Delphi OplossingsCourant

Vol. 10. No. 1, Een gratis electronische publicatie van **Bob Swart Training & Consultancy** (**eBob42**) - http://www.eBob42.com

Helmond, 21 februari 2008,

Welkom bij het nieuwe nummer van de Delphi OplossingsCourant. Vandaag is tevens de dag waarop mijn eerste gratis Delphi / CodeGear RAD Studio seminar in Helmond Brandevoort is gehouden in samenwerking met Barnsten (de vertegenwoordigers van CodeGear in de BeNeLux). Ik schrijf dit voorwoord nog voordat het seminar zelf van start is gegaan, dus kan alleen nog maar de verwachting uitspreken dat het een succes was (wat betreft inschrijvingen was dat zeker het geval: binnen enkele weken zaten we al "vol" en moesten we een wachtlijst opstellen). De resultanten van vandaag gaan we evalueren, maar de kans is zeer groot dat er later dit jaar nog een keer een dergelijk event zal plaatsvinden – hou m'n website en weblog in de gaten voor meer informatie!

In dit nummer van de Delphi OplossingsCourant enkele artikelen over de onderwerpen die tijdens het seminar aan de orde zijn gekomen, zoals Generics, de migratie van data access via de BDE naar dbExpress (en dan met name DBX4). Tijdens m'n seminar heb ik ook een introductie in VCL for the Web en ASP.NET 2.0 gegeven, allebei met verschillende manieren om met AJAX te werken.

Tot slot nog een nieuws van het boekenfront: mijn Delphi 2007 for Win32 Development Essentials cursusboek is nu ook in een Lulu editie te koop van Lulu.com (ruim 100 pagina's voor 19,98 Euro), zie http://stores.lulu.com/drbob42 voor details.

Veel leesplezier met dit nummer van de Delphi OplossingsCourant, en tot de volgende keer.

Groetjes,

Bob Swart
Bob Swart Training & Consultancy

Bob Swart Training & Consultancy

Met mijn bedrijf Bob Swart Training & Consultancy hou ik me met name bezig met de ondersteuning van Delphi ontwikkelaars door middel van consultancy, trainingen en workshops, coaching en support (ook per e-mail), en het leveren van de benodigde licenties, subscriptions en cursusboeken voor zelfstudie (al dan niet onder begeleiding).

Trainingen en Workshops

Mijn Delphi trainingen zijn inclusief dik cursusboek en e-mail ondersteuning (ook achteraf als de training is afgelopen) plus toegang tot het members-only deel van m'n website. Vanaf 2007 worden de trainingen al gegeven in mijn nieuwe cursusruimte in Helmond Brandevoort (ideaal met de trein bereikbaar), zie ook www.eBob42.org.

Alle Win32 en .NET onderwerpen zijn ook als maatwerk training mogelijk in de vorm van een Delphi Workshop in Helmond Brandevoort, of Custom Clinic (op locatie).
Zie http://www.eBob42.com/workshop voor details.

Reseller

Sinds 2007 ben ik ook reseller van CodeGear in de BeNeLux, waardoor ik o.a. Delphi en CodeGear RAD Studio kan leveren, inclusief subscriptions en persoonlijke ondersteuning.
Zie http://www.eBob42.com/CodeGear voor details.

Generic Class Methods

Generics zijn geparametriseerde types, en een onderdeel van het .NET Framework, waardoor ze ook door Delphi for .NET in CodeGear RAD Studio ondersteund worden. Niet door de Win32 versie van Delphi 2007, maar er is inmiddels beloofd dat we in Delphi 2008 for Win32 wel de beschikking zullen hebben over generics.
Omdat niet iedereen het nut of gemak van generics zal zien, volgt hier wederom een artikel over de bruikbaarheid van deze feature. In het vorige nummer van de Delphi OplossingsCourant liet ik zien hoe Generics werken voor classes, en implementeerde ik een TQueue<AnyType>. Sinds die tijd ben ik zelf wat meer gebruik gaan maken van Generics in de vorm van algemeen herbruikbare procedures en functies. Dat kan op verschillende manieren. Maar in ieder geval niet door "losse" procedures of functies te gebruiken. De volgende code levert een compiler fout ("Type parameters not allowed on global procedure or function") op:

```
procedure Swap<AnyType>(var X,Y: AnyType);
```

We kunnen we wel een class omheen schrijven, maar dat werkt niet zo lekker als je dan steeds een instantie nodig hebt om de methode aan te roepen. Gelukkig is het makkelijker als we er een class method van maken. Dan moeten we wel het type meegeven bij de aanroep, maar hebben we geen daadwerkelijke instantie nodig. Dat levert de volgende definitie van de class TGeneric op:

```
type
  TGeneric<AnyType> = class
    procedure Swap(var X,Y: AnyType);
  end;
```

Met een implementatie van de Swap die er als volgt uitziet:

```
  procedure TGeneric<AnyType>.Swap(var X,Y: AnyType);
  var
    Z: AnyType;
  begin
    Z := X;
    X := Y;
    Y := Z
  end;
```

Het gebruik bestaat dan uit het declareren van een TGeneric variabele, waarbij we bij de declaratie van de variabele al moeten aangeven met welk type we willen werken:

```
var
  X: TGeneric<integer>;
  a,b: Integer;
begin
  a := 42;
  b := 7;
  X.Swap(a,b);
end.
```

Op zich werkt dat prima, maar het heeft als nadeel dat de class TGeneric al bij voorbaat heeft vastgelegd dat het met één specifiek type werkt. Dus alle class methods die ik aan TGeneric toevoeg hebben hooguit de keuze uit dat type. Ik kan wel meerdere types toevoegen, maar daarmee wordt de class er niet overzichtelijker op (laat staan de aanroep van de generic class methodes).

Een alternatieve benadering zet het type niet bij de class, maar juist bij de generic class method zelf, als volgt:

```
type
  TGeneric = class
    procedure Swap<AnyType>(var X,Y: AnyType);
  end;
```

De bijbehorende implementatie is dan als volgt (met het type als specifier van de generic class method en niet langer van de class zelf):

```
procedure TGeneric.Swap<AnyType>(var X,Y: AnyType);
var
  Z: AnyType;
begin
  Z := X;
  X := Y;
  Y := Z
end;
```

En het gebruik is dan iets flexibeler: we hoeven alleen maar een variabele van het type TGeneric te declareren, en hoeven pas bij de aanroep van de Swap functie aan te geven met wat voor types we willen werken. Dat is niet alleen makkelijker, het is ook nog eens duidelijker omdat je op het moment van aanroepen expliciet aangeeft van welk type je de generic class method wilt gebruiken:

```
var
  Y: TGeneric;
  a,b: Integer;
begin
  a := 42;
  b := 7;
  Y.Swap<integer>(a,b);
end.
```

Het enige nadeel is nog dat de compiler met een warning komt dat Y niet geïnitialiseerd is. Dat is niet nodig, maar wel een warning. Om daar vanaf te komen, zou je Y van te voren nog even kunnen creëren, al heeft het weinig toegevoegde waarde.

Voor één enkele class methode was het voordeel nog niet zo zichtbaar, maar dat verandert als we meerdere class methodes toevoegen, die ieder met hun eigen specifieke type kunnen werken:

```
type
  TGeneric = class
  public
    class procedure Swap<AnyType>(var X, Y: AnyType);
    class function IIF<AnyType>(const Expression: Boolean;
      TrueValue: AnyType; FalseValue: AnyType): AnyType;
    class function ChooseDef<AnyType>(index: Integer;
      const values: Array of AnyType): AnyType;
  end;
```

De implementatie van de IIF en ChooseDef is hieronder te zien:

```
class function TGeneric.IIF<AnyType>(const Expression: Boolean;
  TrueValue, FalseValue: AnyType): AnyType;
begin
  if Expression then Result := TrueValue
  else
    Result := FalseValue
end;

class function TGeneric.ChooseDef<AnyType>(index: Integer;
  const values: Array of AnyType): AnyType;
begin
  if (index >= Low(values)) and (index <= High(values)) then
    Result := values[index]
  else Result := default(AnyType);
end;
```

Let vooral op de ChooseDef class method, die een default (lege) waarde teruggeeft als de gevraagde index buiten het bereik van het array valt. Hiertoe is er een speciale functie “default” die een lege waarde zoals 0 of nil of een lege string teruggeeft, afhankelijk van het generic type dat gebruikt wordt.

De aanroep van deze twee methodes kan als volgt gedaan worden, waarbij we als resultaat het antwoord 42 te zien krijgen (het maximum van het integer array 1,2,42).

```
var
  Y: TGeneric;
  a,b: Integer;
  c: Array[11..13] of Integer = (1,2,42);

begin
  a := Y.ChooseDef<integer>(12,c); // 0, geen 2
  b := Y.ChooseDef<integer>(2,c); // 42, geen 0
  Y.Swap<integer>(a,b);
  writeln(Y.IIF<integer>(a > b, a, b)); // 42
end.
```

Let op dat de index in het array c van 11 t/m 13 lopen, maar zodra we het array meegeven aan een generic method, dan loopt daarbinnen de index weer van 0 t/m 2 (da's efficiënter), dus de aanroep Y.ChooseDef<integer>(12,c) levert 0 op, en geen 2. Ik weet niet of dat een bug of feature is, maar het is wel onverwacht (alhoewel je beter gewoon al je arrays bij 0 kunt laten beginnen).

Tot nu toe waren de voorbeelden eenvoudig, met het verwisselen van twee waarden of het kiezen uit een lijstje met waarden. Maar het zou handig zijn als we ook iets met de waarde zelf kunnen doen. Dat kan, maar dan moeten we wel expliciet aangeven dat de waarden bijvoorbeeld onderling te vergelijken zijn (door het IComparable interface te implementeren).
Op die manier kunnen we een tweetal generic class methods Min en Max toevoegen, die het minimum of maximum van een array van AnyType terug kunnen geven, mits dit AnyType het IComparable interface implementeert:

```
type
  TGeneric = class
  public
    ...
    class function Min<AnyType: IComparable>(const values: Array of AnyType): AnyType;
    class function Max<AnyType: IComparable>(const values: Array of AnyType): AnyType;
  end;
```

Merk op dat het nu een groot voordeel is dat we de AnyType als specifier van de generic class methods opgeven, en niet als specifier van de TGeneric class, anders zouden we de Swap class method bijvoorbeeld niet meer op gewone records kunnen uitvoeren (als die het IComparable interface niet implementeren). De implementatie van de Min en Max generic class methods is als volgt (het resultaat van CompareTo is positief als het argument kleiner is dan de instantie zelf).

```
class function TGeneric.Max<AnyType>(const values: array of AnyType): AnyType;
var
  item: AnyType;
begin
  if Length(values) >= 1 then
  begin
    Result := values[Low(values)];
    for item in values do
      if item.CompareTo(Result) > 0 then Result := item // item bigger
  end
end;

class function TGeneric.Min<AnyType>(const values: array of AnyType): AnyType;
var
  item: AnyType;
begin
  if Length(values) >= 1 then
  begin
    Result := values[Low(values)];
    for item in values do
      if item.CompareTo(Result) < 0 then Result := item // item smaller
  end
end;
```

Omdat het IComparable interface standaard al in ingebouwde types zoals string, integer en floating points zit, kunnen we nu het maximum van een array of integer als volgt opzoeken:

```
var
  Y: TGeneric;
  a,b: Integer;
  c: Array[11..13] of Integer = (1,2,42);

begin
  a := Y.Max<integer>(c);
  b := 7;
  Y.Swap<integer>(a,b);
  writeln(b);
end.
```

Spannender is het om te kijken of we ook zelf een type kunnen maken waarbij we het ICompatable interface zelf implementeren. Bijvoorbeeld het TSKU record dat zowel de SKU code als de prijs van een CodeGear product bevat.
Om het TSKU record te kunnen gebruiken in de generic Min en Max class methods, moeten we ook het IComparable interface implementeren:

```
  TSKU = record(IComparable)
  public
    SKU: String;
    Price: Double;
    function CompareTo(obj: TObject): Integer;
  end;

function TSKU.CompareTo(obj: TObject): Integer;
begin
  if TSKU(obj).Price > Price then Result := -1 // this instance < object
  else
    if TSKU(obj).Price < Price then Result := 1 // this instance > object
    else
      Result := 0 // equal
end;
```

Een array van TSKU records kunnen we nu als volgt declareren en gebruiken:

```
var
  Reseller: Array[0..14] of TSKU =
    ((SKU: 'Delphi 2007 for Win32 R2 Pro New User'; Price: 774),
     (SKU: 'Delphi 2007 for Win32 R2 Pro Upgrade'; Price: 324),
     (SKU: 'Delphi 2007 for Win32 R2 Pro Subscription'; Price: 235),
     (SKU: 'Delphi 2007 for Win32 R2 Ent New User'; Price: 1724),
     (SKU: 'Delphi 2007 for Win32 R2 Ent Upgrade'; Price: 1124),
     (SKU: 'Delphi 2007 for Win32 R2 Ent Subscription'; Price: 525),
     (SKU: 'CodeGear RAD Studio 2007 Pro New User'; Price: 934),
     (SKU: 'CodeGear RAD Studio 2007 Pro Upgrade'; Price: 414),
     (SKU: 'CodeGear RAD Studio 2007 Pro Subscription'; Price: 315),
     (SKU: 'CodeGear RAD Studio 2007 Ent New User'; Price: 1974),
     (SKU: 'CodeGear RAD Studio 2007 Ent Upgrade'; Price: 1334),
     (SKU: 'CodeGear RAD Studio 2007 Ent Subscription'; Price: 650),
     (SKU: 'CodeGear RAD Studio 2007 Arch New User'; Price: 2614),
     (SKU: 'CodeGear RAD Studio 2007 Arch Upgrade'; Price: 1974),
     (SKU: 'CodeGear RAD Studio 2007 Arch Subscription'; Price: 865));

  SKU: TSKU;

begin
  SKU := TGeneric.Max<TSKU>(Reseller);
  writeln(SKU.SKU);
end.
```

Overigens worden zoals gezegd Generics momenteel alleen ondersteund in Delphi for .NET 2.0, en niet in de Win32 versie van Delphi. Volgens de Delphi Roadmap (zie http://dn.codegear.com/article/36620 komt ondersteuning voor Generics in Delphi for Win32 in de volgende versie van Delphi, met de codenaam Delphi Tiburón (Delphi en VCL development met Unicode en Generics).

Van BDE naar SQL DBMS met Delphi 2007 en DBX4

In de vorige nummers van de Delphi OplossingsCourant ging ik al in op DBX4, de nieuwste versie van de dbExpress data access laag in Delphi 2007 en CodeGear RAD Studio 2007. Deze keer ga ik verder, en laat ik zien hoe bestaande data uit een BDE tabel te migreren is naar een tabel in een echte SQL DBMS zoals SQL Server, InterBase of Blackfish SQL, de nieuwste managed database van CodeGear zelf.
Als binnenkomer volgt hier nog even een overzicht van de deployment mogelijkheden van Blackfish SQL, die niet altijd even makkelijk te vinden zijn.

Blackfish SQL

Blackfish SQL is een managed SQL-compliant embedded database die naast onze Delphi toepassingen kan worden gedeployed. Er zijn drie Blackfish SQL for Windows connection drivers: als eerste DBXClient – een Win32 DBX4 driver om via Delphi en C++Builder naar een remote Blackfish SQL database te praten, als tweede een Local ADO.NET 2.0 Provider (voor een local Blackfish SQL database – de driver en de Blackfish database kernel draaien hierbij in-process), en als derde een Remote ADO.NET 2.0 Provider, waarbij je een .NET remote connection naar een Blackfish SQL database kan maken. Je kan bijvoorbeeld Blackfish SQL als .NET assembly op een web server zetten en er dan ASP.NET toepassingen tegen aan laten werken.
Omdat Blackfish SQL een managed database is, geschreven in C# (als directe vertaling van de Java implementatie van JDataStore), bestaat de deployment uit een .NET assembly die te vinden is in de C:\Program Files\Common Files\CodeGear Shared\RAD Studio\Shared Assemblies\5.0 directory. Het betreft de volgende drie assemblies:

- Borland.Data.BlackfishSQL.Design.dll
- Borland.Data.BlackfishSQL.LocalClient.dll
- Borland.Data.BlackfishSQL.RemoteClient.dll

Let er op dat de laatste versie (na de December 2007 Update) als datum 11 december 2007 heeft, en niet nog 26 augustus 2007. Indien u de December Update heeft gedraaid, maar nog de Blackfish SQL assemblies van augustus 2007 heeft, dan is er iets mis (en een oplossing voor dit probleem wordt beschreven in mijn weblog te http://www.bobswart.nl/Weblog/Blog.aspx?RootId=5:1951 alsmede op Quality Central te http://qc.borland.com/wc/qcmain.aspx?d=57941

CodeGear RAD Studio 2007 Professional bevat een unlimited Blackfish SQL desktop database deployment license voor systemen met 1 CPU, 1 gebruiker of 4 connecties, met databases die 512 MB kunnen worden.

CodeGear RAD Studio 2007 Enterprise en Architect geven een ongelimiteerde Blackfish SQL database deployment license voor systemen met 1 CPU, maximaal 5 gebruikers of 20 connecties en databases die 2 GB groot kunnen worden.
In alle gevallen kunnen extra deployment licenties gekocht worden (ook voor meer gebruikers, connecties, CPUs of grotere databases)– neem contact met me op voor de details als je interesse hebt.

Blackfish SQL gebruik als databases het formaat .jds – hetzelfde als JDataStore, en Blackfish SQL is dan ook de uiteindelijke versie wat destijds als NDataStore begonnen is. Blackfish SQL draait dan ook als managed database onder zowel Java als .NET.
Zie http://dn.codegear.com/blackfish voor meer informatie over Blackfish SQL.

Data Migratie

Zoals (hopelijk) bekend, is de Borland Database Engine (BDE) een bevroren data access technologie die vanaf Delphi 1 beschikbaar was, maar vanaf Delphi 7 het predicaat "frozen" heeft meegekregen, en waar geen onderhoud meer op gepleegd wordt. Voor de grote broer SQL Links (de BDE uitbreiding om met SQL Server databases te werken) geldt zelfs de toevoeging "deprecated", en SQL Links wordt ook niet meer geleverd bij versies van Delphi groter dan 7 (daarbij moet opgemerkt worden dat SQL Links nog wel gebruikt kan worden door een Delphi met een versie groter dan 7 indien de BDE / SQL Links op de ontwikkelmachine aanwezig is – maar het gebruik is daarbij volledig "op eigen risico").
Al vanaf Delphi 6 (en Kylix) heeft Borland dbExpress aangemerkt als de data access architectuur die de voorkeur heeft boven de BDE en SQL Links. Tot nu toe was er nog niet echt haast geboden om van de BDE over stappen (het zit er immers nog altijd in), maar sinds vorig jaar is de laatste BDE engineer van Borland niet meer werkzaam bij CodeGear. En als straks voor Delphi 2008 de stap naar volledige UNICODE wordt gezet, is het nog maar de vraag of de BDE dit zal meemaken. Om niet voor verrassingen komen te staan, zal ik in dit artikel aangeven hoe we BDE aliases kunnen analyseren, hoe we SQL kunnen genereren om vergelijkbare tabellen in SQL DBMSs te creëren en hoe we data uit BDE tabellen kunnen overpompen naar deze SQL DBMSs.

BDE Analyse

Laten we beginnen met een kleine GUI om alle mogelijke BDE aliases strings op te halen die op een machine aanwezig zijn, en voor een gekozen alias alle tabellen op te sommen die daarbinnen te vinden zijn.
Hier hebben we twee TComboBox componenten voor nodig, genaamd cbDatabases en cbTables. Het vullen van de cbDatabases met de aanwezige alias strings gaat met behulp van een TSession component als volgt, bijvoorbeeld in de FormCreate:

```
procedure TFormSDE.FormCreate(Sender: TObject);
begin
  Session1.GetDatabaseNames(cbDatabases.Items);
end;
```

Hierdoor zal de dbDatabases TComboBox gevuld worden met de aanwezige BDE alias strings. Als we eenmaal een alias gekozen hebben, kunnen we aan een TDatabase component vragen welke tables er allemaal bekend zijn voor deze alias. Dat gaat met de GetTableNames functie, en doe ik bij voorkeur in de OnChange event van de cbDatabases combobox, zodat je daarmee meteen de cbTables combobox kunt vullen:

```
procedure TFormSDE.cbDatabasesChange(Sender: TObject);
begin
  Database1.Close;
  Database1.DatabaseName := cbDatabases.Items[cbDatabases.ItemIndex];
  Database1.GetTableNames(cbTables.Items, cbSystemTables.Checked);
end;
```

Eenzelfde truc kunnen we uithalen in de OnChange event van de cbTables combobox, waarbij we de dan gekozen tablename kunnen gebruiken om deze tabel te analyseren.

```
procedure TFormSDE.cbTablesChange(Sender: TObject);
begin
  Table1.Close;
  Table1.DatabaseName := Database1.DatabaseName;
  Table1.TableName := cbTables.Items[cbTables.ItemIndex];
end;
```

Uitgaan de van een BDE TTable waarvan de DatabaseName en TableName zijn opgegeven kunnen we alle benodigde informatie ophalen uit de FieldDefs en IndexDefs collecties. Deze informatie wil ik gaan gebruiken om een SQL CREATE TABLE command te produceren dat de structuur van de table kan reproduceren in een SQL DBMS. Het voordeel is dat we de inhoud van dit SQL CREATE TABLE command kunnen opslaan in een script en (later) op een andere machine kunnen draaien.

Een probleem van deze benadering is dat de SQL CREATE TABLE voor ieder DBMS type een iets andere inhoud zal hebben, omdat de veld types niet altijd hetzelfde zullen zijn. Zo kent SQL Server bijvoorbeeld de veld types IMAGE en TEXT terwijl Blackfish SQL die niet kent (maar daar moeten we VARBINARY voor de IMAGE en VARCHAR voor de TEXT gebruiken). En ook een floating point getal wordt door het ene DBMS als DOUBLE aangegeven terwijl de andere een FLOAT wil zien.
In dit artikel zal ik voor SQL Server en Blackfish SQL een aantal field mappings aangeven, maar laat het aan de lezer over om voor zijn specifeke DBMS de mapping compleet te maken (zie dan de documentatie van de DBMS voor de benodigde details).

SQL CREATE TABLE

De basis syntax van het SQL CREATE TABLE command is kort gezegd als volgt:

```
CREATE TABLE <tablename>
  ( <fieldname> <fieldtype> [size]  [[NOT] NULL] )
```

Waarbij je een of meerdere fields kunt aangeven. De size is optioneel (en wordt voornamelijk bij CHAR of VARCHAR velden gebruikt), net als de NULL of NOT NULL (default is NULL). Zoals eerder aangegeven zullen we bij het fieldtype voor een DBMS specifieke waarde moeten kiezen, waar de rest DBMS onafhankelijk is.

Als we het fieldtype nog even buiten beschouwing laten, dan kan het DBMS onafhankelijke deel van de SQL CREATE TABLE command met een functie Table2SQL als volgt geproduceerd worden:

```
function Table2SQL(const Table: TTable): String;
var
  i: Integer;
  Str: String;

begin
  Table.Open;
  Table.FieldDefs.Update;
  Table.IndexDefs.Update;

  try
    Str := 'CREATE TABLE ' +
      ChangeFileExt(Table.TableName, '') + '(';

    for i:=0 to Table.Fields.Count-1 do
    begin
      Str := Str + '"' + Table.Fields[i].DisplayName + '" ';

      // fieldtype

      if Table.FieldDefs[i].Required then Str := Str + ' NOT';
      Str := Str + ' NULL';
      if (i < Table.Fields.Count-1) then Str := Str + ', ';
    end;
    Str := Str + ')';

    Result := Str;
  finally
    Table.Close;
  end
end;
```

Merk op dat de FieldDefs.Update en IndexDefs.Update meestal niet meer nodig zijn als de Table reeds geopend is (en zonder dat de Table geopend is levert FieldDefs.Update toch niet de gewenste informatie op).

Voor iedere DBMS die we willen ondersteunen bij de migratie, moeten we nu een eigen field mapping schrijven van het BDE / SQL Links type naar het specifieke DBMS type.

Field Mapping

Voor SQL Server en Blackfish SQL heb ik de volgende field mappings gevonden, waarbij ik aan de BDE / SQL Links kant uitga van de waarde van de Table.FieldDefs[i].FieldClass en aan de DBMS kant het corresponderende type heb gekozen:

BDE / SQL Links	SQL Server	Blackfish SQL
TStringField	VARCHAR(size)	VARCHAR(size)
TIntegerField	INTEGER	INTEGER
TSmallIntField	SMALLINT	SMALLINT
TFloatField	FLOAT	DOUBLE
TDateTimeField	DATETIME	TIMESTAMP
TDateField	DATE	DATE
TTimeField	TIME	TIME
TCurrencyField	DECIMAL	DECIMAL
TBooleanField	BIT	BOOLEAN
TMemoField	TEXT	VARCHAR
TGraphicField	IMAGE	VARBINARY
TBlobField	IMAGE	VARCHAR ?

Deze lijst is niet compleet, maar is voldoende om van de DBDEMOS BDE alias alle tabellen om te zetten naar SQL Server of Blackfish SQL, en daar ging het me in eerste instantie om.

Deze tabel heb ik vervolgens vertaald naar een TFieldMapping record structuur die ik meteen heb ingevuld voor de bovenstaande meest voorkomende veld types:

```
type
  TDatabase = (dbSQLServer, dbBlackfishSQL, ...);

  TFieldMapping = record
    FieldClass: TFieldClass;
    SQL: Array[TDatabase] of String;
  end;

const
  FieldMappings: Array[0..11] of TFieldMapping =
    ((FieldClass: TStringField;   SQL: (' VARCHAR(', ' VARCHAR(')), // + size
     (FieldClass: TIntegerField;  SQL: (' INTEGER ', ' INTEGER ')),
     (FieldClass: TSmallIntField; SQL: (' SMALLINT ', ' SMALLINT ')),
     (FieldClass: TFloatField;    SQL: (' FLOAT ', ' DOUBLE ')),
     (FieldClass: TDateTimeField; SQL: (' DATETIME ', ' TIMESTAMP ')),
     (FieldClass: TDateField;     SQL: (' DATE ', ' DATE ')),
     (FieldClass: TTimeField;     SQL: (' TIME ', ' TIME ')),
     (FieldClass: TCurrencyField; SQL: (' DECIMAL ', ' DECIMAL ')),
     (FieldClass: TBooleanField;  SQL: (' BIT ', ' BOOLEAN ')),
     (FieldClass: TMemoField;     SQL: (' TEXT ', ' VARCHAR ')),
     (FieldClass: TGraphicField;  SQL: (' IMAGE ', ' VARBINARY ')),
     (FieldClass: TBlobField;     SQL: (' IMAGE ', ' VARBINARY '))
    );
```

Het stukje ontbrekende code in de voorgaande listing kan dan ook als volgt worden aangevuld voor SQL Server (met de strings voor Blackfish SQL in commentaar daarnaast aangegeven):

```
      if Table.FieldDefs[i].FieldClass = TStringField then
        Str := Str + ' VARCHAR(' + IntToStr(Table.FieldDefs[i].Size) + ')'
      else
      begin
        found := False;
        for FieldMapping in FieldMappings do if not found then
        begin
          if Table.FieldDefs[i].FieldClass = FieldMapping.FieldClass then
          begin
            Str := Str + FieldMapping.SQL[Database];
            found := true
          end
        end;
```

```
      if not found then
        raise Exception.Create('Unsupported field type ' +
          Table.FieldDefs[i].FieldClass.ClassName)
    end;
```

Let op de raise Exception.Create aan het eind, die ons zal vertellen wanneer er een ander field type is gevonden, zodat we dan daarvoor ook op zoek moeten naar een bijbehorende SQL mapping.
Voor de Country tabel uit de DBDEMOS alias, is hiermee de volgende CREATE TABLE geproduceerd:

```
CREATE TABLE country("Name" VARCHAR(24) NULL, "Capital" VARCHAR(24) NULL,
  "Continent" VARCHAR(24) NULL, "Area" FLOAT NULL, "Population" FLOAT NULL)
```

Copy Copy

En dan nu de laatste stap: het overpompen van de data uit de BDE / SQL Links tabel naar de SQL DBMS. Dat kan door gebruik te maken van een aantal DBX4 componenten die de BDE records eerst inlezen in een TClientDataSet, en daarna alles naar het SQL DBMS sturen.
Allereerst is daar de dbExpress SQLConnection component die een verbinding met het SQL DBMS moet maken – hiertoe moet je de juiste connection properties opgeven. Dan is er een SQLDataSet die via zijn SQLConnection property aan de SQLConnection component verbonden is, en een SELECT * FROM uitvoert op de betreffende (nog lege) tabel die we eerder met het CREATE TABLE command hebben aangemaakt. Een TDataSetProvider verbonden met de SQLDataSet en een TClientDataSet verbonden met de DataSetProvider maken het plaatje compleet.
Bij het overkopieren hoeven we nu alleen nog maar door de BDE / SQL Links TTable heen te lopen, en per record een Append te doen in de ClientDataSet (met Append komt ieder volgend record achter het vorige, zodat we dezelfde volgorde aanhouden als in de BDE table). Vervolgens kunnen we van ieder veld de waarde kopiëren door de Fields[i].Value te gebruiken. Wat nog iets beter is (ook voor memo velden) is om Assign te gebruiken om de waarde van de BDE velden toe te kennen aan die van de DBX4 velden. Hierbij ga ik wel uit van het feit dat beide tabellen de velden in exact dezelfde volgorde hebben staan (zodat we niet via FieldByName hoeven te zoeken, maar kunnen volstaan met de mapping van veld i uit de BDE tabel naar veld i van de SQL DBMS tabel).
Omdat de SQLDataSet met een SELECT * FROM werd gestart, zal bij de aanroep van ApplyUpdates van de ClientDataSet automatisch een bijbehorende SQL INSERT worden gegenereerd.

```
procedure TFormSDE.btnCopyClick(Sender: TObject);
var
  i: Integer;
begin
  ClientDataSet1.Close;
  Table1.Open;
  SQlDataSet1.CommandText := 'SELECT * FROM ' +
    ChangeFileEXt(Table1.TableName,'');
  ClientDataSet1.Active := True;
  while not Table1.Eof do
  begin
    ClientDataSet1.Append;
    for i:=0 to Table1.Fields.Count-1 do
      ClientDataSet1.Fields[i].Assign(Table1.Fields[i]);
    ClientDataSet1.Post;
    ClientDataSet1.ApplyUpdates(0);
    Table1.Next;
  end;
  Table1.Close;
end;
```

In de code van de laatste listing wordt deze ApplyUpdates voor ieder record aangeroepen (direct na de Post), maar dit kun je ook na iedere X records doen om het proces een klein beetje te versnellen.
Op het eind van de code laat ik de ClientDataSet op Active staan. Dat doe ik met opzet, zodat een eventueel DBGrid op het conversie form dan meteen de inhoud kan laten zien:

Met de code uit dit artikel kun je zelf de structuur en inhoud van je BDE tabellen migreren naar een SQL DBMS, en daarna met bijvoorbeeld DBX4 componenten en een data module met de data werken.

VCL Data Modules delen met VCL for the Web

VCL for the Web – ook bekend onder de naam IntraWeb – biedt als grootste voordeel het hergebruik van VCL data modules, inclusief niet alleen alle database connecties, tables, queries en stored procedures, maar ook de speciale business rules voor insert, update en deletes bijvoorbeeld (om van de speciale berekeningen voor de calculated fields nog maar te zwijgen). De exacte manier van dit hergebruik is echter niet altijd even duidelijk, dus hieronder beschrijf ik een handige manier om het voordeel maximaal uit te buiten.
Ik ga ervan uit dat er een bestaand VCL Win32 of .NET project is met daarin een VCL data module. Deze data module gaan we hergebruiken in een VCL for the Web toepassing.

Met makkelijkst is altijd om dat vanuit dezelfde projectgroup te doen, dus daar voeg ik meestal mijn nieuwe of bestaande VCL for the Web toepassing aan toe. De VCL for the Web Application Wizard heeft een optie om een User Session te maken, die de basis is voor een data module. Die User Sessie moeten we inderdaad gebruiken, maar niet als onze data module, want deze data module kunnen we natuurlijk niet delen met een "normale" VCL Win32 of .NET toepassing.
Wat we moeten doen is de unit met daarin onze VCL data module toevoegen aan het VCL for the Web project, maar er dan ook voor zorgen dat deze data module niet automatisch gecreëerd wordt. We hebben natuurlijk niets aan één globale data module, maar moeten er een hebben voor iedere gebruiker, oftewel voor iedere User Session. Daartoe moeten we de TIWUserSession uitbreiden met een veld voor onze data module, en deze in de constructor aanmaken, als volgt:

```
constructor TIWUserSession.Create(AOwner: TComponent);
begin
   inherited;
   MyDataModule := DataMod.TDataModule1.Create(Self);
end;
```

Maar nu zijn we er nog niet, want we willen natuurlijk tijdens design-time al de koppelingen maken met de IW data-aware controls, en dat gaat niet door deze uit de UserSession te halen (althans, dat wordt nog niet ondersteund door de designer).
Om dat laatste probleem op te lossen moeten we terug naar de data module unit, en daar de globale instantie variabele aanpassen, om er in het geval van een VCL for the Web toepassing een functie van te maken die de data module uit de UserSession teruggeeft.

```
{$IFDEF IW}
function DataModule1: TDataModule1;
{$ELSE)
var
   DataModule1: TDataModule1;
{$ENDIF}

implementation
{$IFDEF IW}
uses
   ServerController, UserSessionUnit;

   ...

function DataModule1: TDataModule1;
begin
   Result := UserSession.MyDataModule
end;
{ENDIF)
```

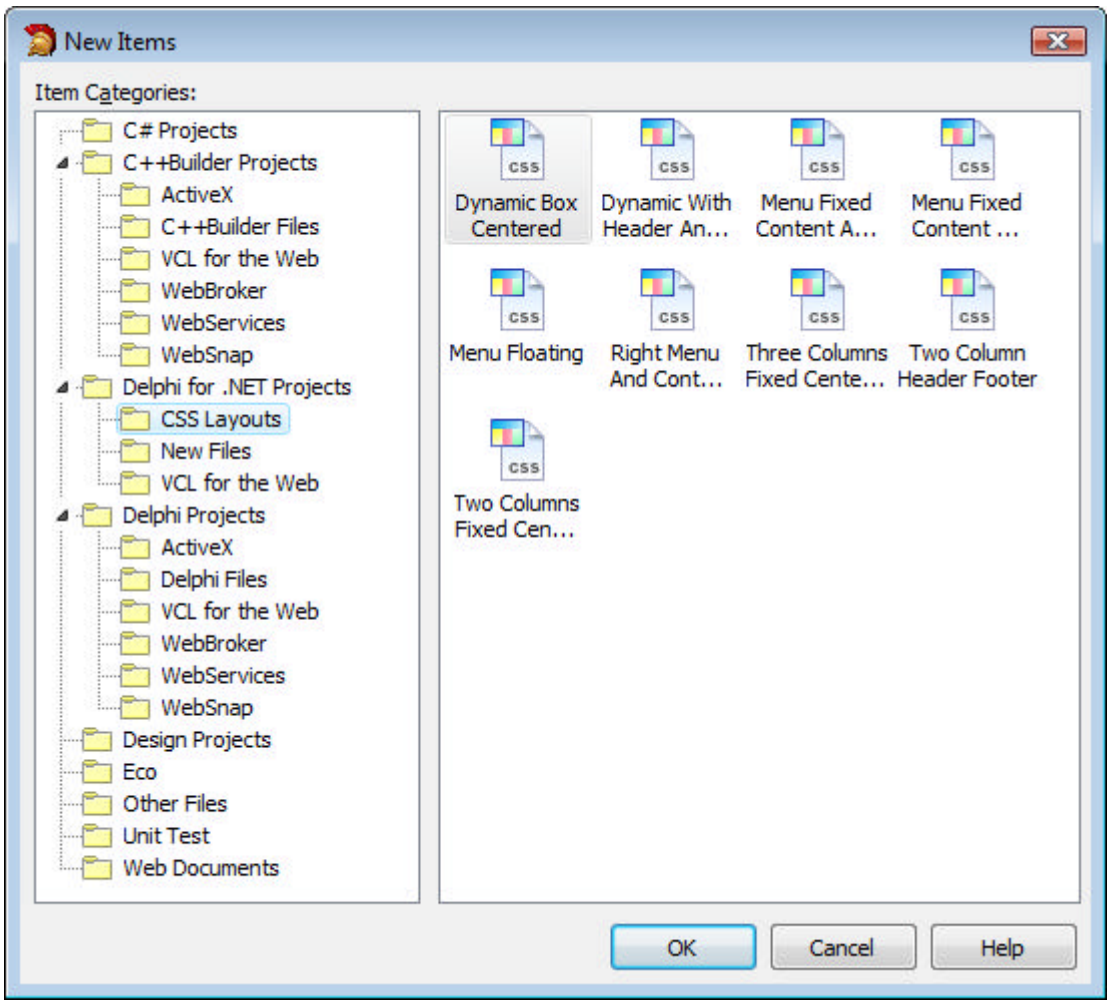
Neem nu IW op in de Conditional Defines van het VCL for the Web project, en builden geeft altijd het goede resultaat (een Build is altijd nodig, want de compiler ziet een conditional define niet als een wijziging in de source code!).

ASP.NET 2.0 CSS Layouts

CodeGear RAD Studio 2007 biedt ondersteuning voor master-content pages. Dat wil zeggen dat je een master page maakt (in een .master file) en een content page die afgeleid is van de master page – een beetje te vergelijken met het concept Form Inheritance van Delphi for Win32, maar dan slechts één niveau diep. Dus met name voor het vaststellen en hergebruik van een design.
Als we een nieuwe master page maken, dan is die leeg en moeten we die zelf vullen met een inhoud. De daadwerkelijke inhoud komt steeds in de contentplaceholder. Daaromheen kunnen we alles neerzetten wat we willen, maar in de contentplaceholder komen de details van de content pagina's:

```
<asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
</asp:contentplaceholder>
```

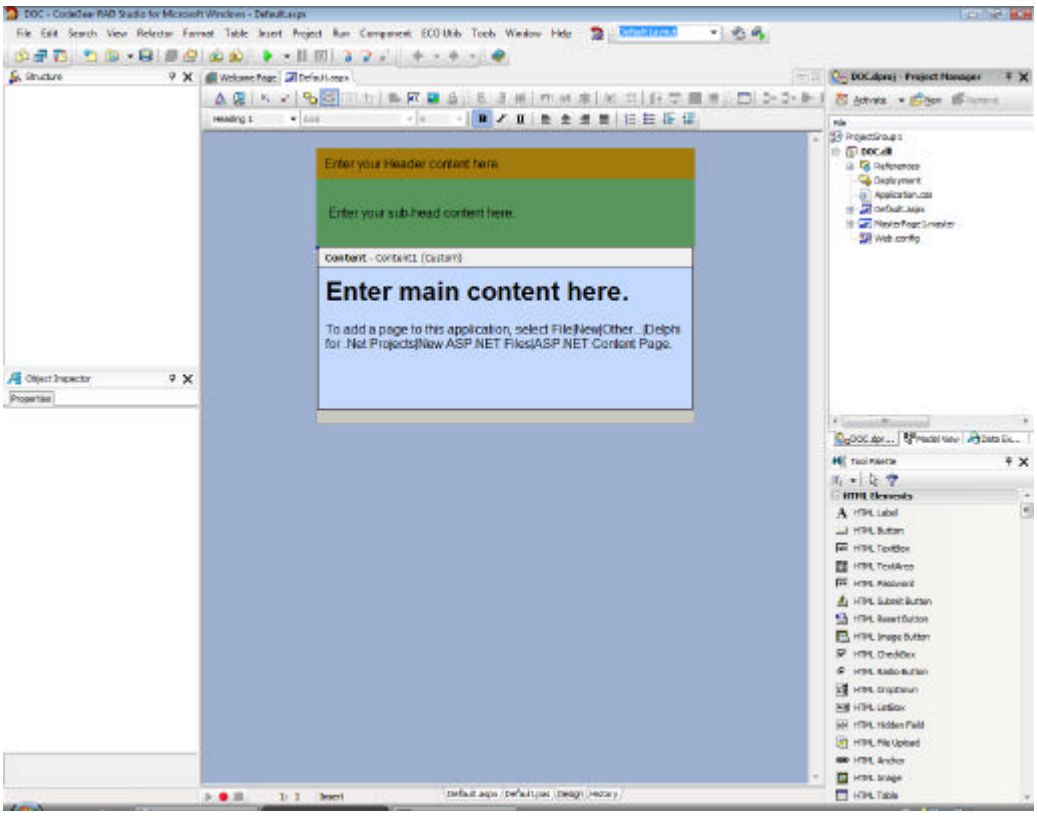
Bij het toevoegen van een nieuwe content pagina kunnen we kiezen uit de lijst met bestaande master pages binnen het project. En zal de inhoud ingevuld worden op de plek van de contentplaceholder, maar de rest staat al vast (en is al bepaalde in de gekozen master page).
Omdat een master page normaal gesproken leeg is, zijn er al een negental speciale CSS Layouts aanwezig in CodeGear RAD Studio die ons vast een vooraf uitgewerkte master page bieden, waar we zelf alleen nog maar content pages voor hoeven te maken. De verschillende CSS Layouts zijn de vinden in de Object Repository:



Helaas zijn er alleen maar negen icoontjes te zien, en is de omschrijving niet altijd duidelijk genoeg om meteen te weten hoe het resultaat eruit zal zien. Daarom volgen hier negen screenshots en een korte beschrijving van de CSS Layouts, zodat het kiezen straks wat makkelijker kan gaan. Overigens zijn acht van de templates oorspronkelijk afkomstig van http://intensivstation.ch/en/templates/ waar ook meer informatie betreffende de stylesheets te vinden is.

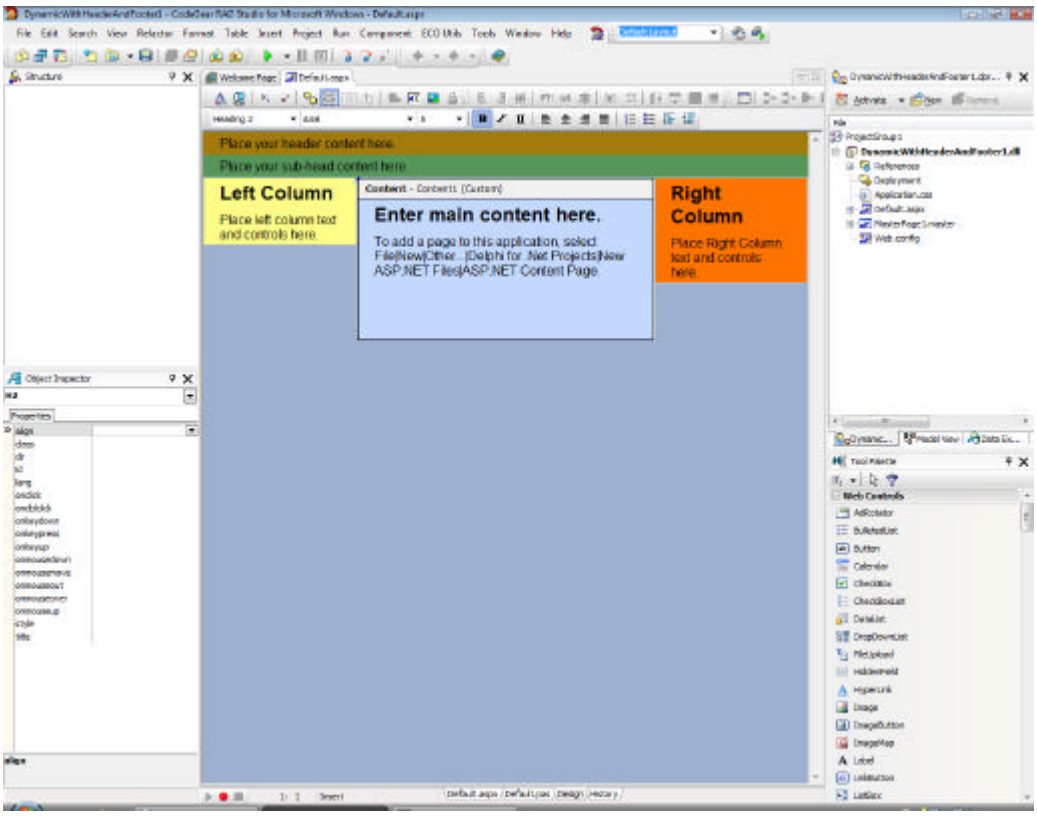
Dynamic Box Centered

Dit bestaat uit een header en subheader gevolgd door de content die we kunnen invullen in onze content pagina's. De linker en rechter ruimte om de content blijft vrij, ook in de browser.



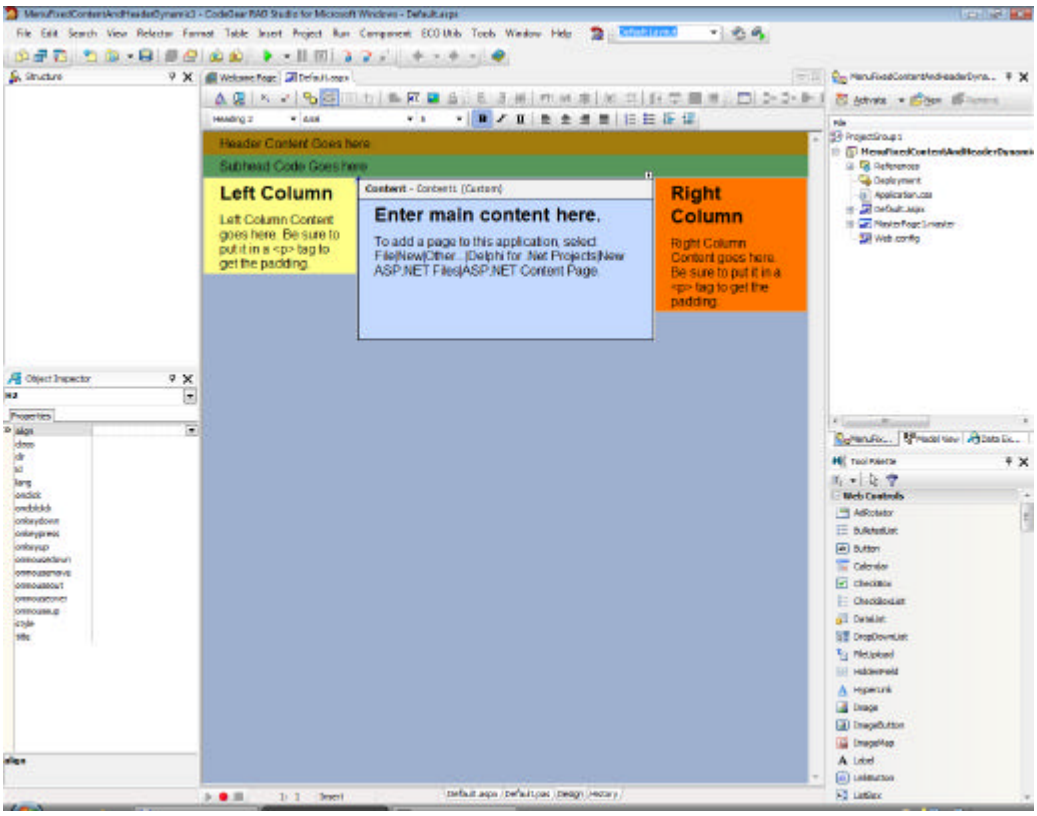
Dynamic with Header and Footer

Dit is een master page met een header en subheader, maar ook een linker en rechter (vaste) kolom die om de content heen staan. Komt over de gehele breedte van de browser.



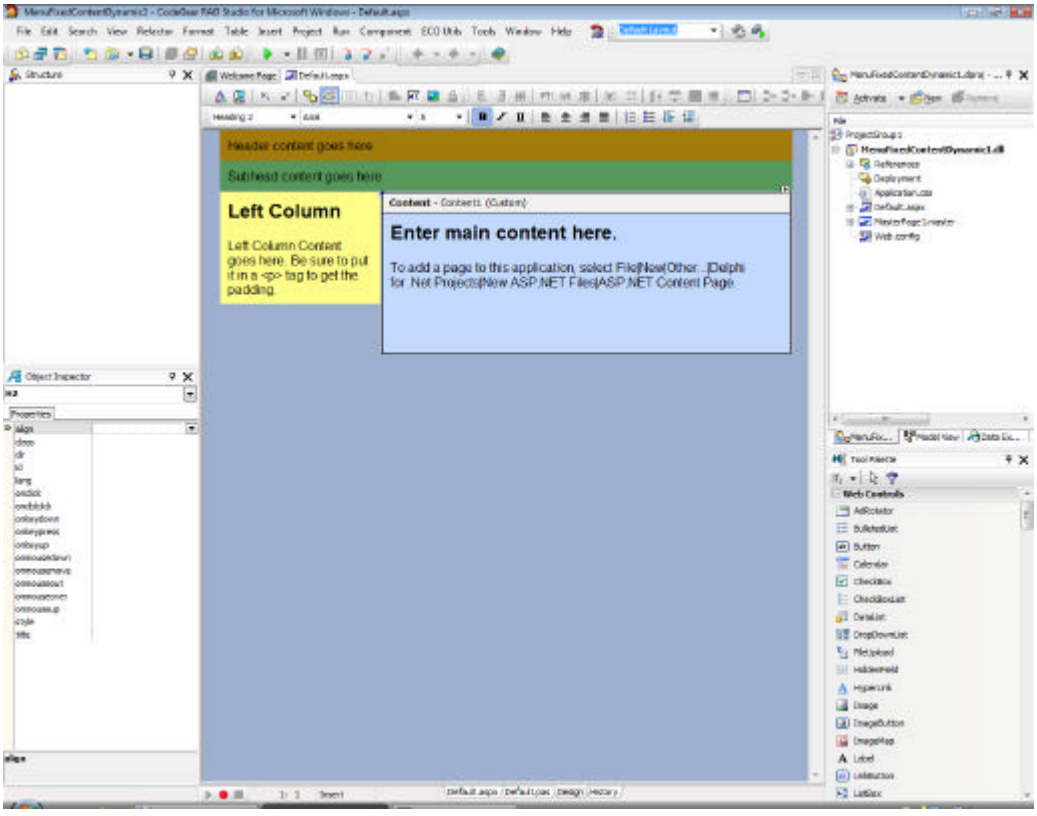
Menu Fixed Content and Header Dynamic

Dit is een master page met een header en subheader, maar ook een linker en rechter (vaste) kolom die om de content heen staan. Eerlijk gezegd hetzelfde als de vorige…



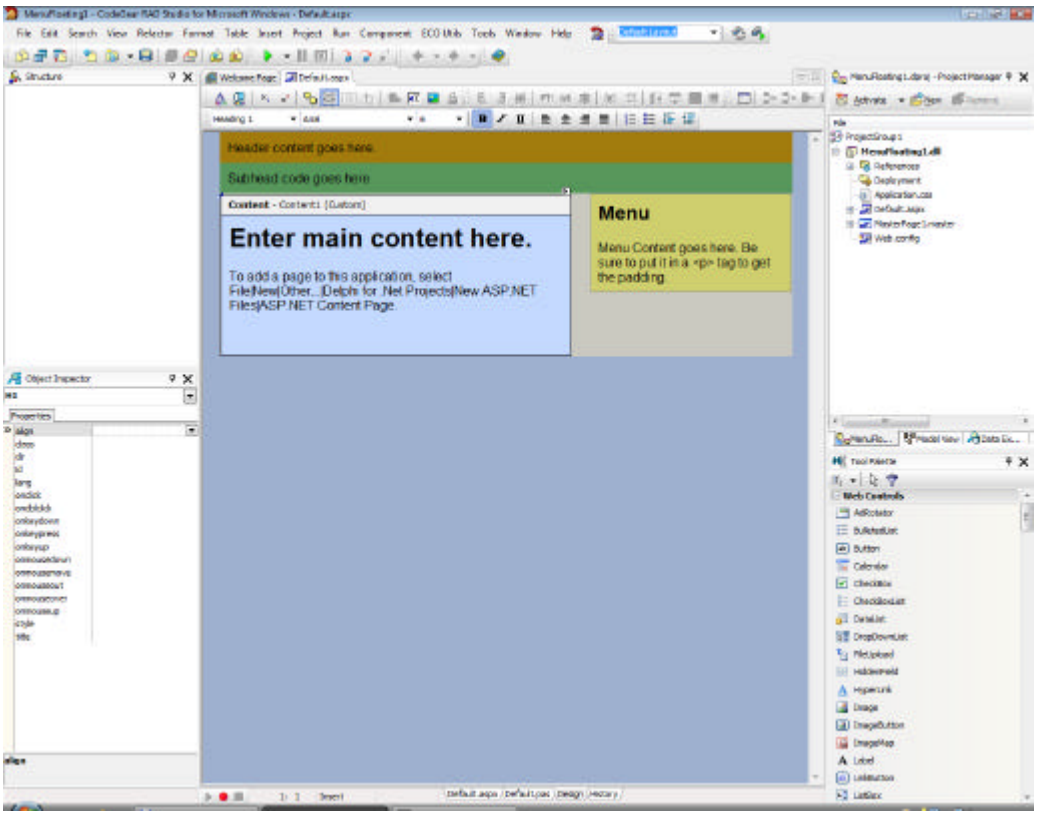
Menu Fixed Content Dynamic

Dit is een master page met een header en subheader, met alleen een linker (vaste) kolom die naast de content staat. Links en rechts een klein stukje leeg.



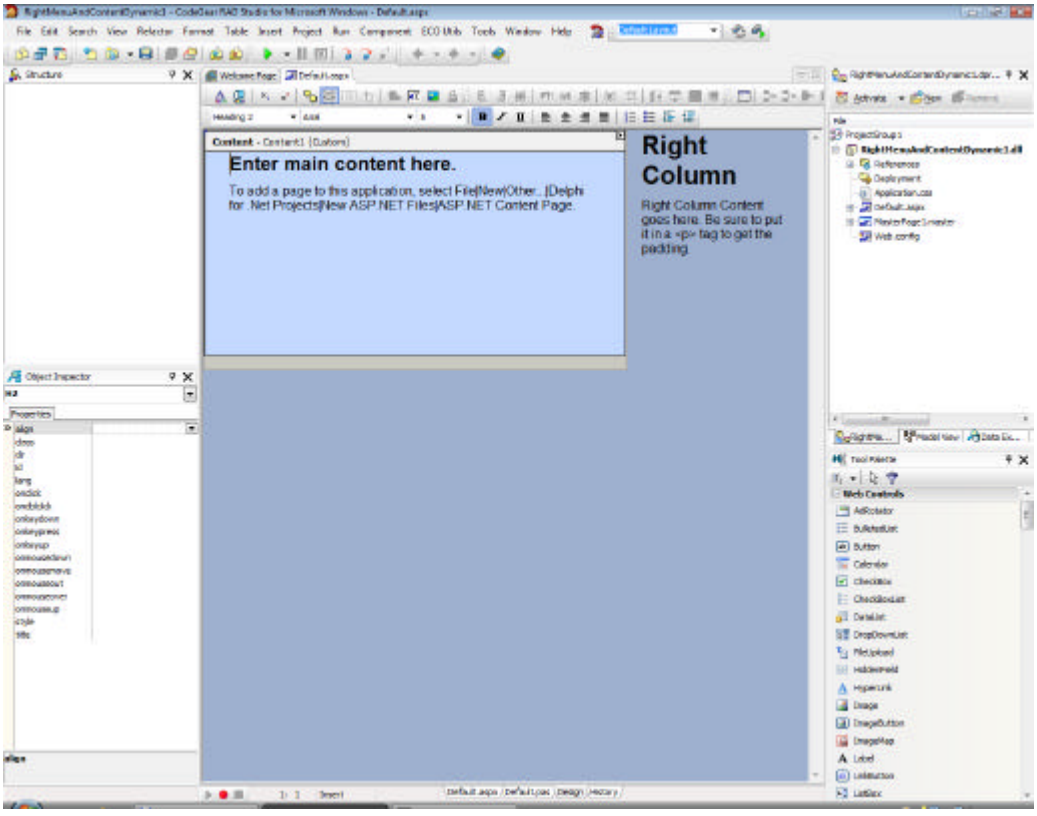
Menu Floating

Dit is een master page met een header en subheader, en een floating menu aan de rechterkant. Links en rechts een klein stukje leeg.



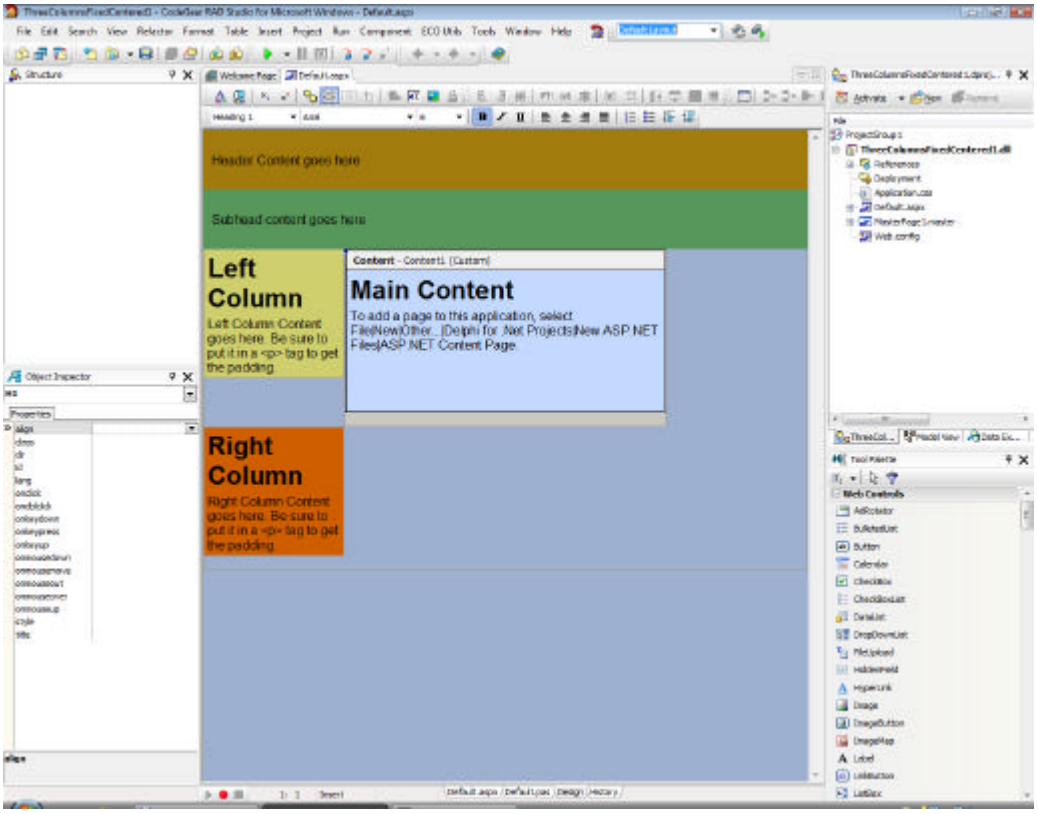
Right Menu and Content Dynamic

Ook hier hebben we aan de rechterkant een menu, maar nu zonder een header of subheader erbij. Vind ik persoonlijk wat minder handig, want ik wil altijd wel een header eigenlijk.

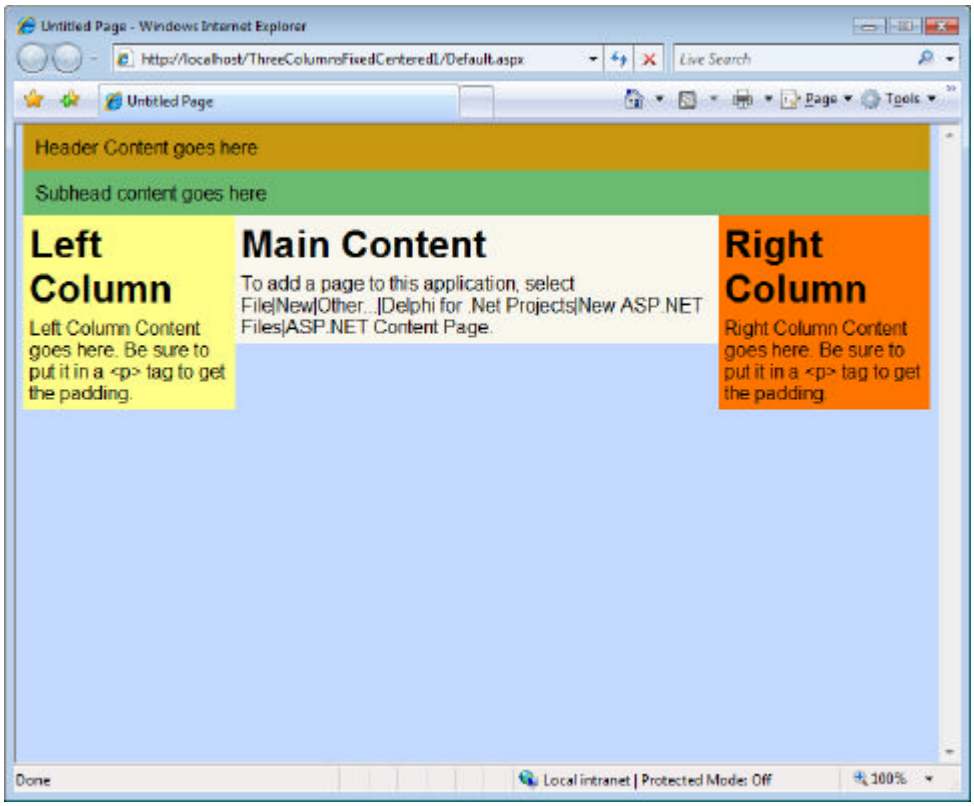


Three Columns Fixed Centered

Dit template lijkt weer op de andere met drie kolommen waarbij de inhoud in het midden onder de header en subheader zit. Tijdens design-time lijkt de rechterkolom vaak aan de linkerkant te komen (omdat de breedte fixed is).

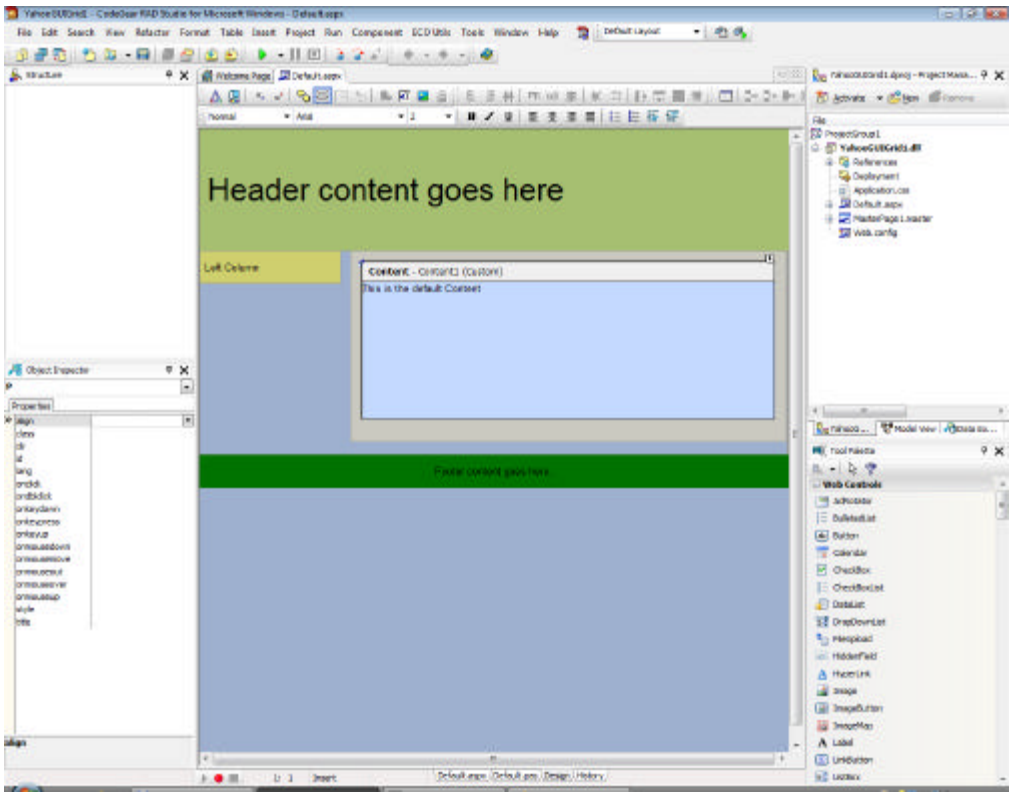


Maar tijdens run-time ziet het er gewoon goed uit:



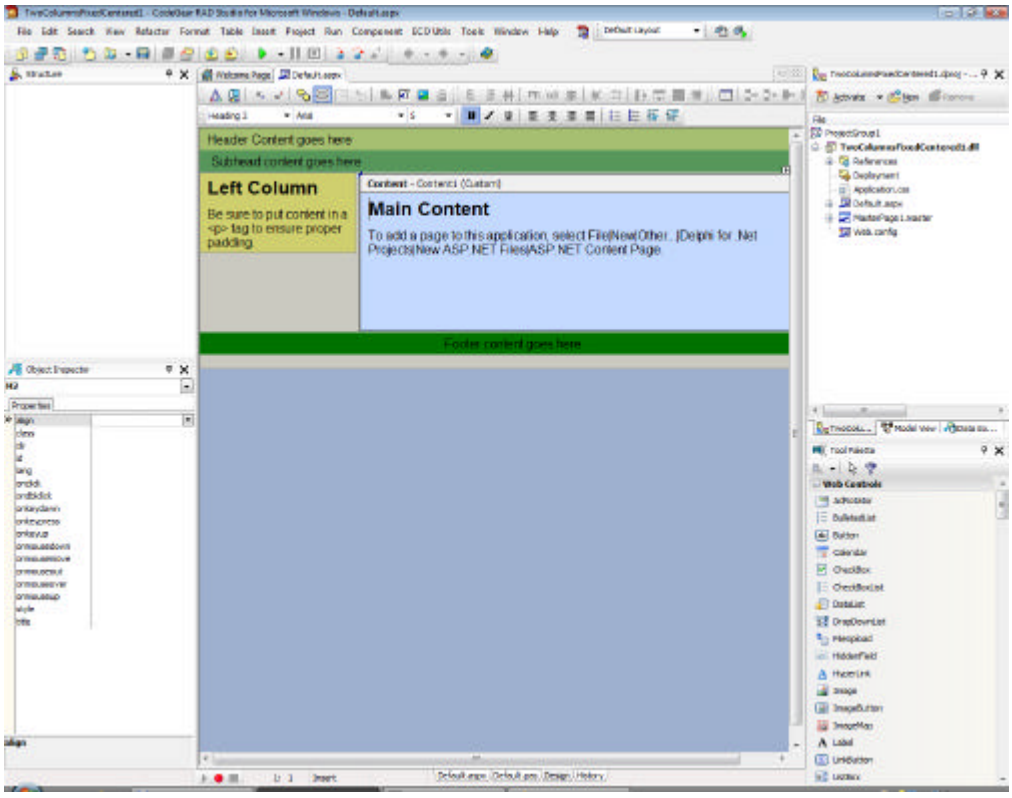
Two Column Header Footer

Bij dit template hebben we een header, linker kolom, een stukje leeg tussen de linker kolom en de content, de main content en een footer. Dit is de enige van de negen die niet afkomstig is van de templates van http://intensivstation.ch/en/templates/



Two Columns Fixed Centered

Dit template is bijna hetzelfde als de vorige, alleen is er deze keer ook een subheader en zijn de twee kolommen weer vast aan elkaar.



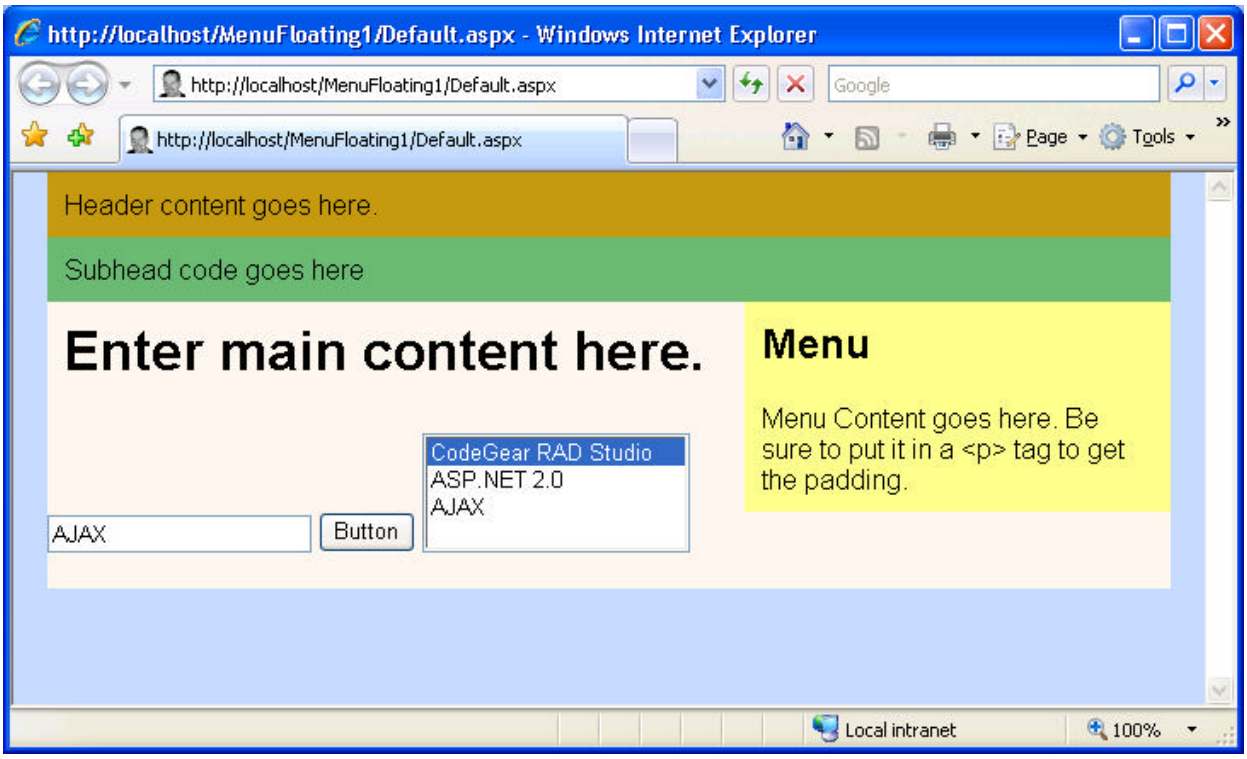
ASP.NET 2.0 en AJAX

Naast de CSS Layouts, bevat CodeGear RAD Studio 2007 ook ondersteuning voor ASP.NET 2.0 en AJAX (de officiële Microsoft AJAX voor ASP.NET controls). Er is hier een speciale wizard voor die een ASP.NET Ajax project maakt.
Het resultaat is een pagina met een ScriptManager en een UpdatePanel, de twee onderdelen van AJAX in ASP.NET. De ScriptManager zorgt voor het uitvoeren van de events, en de UpdatePanel is het visuele onderdeel dat de controls bevat die ge-update worden door de AJAX events:

```
    <form runat="server">
    <asp:ScriptManager id="ScriptManager1" runat="server" />
      <div>
        <asp:UpdatePanel id="UpdatePanel1" runat="server"></asp:UpdatePanel>
      </div>
</form>
```

Als we echter met een bestaand project bezig zijn, zoals een CSS Layout project, dan is de grote vraag: hoe voegen we daar AJAX aan toe? Het gebruik van de ScriptManager en UpdatePanel kan niet zomaar; daar moet je wel wat voor doen.
De details zijn te vinden in de web.config van het ASP.NET AJAX Project. Hier moeten we de volgende secties uit kopiëren. Direct onder de <configuration> node moeten we het <configSections> deel van de AJAX web.config overnemen, en vervolgens in de <system.web> sectie de nodes <pages>, <compilation>, <httpHandlers> en <httpModules>. Dat is voldoende om ook een "normale" ASP.NET toepassing geschikt te maken voor AJAX.

Als laatste stap moeten we de System.Web.Extensions.dll assembly toevoegen aan de References lijst van het project, en daarna is het moment gekomen waarin we ook in het bestaande niet-AJAX ASP.NET project gebruik kunnen maken van de ScriptManager en de UpdatePanel om er AJAX features aan toe te voegen.
De TextBox, Button en ListBox in de volgende screenshot zitten in de UpdatePanel, en als gevolg daarvan zullen alle update acties achter de schermen gebeuren zonder een volledige pagina refresh. Zo eenvoudig is het toevoegen van AJAX aan ASP.NET 2.0 (alhoewel er ook bijzondere situaties zijn, zoals het updaten van controls buiten het UpdatePanel, en daar kom ik de volgende keer op terug).



Delphi boeken op Lulu.com

De afgelopen tijd heb ik enkele Delphi cursusboeken omgezet tot een PDF formaat dat bruikbaar was om er door Lulu een boekje van te laten maken. Ik heb twee verschillende formaten aangemaakt: die van maximaal 88 pagina's met een nietje in de rug (de saddle-stitch versie) en de versie vanaf 68 pagina's (maximaal 680) met een zijkaft waarop de titel gedrukt kan worden.

De volgende Delphi titels zijn nu te koop:

- Delphi 2007 for Win32 Development Essentials (104 pagina's – alleen Win32)
- Delphi 2006 Development Essentials (238 pagina's – Win32 en .NET)
- Delphi 2006 ADO.NET 1.1 Database Development (169 pagina's)
- Delphi 2006 Advanced ASP.NET 1.1 Web Development (135 pagina's)
- Delphi Win32 / .NET Component Development (175 pagina's)

En een paar dunne boekjes, voor 17 Euro per stuk:

- Delphi for Win32 VCL Database Development (88 pagina's)
- Delphi 2006 ASP.NET ECO III Weblog (63 pagina's)

Ik werk ook aan nieuwe Delphi 2007 titels, waaronder eentje over ASP.NET 2.0 Web Development, eentje over Win32 Web Development (met WebSnap en IntraWeb) en eentje over XML, SOAP en Web Services.
Zie http://stores.lulu.com/drbob42 voor een compleet overzicht van deze en de andere publicaties die ik op Lulu.com heb gezet, of http://www.eBob42.com/courseware voor de cursusboeken die nog niet op Lulu.com staan omdat ik er nog regelmatig training mee geef en uitbreidingen op aanbreng.

Delphi support per e-mail

Naast het geven van training & consultancy (en het spreken op seminars en schrijven van artikelen), ben ik per 1 juni ook begonnen met het geven van structurele ondersteuning per e-mail. Dit betekent dat Delphi ontwikkelaars mij per e-mail vragen en problemen kunnen voorlegen. De prijs is 100 Euro per (kalender)maand, met een korting voor wie meteen voor een kalenderjaar betaald; dan is de prijs 1000 Euro, ex. BTW.
Voor die prijs kan een ontwikkelaar mij vragen per e-mail stellen die ik dan "snel" en in detail zal proberen te beantwoorden. Meestal zal "snel" betekenen binnen een paar uur, vrijwel altijd nog dezelfde dag (er zijn uitzonderingen, als ik bijvoorbeeld die dag training geef of in het buitenland ben).
Ik kan helaas geen garantie geven dat ik alle vragen zal kunnen beantwoorden, maar zal wel altijd proberen om relevante informatie of verwijzingen op te zoeken. Mocht de "service" niet bevallen, dan kan er altijd aan het eind van een periode gestopt worden (maar er vind geen restitutie plaats van betaalde bedragen).
Indien een vraag extra uitzoekwerk zou verlangen, dan meld ik dat eerst (plus een schatting hoeveel tijd (en geld) dat extra uitzoeken met zich mee zou brengen). Je kan dan zelf bepalen of het de moeite waard is om mij dat uit te laten zoeken. Iemand heeft mij bijvoorbeeld wel eens laten uitzoeken hoe je een bepaald "vingerafdruk"-scan apparaat kunt gebruiken, waarbij alleen een DLL en een C header file beschikbaar was. Dat is in de vorm van een klein project van enkele dagen uitgevoerd. Tot volle tevredenheid van beide partijen overigens.

Marco Cantù is pas geleden ook met een vergelijkbare service begonnen. Alleen doet hij het niet per e-mail, maar in private nieuwsgroepen, en doet hij het niet alleen maar heeft hij een team van zo'n 10 personen die de vragen beantwoorden.
Bij mij krijgt u altijd alleen antwoord van mijzelf en per e-mail. Omdat ik het alleen doe zijn er echter maar een beperkt aantal "plaatsen" beschikbaar. Mail Bob@eBob42.com voor meer informatie.

Delphi bestellen bij Bob Swart 

Onderstaande prijzen zijn exclusief 19% BTW en gelden voor de Electronic Software Delivery (ESD) edities, die u kunt downloaden en kunt installeren met een bijbehorend serienummer dat u ontvangt. Een ESD licentie is goedkoper en ontvangt u sneller dan wanneer u op een doos + DVD moet wachten.

Delphi for PHP (ESD)

New User	€ 234,=
Subscription	€ 99,= per jaar

Delphi for PHP is een nieuwe ontwikkelomgeving met de look-en-feel van de Delphi IDE, maar PHP als ontwikkeltaal.

Delphi 2007 for Win32 Enterprise R2 (ESD)

New User	€ 1.724,=
Upgrade	€ 1.124,=
Subscription Delphi for Win32	€ 525,= per jaar

Delphi 2007 for Win32 Professional R2 (ESD)

New User	€ 774,=
Upgrade	€ 324,=
Subscription Delphi for Win32	€ 235,= per jaar

De R2 in de naam wil zeggen dat dit Delphi 2007 (of C++Builder 2007) is met Update #3 er al meteen in.
Voor C++Builder 2007 R2 gelden dezelfde prijzen. Maar u krijgt dus maar één product voor deze prijs. Wie ze allebei wil hebben – of allemaal, inclusief de Delphi for .NET 2.0 identiteit – is goedkoper uit om de hele CodeGear RAD Studio aan te schaffen. Zie ook http://www.eBob42.com/CodeGear
CodeGear RAD Studio 2007 is beschikbaar in een Architect, Enterprise en Professional editie.

CodeGear RAD Studio 2007 Architect (ESD)

New User	€ 2.614,=
Upgrade	€ 1.974,=
Subscription CodeGear RAD Studio	€ 865,= per jaar

CodeGear RAD Studio 2007 Enterprise (ESD)

New User	€ 1.974,=
Upgrade	€ 1.334,=
Subscription CodeGear RAD Studio	€ 650,= per jaar

CodeGear RAD Studio 2007 Professional (ESD)

New User	€ 934,=
Upgrade	€ 414,=
Subscription CodeGear RAD Studio	€ 315,= per jaar

Special Aanbieding: Van 1 februari 2008 t/m 15 maart 2008 krijgt u een bij een bestelling van Delphi 2007, C++Builder 2007 of CodeGear RAD Studio 2007 een gratis licentie van Delphi for PHP (ter waarde van 234 Euro ex.BTW).
De gratis Delphi for PHP licentie komt in de vorm van een CodeGear Promotion Coupon met instructies voor het downloaden van de Delphi for PHP setup en het bijbehorende serienummer. Laat deze kans niet aan u voorbijgaan om geheel gratis kennis te maken met Delphi for PHP.

Zie http://www.eBob42.com/CodeGear of stuur e-mail naar Bob@eBob42.com voor meer informatie of om een bestelling te plaatsen