# Delphi OplossingsCourant

**Vol. 7. No. 1,** Een gratis electronische publicatie van **Bob Swart Training & Consultancy** (**eBob42**) - http://www.eBob42.com

Helmond, 9 januari 2005,

Welkom bij het nieuwe nummer van de Delphi OplossingsCourant; exact een jaar geleden sinds het vorige nummer. In plaats van de nodige rust heeft de kerstvakantie helaas voor veel emotie en chaos gezorgd. Desalnietemin (of juist daarom) wens ik iedereen graag een gelukkig en vooral gezond 2005.

Waar het vorige nummer van de Delphi Oplossings-Courant nog in het teken stond van Delphi 8 for .NET daar gaan we in dit nummer in op de opvolger van zowel Delphi 7 als Delphi 8 for .NET, namelijk Delphi 2005. De Diamondback previews werden vertoond vanaf september, de aankondiging van Delphi 2005 volgde op 12 oktober, en Delphi 2005 het kwam uiteindelijk beschikbaar in november met een eerste update al in december. En nu is dan het moment aangebroken dat ik verwacht dat een hoop ontwikkelaars Delphi 2005 zullen oppakken. Wie Delphi 8 for .NET vorig jaar heeft overgeslagen (en liever wou wachten op een meer volwassen .NET ontwikkelomgeving) die kan nu met Delphi 2005 aan de slag voor een eerste kennismaking met .NET. Maar ook wie nog bestaande Win32 projecten heeft en daar voorlopig nog niet vanaf is, kan met Delphi 2005 uit de voeten.
Na de inhoud van mijn nieuwe Delphi Clinics, begin ik dit nummer met nieuws over Delphi updates, Kylix en C++Builder (die we waarschijnlijk als C++ personality in de volgende Delphi IDE terug zullen zien), alsmede Chrome – een Object Pascal achtige compiler en Visual Studio.NET plugin.

Veel leesplezier, en vergeet niet om de websites http://www.drbob42.com en http://ww.eBob42.com in de gaten te houden voor het laatste nieuws.
Wie na lezen van de Delphi OplossingsCourant nog opmerkingen heeft, kan gerust een mailtje sturen naar Bob@eBob42.com- ik stel alle feedback op prijs!

Groetjes,

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

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

## Nieuwe Delphi Clinics in 2005

Het uitkomen van Delphi 2005 betekent ook weer een uitbreiding van het cursusaanbod. Vorig jaar nog werd mijn Delphi 8 for .NET cursusmateriaal door Borland zelf in licentie gebruikt voor alle officiële Borland Delphi trainingen wereldwijd (en daar ben ik best wel een beetje trots op). Het betreft de Delphi 8 Essentials en Delphi 8 ASP.NET Essentials courseware.
Voor Delphi 2005 heb ik het aanbod uitgebreid tot een zestal cursusboeken (inclusief eentje over Delphi 2005 met IntraWeb). De onderwerpen zijn nu zowel Win32 als .NET, en ook al gebruik ik Delphi 2005 in de cursusboeken, bijna alle Win32 onderwerpen zijn ook nog relevant voor Delphi 7, en dat geldt ook voor de meeste .NET onderwerpen en Delphi 8 (met uitzondering van de nieuwe features in Delphi 2005 uiteraard). De nieuwe Delphi cursusonderwerpen die begin 2005 beschikbaar komen zijn als volgt:

**I. Delphi 2005 Development Essentials**
http://www.eBob42.com/training/2005.htm

**II. VCL Database Development Essentials**
http://www.eBob42.com/training/20050127.htm

**III. ADO.NET Database Development Essentials**
http://www.eBob42.com/training/20050210.htm

**IV. ASP.NET Web Development Essentials**
http://www.eBob42.com/training/20050224.htm

**V. XML, SOAP and Web Services Essentials**
http://www.eBob42.com/training/20050310.htm

**VI. IntraWeb 7 Web Development Essentials**
http://www.eBob42.com/training/20050224.htm

De reguliere trainingen zijn niet hands-on, maar de onderwerpen zijn ook als workshop (bij mij thuis in Helmond) of Custom Clinic (bij u op lokatie) te organiseren, en dan kunt u de agenda en gewenste onderwerpen van de dag(en) zelf samenstellen.
Zie http://www.eBob42.com/workshop voor details.

De hoofdstukindeling van de nieuwe Delphi 2005 cursusboeken is als volgt (III en IV ook in C# editie):

### I. Delphi 2005 Development Essentials
1. Intro Microsoft .NET Framework
2. Delphi 2005 IDE
3. Refactoring
4. Language Enhancements
5. Windows Forms Applications
6. VCL and VCL for .NET
7. Migrating Win32 Apps to .NET
8. Unit Testing
9. Interoperability with .NET
10. Win32 Interoperability

### II. VCL Database Development Essentials
1. Relational Databases and SQL
2. Borland Database Engine (BDE)
3. dbGo for ADO (ADOExpress)
4. TClientDataSet
5. Crossplatform: dbExpress
6. Win32 Multi-Tier: DataSnap

### III. ADO.NET Database Development Essentials
1. Relational Databases and SQL
2. .NET Data Access: ADO.NET
3. Borland Data Explorer
4. Borland Data Provider for .NET
5. BDP and .NET Remoting
6. Enterprise Core Objects II

### IV. ASP.NET Web Development Essentials
1. Introduction ASP.NET 1.1
2. ASP.NET Web Forms Applications
3. User and Custom Controls
4. ASP.NET and Data Access (ADO.NET and BDP)
5. ASP.NET and ECO II

### V. XML, SOAP and Web Services Essentials
1. Win32 XML Programming
2. XML in Delphi 2005 for .NET
3. ASP.NET Web Services
4. Web Services Migration from Win32 to .NET
5. Web Services Interoperability Win32 and .NET
6. ASP.NET Web Services and DataSets
7. Intro .NET Remoting

### VI. IntraWeb 7 Web Development Essentials
1. IntraWeb Page Mode
2. IntraWeb Application Mode
3. IntraWeb State Management
4. IntraWeb Database Example
5. IntraWeb Components
6. IntraWeb "3.2" Components
7. IntraWeb Custom Components
8. IntraWeb for .NET vs. ASP.NET

Wie liever zelf een curriculum met onderwerpen wil samenstellen, heeft de mogelijkheid om mij een maatwerk training / workshop te laten organiseren. Dat kan bij mij thuis (voor twee tot maximaal drie personen), maar ik kan ook op bezoek komen.
Zie http://www.eBob42.com/workshop voor details.

Behalve de Delphi Clinics ben ik de eerste drie maanden van 2005 ook te vinden op de HCC Pascal/Delphi Gebruikersdag op 19 maart in IJsselstein, alsmede op 1 april bij een SDE van de Software Developer Network in Ede/Wageningen.

### HCC Pascal/Delphi Gebruikersgroep
Op zaterdag 19 maart behandel ik Delphi 2005, en in het bijzonder de IDE, Refactoring en Unit Testing, en de nieuwe taalelementen die we in Delphi voor zowel Win32 als .NET kunnen gebruiken. Zie ook de agenda op http://www.hcc-pgg.nl voor details.

### Software Developers Network Event
Op vrijdag 1 april hou ik voor de SDN een sessie over Delphi 2005 en .NET Remoting met de nieuwe Borland Data Provider componenten RemoteServer en RemoteConnection in combinatie met DataSync en DataHub.
Zie http://www.sdn.nl voor meer informatie.

## Delphi 7 Update #1
De update van Delphi 7 is te downloaden van de Borland website (voor geregistreerde Delphi 7 gebruikers). Er zijn meer dan 100 bugs opgelost, dus het is zeer de moeite waard! Er waren echter wat problemen met deze update, dus op 11 juni 2004 is een fix voor de update uitgekomen, gevolgd door een database supplemental in augustus.
http://bdn.borland.com/article/0,1410,32400,00.html

## Delphi 8 Update #3
Update 1.1 (SP1) van het .NET Framework 1.1 heeft gevolgen voor de Delphi 8 for .NET compiler. Zo erg zelfs dat lange tijd het advies was om .NET 1.1 SP1 maar niet te installeren (helaas had niet iedereen die mogelijkheid).
Vlak voor Kerst 2004 heeft Borland gelukkig een public beta van Delphi 8 Update #3 uitgebracht die het probleem veroorzaakt door .NET 1.1 SP1 oplost.
http://bdn.borland.com/article/0,1410,32873,00.html

## Delphi 2005 Update #1
De eerste update voor Delphi 2005 is vlak voor Kerst 2004 beschikbaar gekomen voor geregistreerde Delphi 2005 gebruikers. Deze update is voor Delphi 2005 Architect, Enterprise en Professional.
http://bdn.borland.com/article/0,1410,32875,00.html

# Kylix

Tijdens de "Meet The Delphi Team" sessie bij de Borland Conference in San Jose 2004, begon Michael Swindell met de aankondiging van het Kylix Community project. Het doel van dit project is het uitbrengen van verbeteringen in de Kylix drivers (het meer compatible maken met de nieuwste distributies van Linux) alsmede het werken aan CLX. Er zullen geen nieuwe features komen, maar we kunnen wel betere compatibiliteit met moderne versies van Linux verwachten.
Zie http://freeclx.sourceforge.net/ voor meer details.

## Kylix Exec-Shield Fix

Als een van de eerste resultaten van het Kylix Community project is er een Kylix Exec-Shield Fix uitgebracht door Simon Kissel. Deze fix voorkomt dat Linux toepassingen gecompileerd met Kylix crashen op Linux distributies met de 2.6.8 kernel (plus de exec-shield patch), zoals Red Hat Fedora Core 2.
http://crosskylix.untergrund.net/execshield.shtml

# CrossKylix

Behalve met de Kylix Exec Shield Fix is Simon Kissel is in 2004 ook druk bezig geweest met CrossKylix: een toolkit en Delphi 7 IDE plugin voor de Borland Kylix command-line compiler. Het effect hiervan is dat ik vanuit de Delphi 7 IDE een CLX toepassing (GUI zowel als web – dus ook WebBroker, WebSnap en Web Services) kan compileren tot een Windows executable, maar ook met de CrossKylix plugin tot een Linux executable. Die hoef ik dan alleen nog maar te deployen op een Linux machine. Uiteraard is er (nog) geen debugger support, en is de Kylix CD nodig tijdens installatie van CrossKylix (zodat je dus een volledige Delphi alsmede Kylix licentie moet hebben – alhoewel het ook met de trial edities werkt).
CrossKylix biedt mij in staat om toepassingen te bouwen in Delphi 7 onder Windows, en die alsnog als Linux executables te compileren en deployen. Het is gratis, en ik vind het geweldig!
Zie http://crosskylix.untergrund.net/ voor details.

# C++Builder in Delphi

Op 14 december 2005 publiceerde Borland een "Open Letter" voor Borland C++ ontwikkelaars waarin de toekomst van C++Builder (en VCL voor C++) aan de orde kwam. De open letter is te lezen op http://bdn.borland.com/article/0,1410,32845,00.html en belooft min of meer dat de volgende versie van Borland C++Builder deel zal uitmaken van de Delphi IDe familie (waarschijnlijk in de vorm van een C++ persoonlijkheid).

# RemObjects Chrome

Behalve Delphi spin-offs zoals Kylix (Delphi voor Linux) en C++Builder (Delphi voor C++) is er het afgelopen jaar ook een andere Object Pascal speler gekomen. Onder de naam Chrome werken de mensen van RemObjects Software aan een Object Pascal achtige compiler voor .NET en Mono. Deze zal zowel in de vorm van een (gratis) command-line compiler als in de vorm van een Visual Studio.NET plugin beschikbaar komen. Chrome is gebasseerd op Object Pascal met eigen uitbreidingen, waardoor het wel erg lijkt op de Delphi programmeertaal, maar op sommige punten toch (soms behoorlijk) verschilt. Zo zijn er eigen regels voor namespaces, technieken voor het inline declareren van variabelen (binnen een for-loop bijvoorbeeld), en nog veel meer. Zie http://www.chromesville.com of http://www.drbob42.com/chrome voor meer info.

## Chrome Preview

Op dit moment is het mogelijk om een gratis "januari 2005" preview versie van de Chrome command-line compiler voor .NET 1.1/2.0 of voor Mono te downloaden van de Chrome website te http://www.chromesville.com/downloads
Het is nog een beta, en er kunnen nog zowel nieuwe features bijkomen als wegvallen, maar je kan er al hele leuke dingen mee doen.

## Chrome Artikelen

Voor een eerste kennismaking van Chrome is het lezen van het artikel "Introducing the Chrome Command Line Compiler" aan te raden dat te vinden is te http://www.chromesville.com/?ch01.

## Chrome Reflector Plugin

Chrome zou zich niet als volwaardige .NET taal kunnen zien als er geen Reflector plugin voor zou zijn, en dat is nu dan ook het geval. Net als Chrome zelf is de Reflector plugin nog een beta, maar het is nu mogelijk om .NET IL code te converteren naar Chrome source code!
Zie http://www.chromesville.com/downloads

# Chrome Trainingen

Ik zie Chrome niet direct als vervanger van Delphi, maar meer als een welkome uitbreiding van de mogelijkheden voor de Object Pascal ontwikkelaar. Om die reden ben ik bezig om een aantal specifieke .NET trainingsonderwerpen en cursusboeken (zoals ADO.NET en ASP.NET) te schrijven voor Chrome. Pas na het definitief beschikbaar komen van Chrome zal dit trainingsmateriaal beschikbaar komen – zie http://www.eBob42.com/training tegen die tijd voor meer details.

# Borland Delphi 2005

*Op 12 oktober 2004 was het zover: Borland kondigde Delphi 2005 aan. Dit is zowel de opvolger van Delphi 7 (voor Win32) als Delphi 8 (voor .NET), maar ook van C#Builder. De Delphi 2005 ontwikkelomgeving beschikt namelijk over de mogelijkheid om drie verschillende compilers te gebruiken. Twee voor Delphi, en een – van Microsoft –voor C#. Dat is overigens niet de eerste keer voor Borland, want ook Borland Pascal 7 beschikte al over de mogelijkheid om zowel een DOS, DPMI als Windows compiler te gebruiken. Daarbij moet worden aangetekend dat Borland Pascal 7 destijds natuurlijk nog geen RAD designers bevatte, en het dus puur de code editor en compiler was (iets eenvoudiger dan de verschillende designers die nu in Delphi 2005 zitten).*

## Gespleten Persoonlijkheid?

Delphi 2005 als gespleten persoonlijkheid? Je zou er haast hoofdpijn van krijgen, en het is me overkomen dat ik met een project bezig ben, en niet meer 100% zeker wist of ik nu voor Win32 of .NET bezig was. Soms is dat duidelijk, bijvoorbeeld als je met een ASP.NET toepassing bezig bent, maar soms gebruik ik VCL of schrijf een console toepassing, en eerlijk gezegd is het dan niet 100% duidelijk. Dat gevoel kun je helemaal krijgen als je een bestaand project opent: "levert die VCL toepassing nu een Win32 of een .NET toepassing op?"
Als kleine hint is er in de menubalk van Delphi (rechts van het menu) een icoontje zichtbaar met daarin de representatie van de gekozen persoonlijkheid van Delphi 2005. Standaard is dat de gouden helm – het BDS symbool. Voor een Delphi Win32 toepassing is het de bronzen helm met een klein windows logo in de rechteronderhoek. Voor Delphi for .NET is het een ijzeren helm, en voor C# is het het oude C#Builder icoon. De ijzertijd volgt dus op de bronstijd als het ware.

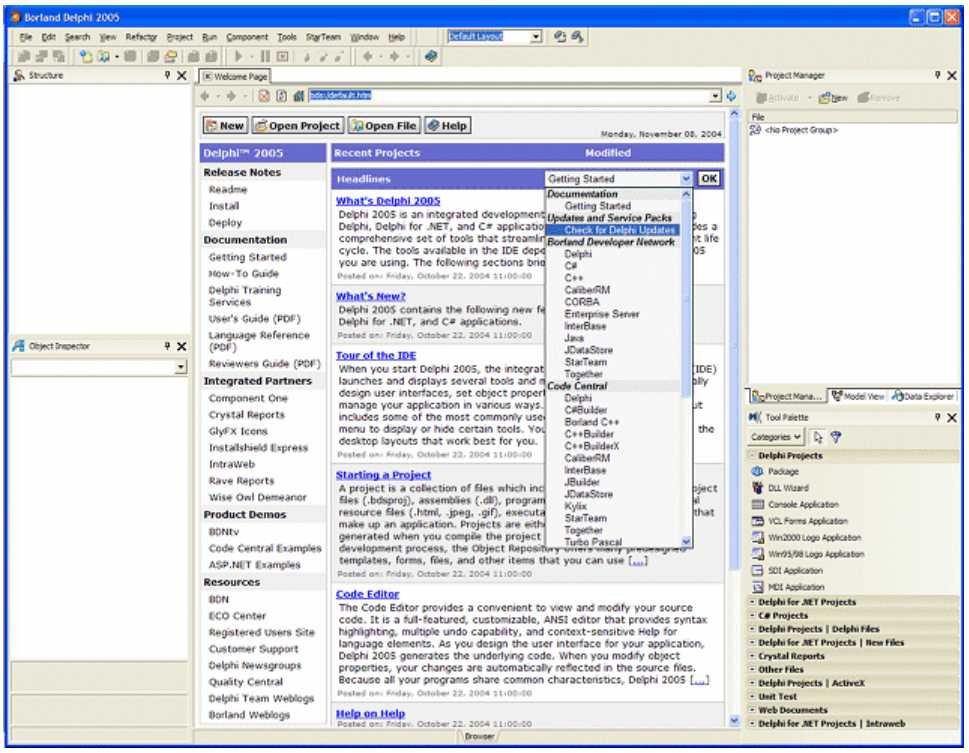
| Icon | Persoonlijkheid |
|---|---|
|  | Delphi 2005 - Borland Developer Studio |
|  | Delphi Win32 |
|  | Delphi for .NET |
|  | C# |

Table 1. Delphi 2005 "persoonlijkheden"

Wie het idee van ijzer (.NET) als moderner dan brons (Win32) niet zo ziet zitten, kan het ook zien als goud (Win32) en zilver (.NET) - maar dat laat ik aan de lezer over.
Op het moment van schrijven is de 30-dage trial versie van de Architect versie van Delphi 2005 net beschikbaar gekomen. In de readme die je bij installatie krijgt wordt overigens ook verwezen naar een Personal editie van Delphi 2005 (voor niet commerciële doeleinden), die wel langer dan 30 dagen zal werken. Het is nog niet bekend welke features er allemaal in de Personal editie zullen zitten, waardoor ik me in dit nummer van de Delphi OplossingsCourant zal beperken tot algemene zaken in de IDE (zoals Refactoring en Unit Testing) en enkele nieuwe taaluitbreidingen voor zowel Win32 als .NET.

### Welcome Page

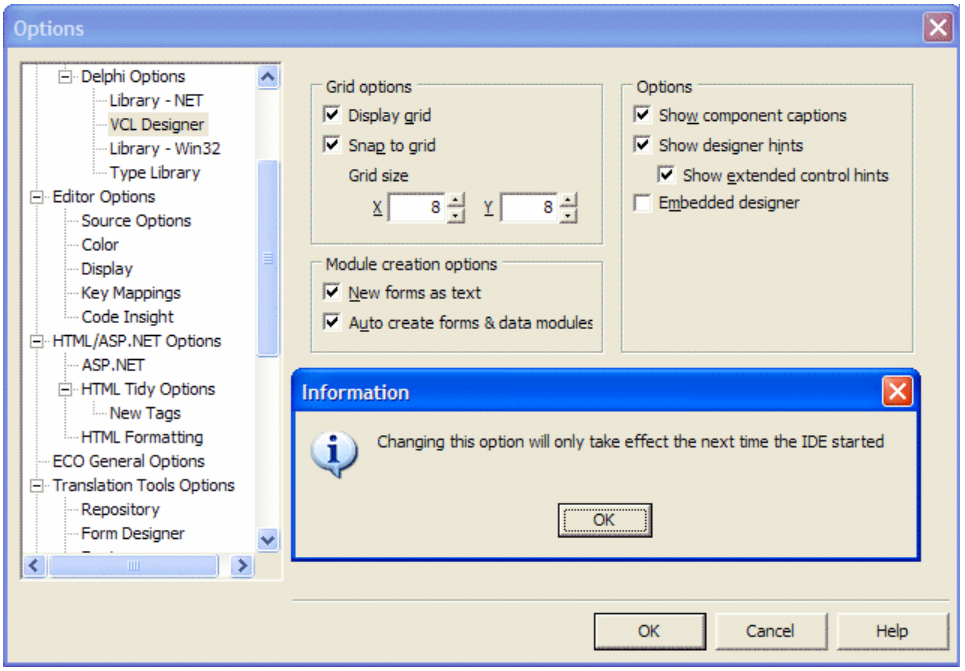
Bij het osptarten van Delphi 2005 krijgen we de Welcome Page (zie figuur 1). Deze bevat niet alleen een overzicht van de laatst geopende projecten, maar ook een lijst met nuttige links naar plaatsen op de Borland website met meer informatie (inclusief een link naar de Reviewers Guide – een document van 121 pagina's met meer informatie dan ik in deze paar pagina's kan vullen). De Welcome Page is met name nuttig als je een internet verbinding hebt, want je kan een keuze maken uit verschillende zgn. RSS-kanalen voor de laatste nieuwtjes van Delphi (bijvoorbeeld de nieuwste updates!) of de artikelen op BDN, of the weblogs van de Borland Delphi teamleden. Erg leuk om te zien, en iedere keer als je Delphi 2005 start wordt dan de RSS opnieuw voor je opgehaald (zodat je iedere keer kunt zien of er bijvoorbeeld weer een nieuwe update beschikbaar is).




*Figuur 1. Welcome Page*

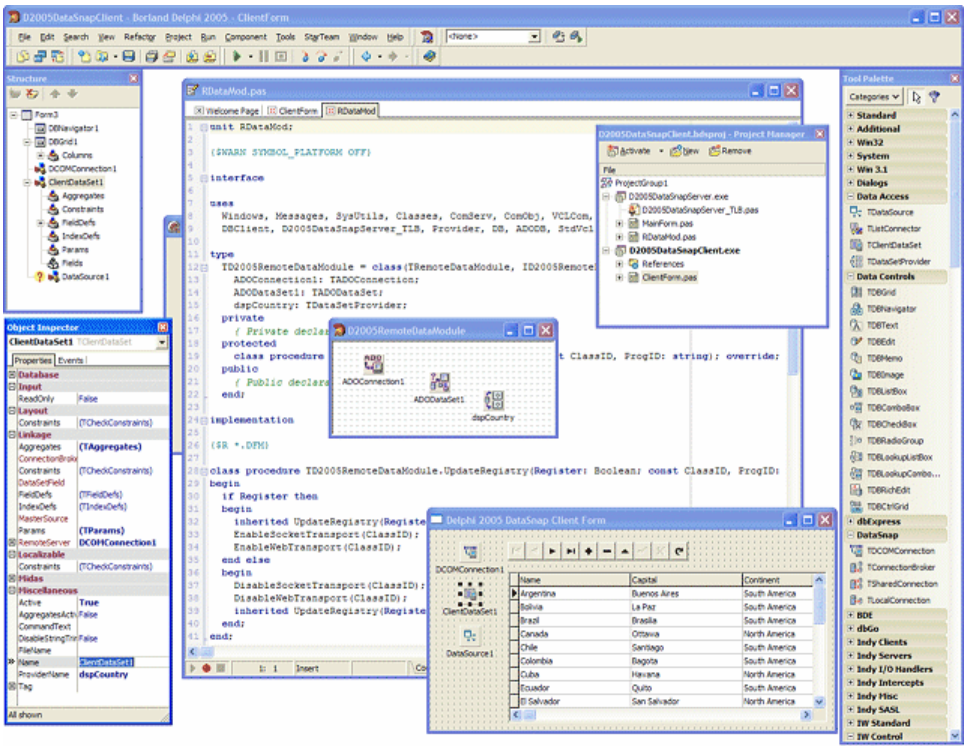
## VCL Designer

Delphi gebruikers zijn altijd gewend geweest om een zogenaamde "floating designer" te hebben. Met Delphi 8 for .NET besloot Borland helaas hiervan af te stappen, alhoewel achteraf bleek dat er een ongedocumenteerde registry setting was om alsnog een soort floating designer voor VCL toepassingen te krijgen. Gelukkig heeft Borland besloten om deze optie in Delphi 2005 nu ook via de reguliere Tools Options dialoog beschikbaar te maken. Nog steeds alleen voor VCL toepassingen, en zoals te zien is in de screenshot van figuur 2 moet je heel Delphi 2005 opnieuw starten, maar dan heb je weer de oude floating designers terug.



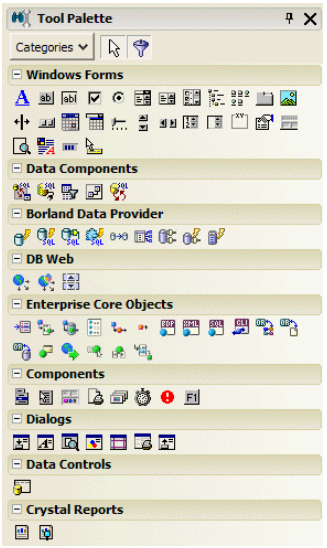

*Figuur 2. No Embedded Designer*

In figuur 3 is een voorbeeld te zien van meerdere Delphi VCL forms alsmede een data module op de floating manier.



*Figuur 3. Floating Form en Data Module*

Let ook op de Tool Palette helemaal aan de rechterkant van de IDE. Waar we vroeger het Component Palette hadden (een regel met vele tabs vol componenten), daar hebben we nu een verticale verzameling categorieën gevuld met componenten. Er is echter wel een nadeel aan de oude manier van werken, want zodra je een component category openklapt krijg je meteen meerdere regels met componenten te zien - op iedere regel zowel de icoon als de naam van het component. Dit kun je instellen op alleen de icoontjes, en dan kun je ze bijna allemaal zien (vooral bij WinForms toepassingen, zie figuur 4).




*Figuur 4. WinForms Componenten*

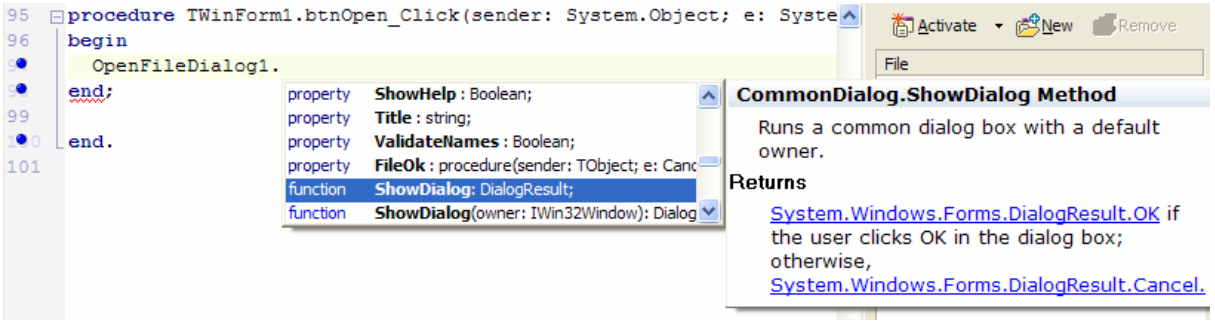
Los daarvan, kun je wel snel een component vinden door de eerste letters van zijn naam in te tikken (bijvoorbeeld "Da" om alle componenten te zien waarvan de naam met "Da" begint - of "TDa", want die T krijg je kado).

# Code Editor

Behalve de IDE, zijn er ook heel wat verbeteringen in de code editor terug te vinden, zoals Help Insight, Error Insight, Sync Edit en Refactoring.

## Help Insight

We kennen allemaal Code Insight, die op verschillende momenten hulpvaardige informatie laat zien in de IDE. Zo is er class completion, die na de naam van een object (class instantie) en een "." (punt) een lijst met mogelijke properties of methodes laat zien. Delphi 2005 biedt daarnaast nu ook Help Insight, die een pop-up window extra informatie laat zien. Bijvoorbeeld voor de elementen in een Code Insight window, zoals in figuur 5 te zien, maar ook als je je muiscursor boven een identifier laat rusten: dan krijg je de plek te zien waar deze identifier (een type, instantie of methode bijvoorbeeld) is gedefinieerd. Erg handig in het dagelijks gebruik, vooral bij die properties en methods waar ik niet altijd uit m'n hoofd de juiste betekenis van weet.

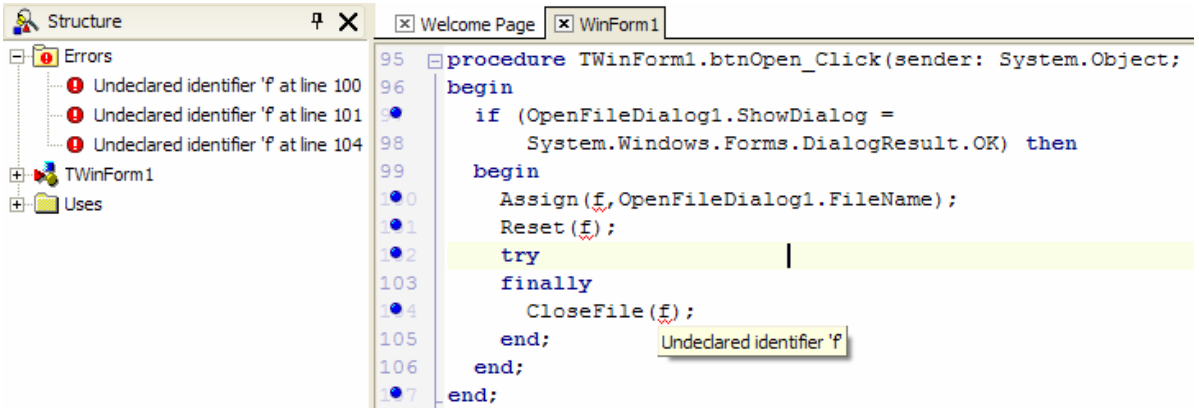



*Figuur 5. Help Insight*

De hyperlinks in de Help Insight pop-up windows kun je ook weer gebruiken voor nog meer (achtergrond)informatie.

## Error Insight

Een andere nieuwe feature doet mij een beetje aan MS Office denken, en dan met name de rode "friemeltjes" die Word altijd onder mijn spelfouten neerzet. Delphi 2005 laat nu vergelijkbare rode golfjes zien onder wat Delphi denkt dat syntax fouten zijn. En dat kan van alles zijn: haakjes die niet gesloten zijn, routines die onbekend zijn, of variabelen die wel gebruikt worden maar nog niet gedeclareerd zijn, zoals in figuur 6 te zien.




*Figuur 6. Error Insight*

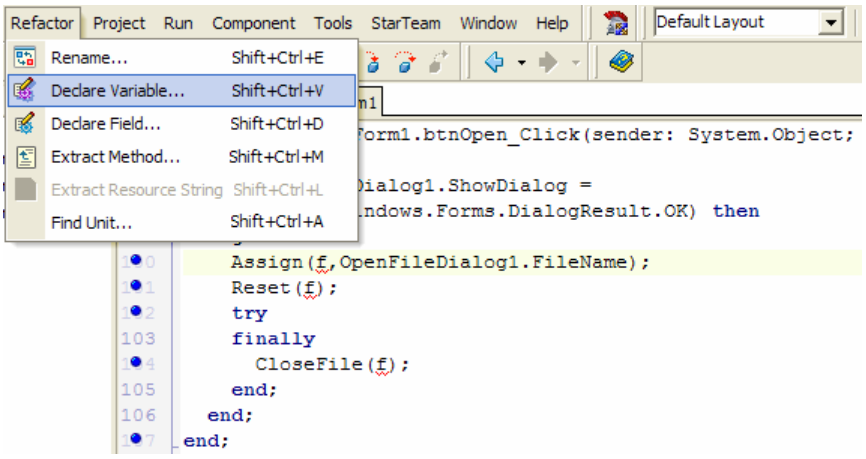
Uiteraard is dit probleem snel op te lossen door een "var f: TextFile;" toe te voegen aan de betreffende routine. Maar soms zit je erg diep in een lange routine, en dan heb je niet altijd zin om helemaal naar boven te scrollen om daar de declartie neer te zetten, en dan weer helemaal naar beneden om verder te gaan met de code waar je bezig was. Ik ben dan regelmatig de draad kwijt, en wil dus eigenlijk doorgaan met het schrijven van code zonder "uitstapjes" te hoeven maken.

## Refactoring

Voor het declareren van variabelen of class velden kun je nu in Delphi 2005 ook gebruik maken van Refactoring. Voor wie niet weet wat Refactoring is: het is de naam van het proces waarin je de source code (her)structureerd, zonder daarbij de werking van de source code zelf te wijzigen. Je bent dus niet nieuwe features aan het toevoegen, maar alleen de structuur, leesbaarheid en vaak ook de onderhoudbaarheid van je source code aan het verbeteren. Althans, dat is de formele definitie van Refactoring.

Er zijn zes verschillende soorten Refactoring in Delphi 2005, die onder te verdelen zijn in Refactoring voor nieuwe code (die helpen dus eigenlijk wel ondersteuning bieden bij het schrijven van nieuwe code), en Refactoring van bestaande code (die met name helpt bij het onderhouden van oude code).

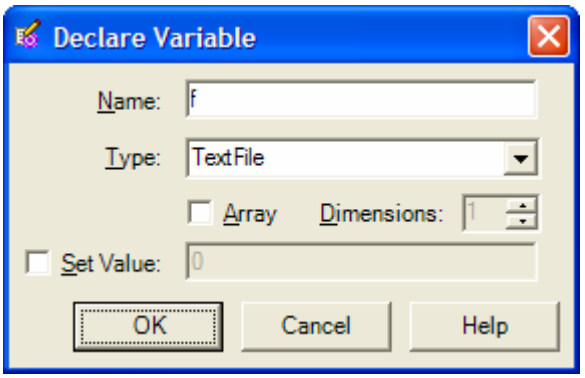
Voor nieuwe code kennen we de Declare Variable, Declare Field en Find Unit. Voor bestaande code kunnen we de Rename, Extract Method en Extract Resource String gebruiken (zie figuur 7).




*Figuur 7. Refactoring*

In ons voorbeeld - te zien in zowel figuur 6 als 7 - hebben we een variabele "f" die door Error Insight als "nog niet gedefinieerd" wordt aangegeven, en kennelijk als TextFile gedeclareerd moet worden. We kunnen dit doen door Shift+Ctrl+V in te tikken, of door het Refactor menu te kiezen (zowel uit het hoofdmenu als via de rechtermuisknop) en daarvan de Declare Variable optie.

In de dialoog die volgt (zie figuur 8) staat de naam al ingevuld, en kunnen we dan het type kiezen. Vaak is het juiste type al geselecteerd (met name bij Integers kan dat goed voorspeld worden door Delphi), maar soms moet je zelf het type kiezen of zelf intikken.




*Figuur 8. Declare Variable*

Een kleine tip: de editbox voor de naam is niet read-only, maar het is niet de bedoeling dat je hier een andere naam intikt dan de naam die al ingevuld is. Anders zal wel de variabele declaratie met de nieuwe naam plaatsvinden, maar blijft de oude variabele in de code (die nog niet gedeclareerd was) gewoon zijn oude naam houden. En die is dan dus nog steeds niet gedeclareerd. Voor het veranderen van naam kun je de Rename Refactoring gebruiken.

Als laaste van de "nieuwe code" Refactoring technieken is er de Find Unit, die je kan gebruiken om van een bepaalde class of type de unit te vinden waar deze in gedefinieerd is, en deze unit dan automatisch aan de uses clause van de interface of implementatie toe te voegen. Erg handig, alhoewel het zoeken in praktijk nog niet helemaal foutloos werkt.
De Rename mogelijkheid van Refactoring lijkt misschien een beetje op een normale "search en replace", maar is er dan wel eentje die het hele project door gaat, en daarbij slim te werk gaat door niet alleen syntactisch, maar juist ook semantisch te kijken of ieder voorkomen van het betreffende woord ook echt de identifier is die van naam veranderd moet worden. Veel veiliger dus dan zomaar een "search en replace", zeker omdat ook de inhoud van de .dfm of .nfm wordt meegenomen!
De Extract Method Refactoring is de meest complexe van allemaal: via deze techniek kun je een stuk code selecteren dat uit het huidige blok wordt gehaald en in een eigen routine wordt neergezet. Inclusief het overhalen van argumenten die nodig zijn voor de aanroep en variabelen die alleen nog maar lokaal nodig zijn. Dit is perfect voor het herstructureren van routines die enkele pagina's lang zijn (ik heb ze gezien, echt waar!). Het volgende artikel in deze Delphi OplossingsCourant komt hier in detail op terug, en laat zien hoe we Refactoring in praktijk kunnen gebruiken voor het onderhoud (of aanpassing) van een bestaand project – ook (of misschien wel juist) indien je daar zelf niet de oorsponkelijke auteur van was.
De laaste Refactoring is de Extract Resource String, die strings kan omzetten naar resource strings (die makkelijk te vertalen zijn indien gewenst). Handig in gebruik, en het schoont flink op.

### Unit Tests

Refactoring was al bekend voordat het in Delphi 2005 werd geïntroduceerd (er zijn hele boeken beschikbaar met tientallen technieken daarin beschreven - Delphi 2005 biedt er een zestal aan op dit moment). Een andere feature die al in praktijk gebruikt wordt, maar nu in de Delphi 2005 IDE geïntegreerd is, is Unit Testing. Dit is een fenomeen dat uit de Extreme Programming (XP) hoek afkomstig is. Het idee is dat je niet tot het eind van je project wacht met het testen of het schrijven van tests, maar dat je dat van het begin af aan doet (sommige extreme XP aanhangers beweren zelfs dat je al moet beginnen de test te ontwerpen voordat je de eerste regels code schrijft - maar dat gaat de meeste mensen toch een beetje te ver). De softwarematige ondersteuning voor Unit Testing begon destijds met JUnit - een framework voor Java. De Delphi versie heet DUnit, en is er inmiddels ook in een Delphi for .NET versie van DUnit. Daarnaast is er ook een taalneutrale .NET versie, NUnit genaamd. Ze hebben allemaal gemeen dat je een test-project maakt naast je eigenlijke project, en dat het test-project wordt gebruikt om de units uit je daadwerkelijke project te testen. En dat met name de classes en de publieke methoden uit die classes. Het is de bedoeling dat je test cases schrijft: routines die de aanroepen van de te testen methoden in source code vastleggen. Per testcase kun je dan controleren of het resultaat is wat je verwacht, en anders wordt er een unit testing error gegenereerd. Alle test cases worden dan automatisch allemaal vanuit je testprogramma aangeroepen, en als alles goed verloopt zie je groen licht, maar voor iedere foute test case krijg je een duidelijke melding welke test er gefaald heeft.
Het grootste voordeel van het gebruik van Unit Testing is het feit dat je je tests kan blijven ontwikkelen terwijl de code in het project zelf ook verder doorontwikkeld wordt. En de testcode die verdwijnt niet, maar is iedere keer opnieuw bruikbaar om regressie te voorkomen.
Delphi 2005 bevat ingebouwde ondersteuning voor Unit Testing in de vorm van DUnit (voor Delphi Win32), DUnit for .NET (voor Delphi .NET) en NUnit (voor zowel C# als Delphi for .NET).

## IntraWeb

Delphi 2005 brengt ook IntraWeb 7.2.14 weer terug voor het bouwen van web toepassingen. Zowel voor Win32 als .NET toepassingen, overigens, maar wel alleen compatible met VCL en VCL for .NET (dus ook compatible met de VCL data access componenten, en niet de ADO.NET of BDP componenten bijvoorbeeld).
Als eerste officiële Nederlandse Authorised IntraWeb trainer verzorg ik ook in 2005 weer IntraWeb 7 trainingen inclusief een nieuw cursusboek (voor zowel Win32 als .NET). De eerste IntraWeb training is op donderdag 24 maart. Wees er snel bij, want de IntraWeb training zit bijna altijd helemaal vol.

# Delphi Taal Uitbreidingen

Er zijn een aantal fijne taaluitbreidingen in Delphi 2005, zowel voor Win32 als .NET. Deze taaluitbreidingen zullen in iedere editie van Delphi 2005 zitten, dus ook in de Personal versie. Het betreft o.a. een nieuwe for-loop syntax (de zgn. for-in-do loop), inlining van code, en - speciaal voor .NET - ondersteuning voor multi-unit namespaces.

## For-In-Do

De nieuwe for-in-do loop stelt je in staat een loop te schrijven zonder expliciet van een index gebruik te maken, maar door het element zelf op te geven. De oude (nog steeds geldige) manier om door een array te lopen is als volgt:

```
var
  x: Array[1..100] of Double;
  som: Double;
  i: Integer;
begin
  som := 0.0;
  for i:=1 to 100 do
    som := som + x[i];
```

Met de nieuwe syntax hebben we de index i niet langer nodig, maar wel een element dat verwijst naar het "huidige" element in het array. Dat ziet er dan als volgt uit:

```
var
  x: Array[1..100] of Double;
  som: Double;
  e: Double;
begin
  som := 0.0;
  for e in x do
    som := som + e;
```

De for-in-do loop zorgt ervoor dat de variabele e automatisch door alle elementen van het array x loopt. En het mooiste is dat we niet hoeven te weten of bij te houden waar het array begint, en waar het eindigt. Dus of het nu van 1 tot 100 of van 0 tot weet-ik-veel loopt, dat wordt automatisch door de compiler uitgezocht.
Daarnaast werkt deze syntax ook voor multi-dimensionale arrays, waarbij de denkbeeldige indices van het array van links naar rechts worden doorlopen:

```
var
  x: Array[1..2,1..2] of Integer;
  y: Integer;
begin
  x[1,1] := 1;
  x[1,2] := 2;
  x[2,1] := 3;
  x[2,2] := 4;
  for y in X do writeln(y); // 1,2,3,4
```

Behalve arrays, werkt de nieuwe for-in-do loop ook op strings (wat in feite een array van characters is), maar ook op andere verzamelingen die een GetEnumerator functie implementeren. Voorbeelden zijn de TStrings of TStringList, en de Collection classes. Om bijvoorbeeld te tellen hoevaak de letter X in de items van een listbox staat, kun je nu de volgende code schrijven:

```
var
  s: String;
  c: Char;
  Score: Integer;
begin
  Score := 0;
```

```
for S in ListBox1.Items do
  for C in S do
    if C = 'X' then Inc(Score);
```

Al met al een erg leuke nieuwe feature, die ik vast nog vaak zal gebruiken.

### Inlining

Een andere functionaliteit is inlining van routines. Dit gaat met het inline keyword, en heeft tot gevolg dat een routine niet meer aangeroepen wordt, maar dat de code die in de body van de routine staat neergezet wordt op de plek waar de routine zou worden aangeroepen. De aanroep (met alle argumenten) wordt dus vervangen door de inhoud van de routine. Met als gevolg dat de uiteindelijke code natuurlijk (een stuk) groter wordt, maar potentieel ook (een stukje) sneller, omdat de aanroep en return niet meer nodig zijn, alsmede het op de stack zetten van de argumenten en het eraf halen van het resultaat. Er moet voor de argumenten nog wel iets speciaals gedaan worden, want de code binnen de routine verwacht nog steeds met argumenten te werken. Hoe meer argumenten, hoe groter de kans dat het er niet beter op wordt. In alle gevallen raad ik met klem aan om een routine die je wilt inlinen ook te controleren (voor en na het inlinen) of de snelheid ook daadwerkelijk is toegenomen.
Inlining is met name zinvol voor Win32, en wat minder voor .NET (alhoewel het daar ook mogelijk is), vanwege het feit dat de .NET JIT (Just-In-Time) compiler zelf ook al inlining en optimalisaties uitvoert.

### Multi Unit Namespaces

Een derde belangrijke uitbreiding van de Delphi taal - deze keer uitsluitend voor .NET - is de zogenaamde multi-unit namespaces. Een belangrijke uitbreiding ten opzichte van Delphi 8 for .NET. In Delphi 8 for .NET geldt het feit dat iedere unit zijn eigen namespace is (de naam van de unit zonder .pas). Dus de unit Bob.Swart.Tools in Bob.Swart.Tools.pas heeft als namespace ook Bob.Swart.Tools. Dat heeft als grote nadeel dat je al snel heel veel relatief kleine namespaces krijgt, die de inhoudelijke structuur en opbouw van je source code laat zien, terwijl ik eigenlijk logische in plaats van fysieke namespaces wil gebruiken. In C# is het mogelijk om meerdere source files in dezelfde namespace te laten zitten.
Delphi 2005 doet het meer op de C# manier. De regel geldt nu dat indien de naam van de unit geen punt bevat, dan is de naam van de unit de naam van de namespace. Dus unit tools in tools.pas resulteert in de tools namespace. Als er wel een punt in de naam zit, zoals Bob.Swart.Tools, dan geldt dat het rechter deel van de naam genegeerd wordt, en de rest is dan de naam van de namespace. Dus zowel Bob.Swart.Tools en Bob.Swart.AndereTools leveren samen een bijdrage aan de Bob.Swart namespace. Dat geeft mij eindelijk de mogelijkheid om een kleine hoeveelheid namespaces over te houden, maar toch een flink aantal units te gebruiken (waarbij ik iedere class weer zijn eigen unit kan geven, zonder meteen voor iedere class zijn eigen namespace te krijgen).
Er is een mogelijk probleem dat kan optreden als verschillende units in dezelfde namespace een identifier (type of globale variabele of methode) met dezelfde naam bevatten. Die identifier komt dan meer dan éénmaal voor in dezelfde namespace, waarbij er "van buiten" geen onderscheid gemaakt kan worden. De resulterende toepassing of .NET assembly zal niet door PEVerify geaccepteerd worden. Maar eigenlijk is het sowieso niet slim om identifiers met dezelfde naam te gebruiken (ook niet in verschillende units), dus hopelijk komt dit probleem niet al te vaak voor. De Delphi 2005 compiler geeft er overigens nog geen foutmelding voor - maar dat zal in de toekomst waarschijnlijk veranderen.

## Conclusie

Delphi 2005 is een enorm uitgebreide nieuwe versie van Delphi, waarmee zowel Win32 als .NET toepassingen gemaakt kunnen worden. Voor .NET toepassingen kun je kiezen tussen C# en Delphi for .NET, en zelfs binnen Delphi for .NET kun je kiezen voor VCL for .NET compatible of WinForms compatible toepassingen (alsmede IntraWeb / VCL for .NET versus ASP.NET voor de web toepassingen). Een zeer ruim spectrum van mogelijkheden, los van de vele database technieken in de vorm van BDE, dbGo for ADO. dbExpress, InterBase Express, ADO.NET en de Borland Data Provider.
De IDE is uitgebreid met ondersteuning voor Refactoring (hier kunnen we in de toekomst nog meer van verwachten) en Unit Testing, waarmee Delphi laat zien aan te sluiten bij recente ontwikkelingen in de wereld van de software engineering.

# Refactoring with Borland Delphi 2005

On October 12th 2004, Borland announced Delphi 2005, the latest version of their RAD development environment. Delphi 2005 consists of three personalities: Delphi for Win32, Delphi for .NET and C# (and it can also compile VB.NET applications).
One of the most impressive set of new features consists of refactoring support in the Delphi 2005 IDE, which is covered in more detail in this article.

## Refactoring

When you apply refactoring you are adding (more) structure to your code, making it clearer to understand, easier to reuse and better maintainable. However, as an important rule of thumb for refactoring, you do all this without adding new functionality to your application. The behavior of the application or piece of code should remain the same.
Borland Delphi 2005 Refactoring support includes a number of very helpful new features, from extracting methods to declaring new variables or fields, extracting resource strings, renaming identifiers and refining the namespace and uses clauses.

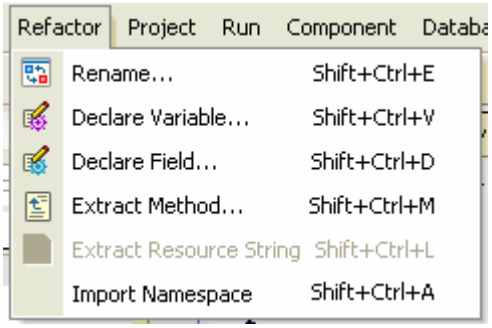



*Figure 1. Delphi 2005 Refactoring Options*

### Rename Symbols

With *Refactor – Rename* you can rename symbols (like fields, methods, properties, variables, etc.). You can view all references before refactoring (to verify all places where the rename will be made, and can remove places if you wish).

### Declare Variable

While writing source code, it may happen that you use variables before you declare them. The Borland Delphi 2005 Refactoring allows you to automatically declare these variables using *Refactor - Declare Variable*, offering you a dialog to enter the specifics, and adding the variable declaration to the current scope.
This feature allows you to focus on the code and algorithm logic, without having to manually navigate to the beginning of the scope to add a variable declaration.

### Declare Field

Similar to declaring undeclared variables, Delphi 2005 offers the ability to declare class fields using *Refactor - Declare Field.*
This feature greatly reduces time to extend your classes with fields while writing your source code, without forcing you to return to your class declaration and add the field definition manually.

### Extract Method

Delphi 2005 allows you to select a portion of source code, and refactor it by turning the code into a method, extracting the selected source code. The refactored method will automatically get a parameter list as well as local variable declarations, and the original section of source code will be replaced by a call to the newly refactored method. Breaking long sections of code into methods increases maintainability and reusability.

### Extract Resource String

There's nothing harder to localize than a portion of source code that uses hardcoded quoted strings inside. Delphi 2005 now allows you to extract these quoted strings and replace them with resource strings (adding the resource string declarations to the implementation section of your code).

### Find Unit / Import Namespace

Sometimes you use classes, methods, fields or types that are defined in another unit's namespace. In order to add the corresponding namespace to the uses (for Borland Delphi) or using (for C#) clause, Borland Delphi 2005 Refactoring offers the ability to automatically import the required namespace for a selected identifier, using *Refactor – Find Unit* or *Import Namespace*.

## Refactoring Example

The six refactoring techniques can be divided in two groups: those who help you write new code (like declare variable, declare field and find unit / import namespace), and those who help you restructure existing code (rename, extract method, extract resource string). We will now use an existing Delphi 2005 demo application to demonstrate the refactoring features in practice.

### SimplePuzzle

Start Delphi 2005, click on the Open Project button or do File | Open Project, then go to the BDS\3.0\Demos\Delphi.NET\VCL\SimplePuzzle directory and open the Puzzle.bdsproj project file. This is an example Delphi puzzle, written by one of the Borland employees, and offers a nice puzzle game. Compile and run it, and I'm sure you will recognize the puzzle (see Figure 2 for an example).

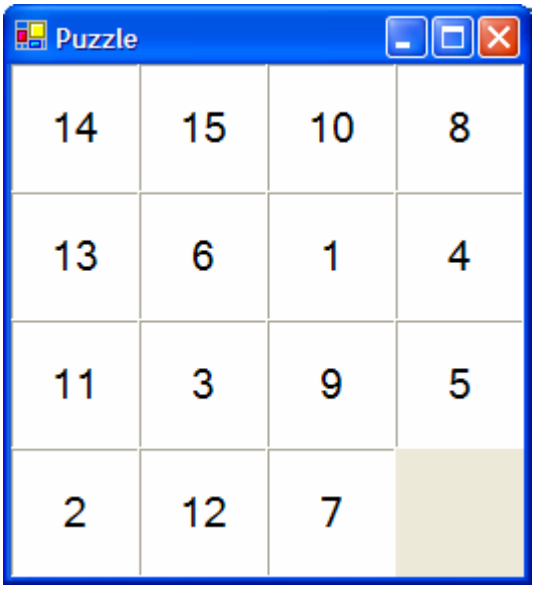



*Figure 2. Puzzle in Action*

However, if you take a look at the source code for the first time, and want to add some features (like the ability to change the number of puzzle pieces), then you may first have to study the source code. Not that it's written in a bad way, but it's just not code written by you (or me for that matter), which is always harder to read than your own code. Which is exactly where Refactoring can help you.

### Rename

First of all, in the class definition of TWinForm1 inside the WinForm1.pas unit, I see the following public section with the definition of size, x1, y1 and mylabels:

```
public
  size:integer;
  x1,y1:integer;
  mylabels:TList;
```

If you look further in the source code, it appears that x1 and y1 are the X and Y location of the free spot on the game board. In order to make sure I understand that from now on, let's rename x1 to FreeSpotX and y1 to FreeSpotY (feel free to use your own names that make sense, of course).
Place the cursor at the location of x1 (either before the x or before the 1) and right-click with the mouse, pick the Refactoring menu, and select the Rename field "x1" option. This will give us the dialog in Figure 3 (where I already specified the new name FreeSpotX):

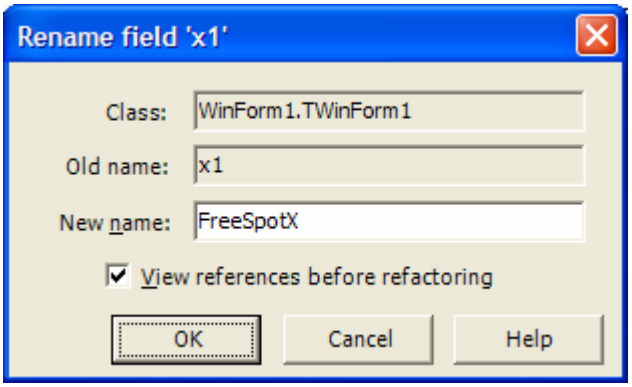



*Figure 3. Rename Field*

Note the option to view the references before refactoring. These references will be shown in a special Refactorings pane at the bottom of the Code Editor:

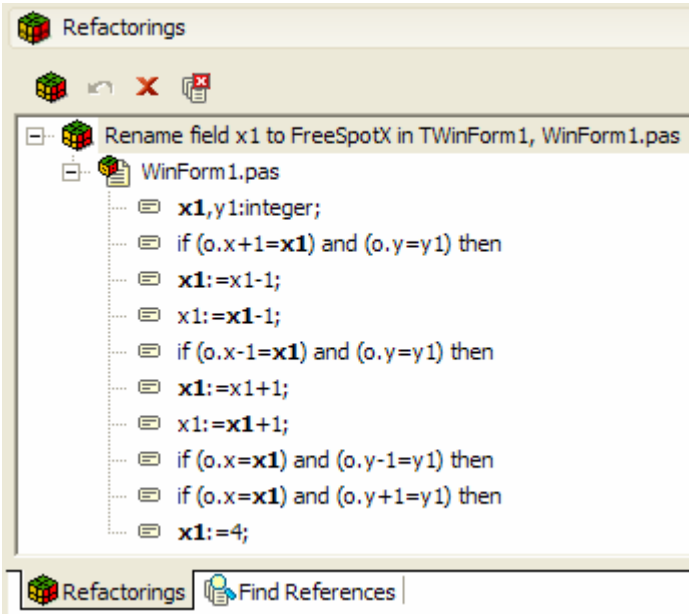



*Figure 4. View References before Refactoring*

As you can see in Figure 4, renaming the field x1 to FreeSpotX is not done in only one location, but in fact throughout the entire unit. In fact, Refactoring makes sure the walk through the entire project to find all occurrences of "x1" to rename to FreeSpotX. This really beats a global search and replace!
Click on the Refactorings Cube on the toolbar to actually apply the refactoring actions. Then, do the same thing with "y1", renaming that field to FreeSpotY.

## Extract Method

Time to scroll a bit down in the source code of the Puzzle project. In the LabelMouseDown method, there's a section of code from line 130 until line 176. Not too long, but it contains four subsections that look almost similar, marked with a "right", "left", "top" and "bottom" comment.

Select lines 139 until 142, and prepare to extract these to a new method. Right-click and select the Refactoring Extract Method choice. This will present you with the Extract Method dialog, which can be seen in Figure 5.

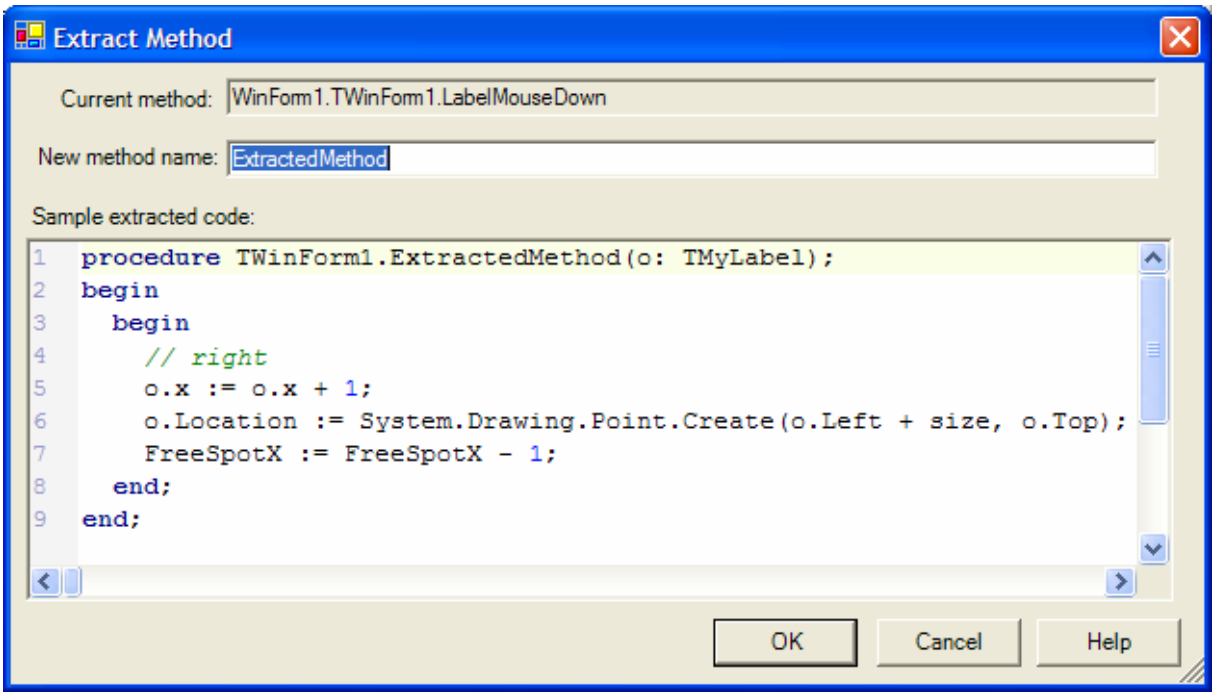



*Figure 5. Extract Method*

Note that the new method is called "ExtractedMethod" by default, but we should change that to something like "MoveRight", to indicate that this is the method that moves a square to the right (and the free spot to the left). Also note that one argument is passed: the "o" variable that we're using here. Obviously, you can use the Rename Refactoring later to rename the "o" in this local routine to something else, a bit more descriptive.
Click on the OK button to extract the selected code and generate the new method "MoveRight". As a result, the selected code in the LabelMouseDown routine will be replaced by a single line, with "MoveRight(o)" inside. Note that no semicolon follows this call, to avoid compilation errors.

## Delphi 2005 Error Insight

However, speaking of compilation errors! By extracting the MoveRight method, we accidentally broke the application: it will no longer compile. The problem is caused by the fact that the new method has an argument of type TMyLabel. And unfortunately, in the interface section of the WebForm1.pas unit, the TMyLabel type definition comes after the TWinForm1 definition. You don't even have to compile the application to see the problem, since numerous red wave lines will be displayed in the Code Editor to alert you to the problem (see Figure 6). A complete list of syntax errors is also displayed in the Structure Pane at the upper-left corner of the IDE.

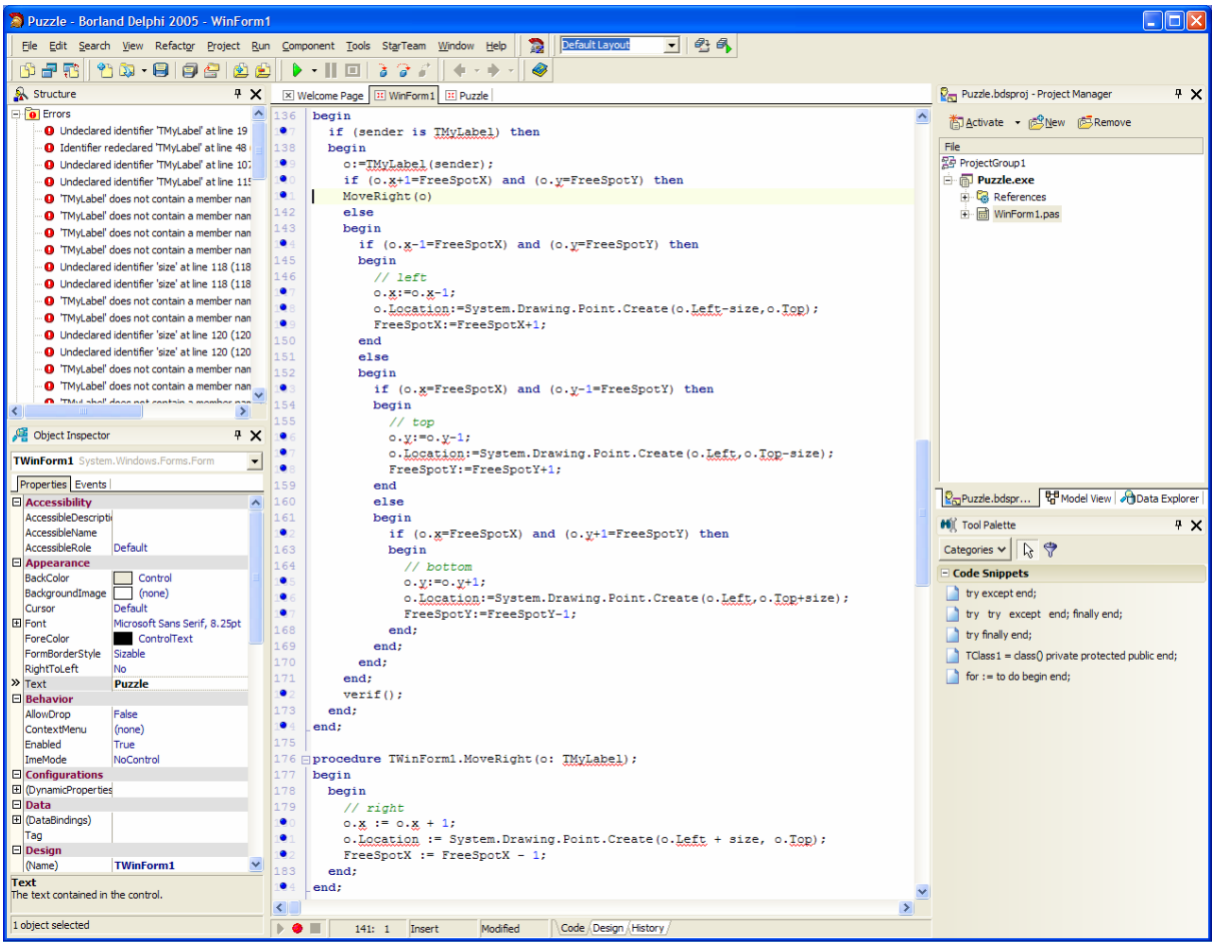



*Figure 6. Error Insight in the Delphi 2005 Code Editor*

Obviously, we should go to the top of this unit, and move the definition of the TMyLabel class before the definition of the TWinForm1 class. That will resolve all errors again.
Once this is done, you can extract the code for the "Left" move as well, if you want.

## Extract Resource String

Apart from extracting methods, we can also extract resource strings. This is useful if you want to translate the application, and display a localized (error) message to the users. In this SimplePuzzle project, there is only one string that's a candidate for extraction as resource string: right at the bottom of the unit in the Verif method. The string is used to display a congratulation message once the puzzle is solved.
You can place your cursor inside the string, right-click and select the Refactoring – Extract Resource String option. This will present you with the dialog shown in Figure 7.

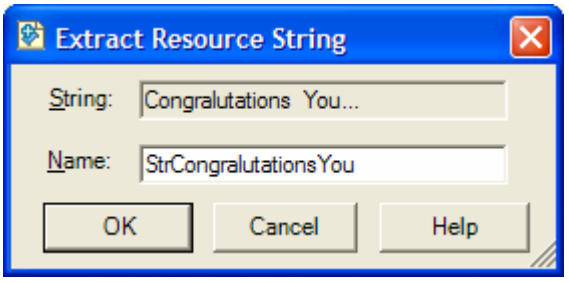



*Figure 7. Extract Resource String*

You can change the name of the resourcestring, and once you press OK, the new resource string is created for you. All extracted resource strings will appear at the start of the implementation section of your unit.

## Add Variable / Field

So far, we've only renamed fields, extracted methods and resource strings – things to do with source code without changing the actual behavior. There are however also Refactoring techniques that can be applied best when adding new features, or changing the behavior (although it's generally recommended not to add features while refactoring), and these include the Add Variable, Add Field and Find Unit / Import Namespace. Let's view these in action now.

The current puzzle, is a 4 by 4 game board, with 15 pieces and one empty spot. The numbers 4 and 15 are "magic" numbers that are hardcoded in the application. And I want to remove the fact that they are hardcoded, so we can change the layout of the gameboard as well as the number of pieces on it.

Let's start with the number 4 – which is currently both the maximum range in the horizontal and vertical dimension. In order to find all occurrences of "4", we can use the new Find in Files feature of Delphi 2005. Although we only have to search in one file, the Find in Files dialog has an option to display the results grouped by file in a nice treeview (see Figure 8).

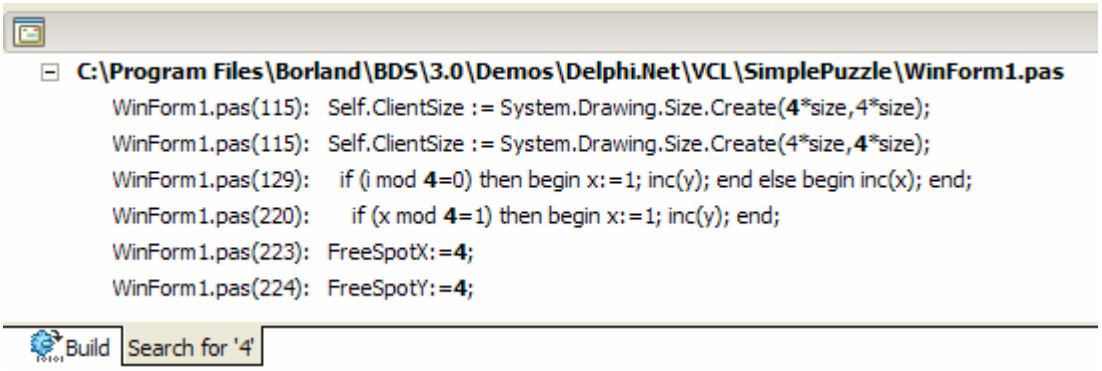



*Figure 8. Search Results for '4'*

We can click on each node in the search results tree to be taken to the corresponding line of source code. Let's take a look at the first statement, which contains two occurrences of '4': one for the maximum horizontal, and one for the maximum vertical pieces. Replace the first 4 by the identifier MaxX. This will immediately allow Error Insight to kick in and mark the MaxX identifier with a red wavy line, to indicate that it hasn't been declared, yet. Something which we'll do right now using Refactoring. Right-click on the MaxX identifier, and select the Refactoring – Add Field option. This will give you the dialog shown in Figure 9.

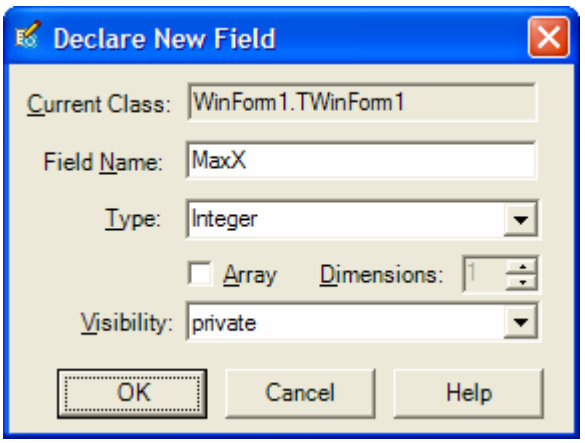



*Figure 9. Refactoring Declare New Field*

Here you can specify the type of the new field (Integer) as well as the visibility specifier (by default set to private). Click on OK to add the declaration to the WinForm1.TWinForm1 class. Note that after you've clicked on OK, you are still at the old location in the Code Editor. You are not taken to the place where the new field declaration has been added to the class. This is actually a very helpful feature, since it allows you to continue working on the source code, declaring new fields and/or new variables without losing your momentum while writing the code. Never do you need to scroll up to the nearest "var" section or the definition area of the class itself. This will allow me to focus on the code, while the Error Insight red wavy lines will alert me when I need to declare a new field or variable.
After we've changed the first 4 to MaxX, we can change the next 4 to MaxY, changing that source line to the following (where MaxY is underlined with a red wavy line, since it hasn't been declared yet):

```
Self.ClientSize := System.Drawing.Size.Create(MaxX*size,MaxY*size);
```

Use Refactoring to create a new private integer field called MaxY to make sure the project compiles again.
There are now four places left that use a hardcoded 4. Two of them use 4 in a "mod 4" expression. In that expression, the 4 is the number of horizontal pieces, so replace the 4 there with the MaxX field.
The last two places are the assignments to FreeSpotX and FreeSpotY, which should be assigned to MaxX and MaxY respectively.
The only thing left to do is to initialize MaxX and MaxY with their original value 4. That can be done in the TWinForm_Load procedure. However, instead of assigning the original values 4 to both MaxX and MaxY, let's assign 8 to MaxX and 2 to MaxY. That's still a gameboard of 16 places, with 15 pieces and one free spot.

```
procedure TWinForm1.TWinForm_Load(sender: System.Object; e: System.EventArgs);
begin
  MaxX := 8; // originally 4
  MaxY := 2; // originally 4
  size:=64;
  init();
end;
```

If you compile and run the application, you can see that we changed the puzzle to an 8x2 format now. Do not try to change MaxX and MaxY to other values just yet, since the above example only worked because 8 * 2 is equal to 4 * 4, resulting in a gameboard with 16 places. And there are still some hardcoded 15's in the project. Search for all occurrences of 15, and replace these with the expression (MaxX * MaxY – 1). This will make sure that no matter which value for MaxX and MaxY is used, the gameboard will be correctly calculated.
As last example, set MaxX to 7 and MaxY to 6, resulting in a gameboard with 42 places and 41 pieces, as shown in Figure 10.


*Figure 10. Puzzle with 7 * 6 gameboard*

I leave it as exercise for the reader to expose the MaxX and MaxY fields through a pop-up menu choice to allow the player to select the size of the gameboard before the game starts. After all, this was just an example application, albeit a very nice one at that.

## Summary

In this article I've explained what Refactoring is, and which Refactoring features are available in Delphi 2005. I have then demonstrated the Rename, Extract Method and Extract Resource String refactoring features using the SimplePuzzle example application. We've restructured the source code without changing the actual behavior of the application.
I then decided to add the functionality to change the gameboard dimensions, adding the MaxX and MaxY fields to the WinForm, and replacing some hardcoded magic numbers by these new fields, using the Refactoring New Field feature.
Refactoring is supported in Delphi for Win32 as well as Delphi for .NET applications. For C# projects, only Rename, Extract Method and Import Namespace are available. The Add New Field and Add New Variable are not available, since you can declare variables almost anywhere in C# code (there is no need to move to a special "var" section). There is also no Extract Resource String functionality in C#, simply because that doesn't exist in C#.
Apart from Refactoring, I've also briefly demonstrated some of the other new Delphi 2005 new features, including Error Insight, and Find in Files results.

**Bob Swart** (aka Dr.Bob – http://www.drbob42.com) is an author, trainer, developer, consultant and webmaster for Bob Swart Training & Consultancy (eBob42) in The Netherlands, who has spoken at Delphi and Borland Developer Conferences since 1993. Bob has written the official Borland Delphi for .NET Essentials and Borland Delphi for .NET and ASP.NET Essentials courseware licensed by Borland.