Delphi OplossingsCourant

Vol 3. No. 3. Een kwartaalpublicatie van **Bob Swart** - namens het voormalig **K**ylix/**D**elphi **O**plossings**C**entrum (KDOC)

Helmond, 29 oktober 2001,

Er is de afgelopen paar maanden heel wat gebeurd. En dan bedoel ik nog niet eens de verschrikkingen die de wereld op 11 september hebben getroffen (waar geen woorden voor zijn).
Dichter bij huis, is per 1 juli het **K**ylix/**D**elphi **O**plossings**C**entrum (KDOC) ontbonden, met als gevolg dat ik ontslag heb genomen en inmiddels bezig ben met de voorbereidingen voor de start van een eenmanszaak per **1-1-2002** (op zolder in Helmond). Ik ga me met name richten op het schrijven van artikelen en boeken, het geven van trainingen (http://www.drbob42.nl/training), en heb daarnaast nog enkele dagen per maand beschikbaar voor advies en ondersteuning bij bouwwerkzaamheden (met Delphi of Kylix).
Ik zal de **D**elphi **O**plossings**C**ourant (DOC) nu op persoonlijke titel voortzetten. Aan artikelen geen gebrek (één van de redenen dat ik ontslag heb genomen is dat ik nog meer wil gaan schrijven).

In dit nummer wil ik vooral even stilstaan bij de nieuwste versie van **Kylix** die op 23 oktober werd aangekondigd (en waarvan de Kylix 2 Enterprise 60-dagen trial editie inmiddels beschikbaar is om de downloaden). Kylix 2 lijkt in vele opzichten op Delphi 6 - zeker als je kijkt naar nieuwe zaken als DataSnap, WebSnap, BizSnap en XML/SOAP.
Vanaf pagina 2 zal ik laten zien dat we nu voor het eerst een gedistribueerde toepassing kunnen schrijven waarbij het ene deel in Delphi 6 is geschreven en het andere deel in Kylix 2. Een cross-language cross-platform oplossing dus!

Wie interesse heeft in een gratis abonnement op de Delphi OplossingsCourant kan een mailtje sturen naar b.swart@chello.nl. Je krijgt dan ieder kwartaal een mailtje met de inhoudsopgave zodra het nieuwe nummer beschikbaar is.

Inhoudsopgave

De Delphi OplossingsCourant (DOC) is een productie van Bob Swart met dank aan Arjan Jansen en Micha Somers.

Eindredactie: Bob Swart e-mail: b.swart@chello.nl

Kylix 2

Op 23 oktober vond de introduktie plaats van Borland® Kylix™ 2, met als een van belangrijkste nieuwe features de ondersteuning voor het gebruik en de bouw van Web Services onder Linux. Een onderzoek van Evans Data geeft aan dat 70 procent van de Linux ontwikkelaars van menig is dat Web Services de Internet applicaties van de toekomst zijn. Kylix 2 gebruikt Web Services en XML technologieën om de e-business capaciteiten van Linux en Apache™ uit te breiden. Door Kylix 2 met Borland® Delphi™ 6 te combineren, kunnen bedrijven cross-platform, hoogwaardige applicaties ontwikkelen die integreren met de industriële standaarden voor Web Services: SOAP, XML en WSDL en XSL. Borland is momenteel het enige bedrijf dat Web Services oplossingen voor zowel Windows en Linux omgevingen biedt.

Kylix 2 in Nederland

Alhoewel er (voor zover ik weet) geen officiele lancering van Borland Kylix 2 in Nederland zal plaatsvinden, zal ik op **woensdag 28 november** een demonstratie van en met Kylix 2 Enterprise geven op het Open Source plein van de Internet in Business beurs in de RAI te Amsterdam.
Daarnaast verzorg ik op **vrijdag 21 december** een eerste **Kylix 2 Clinic** in Eindhoven voor de speciale prijs van slechts 100 gulden per persoon (en dit is zelfs inclusief het zelfgeschreven cursusboek).
Zie http://www.drbob42.nl/training voor meer informatie, of stuur een mailtje voor details.

SOAP & DataSnap

In het vorige nummer van de DOC Courant heb ik al iets geschreven over Web Services in Delphi 6. Inmiddels heb ik al meerdere artikelen over dit onderwerp geschreven, en een nieuwe sectie op mijn website geopend, genaamd Dr.Bob's SOAP Bubbles - zie http://www.drbob42.com/SOAP

Op deze plek zijn de vijf Web Services te vinden die ik de afgelopen tijd geïmplementeerd heb (waaronder eentje die getallen naar romeinse cijfers omzet (en terug), eentje die getallen naar nederlandstalige woorden omzet, en eentje die boter-kaas-en-eieren speelt). Deze keer wil ik echter een bijzonder web service maken, namelijk eentje die vanuit een Kylix toepassing een dataset kan doorgeven aan een Delphi toepassing. Een gedistribueerde toepassing dus - het gebied van DataSnap - maar in dit geval nog cross-language en cross-platform ook! De DataSnap technologie in Delphi 6 bevat een aantal verschillende manier om een verbinding (connection) te maken tussen een Server en een Client, zoals DCom, CORBA, sockets (TCP/IP) en HTTP. Echter voor een cross-platform verbinding is een nieuw component geïntroduceerd: de SoapConnection (die zit dan ook zowel in Kylix 2 als in Delphi 6), en die gaan we dan ook deze keer gebruiken.

Web Services

Een "web service" is niets anders dan een web server toepassing waarvan de interface methoden via SOAP (Simple Object Access Protocol) te benaderen zijn. Communicatie gaat via HTTP en de beschrijving van de interfaces wordt in WSDL (Web Service Description Language) gedaan: een op XML gebaseerde taal.

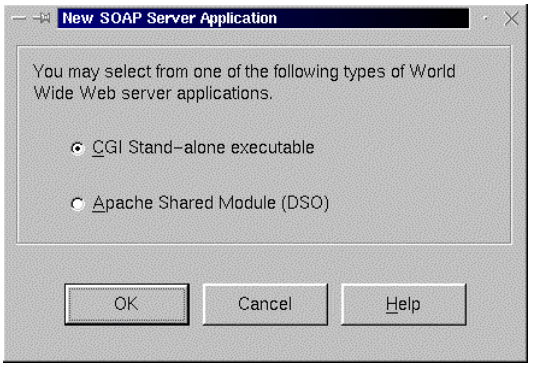
Een client van een web service maakt gebruik van de beschrijving van het interface (in het WSDL document) om uit te vinden hoe de methoden aangeroepen kunnen worden (hoe ze heten, welke parameters ze willen, etc.).

Met behulp van een HTTPRIO (een Remote Invokable Object via HTTP) kan een client een soort (lokale) "instantie" maken van deze (remote) web service en daar vervolgens de methoden van aanroepen.

SOAP Server (Kylix2)

We beginnen met het bouwen van de SOAP Server. En omdat ik nog niet heb laten zien dat we met Kylix 2 daadwerkelijk web services kunnen bouwen, wil ik deze SOAP Server in Kylix 2 Enterprise bouwen (ik maak hierbij gebruik van de 60-dagen trial editie van Kylix 2 Enterprise die te downloaden is van de Borland website - zie ook http://www.drbob42.com/Kylix).

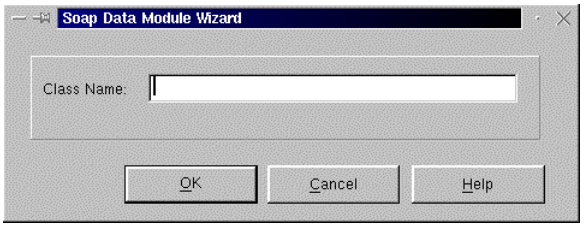
Als eerste moeten we binnen Kylix 2 Enterprise een nieuwe Soap Server toepassing beginnen. Doe *File* | *New*, en ga naar de Soap tab van de Object Repository. Kies voor de Soap Server Application, en klik op OK. De volgende dialoog verschijnt, met de keuze uit een CGI stand-alone executable en een Apache Shared Module (DSO).



Ik kies hier voor een CGI stand-alone executable, alhoewel het voor "real-world" SOAP Servers natuurlijk beter is om ze als Apache DSO module te ontwikkelen, omdat deze niet voor ieder request geladen hoeven te worden.

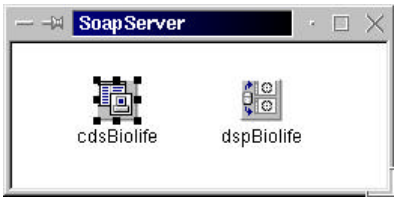
Het resultaat is in beide gevallen een web module met daarop al meteen drie componenten, namelijk HTTPSoapDispatcher, HTTPSoapPascalInvoker en WSDLHTMLPublish. Bewaar de web module als SWebMod.pas en het project als SoapServer42.dpr.

Bovenstaande SOAP Web Module is nodig voor een Web Service toepassing. Echter, ik wil deze keer niet zomaar een web service bouwen, maar een DataSnap web service. En daar heb ik ook nog een SOAP Data Module voor nodig. Doe *File* | *New*, gaan naar de Web Services tab en kies de Soap Data Module. De volgende dialoog verschijnt:

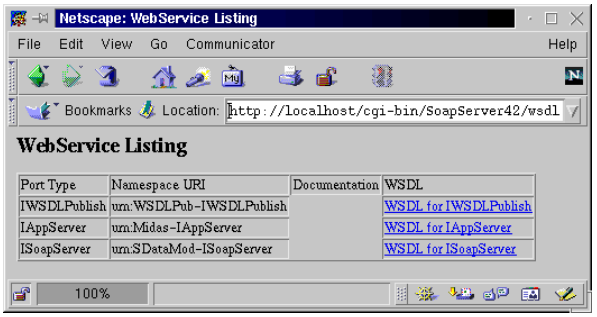


Vul hier de naam SoapServer in, en druk op OK om de SOAP Data Module te krijgen. Bewaar deze in bestand SDataMod.pas.

We kunnen van alles met deze SOAP Data Module doen, maar het uiteindelijke doel is om één of meerdere datasets te exporteren. De eenvoudigste manier om dat te doen is met een ClientDataSet (cdsBiolife) en een DataSetProvider (dspBiolife). Tot slot laat ik de FileName van cdsBiolife naar de biolife.xml tabel wijzen (in MyBase XML formaat).



De tabel hoeft niet geopend te zijn (dat kan de DataSetProvider verzorgen als er een verzoek om data binnenkomt). Rest me nu nog het compileren en deployen van het SoapServer42 project. Het eerste duur twee tellen, en ook het tweede kost weinig moeite, al hoe je hiervoor wel een web server op je Linux machine hebben staan, zoals Apache. Die zit vrijwel bij iedere Linux distributie dus dat zal geen probleem zijn. De SoapServer42 toepassing moet in de cgi-bin directory neergezet worden (in mijn geval de /home/httpd/cgi-bin directory) en kan vervolgens getest worden in een browser zoals Netscape met de string /wsdl achter de URL:



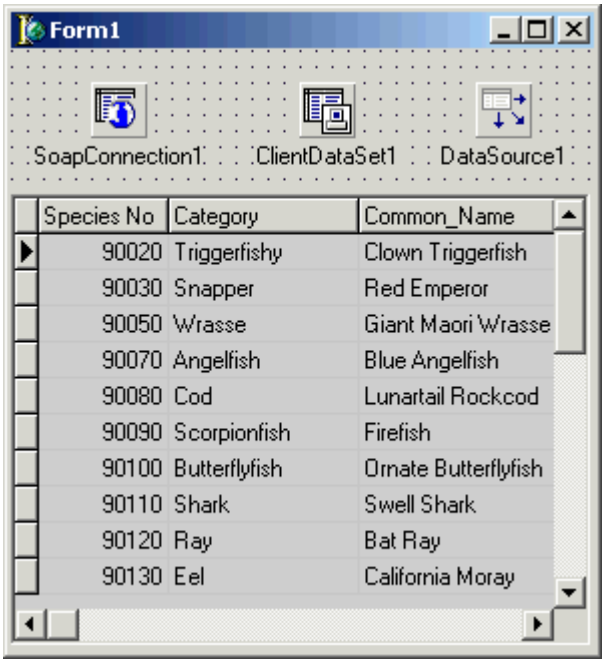
Zoals je ziet vinden we drie WSDL "interfaces" in SoapServer42. De eerste (IWSDLPublish) krijg je altijd bij een Delphi of Kylix web service, de tweede (IAppServer) krijg je omdat we een Remote (DataSnap) Data Module gemaakt hebben, en de derde is specifiek ons ISoapServer interface (afgeleid van de IAppServer).
Daadwerkelijk gebruikmaken van de Remote Data Module, via het SOAP protocol, kunnen we echter pas als we een nieuw Delphi of Kylix project starten. En omdat we tot nu toe in Kylix 2 Enterprise bezig waren, lijkt het me het leukst om de SOAP Client in Delphi 6 Enterprise te bouwen.

SOAP Client (Delphi 6)

Er zal één belangrijk verschil zijn met de test die we zojuist onder Linux hebben gedaan, en dat is de waarde van localhost. Als we een SOAP Server toepassing vanaf een andere machine (dus echt remote) willen benaderen zullen we de machine-naam of het IP-adres van de server moeten weten en moeten gebruiken. Ook dit kun je weer eerst testen in een browser, zodat je zeker weet dat de URL http://linux/cgi-bin/SoapServer42/wsdl ook daadwerkelijk bereikbaar is. Als dat het geval is, dan kun je Delphi 6 Enterprise starten en een nieuw project beginnen. Het maakt niet uit wat voor soort project - we gaan alleen maar een remote dataset ophalen en gebruiken.
De component die de SOAP verbinding tussen de Delphi toepassing en de Soap Server op Linux maakt heet de SoapConnection component, en is te vinden op de WebServices tab van zowel Delphi 6 als Kylix 2. Deze component is er erg goed in om je op het verkeerde been te zetten, dus let op! Wat je moet doen is de URL property invullen met daarin de vindplaats van de Remote Data Module van de Soap Server toepassing. Volgens de on-line help (van Delphi 6 en Kylix 2) zou de waarde van deze URL property de volgende vorm hebben: http://DataHost.org/scripts/AppServer.dll/SOAP
Helaas is dat helemaal fout[1]. De waarde van de URL property moet namelijk niet alleen op /SOAP eindigen, maar moet daarachter nog de naam van het interface van de Remote Data Module hebben, zoals ISoapServer in ons geval. Ook IAppServer kan hier gebruikt worden, alhoewel ik zelf liever expliciet de naam van de Remote Data Module zelf meegeef. In ons geval zal de URL dus als volgt zijn: http://linux/cgi-bin/SoapServer42/soap/ISoapServer
Het bepalen van de juiste URL kan dus even tijd kosten. De SoapConnection component zelf biedt hierbij echter geen ondersteuning. Zelfs bij een URL die niet af is, zoals http://linux is het mogelijk om de Connected property op True te zetten, waarbij de Agent property automatisch de waarde Borland SOAP 1.1 krijgt. Dit is leuk, maar als de URL fout is (of zelfs onzin is), dan had ik toch wel een foutmelding verwacht. Nu loop je het gevaar dat je denkt dat alles goed is, terwijl het later toch niet goed werkt. Als je de juiste URL hebt ingevuld kun je Connected op True zetten.

[1] Bij een Delphi 6 WebApp Debugger SOAP Server kan de URL wel op /SOAP eindigen, maar zal het "server" deel van de URL van de vorm http://localhost:1024 zijn.

De volgende stap bestaat uit het neerzetten van een ClientDataSet component dat de inhoud van de remote biolife.xml tabel op gaat halen. Laat de RemoteServer property van de ClientDataSet wijzen naar de SoapConnection component. Nu wordt het spannend: open de combobox voor de ProviderName property van de ClientDataSet. Als de URL goed was en de SOAP verbinding met de SoapServer42 toepassing daadwerkelijk gelegd kan worden, dan zal hierin de naam van de (remote) DataSetProvider verschijnen, namelijk de dspBiolife die we eerder onder Kylix hadden aangemaakt.
Zet nu de Active property van de ClientDataSet op True (dit kan even duren), en als er dan geen verdere foutmeldingen optreden heb je zojuist de inhoud van de biolife.xml tabel via SOAP van de Linux machine naar de lokale Windows machine opgehaald. Je kan de data zelfs al tijdens design-time laten zien met behulp van een DataSource en een DBGrid.

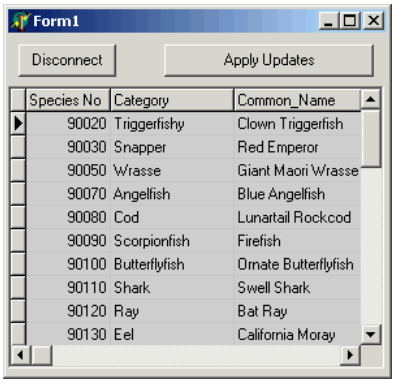


In feit zijn we nu klaar. Echter, voor een mooie afwerking moeten we nog wel een paar extra stappen zetten. Allereerst is het altijd verstandig om de zgn. "Connection" componenten (en ook de ClientDataSet componenten) niet aktief te laten zijn tijdens design-time. Het kan namelijk altijd gebeuren dat je later nog eens een dergelijk project opent, er een verbinding met een server gemaakt moet worden, wat wellicht niet lukt op dat moment, waardoor het laden van het project niet zonder meer lukt.

Omdat het ophalen van de data via SOAP nogal lang kan duren, moeten we altijd zoveel mogelijk proberen te voorkomen dat de ClientDataSet (en de SOAPConnection component) al "aktief" zijn bij het opstarten van de toepassing. Een button zorgt voor het maken van de verbinding.
Los van het "op verzoek" verbinding maken om de data op te halen, zullen we ook iets bijzonders moeten doen om eventuele wijzigingen in de data weer "weg te schrijven" en op te sturen naar de SOAP Server toepassing. Zoals gebruikelijk bij DataSnap toepassingen, kunnen we ook hier de ApplyUpdates methode van de ClientDataSet voor gebruiken. Hiervoor gebruik ik een button met als caption "ApplyUpdates". De source code van beide event handlers is als volgt:

```
procedure TForm1.btnConnClick(Sender: TObject);
begin
  if (Sender AS TButton).Caption = 'Connect' then
  try
    (Sender AS TButton).Enabled := False;
    ClientDataSet1.Open;
    (Sender AS TButton).Caption := 'Disconnect'
  finally
    (Sender AS TButton).Enabled := True
  end
  else
  begin
    ClientDataSet1.Close;
    (Sender AS TButton).Caption := 'Connect'
  end
end;

procedure TForm1.btnApplyClick(Sender: TObject);
begin
  ClientDataSet1.ApplyUpdates(-1)
end;
```



Een druk op de button ApplyUpdates zal nu de wijzigingen doorsturen naar de SoapServer42 toepassing (die nog steeds onder Linux draait).
Wie meer wil weten over DataSnap via SOAP moet binnenkort mijn SOAP Bubbles site maar eens bezoeken te http://www.drbob42.com/SOAP waar ik binnenkort meer artikelen zal publiceren over dit onderwerp (met Delphi 6 en Kylix 2).

WebSnap (in Delphi 6 en Kylix 2)

In dit artikel wil ik een introduktie geven van WebSnap: de nieuwe verzameling wizards en componenten waarmee je vanuit Delphi 6 (en Kylix 2 Enterprise) websites kunt bouwen. Een van de belangrijkste doelstellingen van WebSnap is om een omgeving te scheppen waarin Delphi en Kylix ontwikkelaars kunnen samenwerken met de website designers (die waarschijnlijk geen of weinig verstand hebben van (Object) Pascal, maar wel van JavaScript, en die tools als DreamWeaver of FrontPage gebruiker). Om dit te realiseren is Active Scripting toegevoegd aan Delphi/Kylix (een feature die ook wel server-side scripting genoemd wordt), samen met zogenaamde Script-enabled componenten. De Delphi/Kylix ontwikkelaar kan de bouwstenen opleveren (en een rudimentaire HTML output genereren), terwijl de JavaScript ontwerper er verder mee kan gaan om er iets echt moois van te maken (met behulp van de WebSnap Design Surfaces zoals de Preview, HTML Result, HTML Script, XML en XSL tabs die we straks nog zullen zien).

Het goede nieuws is dat WebSnap gebruik maakt van dezelfde componenten die we als Delphi ontwikkelaar al jaren kenen uit de WebBroker en InternetExpress wereld. WebSnap werkt nog steeds met PageProducers en Web Modules, alhoewel we deze keer behalve Dispatchers ook Adapters gebruiken, en in plaats van één Web Module meerdere Page Modules gebruiken. Een nadeel is echter dat, alhoewel het wel mogelijk is om vanuit een nieuwe WebSnap toepassing de "oude" technieken te gebruiken (bestaande WebBroker en InternetExpress kennis blijft bruikbaar), het niet mogelijk is om een "oude" (legacy) WebBroker of InternetExpress toepassing zomaar om te zetten naar WebSnap of te laten profiteren van de WebSnap componenten. Het is als een DVD speler die ook muziek CD's kan afspelen, maar je kan geen DVD in je autoradio/cd-speler afspelen (en da's maar goed ook, denk ik).

Documentatie

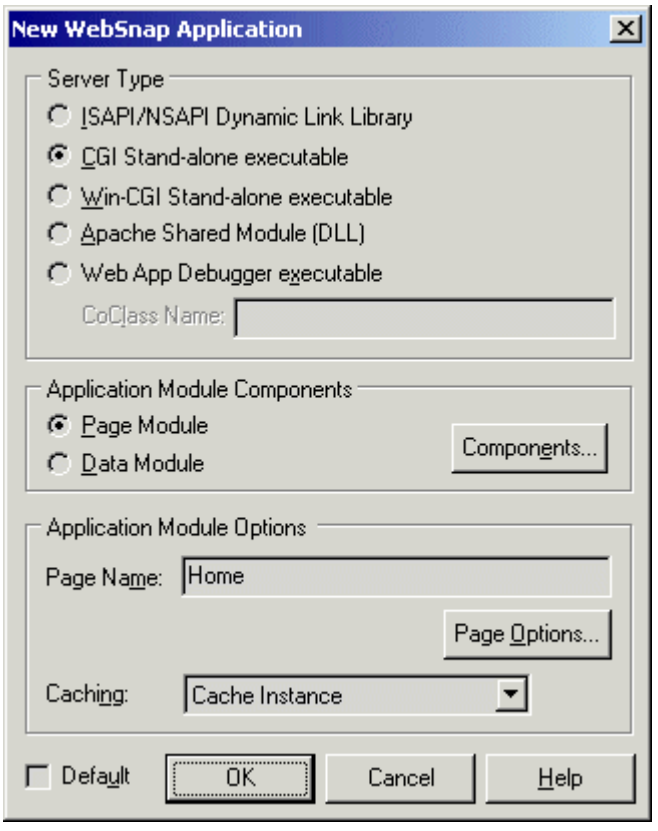
Alhoewel WebSnap geheel nieuw is in Delphi 6, laat de documentatie nogal te wensen over. Er zijn een heleboel verzachtende omstandigheden, maar het resultaat is wel dat er vrij weinig op papier staat betreffende de WebSnap tools en de 17 componenten op de WebSnap tab. Gelukkig zijn er inmiddels wat updates op de Borland Community website te vinden.

Demonstratie

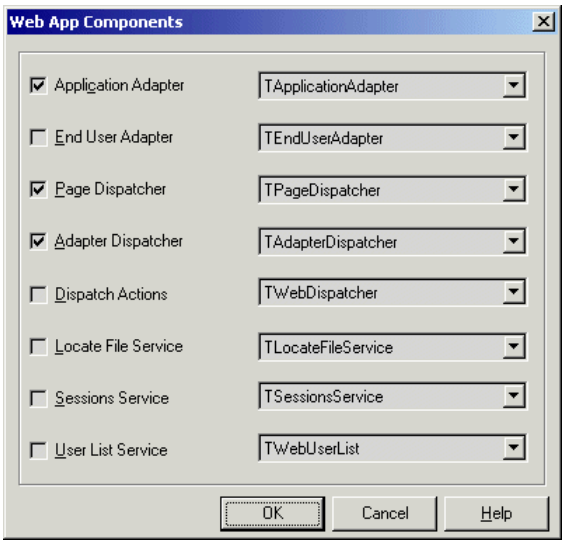
De beste manier om te zien wat WebSnap kan, is om het gewoon maar te doen. Toen Delphi 5 in 1999 beschikbaar kwam heb ik meermalen laten zien hoe je met de (toen) nieuwe InternetExpress de beroemde customer-orders combinatie in een webbrowser kan laten zien. Met WebSnap zien we een aantal bekende dingen terug, maar meer nog nieuwe zaken en ontwikkelingen. Ik zal overigens in dit artikel Delphi 6 Enterprise gebruiken, maar de meeste zaken werken op een vergelijkbare manier in Kylix 2 Enterprise.

We starten een WebSnap application door *File* | *New* | *Other* en dan naar de WebSnap tab van de Object Repository te gaan. Daar bevinden zich drie icoontjes: de WebSnap Application, een WebSnap Data Module en een WebSnap Page Module. Van de eerste heb je er maar één nodig, van de tweede meestal ook (tenzij je je gebruik wilt maken van meerdere data modules), en van de derde kun je er zoveel gebruiken als je maar wilt. In principe kun je iedere pagina van de (dynamische) website een eigen Page Module geven.

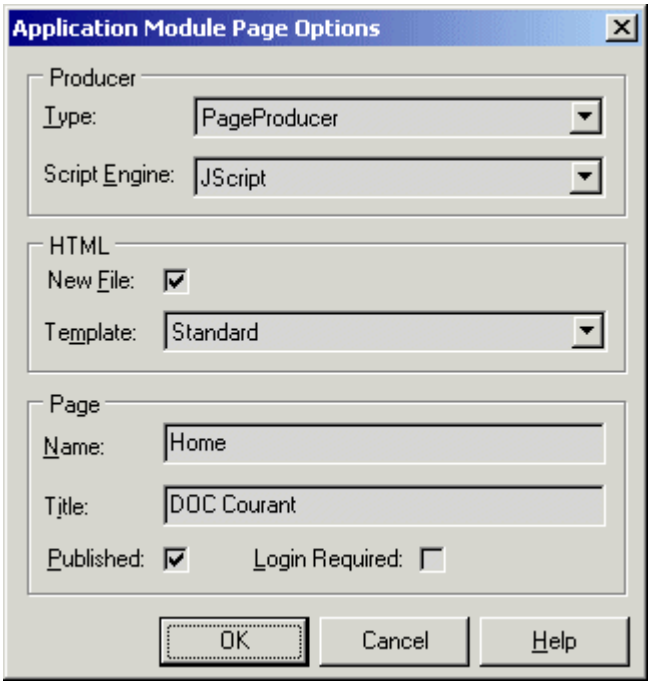
We beginnen met de WebSnap Application, en krijgen de volgende dialoog (wie Delphi6 of Kylix 2 Enterprise heeft kan nu "meespelen" en zijn eigen keuzes invullen, uiteraard):



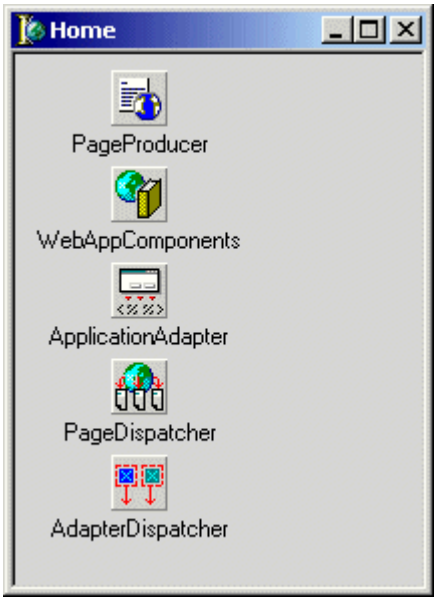
Ik kies nu even voor een CGI toepassing (die is het makkelijkst te testen en weer uit het geheugen te halen). Voor een productieomgeving zou ik altijd voor een ISAPI/NSAPI of Apache DLL kiezen (in Kylix 2 Enterprise voor een Apache DSO Shared Module). Merk op dat we in Delphi 6 nu ook een speciale Web App Debugger executable kunnen kiezen: daarmee krijgen we een COM schil om de web module heen die je kan gebruiken om de web server toepassing in de Delphi IDE te debuggen. Zit ook in Delphi 6 Professional, dus zou wel eens het eind van IntraBob kunnen betekenen, maar da's een ander verhaal.
De Components knop in de "Application Module Components" kan gebruikt worden om aan te geven welke hulp-componenten we nodig verwachten te hebben. Per optie kunnen we uit een drop-down combobox kiezen. Op dit moment zijn de keuzes nog beperkt (de EndUserAdapter kan een EndUserSessionAdapter worden), maar zodra je zelf speciale componenten maakt voor WebSnap dan komen die ook hier in de comboboxen te hangen.



Default staan alleen de eerste, derde en vierde checkbox aan; de rest uit. De tweede knop op de WebSnap Aplication Wizard dialoog kan gebruikt worden om de Page Options op te geven. Hierin kunnen we aangeven wat voor soort PageProducer gebruikt moet worden (de keuze bestaat uit PageProducer, DataSetPageproducer, AdapterPageProducer, INetXPageProducer of de XSLPageProducer). Ook kan je hier de script-engine kiezen (default op JScript) en aangeven wat voor soort template gemaakt moet worden. Het enige dat ik echter invul is de naam van de pagina (Home) en de titel (DOC Courant).

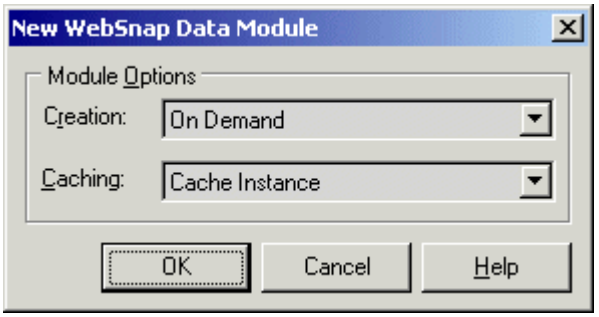


De optie Published zorgt ervoor dat we de pagina ook echt zien. en de Login optie zou ervoor zorgen dat de bezoekers moeten inloggen voordat ze de pagina mogen zien (de Home pagina wil je altijd Published hebben, en over het algemeen zonder directe Login - die volgt indien nodig pas bij de overige pagina's).
Als we dan uiteindelijk op de OK knop van de WebSnap Application Wizard dialoog klikken krijgen we de WebSnap web module, met daarin een vijftal componenten. De PageProducer, de WebAppComponents (met properties die wijzen naar de rest), en de drie componenten die we in de Web App Components dialoog hadden geselecteerd.



WebSnapDataMod

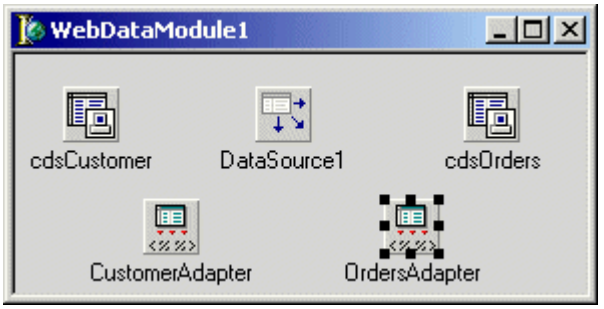
Ik bewaar mijn project in WebSnapCGI.dpr, en de web module in WebSnapWebMod.pas.
Nu is het tijd om één (of meerdere) data module toe te voegen aan de WebSnap toepassing. Ook hiervoor moeten we in de WebSnap pagina van de Object Repository zijn, en dan voor de WebSnap Data Module kiezen:



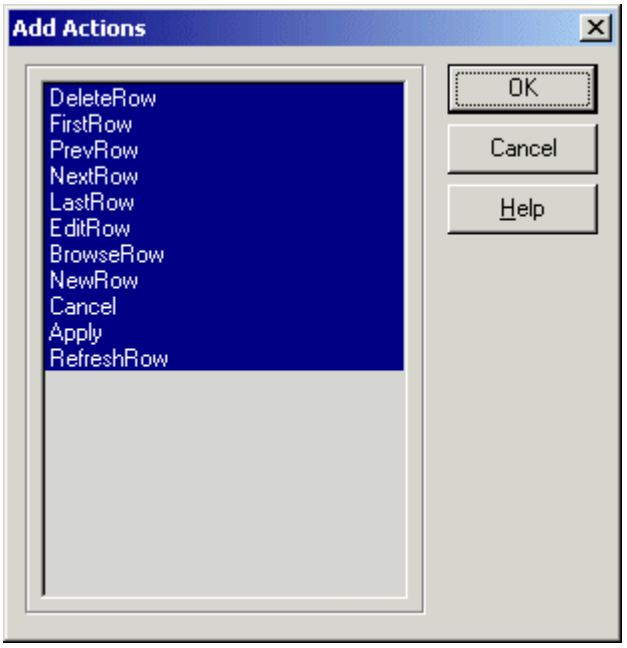
Behalve de keuze "On Demand" kunnen we de data module ook "Always" aanwezig laten zijn. Voor de CGI toepassing die ik nu aan het bouwen ben maakt het niet zoveel uit (die wordt toch na iedere request weer uit de lucht gehaald), maar voor een ISAPI of Apache web toepassing is deze optie wel van belang. Net als de Caching optie, die behalve de waarde "Cache Instance" ook "Destroy Instance" kan krijgen. Let erop dat bij de keuze "Cache Instance" de data module niet opnieuw in de OnCreate event handler zal komen, en derhalve zijn oude "status" blijft houden (dus queries en/of tabellen blijven gepositioneerd op de plek waar de vorige gebruiker deze heeft achtergelaten).
Nadat ik op OK heb geklikt, bewaar ik de nieuwe data module in bestand WebSnapDataMod.pas. Ik plaats er tevens twee ClientDataSet components op (cdsCustomer en cdsOrders), en laat de ene verwijzen naar customer.xml en de tweede naar orders.xml (de aloude DBDEMOS tabellen zijn nu ook in binaire .cds of MyBase .xml formaat beschikbaar). Een master-detail relatie is zo gemaakt met behulp van een DataSource component dat naar cdsCustomers wijst, en als MasterSource door cdsOrders wordt gebruikt. De MasterFields property kan vervolgens gebruikt worden om de CustNo velden aan elkaar te koppelen:
Met WebSnap kunnen we stateless web server toepassingen maken. Dat wil dus zeggen dat de WebSnap web server toepassing niet voor alle clients (browers) gaat bijhouden bij welk record ze zijn. Dat moet de client zelf doen, en ook zelf doorgeven aan de WebSnap server. Voor de beide cdsCustomer en cdsOrders datasets moeten we hiertoe duidelijk aangeven wat de key velden zijn. Selecteer eerst cdsCustomer, klik met de rechter muisknop om de Fields Editor te krijgen, en klik weer met de rechter muisknop om Add all Fields te kunnen kiezen. Nu zijn alle velden van de customer tabel aanwezig. Het CustNo veld is in dit geval de unieke key. Selecteer dit veld, en ga naar de Object Inspector waar je van de ProviderFlags property de sub-property pfInKey op True moet zetten. Doe hetzelfde voor de cdsOrders dataset, maar kies daar voor het OrderNo veld. De WebSnap toepassing zal nu altijd de juiste velden meegeven om de records waar we mee willen werken uniek te kunnen identificeren.

Adapters

We moeten nu nog een ding doen, en dat betreft het plaatsen van twee Adapter componenten (beide een DataSetAdapter in dit geval - het derde component van de WebSnap tab) om de velden uit zowel de Customer als de Orders dataset beschikbaar te maken (als AdapterFields) voor de rest van de WebSnap toepassing. Ondanks het feit dat er al een master-detail relatie gebruikt wordt, zullen we twee verschillende DataSetAdapters moeten gebruiken: eentje voor de cdsCustomer (CustomerAdapter) en eentje voor cdsOrders (OrdersAdapter). De master-detail relatie tussen de twee Adapters kunnen we vervolgens weer aangeven door CustomerAdapter te selecteren als MasterAdapter voor de OrdersAdapter.



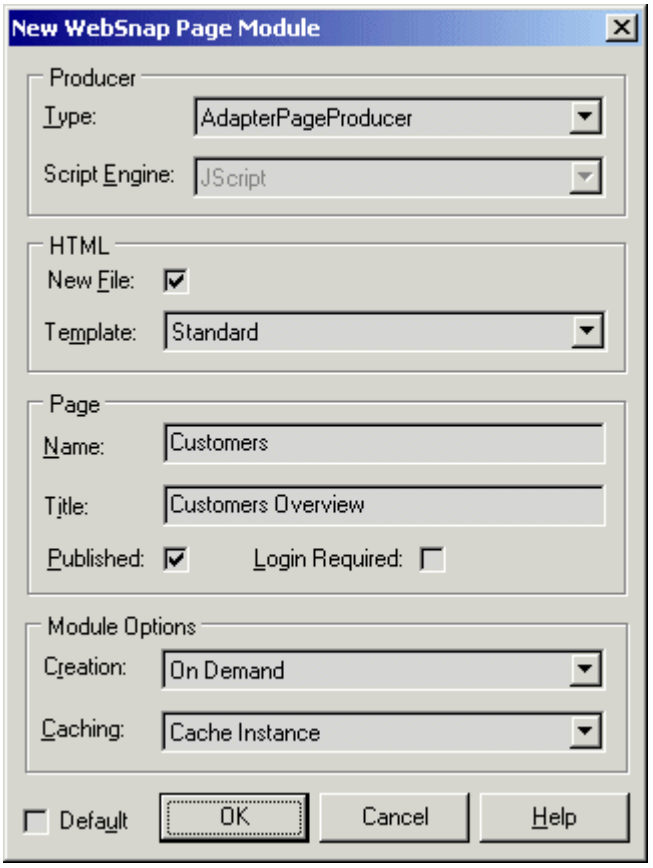
In de Object Treeview kunnen we de beide DataSetAdapter componenten openklappen, en dan zien we de Actions en Fields subproperties. Door met de rechter muisknop op Actions te klikken kunnen we uit alle 11 mogelijke actions juist alleen diegene toevoegen die we toe willen staan op de velden uit de Customer en Orders datasets.



Stel dat je bijvoorbeeld nu al weet dat je de data nooit wilt laten editen, dan is dit het moment om te zorgen dat de EditRow action niet wordt toegevoegd aan de lijst. In dit geval wil ik alles toestaan, dus ik had net zo goed "Add All Actions" kunnen kiezen.
We zijn nu klaar met de data module, die ik onder de naam WebSnapDataMod.pas bewaar.

Eerste WebSnapPageMod

Nu we de WebSnap toepassing en data module hebben gemaakt wordt het tijd voor enkele WebSnap Page Modules: de hulpmiddelen om de daadwerkelijke (dynamische) webpagina's te genereren. Wederom naar de WebSnap tab van de Object Repository om deze keer voor WebSnap Page Module te kiezen:
Deze dialoog lijkt erg op die van de nieuwe WebSnap Application, maar heeft geen speciale Component en Page Options knoppen meer. We kunnen wel aangeven wat voor soort PageProducer component gebruikt moet worden voor deze pagina. In de screenshot (in de rechter kolom) is te zien dat ik een AdapterPageProducer heb gekozen (die kan automatisch de scripting code genereren voor een Adapter), zodat ik me dus zelf niet zoveel bezig hoef te houden met het schrijven van het active script (dat zal ik een andere keer laten zien). Wie dat nu wel wil, had ook een andere PageProducer kunnen kiezen.
Daarnaast heb ik ook de naam (Customers) en Titel (Customers Overview) van de pagina opgegeven.

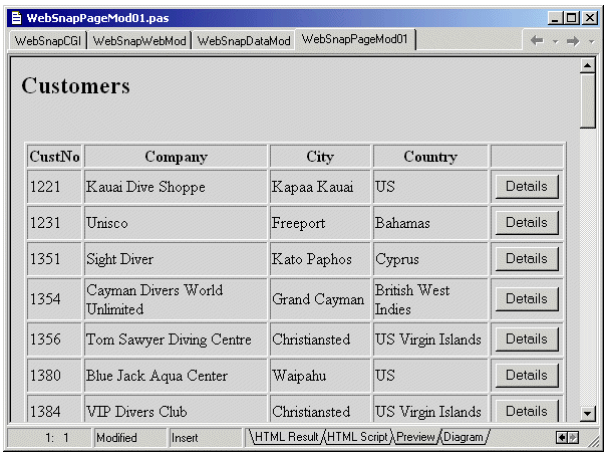


Als je op OK klikt krijg je een nieuwe Page Module, met automatisch al een AdapterPageProducer component (zoals we aangaven). Voor we verder gaan wil ik deze unit als WebSnapPageMod01.pas opslaan (er komt straks ook nog een tweede).
Verder zullen we gebruik willen maken van de CustomerAdapter in de WebSnap Data Module, dus moeten we de unit WebSnapDataMod aan de uses clause toevoegen (bijvoorbeeld met *File | Use Unit*). Als dat geregeld is, kunnen we de AdapterPageProducer in de Object Treeview selecteren en openklappen. De property WebPageItems verschijnt (wat bij sommige een herinnering kan oproepen aan InternetExpress). Vanaf hier kunnen we nieuwe componenten toevoegen door met de rechter muisknop op een component in de Object Treeview te klikken en voor "New Component" te kiezen. Dit geldt voor de WebPageItems en alles wat daar straks onder komt. We beginnen met een AdapterForm direct onder de WebPageItems, en op dit AdapterForm een AdapterGrid. Hier wil ik een overzicht geven van de Customers - zonder orders.
Wie in de tussentijd een blik op de Code Editor heeft geworpen zal het niet ontgaan zijn dat er in plaats van een Code tab (en een Diagram tab die ook nieuw is in Delphi 6) er plotseling een aantal

extra tabs te zien zijn, zoals WebPageMod01.pas, WebPageMod01.html, HTML Result, HTML Script en Preview. De eerste twee tabs zouden nog Unit1.pas en Unit1.html kunnen heten (de namen van de unit voordat ze bewaard werden). Da's een klein foutje in Delphi 6, maar je kunt je juiste namen in de tabs krijgen door even naar een ander source bestand te gaan en weer terug naar de WebPageMod01 (zodat de tabs hun nieuwe - juiste - namen krijgen). De inhoud van de eerste twee tabs is overigens gewoon te editen, de derde en vierde zijn echter read-only en laten zien hoe de HTML en scripting die gegenereerd worden eruit komen te zien. De vijfde tab geeft een echte preview alsof we een browser gebruiken. Dit is de pagina die op dit moment heel duidelijk een design-time warning geeft dat de Adapter property van de AdapterGrid1 nog nil is. Dat kunnen we oplossen door terug te gaan naar de Object Treeview, de AdapterGrid1 te selecteren en in de Adapter property de CustomerAdapter van WebSnapDataMod te kiezen (die kunnen we pas kiezen als de WebSnapDataMod in de uses clause staat).

Als je vervolgens ook tijdens design-time echte data wilt zien, moet je zorgen dat cdsCustomer geopend is (Active property op True). We zien nu alle 13 velden van cdsCustomer, en dat vind ik persoonlijk wat veel voor een gewone overview waarin we alleen maar de juiste customer willen selecteren om vervolgens pas alle details te zien (inclusief bijbehorende orders). Kortom: ik wil velden verwijderen. Daarvoor moet ik niet terug naar de WebSnap Data Module (voor wie ze uit de cdsCustomer wil verwijderen), want dan heb ik ze straks bij de detail view ook niet meer. Nee, ik moet gewoon de AdapterGrid weer selecteren, met de rechter muisknop klikken en kiezen voor "Add All Columns". Als ik ze eenmaal alle 13 zie, kan ik er rustig enkele verwijderen (zoals het adres enzo). Tip: om een Col te verwijderen in de TreeView moet je op de "delete" knop in de TreeView klikken, of met de rechter muisknop op de columnnaam klikken en dan via Edit de keuze Delete maken. De delete toets werkt pas (matig) na het installeren van de Delphi 6 Update 1 patch.
Uiteindelijk hou ik nog maar vier velden over: ColCustNo, ColCompany, ColCity en ColCountry. Ik wil echter nog wel een vijfde column toevoegen in het AdapterGrid, waar ik dan een speciale knop in wil plaatsen die naar de "detail" view gaat.

Dit doe ik door weer met de rechter muisknop op de AdapterGrid te klikken en voor New Component te kiezen. De keuze valt op een AdapterCommandColumn, die default met alle mogelijke command knoppen begint (DeleteRow, FirstRow, PrevRow, NextRow, LastRow, EditRow, BrowseRow, NewRow en RefreshRow). Ik wil er eigenlijk maar eentje zien, dus klik ik weer met de rechter muisknop, deze keer op de AdapterCommandColumn en kies weer New Component, en kies deze keer één enkele AdapterActionButton. Zodra ik deze keuze heb gemaakt komt er weer een design-time warning in beeld: de AdapterActionButton heeft een lege Action! Door voor de EditRow action te kiezen lossen we dat op. Ik zou dan wel de caption veranderen in "Details" of "Edit Details" of zoiets (dat staat beter dan "EditRow").



Tot slot kunnen we via de PageName property aangeven naar welke pagina er gesprongen moet worden als de gebruiker daadwerkelijk op de "Details" knop drukt. Vul hier CustomerOrders in. Die Page Module bestaat nog niet, maar die gaan we dan ook meteen maken. Ik had eigenlijk wel verwacht dat ik uit een lijst met bestaande PageNames zou kunnen kiezen (maar misschien is dat iets voor een custom property editor die ik hier binnenkort voor kan schrijven).

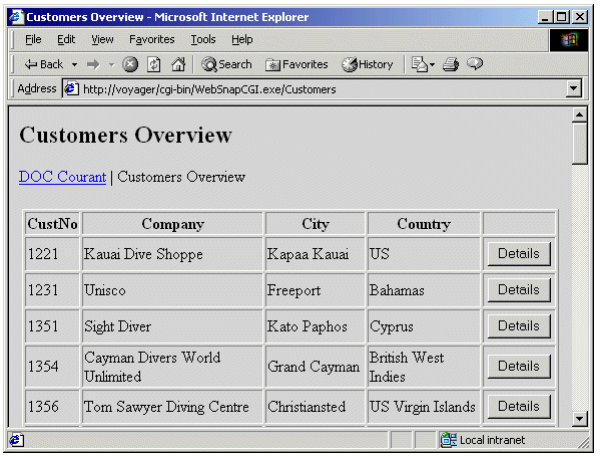
Nog een WebSnapPageMod

Tijd voor de tweede WebSnap Page Module. Gebruik weer dezelfde wizard uit de WebSnap tab van de Object Repository, en kies wederom voor een AdapterPageProducer. Als naam moeten we dus CustomerOrders opgeven, als titel heb ik Customer Orders Detail View gebruikt.
Zorg ervoor dat de Published checkbox uit staat, zodat we niet direct vanuit de Home pagine op deze pagina kunnen komen (we willen de detail

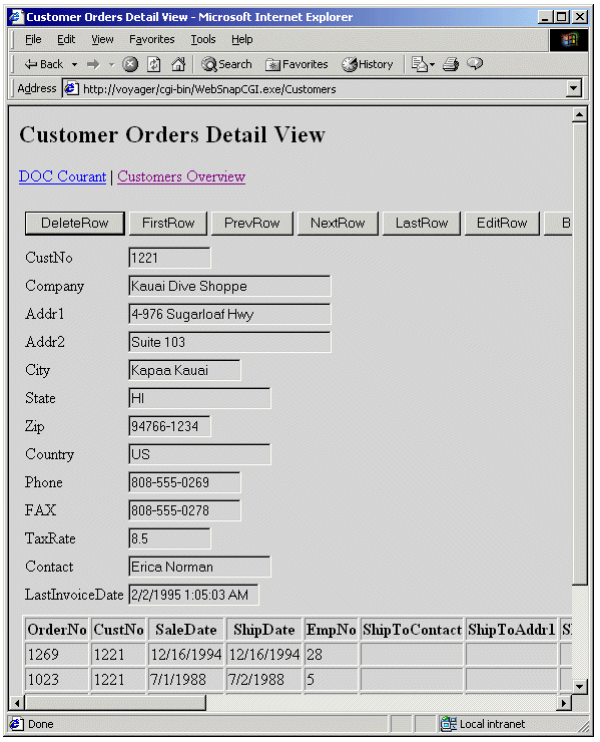
pagina immers alleen toegankelijk laten zijn vanuit de overview pagina)
Ook deze Page Module (WebSnapPageMod02.pas) heeft de WebSnapDataMod nodig in de uses clause om bij de CustomerAdapter en OrdersAdapter te kunnen komen. En wederom kunnen we een min-of-meer visueel ontwerp maken van de webpage door in de Treeview verschillende componenten toe te voegen aan de WebPageItems property. Ik start weer met een AdapterForm, maar deze keer stop ik daar een AdapterCommandGroup, AdapterFieldGroup en AdapterGrid op.
De DisplayComponent property van de AdapterCommandGroup koppel ik aan de AdapterFieldGroup. De Adapter property van de AdapterFieldGroup koppel ik aan de CustomerAdapter van de WebSnap Data Module, en de Adapter property van de AdapterGrid koppel ik aan de OrdersAdapter. Deze keer wil ik in de detail view wel alle velden zien, en ook alle mogelijke akties toestaan (zoals het editen of verwijderen van records).
Als laatste speeltje kun je tijdens design-time vast een voorproefje krijgen van de verschillende manieren waarop de data gerepresenteerd kan worden. Kijk maar eens naar de AdapterMode property van de AdapterFieldGroup component. De mogelijke waarden zijn Browse, Edit, Insert en Query. Omdat wij via de EditRow in deze pagina zullen komen, zou je alvast kunnen vermoeden wat de AdapterMode zal zijn als we voor het eerst op de pagina komen.

Aktie!

We zijn klaar. En hebben niet één regel code geschreven! Alleen maar klikken en koppelen eigenlijk. Even compileren en draaien maar. Of toch niet? We moeten eerst nog even zorgen dat we niet alleen de WebSnapCGI.exe maar ook de html bestanden voor de Home, Customer en CustomerOrders pagina's goed gedeployed worden. Zonder gebruik te willen maken van de LocateFileService component moet je die html bestanden naast de executable in de scripts directory neer moeten zetten.
De eerste pagina is nog niet zo spannend (alleen maar DOC Courant als titel en een enkele link: naar Customers Overview - merk op dat de Customer Orders Detail View niet direkt beschikbaar is). De tweede pagina, Customers Overview is leuker, en geeft ook daadwerkelijk het gewenste Customers Overview:



Als we op een van de Details knoppen drukken krijgen we inderdaad de details van die betreffende rij te zien. Alleen dan wel in Edit mode, want we hebben voor de EditRow aktie gekozen:



De pagina is ook terug te zetten in de Browse mode door op de BrowseRow knop te drukken. En natuurlijk kun je de nodige velden verwijderen als ze in de weg staan. En de titels of captions kun je ook wijzigen. Waar het om gaat is dat we met behulp van WebSnap en Adapter componenten in een paar minuten een eerste webtoepassing in elkaar hebben gedraaid.

Op de Delphi 6 Clinic van 30 november ga ik uiteraard nog veel dieper in op WebSnap. Zie http://www.drbob42.nl/training voor details.

Conferenties en Evenementen

Tot het eind van het jaar zal ik nog een drietal Delphi en Kylix Clinics geven. Allereerst zal ik op **vrijdag 9 november** demonstreren hoe je een echte e-business website kunt bouwen met behulp van een combinatie van WebSnap en BizSnap (XML Document Programming en SOAP).
Op **vrijdag 30 november** volgt dan op speciaal verzoek de herhaling van mijn WebSnap Clinic, waarin alle componenten en hulpmiddelen aan de orde zullen komen (inclusief enkele (nog) niet gedocumenteerde zaken).
Tot slot zal ik op **vrijdag 21 december** het seizoen afsluiten met de eerste Kylix 2 Clinic. Zoals eerder vermeld bevat Kylix 2 Enterprise een hoop nieuwe feaures zoals DataSnap, WebSnap en BizSnap, maar ik zal ook dbExpress en WebBroker (nu in Kylix 2 Professional) aan de orde laten komen.

Los van deze drie Delphi en Kylix Clinics zal ik ook op **woensdag 28 november** een demonstratie van Kylix 2 Enterprise geven op het Open Source plein tijdens de Internet in Business beurs in de RAI te Amsterdam.
Zie http://www.drbob42.com/events voor meer informatie.

Delphi 6 Clinics in 2002

Ook in 2002 zal er weer eens in de drie weken een Delphi 6 Clinic plaatsvinden in Eindhoven. Deze keer niet meer op de vrijdag maar op **donderdag**. De data en onderwerpen zijn als volgt:

10 januari - **dbExpress & DataSnap**
31 januari - **WebBroker & InternetExpress**
21 februari - **WebSnap & Adapters**
14 maart - **BizSnap & Web Services**
4 april - **WebSnap & BizSnap** (in praktijk)

De inschrijving voor het seizoen 2002 begint medio november 2001 (met de gebruikelijke extra kortingen voor wie zich voor eind dit jaar nog inschrijft). Zie http://www.drbob42.nl/training

BorCon 2002

Tot slot zal ik ook aanwezig zijn bij de officiële Borland Conferentie in de USA van 18 t/m 22 mei in Anaheim. Ik zal daar twee presentaties verzorgen (allebei tweemaal) over *Introduction to dbExpress and ClientDataSets* en *Multitier Application Development using DataSnap.*
Zie wederom http://www.drbob42.com/events voor meer informatie.

Boeken en White Papers

Afgezien van de artikelen die ik regelmatig schrijf voor o.a. The Delphi Magazine, Delphi Developer, UK-BUG Developer's Magazine, SDGN Magazine, Blaise en de Borland Community, TechRepublic en DevX websites, schrijf ik ook regelmatig white papers en hoofdstukken voor boeken (over Delphi, Kylix of C++Builder).

Migratie van Delphi 5 naar Kylix

Vlak na het uitkomen van Kylix heb ik een white paper geschreven met daarin aandacht voor het migreren en omzetten van Delphi 5 projecten naar Kylix. Omdat Delphi 5 geen CLX ondersteunt, was dit eenrichtingsverkeer. Inmiddels ben ik bezig met de opvolger van dit white paper, betreffende cross-platform development met zowel Delphi 6 als Kylix 2 (en uiteraard staat het gebruik van CLX daar centraal).
Zie http://www.drbob42.com/Kylix voor details.

Kylix Developer's Guide

Dit boek zal rond eind november / begin december verschijnen. De hoofdauteur is Charlie Calvert, vroeger bij Borland werkzaam. Ik heb voor dit boek al drie hoofdstukken geschreven over web server toepassingen met WebBroker, en zal binnenkort uitbreidingen schrijven over Kylix 2 met WebSnap en BizSnap (zowel XML Document Programming als SOAP).
Zie http://www.drbob42.com/books voor details.

Delphi 6 Developer's Guide

Ook dit boek zal waarschijnlijk pas ergens in December verschijnen. Uiteraard zijn Xavier Pacheco en Steve Teixeira de hoofdauteurs, maar Bob Swart en Micha Somers hebben samen een hoofdstuk geschreven over Active Server Pages and Active Server Objects in Delphi 6.
Zie http://www.drbob42.com/books voor details.

C++Builder 5 Developer's Guide

En tot slot wil ik nog even de aandacht vragen voor een boek over C++Builder 5. Het enige engelstalige boek over C++Builder 5 zelfs. Het is door een team van ruim 30 auteurs is geschreven (ikzelf heb ruim 120 pagina's voor mijn rekening genomen, met name over MIDAS en internet technologieën, en dit is het eerste boek met mijn naam op de omslag).
Zie http://www.drbob42.com/books/bcb5.htm voor de inhoudsopgave, een bespreking en een voorbeeldhoofdstuk.