Stef van Hooijdonk is architect bij Tam Tam BV

Maak je documentatie compleet

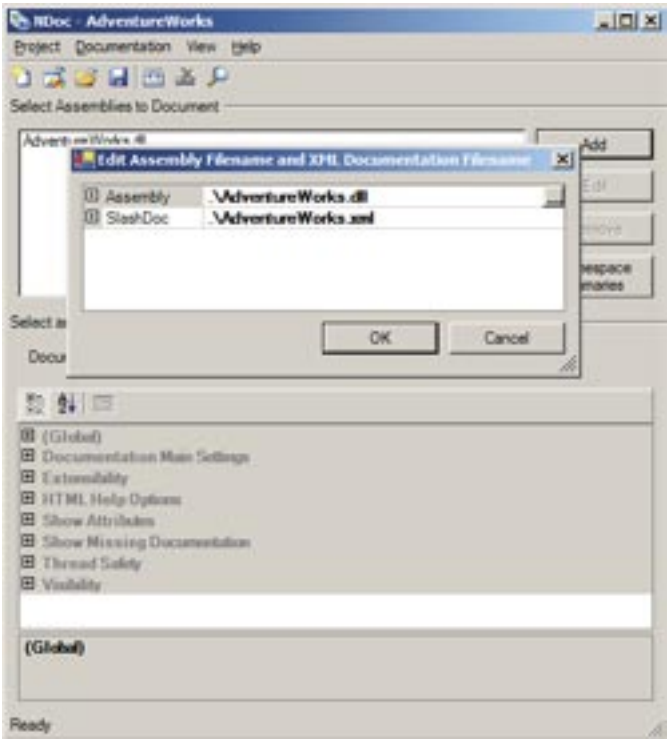
MET NDOC DE CODE- EN SQL-COMMENTS TOT ÉÉN GEHEEL SMEDEN

Vraag het elke ontwikkelaar en ze zullen allemaal erkennen dat documenteren van code een goede zaak is. Hoe groter de omvang van het project hoe belangrijker goede documentatie is. Dit betekent wel dat zoveel mogelijk onderdelen van jouw oplossing zo goed mogelijk moeten worden beschreven. In dit artikel beschrijft de auteur het belang van documenteren van code, waarbij hij commentaar in code en SQL Server via NDoc genereert.

Met de release van Visual Studio 2002 kwamen de beroemde 'triple slashes', waarmee je elke *class*, *method* en *property* kon voorzien van een omschrijving. Met daarbij een functie die al het commentaar van de code van jouw oplossing in mooi opgemaakte HTML-pagina's kon samenvatten: zogeheten 'Build Comment Web Pages.' Nu kon je gemakkelijk al je code voorzien van een eenduidig en beschrijvend commentaar, en dat op een redelijke manier bekijken of zelfs opleveren bij je project of product.

NDoc

Toch bleek al snel dat de Comment Web Pages die Visual Studio genereerde niet optimaal waren. De mensen achter het open source NDoc zijn hier op ingesprongen. NDoc is een tool die aan de hand van de assemblies documentatie kan genereren; zie afbeelding 1. Voor een optimaal resultaat moet het project zo zijn ingesteld dat ook het bijbehorende slashdoc gegenereerd kan worden. Slashdoc is een XML-bestand waarin alle code-comments zijn opgenomen die bij de types, properties en methods zijn opgenomen in de broncode.



Afbeelding 1. NDoc genereert documentatie aan de hand van de assemblies

SQL Server

In Microsoft SQL Server, en andere bekende databasesystemen, kan je sinds jaar en dag je 'code' ook al voorzien van commentaar en uitleg. Elke *table*, *field*, *view*, *stored procedure* en *user defined function* kan worden voorzien van dit belangrijke commentaar. Daarnaast bevat SQL Server natuurlijk een hoop informatie over het datamodel van de database die bij jouw oplossing hoort, zoals veldtypes, beperkingen en parent child-relaties.

SQL Server-commentaar opnemen in je NDoc

Tot zover eigenlijk niets nieuws. Wat we nu willen gaan doen, is deze twee bronnen van informatie, de code-comments en SQL-comments, tot één geheel smeden. NDoc gebruikt .Net-assemblies als bron voor het genereren van documentatie. Wanneer we het datamodel van de database kunnen samenvatten in een assembly, met bijbehorende slashdoc, kunnen we alle features van NDoc gebruiken om de code-comments aan te vullen en uit te breiden met de SQL-comments. We gaan een consoleapplicatie schrijven die met een aantal system stored procedures binnen SQL Server het datamodel gaat inlezen en dan zo'n assembly en slashdoc.xml kan genereren.

Het genereren gaat in een aantal stappen:

1. het inlezen van het datamodel naar objecten
2. deze objecten gebruiken voor het genereren van een C#-bronbestand
3. dit C#-bronbestand compileren tot een assembly
4. deze objecten gebruiken voor het genereren van een slashdot.xml-bronbestand.

Datamodel uit SQL halen

Voor de eerste stap, het inlezen, hebben we nodig:

- een class voor een veld/kolom (zie codevoorbeeld 1)
- een class voor een tabel/view (zie codevoorbeeld 1)
- een class voor een stored procedure/udf
- en een class voor de database zelf.

Tabellen en views

Als eerste gaan we alle tabellen en views uit de database inlezen en omzetten naar objecten in onze consoleapplicatie. De tabellen en views komen uit dezelfde stored procedure 'sp_tables'. Een 'type' kolom geeft aan of het om een view of een tabel gaat. Daarnaast worden ook de primary keys uitgelezen. Ik gebruik de Enterprise Library v1.1 voor toegang tot de database, zie codevoorbeeld 2.

Stored procedures en user defined functions

Voor het inlezen van de stored procedures en de user defined functions zijn vergelijkbare system stored procedures aanwezig. In tabel

```
/// <summary>
/// Beschrijft een database veld.
/// </summary>
public class SqlField{
  public string Name;
  public string SqlName;
  public string Type;
  public bool Required;
  public int MaxLength;
  public bool Primairy;
  public string Remarks;
}

/// <summary>
/// Beschrijft een tabel uit een database.
/// </summary>
public class SqlTable {

  /// <summary>
  /// De velden van een tabel,
  /// bevat <c>SqlField</c> objecten.
  /// </summary>
  public SortedList Fields = new SortedList();

  /// <summary>
  /// De primairy keys van een tabel,
  /// bevat <c>SqlField</c> objecten.
  /// </summary>
  public SortedList PrimairyKeys = new SortedList();

  /// <summary>
  /// De foreign keys van een tabel,
  /// bevat <c>SqlForeignKey</c> objecten.
  /// </summary>
  public SortedList RelatedTables = new SortedList();

  public string Name;
  public string SqlName;
  public string Owner;
  public string Remarks;
}

/// <summary>Bevat alles van een Sql Database,
/// tabellen, views en sprocs</summary>
public class SqlDatabase {

  /// <summary>Tabellen in deze database</summary>
  public SortedList Tables = new SortedList();

  /// <summary>Views in deze database</summary>
  public SortedList Views = new SortedList();

  /// <summary>Storedprocedures in deze database</summary>
  public SortedList Sprocs = new SortedList();

  /// <summary>SQLName of deze Database</summary>
  public string Name;

  /// <summary>Versienummer van
  /// de gebruikte SQL Server</summary>
  public string SqlVersion = "";

  /// <summary>De Displayname, wordt gebruikt
  /// als Namespace en C# Class naam</summary>
  public string DisplayName;
```

Codevoorbeeld 1.

```
SqlDatabase result = new SqlDatabase();
result.Name = databasename;
result.DisplayName = displayname;

Database sourcedb = DatabaseFactory.CreateDatabase( databasename );
// we hebben het versie nummer later nodig om een verschil
// tussen SQL Server 2000 en SQL Server 2005 te omzeilen
using( IDataReader idr = sourcedb.ExecuteReader(
     CommandType.Text, "SELECT @@VERSION as version" ) ) {
  if ( idr.Read() ) {
     result.SqlVersion = DataReading.ReadStringDataField(idr, "version" );
  }
}
// lees de tabellen van de database
using( IDataReader idr = sourcedb.ExecuteReader(CommandType.Text,
     "sp_tables" )) {
  while( idr.Read() ) {
     string type = DataReading.ReadStringDataField(idr, "TABLE_TYPE" );

     SqlTable table = new SqlTable();
     table.SqlName = DataReading.ReadStringDataField(idr, "TABLE_NAME" );
     table.Name = TranslateSqlName( table.SqlName );
     table.Owner  = DataReading.ReadStringDataField(idr, "TABLE_OWNER" );
     // haal alle kolomen/velden op voor deze tabel
     using( IDataReader idr_fields = sourcedb.ExecuteReader(
          CommandType.Text, "sp_columns " + table.Name ) ) {
       while( idr_fields.Read() ) {
          SqlField field = new SqlField();
          field.SqlName  = DataReading.ReadStringDataField(idr_fields,
               "COLUMN_NAME");
          field.Name   = TranslateSqlName( field.SqlName );
          if ( field.Name.ToLower() == table.Name.ToLower() )
            field.Name = "_" + field.Name;
          field.Type   = DataReading.ReadStringDataField(idr_fields,
               "TYPE_NAME" );
          field.MaxLength= DataReading.ReadIntegerDataField(idr_fields,
               "LENGTH" );
          field.Required = DataReading.ReadBooleanDataField(idr_fields,
               "NULLABLE" );
          // voeg dit veld toe aan de tabel
          table.Fields.Add( field.SqlName, field );
       }
     }
     // haal alle primary key's op voor deze tabel
     using( IDataReader idr_constraints = sourcedb.ExecuteReader(
          CommandType.Text, "sp_pkeys " + table.Name) ) {
       while( idr_constraints.Read() ) {
          string p = DataReading.ReadStringDataField(idr_constraints,
               "COLUMN_NAME");
          foreach( SqlField f in table.Fields.Values ) {
            if (f.SqlName == p ){
               f.Primairy= true;
               table.PrimairyKeys.Add(f.SqlName, f );
            }
          }
       }
     }
     if ( type == "TABLE" )
       result.Tables.Add(table.SqlName, table );
     if ( type == "VIEW" )
       result.Views.Add(table.SqlName, table );
  }
}
```

Codevoorbeeld 2.

```
/// <summary>
/// Sql Statement voor het inlezen van descriptions
/// op tables, views, sprocs and hun kolommen/params
/// </summary>
/// <remarks>SQL 2000 versie</remarks>
public const string GetDescriptions = @"SELECT
   sysobjects.xtype,
   sysobjects.name AS TABLE_NAME,
      desc2.[value] AS TABLE_DESCRIPTION,
   syscolumns.name AS COLUMN_NAME,
      desc1.[value] AS COLUMN_DESCRIPTION
FROM
   sysobjects
      LEFT JOIN syscolumns ON syscolumns.id = sysobjects.id
      LEFT JOIN sysproperties desc1 ON (
         ( desc1.smallid = syscolumns.colid ) AND
         ( desc1.id = syscolumns.id ) AND
         ( desc1.[Name] = 'MS_Description' ) )
      LEFT JOIN sysproperties desc2 ON (desc2.id = sysobjects.id
         and desc2.smallid = 0 and
         desc2.[Name] = 'MS_Description')
where
   sysobjects.xtype = 'P' or
   sysobjects.xtype = 'U' or
   sysobjects.xtype = 'V'
ORDER BY
   sysobjects.name,syscolumns.name";

/// <summary>
/// Sql Statement voor het inlezen van descriptions
/// op tables, views, sprocs and hun kolommen/params/// </summary>
/// <remarks>SQL 2005 versie</remarks>
public const string GetDescriptions2005 = @"SELECT
   sysobjects.xtype,
   sysobjects.name AS TABLE_NAME,
      desc2.[value] AS TABLE_DESCRIPTION,
   syscolumns.name AS COLUMN_NAME,
      desc1.[value] AS COLUMN_DESCRIPTION
FROM
   sysobjects
      LEFT JOIN syscolumns ON syscolumns.id = sysobjects.id
      LEFT JOIN sys.extended_properties desc1 ON (
         ( desc1.minor_id = syscolumns.colid ) AND
         ( desc1.major_id = syscolumns.id ) AND
         ( desc1.[Name] = 'MS_Description' ) )
      LEFT JOIN sys.extended_properties desc2 ON (
         ( desc2.major_id = sysobjects.id ) AND
         ( desc2.minor_id = 0 ) AND
         ( desc2.[Name] = 'MS_Description' ) )
where
   sysobjects.xtype = 'P' or
   sysobjects.xtype = 'U' or
   sysobjects.xtype = 'V'
ORDER BY
   sysobjects.name,syscolumns.name ";
```

Codevoorbeeld 3

```
string getDescriptions = SqlDatabase.GetDescriptions;

// check of we met een SQL2005 Server verbinding hebben
if ( result.SqlVersion.ToUpper().StartsWith(
     "MICROSOFT SQL SERVER 2005" ) )
   getDescriptions = SqlDatabase.GetDescriptions2005;

// begin met het lezen van de descriptions
using( IDataReader idr = sourcedb.ExecuteReader(
     CommandType.Text, getDescriptions ) ) {
   while( idr.Read() ) {
   }
}
```

Codevoorbeeld 4

1 staat een overzicht van de gebruikte stored procedures voor het inlezen van het datamodel. Wat nu nog ontbreekt, is het inlezen van alle omschrijvingen die we hebben opgegeven in SQL Server. En daar hebben we het versienummer van de SQL Server voor nodig. In codevoorbeeld 3 staan twee varianten voor het inlezen van deze omschrijvingen: een voor SQL Server 2000 en een voor SQL Server 2005, In codevoorbeeld 4 staat de code die wordt toegevoegd aan het inlezen.

Generen van de assembly

Na het inlezen van het datamodel uit SQL Server beschikken we over alle objecten die we nodig hebben voor het genereren van onze assembly. Tabellen en views in SQL hebben velden, iets wat goed overeenkomt met een class met properties. De udf's en sprocs hebben parameters wat weer goed overeenkomt met methods. Maar methods hebben een class nodig. Voor deze eerste versie van de consoleapplicatie gaan we uit van drie namespaces:

- Databases.Sql.AdventureWorks.Tables, waarin alle tabel classes zitten
- Databases.Sql.AdventureWorks.Views met daarin alle view classes en
- Databases.Sql.AdventureWorks met daarin de class met de methods voor de sprocs en udf's.

Het genereren van een assembly gaan we doen met een tussenstap: een C#-bronbestand. Er gelden echter wel verschillende regels in C# en SQL als het gaat om wat er mag op het gebied van naamgeving van objecten en classes. Daarvoor worden de karakters die niet toegestaan zijn in C# vervangen door een underscore in de namen van de SQL-objecten. De oorspronkelijke naam die in SQL Server wordt gebruikt, wordt in de slashdoc wel uitgeschreven.

In codevoorbeeld 5, waarin we alle tabellen uitschrijven als classes, zie je hoe we een nieuw tekstbestand aanmaken en daarin de C#-classes gaan uitschrijven. De classes die overeenkomen met de tabellen krijgen properties die overeenkomen met de kolommen die we hebben ingelezen. Natuurlijk kan hetzelfde gedaan worden voor de views. Het bronbestand bevat nu dus twee namespaces met daarin classes voor alle tabellen en alle views. Nu gaan we in dezelfde C#-file een derde namespace toevoegen en daarin de laatste class uitschrijven. Deze class geven we de methods die overeenkomen met alle stored procedures en de user defined functions; zie de downloadbare voorbeeldoutput voor het eindresultaat van het uitschrijven van al onze SQL-objecten in het C#-bronbestand gebaseerd op de AdventureWorks-database van SQL Server 2005. Nu hebben we dus een bronbestand dat gecompileerd kan worden, en dat gaat aan de hand van codevoorbeeld 6. Een ICodeCompiler (in ons geval een C#CodeCompiler) heeft eigenlijk maar drie belangrijke parameters:

- welke referenced assemblies hebben we nodig?
- wat wordt de uitvoer? en
- wat is het bronbestand?

Na het compileren hebben we dus twee bestanden: het C#-bronbestand en een assembly.

Stored procedures	Omschrijving
sp_tables	Inlezen van de tabellen en views van een database
sp_columns	Inlezen van de kolommen van een tabel of view
sp_pkeys	Inlezen van de primary keys van een tabel
sp_fkeys	Inlezen van de foreign keys van een tabel
sp_sproc_columns	Inlezen van de stored procedures, user defined functions en de parameters daarvoor.

Tabel 1. Gebruikte stored procedures voor het inlezen van het datamodel:

```
/// <summary>
/// Generate a C# File with classes and methods and properties
/// for the supplied SqlDatabase
/// </summary>
/// <param name="database">Source database</param>
/// <returns>the filename</returns>
public string MakeCSFile( SqlDatabase database ) {
   csnamespace       = "Databases.Sql." + database.DisplayName;
   csnamespacetables = csnamespace + ".Tables";
   csnamespaceviews  = csnamespace + ".Views";
   // Creates a text file to store the new classes
   string csfile = string.Format( "{0}.cs", csnamespace );
   using( StreamWriter streamWriter = File.CreateText( csfile ) ) {
      // usings
      streamWriter.WriteLine ("using System;");
      streamWriter.WriteLine ("using System.Reflection;");
      streamWriter.WriteLine ("using System.Runtime.CompilerServices;");
      streamWriter.WriteLine ("using System.Data;");
      streamWriter.WriteLine ("using Microsoft.Practices.EnterpriseLibrary.
         Data;");
      streamWriter.WriteLine ("");
      // assembly header
      streamWriter.WriteLine ("[assembly: AssemblyTitle(\"{0}\")]",
         csnamespace );
      streamWriter.WriteLine ("[assembly:
            AssemblyVersion(\"{0:yyyy.MM.dd.HHmm}\")]",
         DateTime.Now );
      streamWriter.WriteLine ("[assembly: AssemblyCompany(\"{0}\")]",
         database.DisplayName );
      streamWriter.WriteLine ("");

      streamWriter.WriteLine("/// <summary>");
      streamWriter.WriteLine("/// Tables in Database {0}",
         database.DisplayName);
      streamWriter.WriteLine("/// </summary>");
      streamWriter.WriteLine("namespace {0} {{", csnamespacetables );
      foreach( SqlTable table in database.Tables.Values ) {
         streamWriter.WriteLine (" /// <summary> " );
         streamWriter.WriteLine (" /// Table with the name {1}. Owner
            is {0}", table.Owner, table.SqlName );
         streamWriter.WriteLine (" /// </summary> " );
         streamWriter.WriteLine (" public class {0}_{1} {{ ",
            table.Owner, table.Name);
         int u = 0;
         foreach( SqlField field in table.Fields.Values ) {
            string type = TranslateSqlFieldTypeToCode(field.Type );
            streamWriter.WriteLine(" private {1} f_{0}{2}; // database field",
               field.Name.ToLower() , type, u );
            streamWriter.WriteLine("  /// <summary> " );
            streamWriter.WriteLine("  /// Field with the name {1} for {0}.",
               table.SqlName, field.SqlName );
            streamWriter.WriteLine("  /// </summary> " );
            streamWriter.WriteLine("  public {1} {0} {{", field.Name, type );
            streamWriter.WriteLine("    get {{ return f_{0}{1}; }}",
               field.Name.ToLower() ,u );
            streamWriter.WriteLine("  }" );
            u++;
         }
         streamWriter.WriteLine (" }"); // end of table class
      }
      streamWriter.WriteLine (" }"); // end of namespace
   }
}
```

Codevoorbeeld 5

```
// Create the C# compiler
CSharpCodeProvider csCompiler = new CSharpCodeProvider();
ICodeCompiler iCodeCompiler = csCompiler.CreateCompiler();

// input params for the compiler
CompilerParameters compilerParams = new CompilerParameters();
compilerParams.OutputAssembly = string.Format(
   "Databases.Sql.{0}.dll", displayname );
compilerParams.ReferencedAssemblies.Add("system.dll");
compilerParams.ReferencedAssemblies.Add("system.data.dll");
compilerParams.ReferencedAssemblies.Add("EnterpriseLibraryAddOn2005.dll");

compilerParams.GenerateExecutable = false;
compilerParams.GenerateInMemory = false;
// Run the compiler and build the assembly
CompilerResults cr = iCodeCompiler.CompileAssemblyFromFile(compilerParams,
csfile);
```

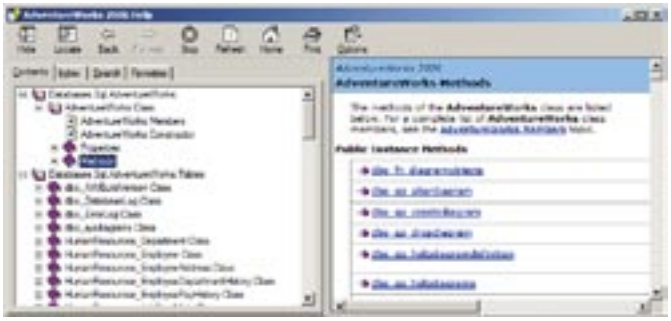
Codevoorbeeld 6

Genereren van het slashdoc.xml

De assembly bevat alleen de structuur en de naamgeving van de verschillende SQL-objecten. Ons SQL-commentaar of de extra informatie die we hebben opgehaald over de SQL type-info en de relaties zien we daarin nog niet terug. Het uitschrijven van een slashdoc XML-bestand gaat ongeveer op dezelfde manier. Voor elke class, method en property wordt een <member>-element opgenomen in het slashdoc.xml. Nu hebben we bij het uitschrijven van het C#-bronbestand een bepaalde structuur aangehouden voor ons datamodel. We zullen dit model dus weer precies moeten gaan volgen om het juiste commentaar aan de juiste class in ons C#-bronbestand te koppelen. Wat NDoc namelijk doet, is via reflection en de type-info door alle types in de assembly heen itereren, en elk type en de properties, de fields en de methods daarvan worden opgenomen in de uitvoer van NDoc. Ons slashdoc-bestand moet daar dus het juiste commentaar voor gaan genereren.

In de <summary> van tabellen schrijven we wat alle primary en foreign keys van die tabel zijn. In de <summary> en de <remarks> van de velden schrijven we het SQL-datatype, de maximale lengte en de verplichtheid uit. Het gevonden commentaar schrijven in beide gevallen als eerste paragraaf in de <remarks>. Ook kunnen we nu als <example> een SQL-statement uitschrijven die een record selecteert en een update uitvoert. Voor de views doen we hetzelfde. Voor de sprocs en udf's gaan we de parameters beschrijven. Ook het soort udf - is het resultaat een tabel of een waarde - wordt opgenomen.

Bij het uitschrijven van het <member>-element voor een tabel in onze database en in ons C#-bronbestand kunnen we in de in de beschrijving opnemen welke foreign keys we hebben. In codevoorbeeld 7 is te zien dat de derde paragraaf van het <summary>-element verrijkt is met informatie over foreign keys. De <remarks> zijn gevuld met het gevonden commentaar uit de database. Nu hebben we dus een assembly en een slashdoc.xml-bestand. In ons NDoc-projectfile kunnen we dus deze twee gaan toevoegen. Nu



Afbeelding 2. Het resultaat van de uitvoer van NDoc op basis van de AdventureWorks-database.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<doc>
   <assembly>
      <name>AdventureWorks</name>
   </assembly>
   <members>
      <member
         name="T:Databases.Sql.AdventureWorks.Tables.Person_Address">
         <summary>
            <para>Table with the name Address. Owner/Schema is Person.</para>
            <para>No primairy keys defined!</para>
            <para>Foreign key at dbo.OtherTable.OtherField to
                  this Table's field SomeField</para>
         </summary>
         <remarks>Street address information for customers,
                     employees, and vendors.</remarks>
      </member>
   </members>
</doc>
```

Codevoorbeeld 7

bevat het NDoc-projectfile assemblies van jouw solution en van de assemblies die we hier gegenereerd hebben. In afbeelding 2 zie je het resultaat van de uitvoer van NDoc op basis van de AdventureWorks-database die met SQL Server 2005 wordt meegeleverd.

Automatisch genereren

Naast een gewone versie van NDoc is ook een consoleversie beschikbaar. Deze versie is ideaal als je NDoc als build-stap in je build-proces wilt opnemen. Wanneer je het uitschijven van je datamodel nu ook in je build-proces opneemt, wordt bij een documentation-build ook de laatste versie van je datamodel in je documentatie opgenomen.

Tot slot

Een bijkomend voordeel van het genereren van een assembly is dat je nu een assembly hebt waarin alle objecten overeenkomen met jouw datamodel in SQL Server. Wanneer je in een Visual Studio-project deze assembly nu als reference toevoegt, kan je bij je methods en functions ook in je eigen codecommentaar een referentie (“<see cref/>”) opnemen naar de bijbehorende tabel of sproc die in de code wordt gebruikt. Een tabel is bijvoorbeeld te vinden in de namespace Databases.Sql.AdventureWorks.Tables. Door een <see cref = ”Databases.Sql.AdventureWorks.Tables.MyTable” /> op te nemen in je <summary> tag in het commentaar kan de lezer van je documentatie meteen zien welke tabel jouw code gebruikt.

Op de website van het .NET Magazine (www.microsoft.nl/netmagazine12) kun je de voorbeeldcode downloaden. De code bevat een project en een aantal van de assemblies van de Enterprise Library. Wanneer je deze al geïnstalleerd hebt, kan je de referenties in de projectfile veranderen. De downloadbare voorbeeldoutput bevat de output van deze consoleapplicatie die gebaseerd is op de AdventureWorks-database van SQL Server 2005.

Stef van Hooijdonk is architect bij Tam Tam BV (www.tamtam.nl). Naast de rol van architect van Tam Tam en Lead developer, coached Stef de jongere ontwikkelaars binnen Tam Tam. Zijn blog is te lezen op http://blogs.tamtam.nl/stef. Voor vragen is Stef te bereiken via email stef.van.hooijdonk@tamtam.nl

Referenties

SQL Server System Stored Procedures: msdn.microsoft.com/library/en-us/tsqlref/ts_sp_00_519s.asp
NDoc homepage: ndoc.sourceforge.net/
Voorbeeldcode: www.microsoft.nl/netmagazine12