Delphi OplossingsCourant

Vol 3. No. 1. Een publicatie van het Everest **K**ylix/**D**elphi **O**plossings**C**entrum (KDOC) http://www.kdoc.nl

Best, 29 maart 2001,

Op het moment dat ik dit welkomstwoord schrijf is Kylix niet alleen gelanceerd, maar ook gearriveerd. En alhoewel op dit moment de vraag naar Kylix expertise nog wat achter lijkt te blijven op de verwachtingen, zijn we als DOC toch al lekker bezig met Kylix. Vooral de cross-platform zaken die we straks ook in Delphi 6 terug kunnen zien (zoals dbExpress) hebben onze aandacht. Daarover een volgende keer meer. We hebben in ieder geval onze naam uitgebreid tot Kylix/Delphi OplossingsCentrum (KDOC) en onze website is nu te vinden als http://www.kdoc.nl

Het conferentie-seizoen is weer bijna aangebroken. Begin april en mei verzorg ik in samenwerking met de UK Borland User Groep een tweetal MIDAS seminars in respectievelijk London en Edinburgh. In juni is de Delphi Conference 2001 aan de beurt: DCon 2001, georganiseerd door UK-BUG en The Delphi Magazine. Voor deze conferentie zijn de beste Delphi spekers uitgenodigd (een 12-tal in totaal, waaronder Micha Somers en Bob Swart van het Delphi OplossingsCentrum). Voor meer informatie over de ruim 45 Delphi sessies, zie http://www.dcon.co.uk.

En natuurlijk is er in de zomervakantie weer de jaarlijkse Borland Conferentie (BorCon) in de USA, dit jaar in Long Beach, waar Bob Swart een aantal sessies zal doen over Delphi en Kylix (meer informatie volgt in het volgende nummer van de Delphi OplossingsCourant). Mocht u als lezer geen (toekomstig) nummer willen missen, dan kunt u zich vrijblijvend aanmelden door een e-mailtje te sturen naar info@kdoc.nl onder vermelding van "abonnement DOC".

Inhoudsopgave

De **Delphi OplossingsCourant** is een productie van het Everest Kylix/Delphi OplossingsCentrum, met medewerking van Arjan Jansen, Arnim Mulder en Micha Somers.

Eindredactie: Bob Swart - e-mail: info@kdoc.nl

O'Reilly Boekenbal

Vanaf nu zullen we in ieder nummer van de Delphi OplossingsCourant een boek van O'Reilly verloten onder de lezers. Deze keer is een gesigneerd examplaar van het boek "Running Linux" gewonnen door Jim Bosman van Conplacer uit de Meern.

Debug Active Server Objects met MTS

Delphi 5 beschikt over een Active Server Object wizard waardoor het redelijk eenvoudig is om Active Server Objecten te ontwikkelen. Het debuggen van Active Server Objecten daarentegen kan lastig zijn. Vaak worden primitieve debug methodes toegepast (zoals wegschrijven naar log bestanden, tonen van messages, etc.) Dit artikel zal laten zien hoe we zowel onder Windows NT als Windows 2000, Micorsoft Transaction Server (MTS) kunnen gebruiken om het debuggen van Active Server Objecten aanzienlijk vereenvoudigen.

DebugDemoObject

We zullen eerst een simpel voorbeeld ontwikkelen waarmee we vervolgens de debug mogelijkheden zullen bekijken. Om een Active Server Object te creëren, zullen we eerst een ActiveX Library aanmaken.

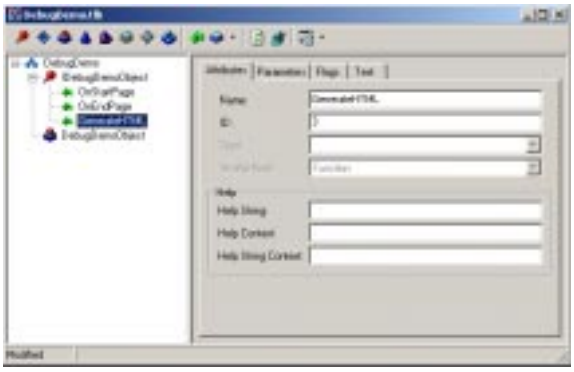
Kies `File|New`, ga vervolgens naar de ActiveX tab en dubbelklik op ActiveX Library. Bewaar de library (bv. DebugDemo.dpr).

Vervolgens maken we in deze library een nieuw Active Server Object aan. Kies daartoe `File|New`, ga vervolgens naar de ActiveX tab en dubbelklik op Active Server Object om de Active Server Object wizard op te starten.

Eerst moeten we een CoClass name opgeven. In ons voorbeeld vullen we in: DebugDemoObject. De overige velden zullen we niet wijzigen. Nadat we op OK hebben geklikt, zal een Active Server Object (met de naam DebugDemoObject) en een template test script worden aangemaakt en wordt de Type Library Editor gestart.

Met behulp van de Type Library Editor kunnen we methoden en properties voor ons Active Server Object aanmaken. Deze kunnen we vervolgens vanuit een Active Server Page aanroepen.

Selecteer IDebugDemoObject en klik op  (New Method) om een nieuwe method te creëren. In ons voorbeeld zullen we de method GenerateHTML noemen. Vanuit een Active Server Page (het door Delphi gegenereerde test script) zullen we deze methode aanroepen om zodoende dynamisch HTML code te genereren.



Door op  (Refresh implementation) te klikken zal Delphi de benodigde code voor deze method genereren. Binnen deze method zullen we als voorbeeld wat code schrijven om de server variabelen en hun waarden te tonen.

```
procedure TDebugDemoObject.GenerateHTML;
var
  s: String;
  i: Integer;
  SessID: string;
  ServerVars: String;
  VarKey: String;
  VarValue: String;
begin
  SessID := Session.SessionID;
  ServerVars := '<table>';
  for i := 1 to
    Request.ServerVariables.Count do
  begin
    VarKey :=
      Request.ServerVariables.Key[i];
    VarValue :=
      Request.ServerVariables[VarKey];
    ServerVars := ServerVars + '<tr><td>' +
                  VarKey + '</td><td>' +
                  VarValue + '</td></tr>';
  end;
  ServerVars := ServerVars + '</table>';
  s := '<html><body>' +
       '<h1>Demo Debugging '+
       'Active Server Objects</h1><br>' +
       'Current session ID: ' + SessID +
       '<br>' + ServerVars +
       '</body></html>';
  Response.Write(s);
end;
```

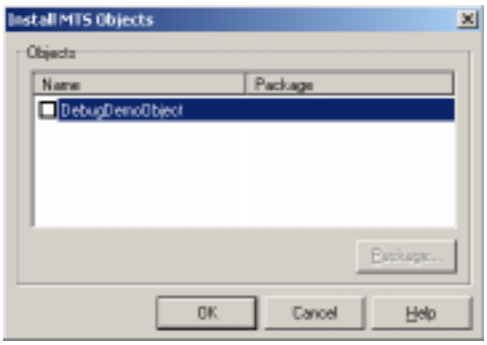
Tevens zullen we het gegenereerde .asp bestand aanpassen door in de code *{Insert Method name here}* te vervangen door *GenerateHTML*:

```
<HTML>
<BODY>
<TITLE> Testing Delphi ASP </TITLE>
<CENTER>
<H3> You should see the results of your
Delphi Active Server method below </H3>
</CENTER>
<HR>
<% Set DelphiASPObj =
  Server.CreateObject("DebugDemo.DebugDemoObject")
  DelphiASPObj.GenerateHTML
%>
<HR>
</BODY>
</HTML>
```

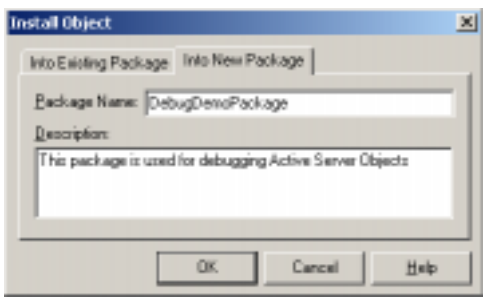
Kies `File|Save All` om dit project te bewaren. Bewaar de unit met het Active Server Object onder de naam DebugDemoObj.
Het script benaderen we via onze webbrowser (bv. http://localhost/DebugDemoObject.asp) waarbij we gebruik maken van IIS als webserver. Bewaar dit bestand daarom in een directory (met scripting rights) die door de webserver makkelijk te benaderen is (bv. c:\inetpub\wwwroot\DebugDemoObject.asp).

Registratie

Voordat we een Active Server Object kunnen gebruiken, zal het eerst moeten worden geregistreerd. Dit kan op verschillende manieren worden gedaan. Vaak wordt het commando *regsvr32 <dllname>* gebruikt om het Active Server Object te registreren. Echter, omdat we ons Active Server Object willen debuggen met behulp van MTS, zullen we het object installeren en registreren in MTS.
Dit kan zowel in Delphi als in de Microsoft Mangement Console. In het geval we Delphi gebruiken, kies `Run|Install MTS Object`.



Selecteer het object dat we willen registreren.
Vervolgens zal gevraagd worden in welke package het object moet worden geïnstalleerd.
Hier kan een bestaande package opgegeven worden of kan het in een nieuwe package worden geïnstalleerd. In ons voorbeeld zullen we een nieuwe package aanmaken door in de "Into New Package" tab als package name in te vullen: DebugDemoPackage.



Het object wordt geinstalleerd door op OK te klikken en in het "Install MTS Objects" dialoogvenster ook op OK te klikken.
Om ons Active Server Object te kunnen debuggen binnen Delphi, zullen we een host application moeten opgeven. Windows NT verschilt hierin echter met Windows 2000.

Debuggen met WinNT + Option Pack

Aangezien we MTS gebruiken, zullen we op een of andere manier aangeven dat MTS moet worden gebruikt als host application. Aangezien MTS reeds gestart is, zullen we MTS eerst moeten afsluiten. Normaal gebruik je hier een aparte debug omgeving voor (bv. een lokale machine met IIS en MTS). Uiteraard is het niet wenselijk om MTS af te sluiten in een productie omgeving omdat hiermee alle MTS services worden gestopt. De Microsoft Management Console (MMC) kan worden gebruikt om MTS te stoppen. Start daartoe MMC, open Computer (onder Microsoft Transaction Server) en klik met de rechter muisknop op My Computer. Kies "Shut Down Server Processes". Nu het MTS server proces is gedeactiveerd, kunnen we terug keren naar Delphi om MTS als hosting application van ons Active Server Object opgeven. Kies `Run | Parameters`. Vul als hosting application de MTS executable inclusief het volledige pad. In mijn geval is dit: d:\winnt\system32\mtx.exe
Tevens moeten we opgeven welke package gebruikt moet worden. Doe dit door bij Parameters /p:"DebugDemoPackage" op te geven.

Debuggen met Windows 2000

MTS is volledig geintegreerd in Windows 2000. Hierdoor is er geen mtx.exe meer beschikbaar. Gelukkig kunnen we onder Windows 2000 gebruik maken van een andere applicatie als hosting application, nl. dllhost.exe (te vinden in de system32 directory). Kies `Run | Parameters`. Vul als hosting application de dllhost executable inclusief het volledige pad, zoals mijn d:\winnt\system32\dllhost.exe
Daarnaast moeten we nog aangeven welk proces we gaan debuggen. Dit gebeurt door bij Parameters het ProcessID op te geven in de vorm van: /ProcessID:<process id>. Dit ProcessID is te vinden door onder Windows 2000 Component Services te starten (onder

`Programs | Administrative Tools`) en vervolgens met de rechter muisknop de properties te openen van de DebugDemo package (onder `Component Services=>Computers=>COM+ Applications`). In het dialoogvenster wordt o.a. een Application ID getoond dat moet worden opgegeven onder Parameters. Houdt er rekening mee dat dit Application ID wordt gegenereerd bij aanmaak van de package en bij iedereen verschillend zal zijn.

In mijn geval moet bij Parameters de volgende waarde worden ingevuld:
/ProcessID:{DCF820E2-6933-4F34-9EEB-F4CA5B671A80}

Start Debuggen

Na de hosting application te hebben opgegeven zijn we in staat om ons Active Server Object te debuggen in de Delphi IDE. Zo kunnen we breakpoints plaatsen in onze code.
In ons voorbeeld zullen we waarom een breakpoint plaatsen op de eerste regel in de code van de method GenerateHTML .

Run vervolgens de ActiveX library. Dit zal tot gevolg hebben dat de host applicatie wordt opgestart (mtx.exe onder Windows NT, dllhost.exe onder Windows 2000). Vervolgens kunnen we ons Active Server Object activeren door de betreffende .asp pagina (bv. http://localhost/DebugDemoObject.asp) in een webbrowser te openen .
Dit heeft tot gevolg dat de GenerateHTML methode van ons DebugDemoObject wordt aangeroepen en Delphi vervolgens stopt op het breakpoint dat we geplaatst hebben. In principe zijn we nu in staat om het DebugDemoObject te debuggen door gebruik te maken van de uitgebreide debug faciliteiten (inspect, watches, etc.) die Delphi biedt. Zo kunnen we bv. het Session ID bekijken door met behulp van de rechtermuisknop via `Debug | Add watch at cursor` een watch toevoegen voor de variabele SessID.
We kunnen vervolgens stap voor stap door het programma heen lopen met behulp van de functietoetsen (F7, F8, F9) om zodoende alle benodigde informatie te achterhalen. Als alle code is uitgevoerd, zal het resultaat worden getoond in de webbrowser.

Beëindigen Debug Sessie

Merk op dat Delphi zich nog steeds in debug mode bevindt omdat onze host applicatie nog actief is. Om de debug sessie af te sluiten, zullen we ervoor moeten zorgen dat de host applicatie wordt afgesloten.
In het geval dat Windows NT wordt gebruikt, sluiten we mtx.exe af door in MMC met de rechter muisknop te klikken op `My Computer` en vervolgens MTS te stoppen.
Onder Windows 2000 is het voldoende om de package met het betreffende Active Server Object af te sluiten (klik met de rechtermuisknop op DebugDemoPackage en selecteer `Shut Down`).

WAPPEN met Delphi (2)

In de vorige Delphi OplossingsCourant (van December 2000) heb ik het een en ander uitgelegd over WAP. Mijn Nokia WAP Toolkit draait nog steeds en dat is niet slecht voor een trial versie die maar 30 dagen had moeten werken.

Deze keer gaan we een WAP applicatie maken in Delphi. We gaan een eenvoudige telefoongids bouwen zodat collega's via de WAP telefoon telefoonnummers van andere collega's kunnen vinden.

We starten een nieuwe project (File, New, Web Server Application) en kiezen de optie CGI Stand-alone executable. Zo kunnen we onze applicatie testen op een webserver zonder dat we de webserver iedere keer down moeten brengen (een ISAPI DLL blijft immers geladen zolang de webserver draait). We zien nu een lege webmodule op het scherm. We voegen nu een action toe aan de webmodule (en maken deze tevens default) zodat de applicatie weet welke code uitgevoerd moet worden wanneer het wordt opgestart

Op de module plaatsen we een TTable. De telefoonnummers worden opgeslagen in en opgehaald uit een tabel. Ik heb hiervoor een simpel Paradox tabelletje gemaakt met de velden "Naam" en "Nummer". Wanneer een verzoek binnenkomt moet het telefoonnummer worden opgezocht in de tabel en het resultaat worden teruggegeven in het OnAction event. De Response parameter van dit event bevat alle informatie die aan de client wordt teruggegeven. De webserver moet worden ingelicht wat voor soort tekst wordt teruggegeven en in dit geval is dat natuurlijk wml.

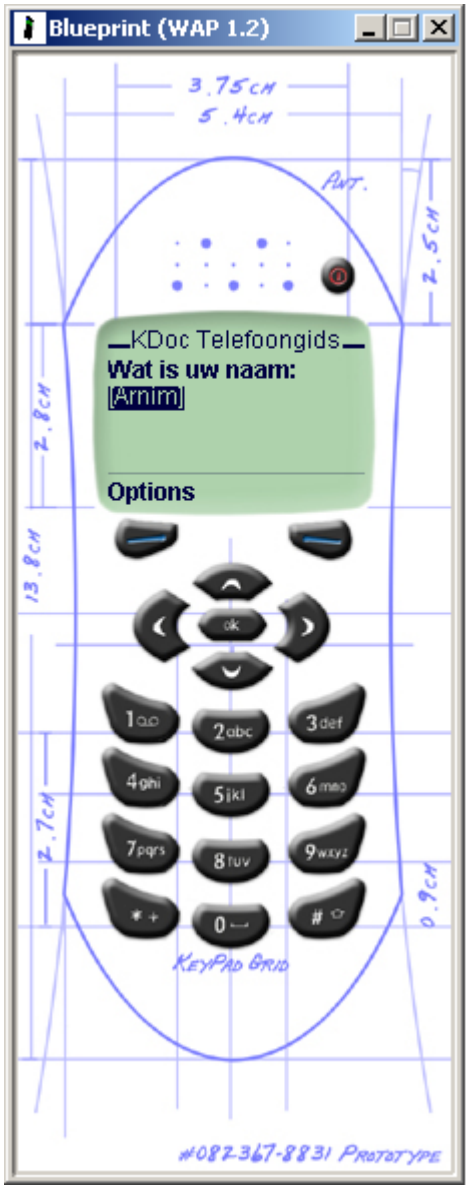
```
Response.Content := 'text/vnd.wap.wml';
```

Daarna kan het resultaat worden teruggegeven:

```
Response.Content :=
  '<?xml version="1.0"?>' +
  '<!DOCTYPE wml PUBLIC '+
  '"-//WAPFORUM//DTD WML 1.1//EN"'+
  '"http://www.wapforum.org/DTD/wml_1.1.xml">'+
// standaard header
  '<wml>'+
  '<card title="Zoekresultaat" id="main">'+
  '<do type="accept" '+
  'label="Back"><prev/></do><p><br/>';

Naam :=
  Request.ContentFields.Values['naam'];
if tblAdres.Locate('Naam',
    Naam, [loCaseInsensitive]) then
  Response.Content := Response.Content +
    'Het telefoonnummer van ' + Naam +
  ' is ' +
  tblAdres.FieldByName('Nummer').AsString
else
  Response.Content := Response.Content +
   'De naam is niet gevonden in het '+
   'telefoonboek';
```

Wanneer nu vanuit een wml pagina de executable wordt aangeroepen zal bovenstaande actie worden uitgevoerd. Zoals je ziet is het maken van een WAP applicatie niet veel verschillend dan dat van een applicatie voor een gewone webserver. Het belangrijkste is dat de ContentType op de juiste waarde gezet wordt (en uiteraard het resultaat volgens de WML standaard) zodat een WAP telefoon de ontvangen pagina kan interpreteren.



Red Hat Linux 7 Unleashed

Authors: Bill Ball, David Pitts, et al.
Publisher: SAMS
ISBN: 0-672-31985-3
Price: US$ 49.99, UK 36.50

Red Hat Linux 7 Unleashed contains information that helps you install, configure and maintain the Red Hat distribution of Linux 7. The book contains a copy of Red Hat Linux 7 itself on the three accompanying CDs bound in the back and front, so you don't really need anything else, which is a great way to get started with Linux (cheaper than buying Linux in the stores).
The book is divided in five parts. The first part is mainly about the installation of Linux and the most essential user services. The first two chapters introduce Red Hat Linux and help you through a step-by-step installation of the Red Hat system. This is a helpful resource next to the Red Hat Linux CDs themselves. Chapter 3 through 8 cover the installation of the LILO boot manager, the X windows system and window managers.

The second part of the book focuses more on the configuration of local and network services such as system startup and shutdown, internet services such as e-mail (SMTP and POP3), FTP, the Apache web server, news, a domain name service (DNS) and dynamic host configuration protocol (DHCP). A final service that's quite helpful to configure if you want to use your Linux machine as a server that other computers can connect to, is the Samba server. The book spends 50 helpful pages on this topic alone, which enabled my Linux computer to be a secure file and printer server.

The third part of the book is probably not meant for the average Linux user, since it's more about system administration and management. Chapters in this part cover (background) information on the Linux filesystem, disk and other devices, printing, TCP/IP networking, backup/restore and system security. It's nice to read about relative new features such as USB support as well.

Part four of the book is about development and productivity using C/C++ programming tools, shell scripting, etc. giving merely a list of available tools. The biggest chapter in this part is about configuring and rebuilding Linux kernels, which is not a task that everyone will perform (at least I won't), but is interesting to read - if only as background information.

The last part of the book contains four appendices. The first is about the Linux Documentation project, the second about the most used (or useful) Linux commands and utilities. The third appendix spends 10 pages to list the GNU General Public License. The last appendix lists the Red Hat Linux RPM packages with a one line description each (quite a list, by the way - more than 40 pages).

The index is detailed (more than 80 pages) and well organised: the few times that the chapters weren't detailed enough for me to help find a certain topic, the index could help me pin-point it right away.
This book has been a great help in installing Red Hat Linux 7 on my machine. Apart from that, it contains very helpful information on configuring and maintaining a running and efficient Red Hat Linux system. The development part is a bit too short for my taste (only 160 pages), but this book is not meant to be a developers book - there are other Linux development books for those purposes.
I can recommend this book to anyone with little or moderate Linux experience who wants to step up to Red Hat Linux 7 and/or maintain a Red Hat Linux 7 system. And given the level of detail, I'm confident this book will be useful even when the next releases of Red Hat Linux will be made available in the future.

Kylix!

Welcome to the latest revolution from Borland. I can't say how satisfied I am that Borland has again been able to change the way I'm working - in a positive way - by adding RAD to Linux.

As a long-time software engineer, I started my career with a copy of Turbo Pascal version 2 for PC-DOS. As my first real programming language, Pascal sticked with me for the years that followed. And when the Object Oriented revolution came, I could join in with Turbo Pascal 5.5, which had the subtitle "With Objects".

An even bigger milestone came when Borland released Turbo Pascal for Windows. Now we could write native 16-bit Windows applications. I ported many old DOS applications to Windows, and life was good. Especially when Borland Pascal 7.0 was released, which supported 16-bit DOS, DOS DPMI and 16-bit Windows development in one Windows (and DOS if you still wanted) Integrated Development Environment.

A few years went by, and Borland Pascal was great. But slowly, Visual Basic began to move in. My clients began to question the life-span of a proprietary tool like Borland Pascal, and even Borland in itself was under fire. I almost felt myself forced to consider C++ as an alternative language, when Borland started to work on the development environment that really changed my life: Borland Delphi. Not just RAD, but 2-way RAD combined with Database Connectivity and a the fastest native Windows compiler you could get! And it didn't end with just Delphi, because just over a year later, Borland released a 32-bit version of Delphi for Win32. And today, Delphi 5 is one of the major development environments for Windows. Sure, we still have competition from Visual Basic, but who still uses PowerBuilder or Visual Objects?

In 1999, Borland hosted a Linux Survey (answered by tens of thousands of developers), and announced its commitment to Linux. This was the start of a new revolution. Not just Linux, which is a power in itself that cannot be stopped, but Rapid Application Development for Linux. Codename Project Kylix. The goal of Project Kylix was to produce a high-performance Rapid Application Development (RAD) development tool for Linux. RAD here meant component-based, two-way visual development of your user-interface (GUI), database, internet, and server applications. Development tool meant a high-speed native Delphi/C/C++ compiler for Linux. Project Kylix would also simplify the porting of existing Delphi and C++Builder applications between Windows and Linux. The latter is realised by the new cross-platform component library called CLX, which will be available in the next version of Delphi for Windows as well. So pay attention when reading future CLX articles in this newsletter, because this is the way of the future. Especially dbExpress, the CLX data access layer is a powerful new technology for cross-platform database development. And in the Server Edition of Kylix, we have the NetCLX with Apache CGI and DSO support to develop powerful web server applications.

But that's not all. Kylix bring more than just a native RAD IDE to the Linux world. For example: Kylix brings a whole army of skilled Delphi developers, and their applications which can be ported from Windows to the Linux world (just like I did with most of my old DOS utilities when Turbo Pascal for Windows shipped). And to make all this possible, let's not forget the over 500 third-party tool and component vendors, who've been working with Kylix pre-released since the Kylix Kick Start on March 20th of last year 2000. If that's not a boost for the Linux world, I don't know what it...

Kylix was launched on January 21st, 2001, and has been available since March 2001. By the time you read this, the Open Source edition of Kylix should be available as well. Yes, the RAD Revolution for Linux is here. And this time, we won't see Visual Basic, but we welcome all Visual Basic developers (and PowerBuilder or Visual Object developers). Welcome to the world of Kylix. Welcome to native RAD on Linux.