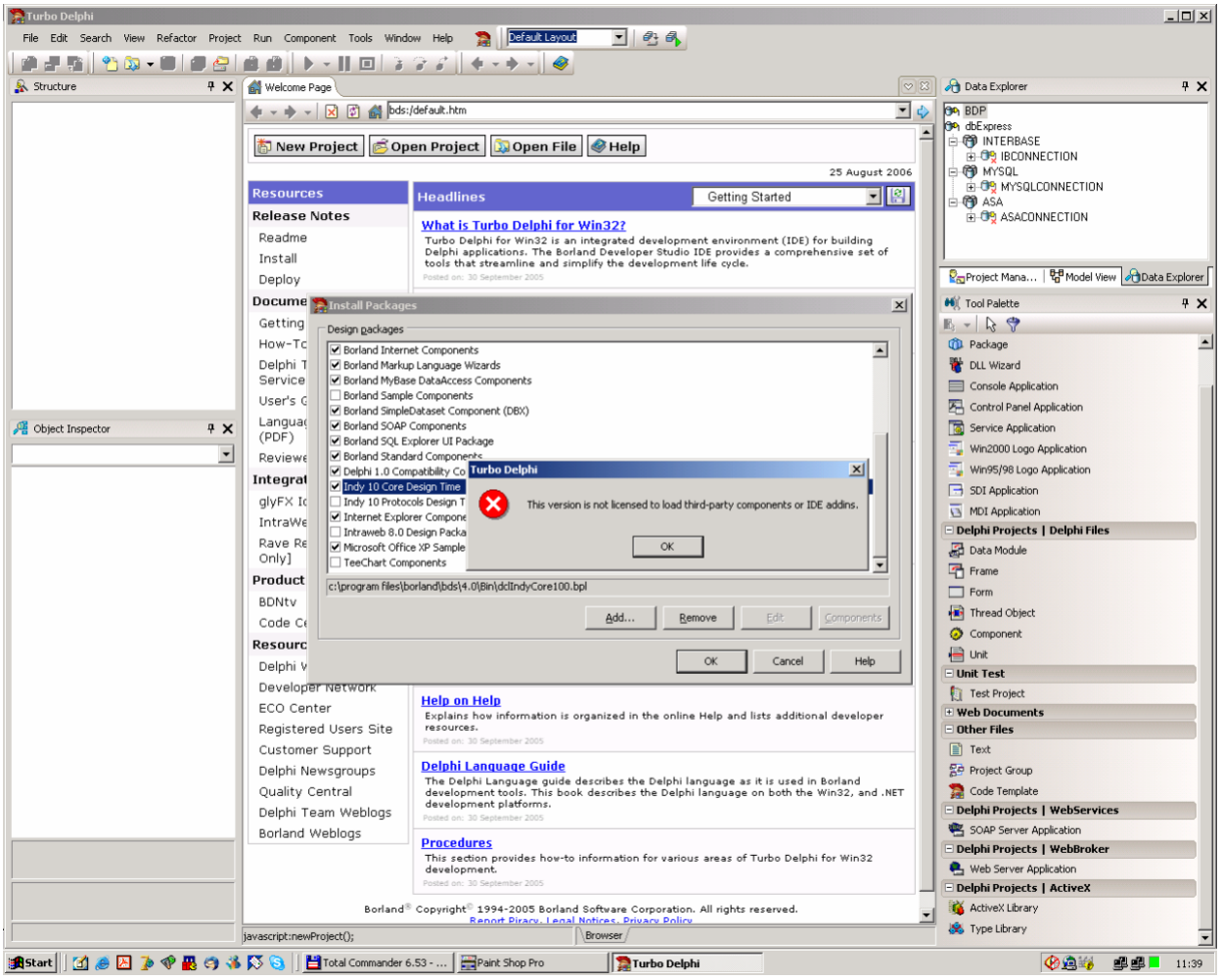
Turbo Professional

De Professional edities van Turbo Delphi, Turbo Delphi for .NET, Turbo C++ en Turbo C# zijn – zoals de naam al zegt – te vergelijken met de Professional editie van de Borland Developer Studio 2006. Alleen bevat de laatste dus alle vier persoonlijkheden, en een Turbo editie maar ééntje. Het betekent ook dat Turbo Delphi en C++ dus geen DataSnap functionaliteiten bevatten, en IntraWeb 8 slechts in Page Mode. Turbo Delphi for .NET en Turbo C# bevatten geen DataSync, DataHub of DB Web controls, wel ECO III, maar slechts met XML Persistence mogelijkheden, en geen ECO III voor ASP.NET.

Turbo Delphi Explorer

Turbo Explorer is de gratis editie van de Turbo-lijn. Er zijn echter wel enkele beperkingen aan de gratis Explorer edities. De belangrijkste beperking is het feit dat je geen third-party componenten of add-ins aan de IDE kunt toevoegen. En als third-party component geldt hierbij alles wat niet door Borland zelf gemaakt is.

In het geval van Turbo Delphi Explorer voor Win32 geldt dit ook voor Indy, IntraWeb en TeeChart. Deze componenten worden wel geïnstalleerd op disk, maar zijn niet terug te vinden in de Tool Palette (de design-time packages zijn disabled, en kunnen ook niet enabled worden, zoals in de volgende screenshot te zien is waar ik probeerde de Indy 10 Core Design Time package te enablen).



Het is wel mogelijk om de Indy en TeeChart componenten tijdens run-time te creeeren en te gebruiken, maar je kan ze dus niet op je form zetten. IntraWeb toepassingen zijn vrijwel niet te ontwikkelen in Turbo Delphi Explorer. Voor andere third-party componenten zul je zelf moeten experimenten of het te doen is ze tijdens run-time aan te maken en te gebruiken. In praktijk zijn de gratis Turbo edities bedoeld om ontwikkelaars kennis te laten maken met de IDE en de normale componenten in het Tool Palette. Desalnietemin is het ook toegestaan om (zelfs commerciële) toepassingen te ontwikkelen met de Turbo edities. Zolang je uit de voeten kunt met de normale verzameling VCL componenten kun je met Turbo Delphi Explorer voor Win32 dus profiteren van een gratis IDE met alle taaluitbreidingen en IDE features die in Delphi 2006 zijn aangebracht. Dus inclusief de nieuwe for-in-loop en andere taaluitbreidingen die later in dit nummer van de Delphi OplossingsCourant aan de orde komen.

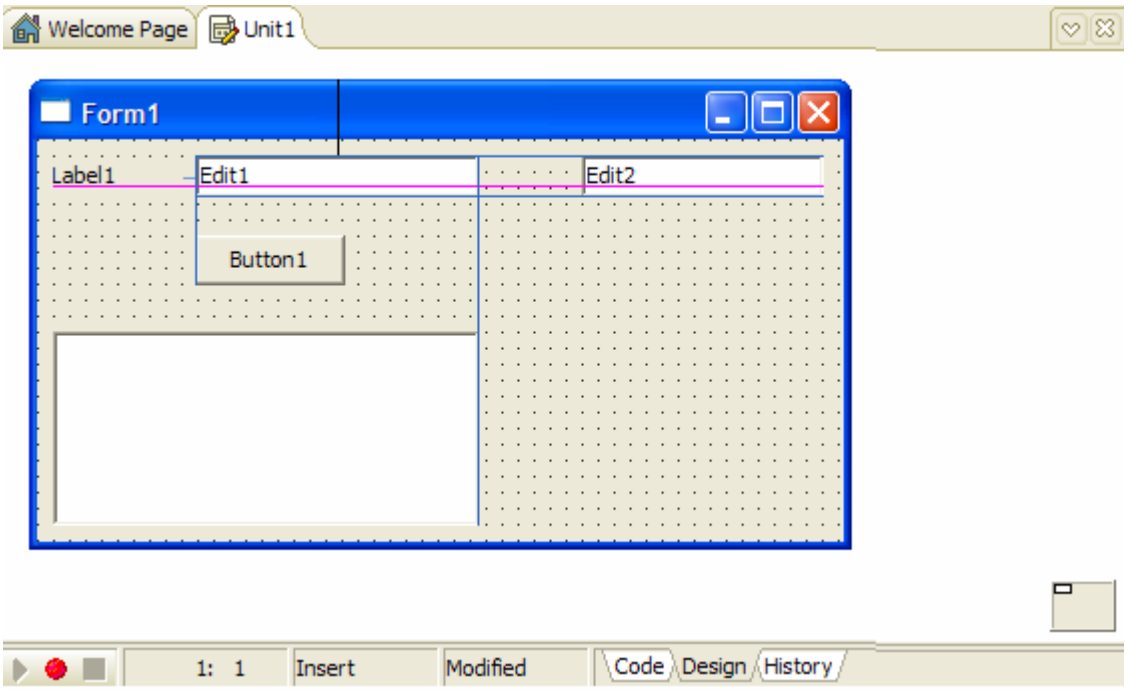
Lastig is dat de Turbo Explorer edities net als de Turbo Professional edities alleen met de IDE compiler worden geleverd, en niet met de command-line compiler. Mogelijk komt de command-line compiler wel beschikbaar voor ontwikkelaars die de Turbo Professional editie hebben aangeschaft.

Installatie

Let op: ook al kun je ervoor kiezen om bijvoorbeeld alleen Delphi 2006 voor Win32 te starten, de IDE zal toch het .NET Framework nodig hebben om te kunnen draaien. Vandaar ook dat tijdens installatie wordt gecontroleerd of het .NET Framework versie 1.1 aanwezig is, en desgewenst wordt geïnstalleerd op je machine. Dat geldt uiteraard niet voor de Win32 toepassingen zelf die je met Delphi 2006 voor Win32 maakt: die hebben alleen Win32 nodig. Wellicht is het hierbij nuttig op te merken dat de ondersteuning voor Windows 95, Windows 98, Windows NT en Windows ME officieel niet meer aanwezig is. Delphi 2006 Win32 toepassingen hebben officieel minimaal Windows 2000, XP of 2003 nodig voor deployment. Natuurlijk draait het vaak nog best onder NT, 98 of zelfs 95, maar dat is allemaal "op eigen risico", en wordt officieel niet meer ondersteund.
Voor .NET toepassing geldt dat Borland Developer Studio 2006 alleen het .NET Framework versie 1.1 ondersteunt, en niet de recent uitgekomen .NET Framework versie 2.0 (of de .NET 3.0 preview).

VCL Designer

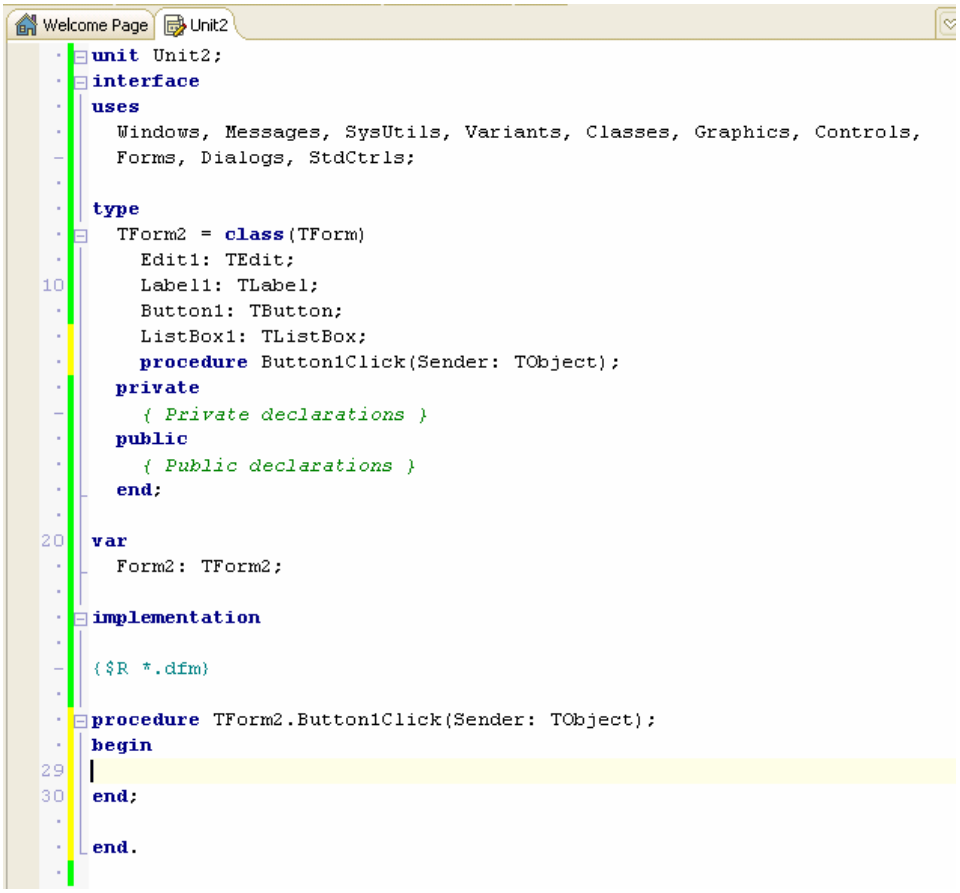
De VCL Designer van Turbo Delphi bevat zogenaamde Live Guidelines, die je vertellen wanneer controls netjes op een lijn staan (zowel boven, onder, links of rechts), een redelijke afstand tot elkaar of de border van het form hebben, en dat de tekst van een label en edit op een lijn staan. De zwarte lijnen geven aan dat de componenten op een goede afstand van de border of het frame afstaan, de blauwe lijnen geven aan dat de componenten aan de boven, onder of zijkant netjes gelijk staan, terwijl een horizontale roze lijn aangeeft dat de tekst zelf (binnen de TLabel en TEdit etc.) op gelijke hoogte staat. Behalve de snap-to-grid functionaliteit die we al kende, hebben we nu ook de mogelijkheid om de componenten naar een blauwe of roze lijn te laten "snappen". Net als bij BDS 2006 zelf.



Een andere fijne feature is rechtsonder de VCL Designer te vinden, en kunnen we gebruiken om de default positie van de VCL forms te specificeren. Het betreft een klein vierkantje dat de desktop representeert, en waarbinnen een nog kleiner vierkantje het VCL form aangeeft (zie ook de screenshot hierboven, waarin beide vierkantjes te zien zijn). Standaard staat het kleintste vierkantje linksboven, maar door nu het kleinste vierkantje te verslepen naar het midden of ergens anders in het kleine vierkantje, kun je dus de default positie van je form bepalen. Het effect is dat de Top en Left properties van je VCL Form gezet worden (en dat had je ook handmatig kunnen doen, maar op deze manier krijg je er een beter gevoel bij wat het eindresultaat zal zijn).
Al deze features zijn overigens alleen beschikbaar voor de VCL en VCL for .NET Forms Designer.

Code Editor

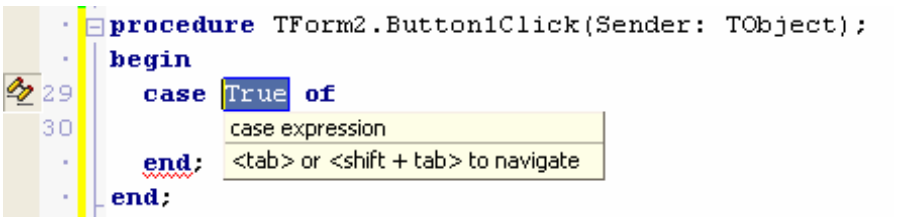
Als we naar de Code Editor kijken, zien we een aantal zaken die meteen opvallen. Zo heeft niet elke regel meer een zichtbaar regelnummer, maar wordt dat om de 10 regels gedaan, plus de huidige regel zelf, met om de vijf regels een streepje in plaats van een punt. Dit geeft een rustiger overzicht, waarbij je echter nog steeds snel kunt zien waar je bent (of welk regelnummer je ook zoekt).



Daarnaast zien we soms gele en groene kleuren links van de code. Die geven aan wat de status is van de editor buffer. Groen will zeggen dat het nieuwe code is, die inmiddels ook op disk is bewaard. Geel wil zeggen nieuwe code die echter nog niet naar disk is weggeschreven. Dus als op dat moment de stroom uitvalt (of de IDE zou vastlopen), dan weet je in ieder geval dat de groene code velig is, maar de gele code weg is. Kun je altijd zelf inschatten hoe groot het riciso is van het niet even opslaan van je code.

Block Completion en Live Templates

De Delphi IDE is ook wat hulpvaardiger als het gaat om het genereren van code (een feature die ze Block Completion noemen). Als je "begin" intikt en daarna een return, dan krijg je de "end "kado. Dat is soms handig. Wat al wat leuker is, is het intikken van "try", waarna je een "finally" en "end" kado krijgt. Het leukste is echter het gebruik van het case statement. Als je case variabele daarbij een enumerated type is, dan is een druk op de tab toets voldoende om alle mogelijke waarden als case labels te krijgen.
Neem bijvoorbeeld de Position property van de TForm. Als je "case" intikt gevolgd door een spatie, krijg je meteen het volgende te zien:

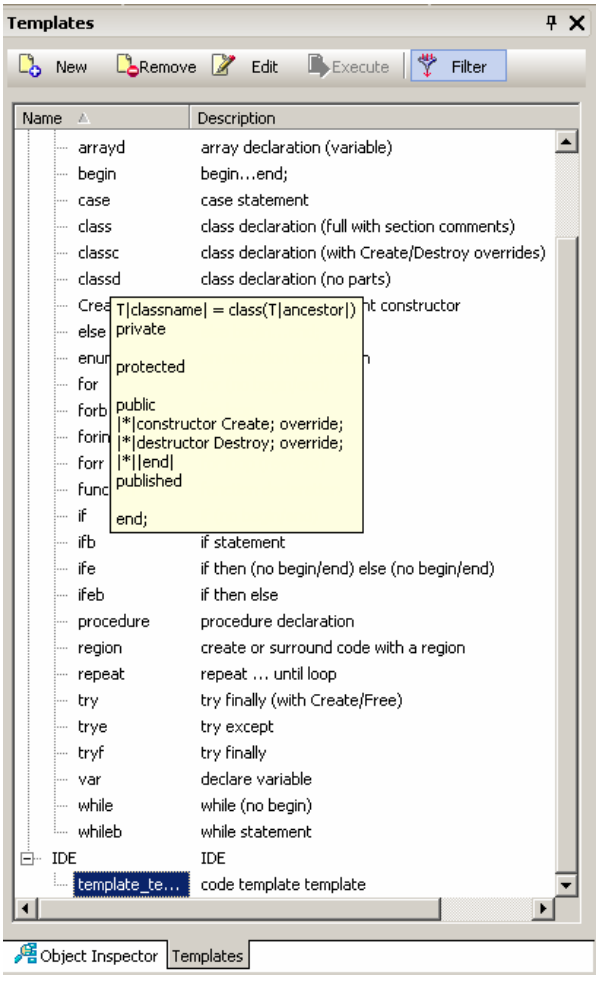


Als je op de plek van True dan "Position" intikt, gevolgd door een tab, krijg je de volgende code:

```
case Position of
  poDesigned: ;
  poDefault: ;
  poDefaultPosOnly: ;
  poDefaultSizeOnly: ;
  poScreenCenter: ;
  poDesktopCenter: ;
  poMainFormCenter: ;
  poOwnerFormCenter: ;
end;
```

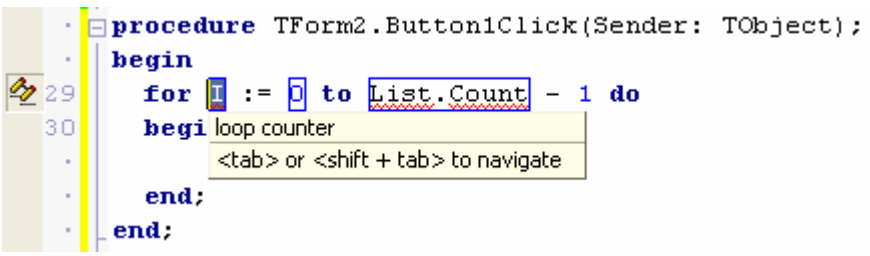
Als je dit een paar keer hebt gebruikt, dan merk je pas hoe prettig het is (nooit meer een gemist case label, of het in hoeven tikken van alle mogelijke waarden).

Een andere nieuwe feature, ook beschikbaar in Turbo Delphi, is Live Templates genaamd. Een overzicht van de beschikbare Live Templates voor de actieve persoonlijkheid van BDS kun je zien met View | Templates. Let op: dit levert geen dialoog op, maar resulteert in een extra tab in de Object Inspector met daarin de aanwezige templates.



Live Templates start je door het betreffende keyword in te tikken, gevolgd door een spatie (zoals ik eerder al bij het case statement vertelde). Als een Live Template aktief is, kun je met de tab toets door de verschillende items lopen en die desgewenst verder invullen. Met Escape beëindig je deze toestand.
Je kan zelf ook Live Templates toevoegen aan deze lijst. Het voordeel van Live Templates ten opzichte van de "oude" Code Templates die je met Control+J kon activeren, is dat Live Templates een stuk code genereren waarin je met de tab toets nog doorheen kunt lopen om bepaalde elementen in te vullen. Je kan nog steeds de "oude" Code Templates krijgen voor Control+J te blijven gebruiken, maar nu krijg je na de naam van de template gevolgd door een spatie de nieuwe Live Template.

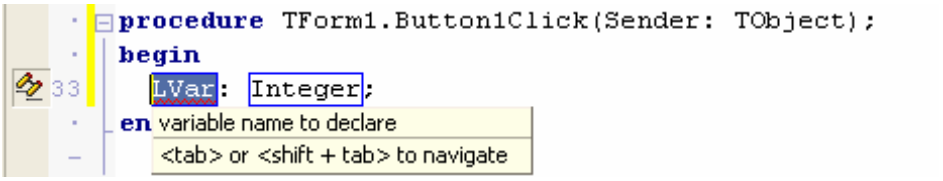
Voor een for loop, levert dit het volgende resultaat na het intikken van "forb" en een spatie:



Met de tab kun je nu van de "I" naar de "0" of de "List.Count" springen om die aan te passen. Met een druk op Escape kun je voortijdig uit de "live" mode komen, maar als je met de tab toets doorloop tot het eind, dan krijg je zelf de variabele declaratie voor de "I" kado, wat tot het volgende resultaat komt:

```
procedure TForm2.Button1Click(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to List.Count - 1 do
  begin
    |
  end;
end;
```

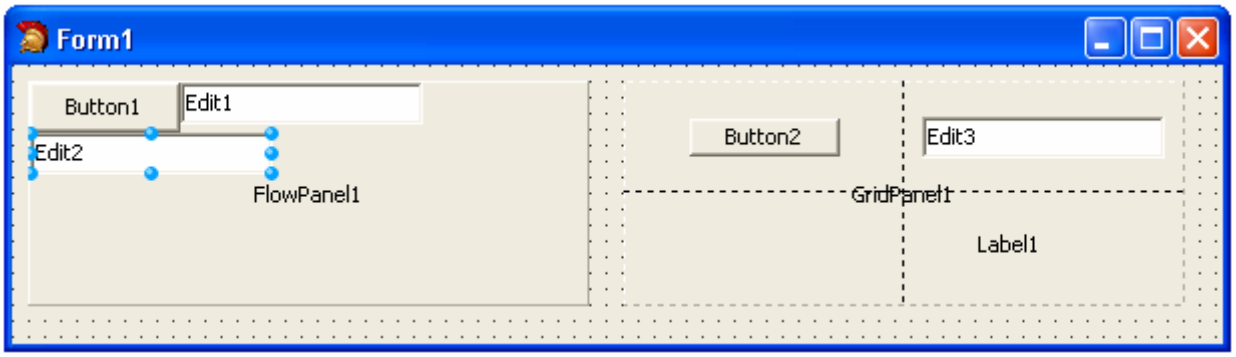
De cursor staat na afloop tussen de begin en end, klaar om verder te gaan. De automatische variabele declaratie kun je trouwens ook "los" krijgen als de "in-place variabele declaration" option. Tik ergens midden in je code gewoon "var" in gevolgd door een tab. Daarna krijg je het volgende beeld, en kun je de naam van de variabele en het type intikken.



Na nog een tik op de tab toets is de variabele declaratie naar de juiste plaats getransporteerd, Wederom zonder dat je je huidige plek in de source code verliest, zodat je meteen verder kunt gaan waar je mee bezig was. Al met al best handig.

Turbo Delphi voor Win32

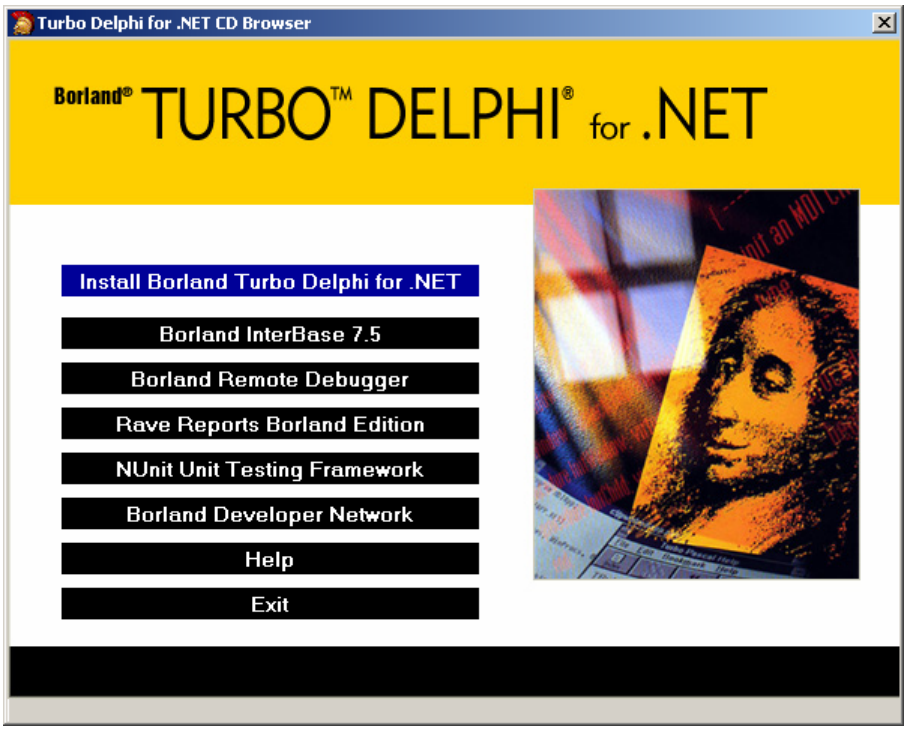
Voor Win32 ontwikkelaars is het goed te weten dat er een stuk of 20 FastCode routines in de Win32 RTL zijn opgenomen, die de snelheid van bestaande Win32 Delphi toepassingen kan doen toenemen door ze alleen maar te hercompileren met Turbo Delphi.
Er zijn een paar nieuwe VCL componenten, zoals de TFlowPanel en TGridPanel, die een beetje werken zoals de ASP.NET designer in flow layout of grid layout. Bij een FlowPanel kun je de FlowStyle zetten (bijvoorbeeld op LeftRightTopBottom), en vervolgens zullen de componenten van links boven naar rechts onder naast elkaar gezet worden. Je kan daarbij de positie van de componenten niet meer aanpassen door ze te verslepen: ze springen steeds terug naar de natuurlijke flow-positie.



Bij het TGridPanel kunnen we opgeven in rows en columns we het grid willen opdelen (default 2 rows en 2 columns), en daarna kunnen we in iedere “cell” precies één componen plaatsen.
Wat handiger en zeker leuker is de TTrayIcon component, die de mogelijkheid biedt om een TrayIcon te laten zien, inclusief animatie, zonder één regel code te hoeven schrijven. Je kunt via een imagelist een lijst van icoontjes meegeven, en tijdens runtime het zichtbare icoontje bepalen. Ook animatie van de icoontjes is mogelijk, net als ballon helpteksten. Werkt prima in praktijk.

Turbo Delphi for .NET

Turbo Delphi for .NET is IDE met daarin alleen de .NET versie van Delphi. Wie bij Delphi 8 for .NET is blijven steken zou dus zeker moeten overwegen om naar Turbo Delphi for .NET over te stappen, om in ieder geval op het niveau van Delphi 2006 for .NET te kunnen werken. Alle WinForms, ASP.NET Web Forms en zelfs VCL for .NET componenten zijn beschikbaar om .NET toepassingen te kunnen bouwen.
Helaas bevat Turbo Delphi for .NET niet de Deployment Manager die zo handig is voor ASP.NET toepassingen (of IntraWeb toepassingen), dus dat moet weer met de hand. In Turbo Delphi for .NET Professional kunnen we met de ECO Designer UML class models maken, met associates en inheritance tussen de classes, en attributes, operations, etc. Dit model kunnen we vervolgens in de persistent maken in een XML document. Het is echter niet mogelijk om de EcoSpace niet een database te schrijven. Het is ook niet mogelijk om met Turbo Delphi for .NET ECO in combinatie met ASP.NET te laten werken. Daar zul je de Enterprise or Architect editie van de volledige Borland Developer Studio voor moeten aanschaffen.
Wat wel heel fijn is dat de Database Explorer, Reflection en zelfs de WinForms Control Import Wizard worden bijgeleverd bij Turbo Delphi for .NET (de WinForms Control Import Wizard zal het niet doen in de gratis Explorer versie vanwege de beperking dat er geen components aan het palette kunnen worden toegevoegd, maar in de Professional editie zit alles er gewoon bij). Je kan bij de installatie ook uit de andere extra tools kiezen die bij Borland Developer Studio worden meegeleverd, zoals InterBase 7.5, Rave Reports Boralnd Edition, and the NUnit Unit Testing Framework.



Conclusie

Borland Developer Studio is de volledige IDE met vier verschillende persoonlijkheden (Delphi, Delphi for .NET, C++ en C#), en in Professional, Enterprise en Architect edities. Speciaal voor de onderkant van de markt zijn er nu van de Professional editie vier verschillende “losse” versies gemaakt: Turbo Delphi, Turbo Delphi for .NET, Turbo C++ en Turbo C#. En wie helemaal geen geld wil of kan uitgeven, kan met de gratis Turbo Explorer edities uit te voeten. Vrijwel volledig functioneel, met als beperking dat er geen componenten of add-ins aan de IDE kunnen worden toegevoegd. Maar zonder 30-dagen trial, dus wat dat betreft kun je die prima gebruiken voor je studie, voor je hobby, zelfs voor je commerciële toepassingen, of om kennis te maken met de mogelijkheden van (Turbo Delphi for) .NET bijvoorbeeld.
Wie nog vragen of opmerkingen heeft over dit artikel of (Turbo) Delphi in het algemeen kan me altijd een e-mailtje sturen of feedback achterlaten in mijn weblog te http://www.eBob42.com/blog

(Turbo) Delphi Taaluitbreidingen

Nu de gratis Turbo Explorer edities van Delphi en Delphi for .NET beschikbaar zijn, kan iedereen profiteren van de Delphi taaluitbreidingen die de laatste jaren aan de taal zijn toegevoegd. In dit artikel wil ik dan ook enkele van de nieuwe taaluitbreidingen aan de orde laten komen. Niet beschikbaar in Delphi 7 of eerder, maar wel in Delphi 8 en later, en dus ook in de Turbo Delphi Explorer versies.

sealed class

Als eerste "feature" wil ik de sealed class bespreken. Dit biedt de mogelijkheid om van een afgeleide class type aan te geven dat er niet verder van afgeleid mag worden. We kunnen dit als volgt uitdrukken:

```
type
  TEindeOefening = class sealed(TObject)
    ...
  end;
```

Behalve van TObject kun je alle classes gebruiken als voorvader - behalve een class die zelf als sealed class is gedefinieerd natuurlijk. Als je probeert van een sealed class een nieuwe class af te leiden, dan krijg je een compiler error (E2353: Cannot extend sealed class).
Wat is het nut van een sealed class? Welnu, het kan een zinvol idee zijn voor componenten bouwers die vaak niet willen dat er nieuwe componenten worden afgeleid van hun werk. En zonder de broncode, is het niet mogelijk om van een sealed class een afgeleide class te maken (en zelfs met de broncode zul je eerst handmatig het sealed keyword moeten weghalen en dan de oorspronkelijke componenten opnieuw moeten compileren).
Ook kan het soms nuttig zijn in een project met meerdere personen om af te spreken dat sommige classes als sealed class worden gedefinieerd. Dat voorkomt dat er te pas en te onpas nieuwe classes worden afgeleid, waardoor je soms door de bomen het bos niet meer ziet (de mate van RAD van Delphi is dermate hoog, dat de verleiding vaak groot is om zelf maar iets te bouwen in plaats van te zoeken naar een bestaande oplossing).

final methods

Gerelateerd aan de sealed class, is er nu ook de mogelijkheid om een method als final method te bestempelen. Dat heeft tot gevolg dat niet de gehele class, maar alleen de betreffende method als "eindstation" wordt aangeduid. Dit kan bijvoorbeeld als volgt worden aangegeven (lees de online help maar na over het nut van de CompareStrings method van de TStrings en TStringList classes):

```
TMyStringList = class(TStringList)
protected
  function CompareStrings(const S1: string; const S2: string): Integer;
    override; final;
end;
```

Op bovenstaande manier geef ik aan dat ik een nieuwe class TMyStringList maak, afgeleid van de class TStringList, en dat ik de method CompareStrings override, maar tevens geef ik aan dat ik de laatste ben die dat kan doen, omdat dit de "final" versie van de CompareStrings method is. Alhoewel TMyStringList geen sealed class is, en er dus rustig nieuwe classes van afgeleid kunnen worden, zal het niet mogelijk zijn om een nieuwe class van TMyStringList af te leiden en daarbij tevens de method CompareStrings opnieuw te overriden en te implementeren.
Ook deze techniek kan nuttig zijn voor component ontwikkelaars of voor software ontwikkelaars die in teams werken waarbij voorkomen dient te worden dat bepaalde methods elders aangepast worden. In sommige situaties is een sealed class het meest voor de hand liggend, maar het kan ook voorkomen dat alleen enkele final methods voldoende zijn. In het geval van een sealed class kan het overigens geen kwaad om eventuele methods van deze sealed class als final te bestempelen (dan kan de compiler over het algemeen iets efficiëntere code genereren).

for-in-do

En derde maar zeker niet mindere uitbreiding in de Delphi programmeertaal betreft een nieuwe for loop. We maken er allemaal gebruik van: for i:=0 to … do … waarbij de I dan de teller is die binnen de loop gebruikt kan worden om bijvoorbeeld elementen uit een array of andere datastructuur te benaderen. Op de traditionele manier gaat dit als volgt:

```
for i:=0 to SL.Count-1 do
  // gebruik SL[i]
```

Maar met de nieuwe for-in syntax zou dat ook als volgt kunnen:

```
for S in SL do
  // gebruik S
```

We hebben hierbij geen iterator meer nodig, maar kunnen over de items van de verzameling lopen (in de juiste volgorde). Wel moeten we rekening houden dat de items slechts read-only benaderd kunnen worden: wijzigingen zijn niet toegestaan (daardoor zou wellicht de volgorde moeten wijzigen bijvoorbeeld).
Wanneer kunnen we de nieuwe for-in loop gebruiken? Welnu, in alle gevallen waarbij er sprake is van een verzameling die uit onderdelen bestaat, zoals de String die in een verzameling TStringList zit, of de Char waar de String mee is opgebouwd. Als je dus alle letters 'a' wilt tellen in de items van een TListBox bijvoorbeeld, kun je nu de volgende code gebruiken:

```
var
  S: String;
  C: Char;
  teller: Integer;
begin
  teller := 0;
  for S in ListBox1.Items do
    for C in S do
      if C = 'a' then Inc(teller);
end;
```

Let op dat de S en C loop-variabelen in bovenstaand stukje code niet gewijzigd mogen worden binnen de loop. Je kan dus niet op deze manier een string aanpassen (de 'a' vervangen door ee '@' bijvoorbeeld). Dezelfde beperking geldt normaal ook voor de loop index variabele (de i in het eerste voorbeeld).
Nog een voorbeeld: wie van alle TButtons op een Form de caption op 'Klik Mij' wil zetten, kan dat als volgt doen:

```
var
  Comp: TComponent;
begin
  for Comp in Self do
    if Comp is TButton then
      TButton(Comp).Caption := 'Klik Mij!'
end;
```

En alhoewel we de loop variabele Comp niet mogen wijzigen, kunnen we wel de properties ervan wijzigen (in dit geval de Caption property).

Ook arrays lenen zich voor het gebruik binnen de for-in loop. En als je ooit in de situatie komt dat de for-in loop niet gebruikt kan worden, dan kun je altijd nog de traditionele for-loop gebruiken, of zelf de for-in loop inbouwen met behulp van een aantal ondersteunende routines - maar dat bewaar ik tot de volgende keer.

Custom DB Web Controls

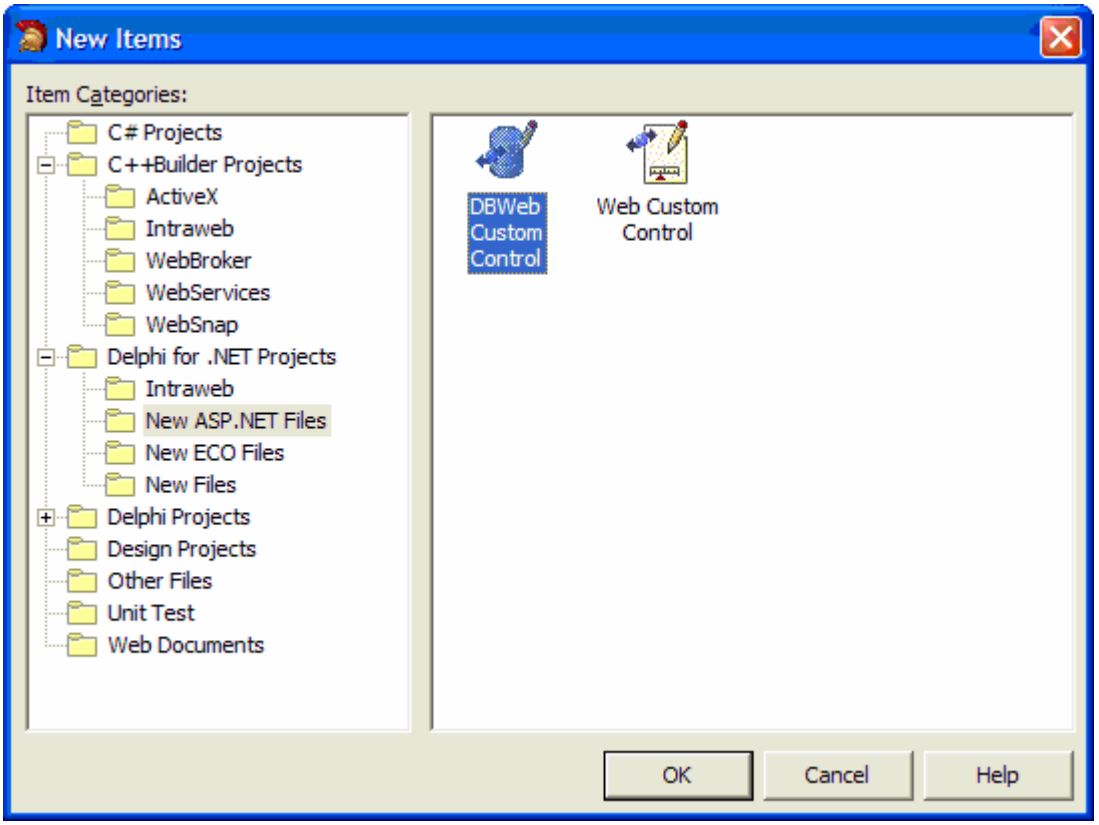
Ik laat in dit artikel zien hoe we Custom ASP.NET DBWeb Web Controls kunnen maken met Delphi 2006. Dit artikel bevat informatie die ik ook tijdens mijn sessie bij de Software Developers Event op 15 september 2006 en tijdens mijn ASP.NET training op 19 oktober 2006 in meer detail zal presenteren

DB Web Controls

ASP.NET DB Web controls zijn te vinden in de Enterprise editie van Delphi for .NET (bijvoorbeeld Delphi 2006, die ik in dit artikel zal gebruiken). Een ASP.NET DB Web control is een beetje te vergelijken met een VCL data-aware control. Via een DataSource kan het visuele component zich voorzien van data, door een tablename en optioneel fieldname property. En waar we vanaf Delphi 1.0 met behulp van een TDataSource de data-aware componenten zoals TDBGrid en TDBEdit kunnen voorzien van data, zo kunnen we in Delphi for .NET met een DBWebDataSource de DBWeb controls zoals de DBWebGrid en DBWebTextBox voorzien van data.

Custom DB Web Control

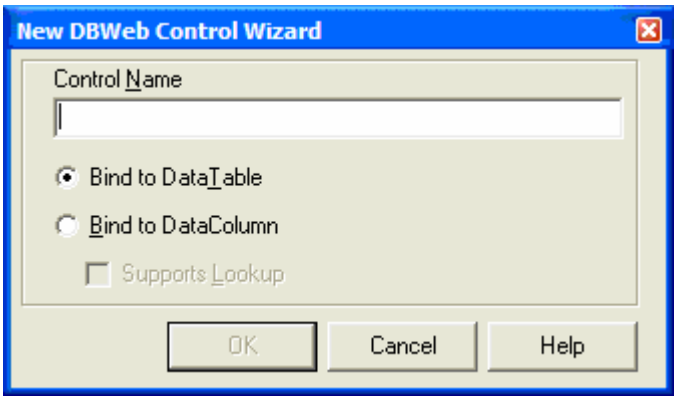
Voor alle .NET custom controls moeten we eerst een Delphi for .NET Package project starten, om uiteindelijk in een .NET Assembly te resulteren. Speciaal voor custom DB Web controls heeft Delphi een DBWeb Custom Control wizard, te vinden in de New ASP.NET Files category in de Object Repository (zei Figuur 1). Deze wizard is overigens ook in Turbo Delphi for .NET terug te vinden (zowel in de Professional als de Explorer editie), maar heeft weinig zin omdat de DB Web assembly zelf geen onderdeel uitmaakt van Turbo Delphi for .NET. Je hebt echt de Enterprise editie van BDS 2006 nodig om met DB Web controls te kunnen werken en/of ze zelf te kunnen maken.



Figuur 1. Object Repositor

Hiermee krijg je de New DBWeb Control Wizard, waar je behalve de naam van de nieuwe control, ook een aantal keuzemogelijkheden hebt, zoals in Figuur 2 (op de volgende pagina) te zien is.

De default keuze geeft je een nieuwe DB Web control die aan een hele DataTable bindt. Dat is iets dat je bij een DataGrid bijvoorbeeld ziet. Het alternatief is een DB Web control die aan een DataColumn (alsmede een DataTable) bindt, zoals een TextBox control. De laatste keuze is ook mogelijk in combinatie met een loopup link (zoals in een DBLookupCombobox).



Figuur 2. New DBWeb Control Wizard

Bij een keuze voor de DataTable zal het nieuwe control een DBDataSource en TableName property hebben, en bij keuze voor de DataColumn zal er behalve de DBDataSource en TableName ook een ColumnName property zijn. Dit wordt afgedwongen door respectievelijk het IDBWebDataLink en IDBWebColumnLink interface uit de Borland.Data.Web namespace.
Tijdens de SDE van 15 september zal ik een aantel verschillende nieuwe DB Web controls bouwen, waaronder een DBWebDataTableInfo die de DataTable binding doet, en een DBWebSpinEdit die de DataColumn binding doet. De DBWebSpinEdit kan overigens op verschillende maniere, en legt nog een klein probleem bloot met de door de New DB Web Control Wizard gegenereerde code. Maar daarover meer tijdens de SDE. Het voorproefje in dit artikel gaat slechts over de DBWebDataTableInfo.

DBWebDataTableInfo

Voor de DBWebDataTableInfo moeten we de keuze op Bind to DataTable laten staan. De gegenereerde unit bevat de class DBWebDataTableInfo, afgeleid van System.Web.UI.WebControls.WebControl, met de interfaces IPostBackDataHandler en IDBWebDataLink. Het IDBWebDataLink interface zorgt dat we de DBDataSource en TableName properties tot onze beschikking hebben.

Delegate

Als we de gegenereerde source code van de DBWebDataTableInfo bekijken, dan valt al snel op dat er een FMyDelegate veld in voorkomt. In commentaar. Het idee is dat we of deze default FMyDelegate gebruiken, of zelf een veld definiëren dat de waarde alsmede (re)presentatie van de waarde voor zijn rekening neemt. De FMyDelegate is standaard een TextBox control. Maar in ons geval wil ik eigenlijk liever een (read-only) Label gebruiken. Dus alle plaatsen waar de FMyDelegate staat moeten we die uit commentaar halen, maar wel van een TextBox in een Label veranderen.
In de class declaratie van de DBWebDataTableInfo is dat als volgt:

```
  strict private
    FMyDelegate: &Label; // BS
```

Vervolgems moeten we in de constructor het FMyDelegate veld als type Label aanmaken, als volgt:

```
constructor DBWebDataTableInfo.Create;
begin
  inherited;
  FMyDelegate := &Label.Create; // BS
  FDataLink := DBWebDataLink.Create(self);
  FIDataLink := (FDataLink as IDBWebDataLink);
end;
```

En in de GetTextt moeten we alleen het gebruik van FMyDelegate uit het commentaar halen.

```
function DBWebDataTableInfo.GetText: string;
var
  sw: StringWriter;
  tw: HtmlTextWriter;
begin
  sw := StringWriter.Create;
  tw := HtmlTextWriter.Create(sw);
  DataBind();
  FMyDelegate.RenderControl(tw); // BS
  Result := sw.ToString();
end;
```

De implementatie van de GetText verzorgt meteen de HTML rendering voor ons. Straks gaan we daar in de DataBind een stukje extra informatie aan toevoegen. Maar eerst nog een laatste stap die noodzakelijk is voor read-only DB Web controls. We moeten voor read-only controls de aanroep naar de RegisterHiddenField in de OnPreRender method in commentaar zetten of helemaal verwijderen. In ons geval is de DBWebDataTableInfo inderdaad read-only, dus moeten we de OnPreRender als volgt aanpassen:

```
procedure DBWebDataTableInfo.OnPreRender(args: EventArgs);
begin
  inherited OnPreRender(args);
  // You need to register the hidden field to identify the key for
  // read-write controls only. Remove this call if control is read-only.
  // Page.RegisterHiddenField(DBWebDataSource.IdentPrefix +
  //   DBWebConst.Splitter + IDataLink.TableName, self.ID);
  DataBind;
end;
```

Waarna we uiteindelijk bij de DataBind aankomen, waar de inhoud van het DB Web control wordt weggeschreven naar de FMyDelegate's Text property. In de DataBind method moeten we de data verzorgen, maar de HTML opmaak wordt over het algemeen in de GetText method gedaan. In dit eenvoudige voorbeeld wijk ik af van die regel, en stop ik de HTML opmaak in de DataBind (het is toch maar een read-only control, en op deze manier is de opmaak sneller gedaan).

DataBind – Analysing DBDataSource

In de DataBind method kunnen we de DBDataSource property analyseren, en en daar dan het resultaat van weergeven. Alleereerst moeten we controleren of de DBDataSource wel ge-assigned is, wat bijvoorbeeld nog niet het geval is als de DBWebDataTableInfo net op een form is gezet. Alleen als de DBWebDataSource property een waarde heeft, kunnen we er gebruik van maken op de GetDataSourceName method aan te roepen. Dit geeft ons de naam van de DBWebDataSource zelf. Daarnaass kunnen we de GetDataSource method aanroepen, die niet de DBWebDataSource, maar juist de DataSet, DataTable of DataView teruggeeft waar de DBWebDataSource mee verbonden is.
Als we eenmaal de DataSource hebben, dan kunnen we kijken of er DataTables in zitten, en daarvan dan de kolommen (de meta data), waarbij we van iedere kolom de ColumnName, DataType en de MaxLength waarde kunnen weergeven. Een soort analyse van de data structuur binnen de tabellen dus.

```
procedure DBWebDataTableInfo.DataBind;
var
  DS: DataSet;
  DT: DataTable;
  DC: DataColumn;
```

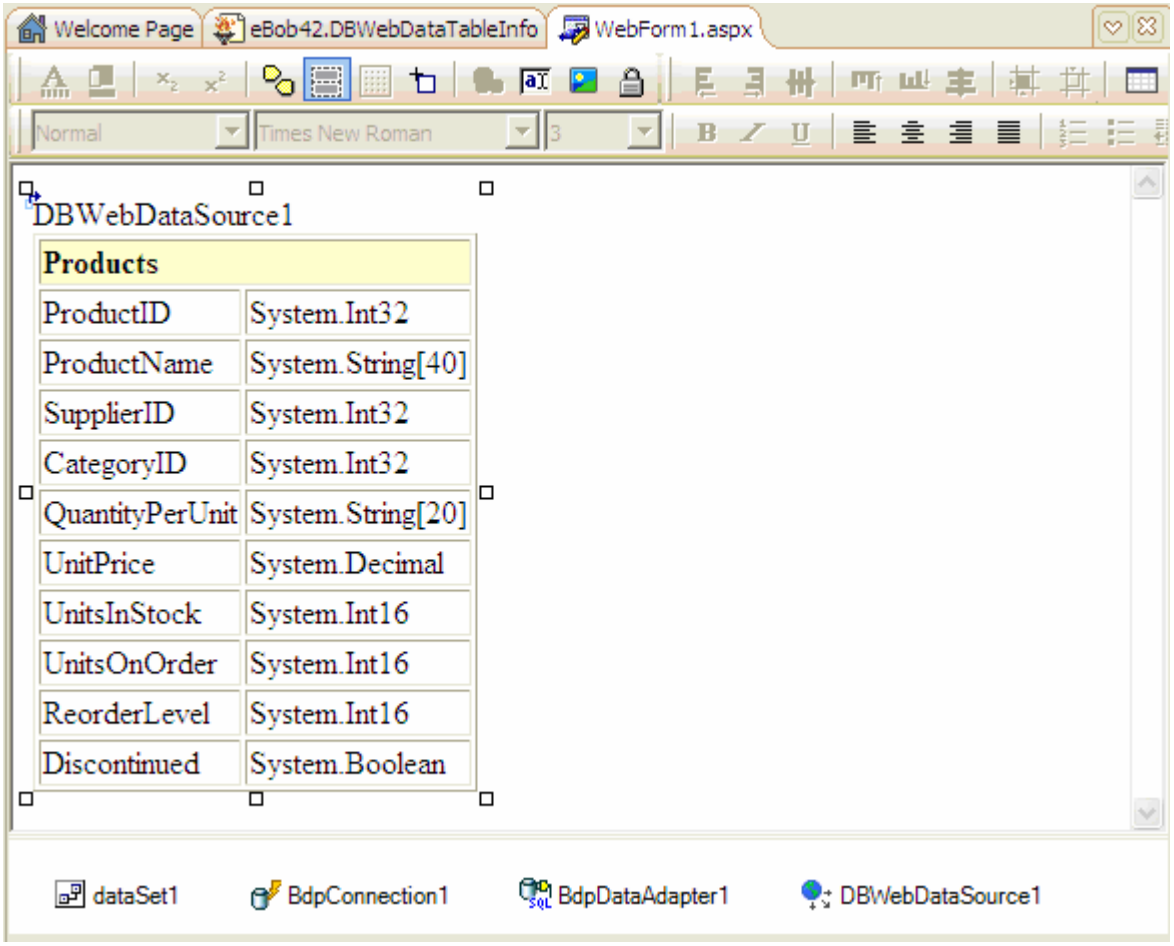
```
begin
  try
    // call to inherited triggers setting of DBDataSource.
    inherited DataBind;

    if Assigned(FIDataLink.DBDataSource) then
    try
      FMyDelegate.Text :=
        FIDataLink.DBDataSource.GetDataSourceName(Page) +
        '<br><table border="1">';
      if FIDataLink.DBDataSource.GetDataSource(Page) is DataSet then
      begin
        DS := FIDataLink.DBDataSource.GetDataSource(Page) as DataSet;
        for DT in DS.Tables do
        begin
          FMyDelegate.Text := FMyDelegate.Text +
            '<tr><td bgcolor=ffffcc colspan=2><b>' + DT.TableName + '</td></tr>';
          for DC in DT.Columns do
            if DC.MaxLength > -1 then
              FMyDelegate.Text := FMyDelegate.Text +
                '<tr><td>' + DC.ColumnName +
                '</td><td>' + DC.DataType.ToString +
                '[' + DC.MaxLength.ToString + ']</td></tr>'
            else
              FMyDelegate.Text := FMyDelegate.Text +
                '<tr><td>' + DC.ColumnName + '</td><td>' +
                DC.DataType.ToString + '</td></tr>'
        end
      end
      else
      begin
        if FIDataLink.DBDataSource.GetDataSource(Page) is DataView then
          DT := (FIDataLink.DBDataSource.GetDataSource(Page) as DataView).Table
        else
          if FIDataLink.DBDataSource.GetDataSource(Page) is DataTable then
            DT := FIDataLink.DBDataSource.GetDataSource(Page) as DataTable
          else DT := nil;

        if Assigned(DT) then // DataTable
        begin
          FMyDelegate.Text := FMyDelegate.Text +
            '<tr><td bgcolor=ffffcc colspan=2><b>' + DT.TableName + '</td></tr>';
          for DC in DT.Columns do
            if DC.MaxLength > -1 then
              FMyDelegate.Text := FMyDelegate.Text +
                '<tr><td>' + DC.ColumnName +
                '</td><td>' + DC.DataType.ToString +
                '[' + DC.MaxLength.ToString + ']</td></tr>'
            else
              FMyDelegate.Text := FMyDelegate.Text +
                '<tr><td>' + DC.ColumnName +
                '</td><td>' + DC.DataType.ToString + '</td></tr>'
        end
      end;
      FMyDelegate.Text := FMyDelegate.Text + '</table>'
    except
      on E: Exception do
        FMyDelegate.Text := FMyDelegate.Text + '<br>' + E.Message + '</table>'
    end
    else
      FMyDelegate.Text := 'DBWebDataTableInfo'
```

```
    except
      on E: Exception do
      begin
        if (not ClassUtils.IsDesignTime(Page)) then
          Page.Response.Write(ClassUtils.GetInternalError(Page, FIDataLink, E, Self.ID))
      else
        Raise Exception(E);
    end;
  end;
```

Merk op dat de DBWebDataSource aan zowel een DataSet, DataTable als DataView verbonden kan zijn, en de code moet dan ook met alledrie de situaties rekening houden.
Het resultaat is te zien in Figuur 3, waarin de DBWebDataTableInfo tijdens design-time te zien is als een HTML table, verbonden met de DBWebDataSource1 (de naam boven de table), die op zijn beurt is verbonden met een DataSet met daarin de Products table, die de velden ProductID, ProductName, etc. heeft. Van ieder veld is de naam en het type te zien. Op die manier kunnen we zowel tijdens design-time als tijdens run-time de meta data van een DataTable verbonden met de DBWebDataSource zien.



Figuur 3. DBWebDataTableInfo

Tijdens de sessie op de SDE zal ik behalve de DBWebTableInfo ook een wat uitgebreide DBWebSpinEdit laten zien. Op twee verschillende manieren overigens, waarvan eentje leidt tot het vinden van een bug in de code die de New DBWeb Control Wizard genereerd – plus een oplossing uiteraard.
Daarnaast zal ik tijdens deze sessie op de SDE van 15 september ook ingaan op de deployment details.

Wie (nog) meer wil weten van DB Web controls is welkom tijdens mijn Delphi 2005 ASP.NET 1.1 Web Development training van 19 oktober of de Advanced ASP.NET training op 30 november.