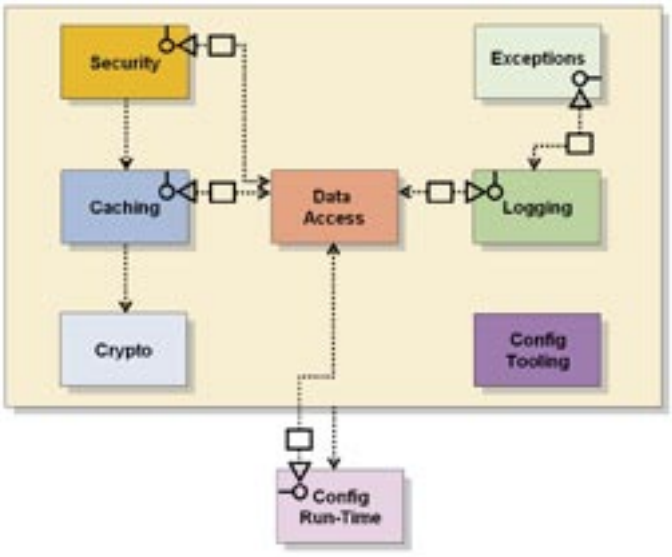
Dennis Mulder is senior consultant bij Avanade.

Application blocks klaar voor de enterprise

BASEER NIEUWE APPLICATIES OP ENTLIB 1.0

De Microsoft Patterns & Practices Group heeft goed geluisterd naar de behoeften van de ontwikkelaarsgemeenschap. Met de komst van Enterprise Library (EntLib) versie 1.0 heeft Microsoft een groep veel gebruikte application blocks herschreven met als doel ze beter met elkaar te laten samenwerken, uitbreidbaar en consistenter te maken en vooral ook om applicaties beter en makkelijker te kunnen configureren. Niet alleen zijn er zaken verbeterd, ook zijn nieuwe ideeën onderdeel geworden van een geïntegreerde library die de productiviteit van iedere programmeur verhoogt. Waar moet je rekening mee houden als je applicaties gaat ontwikkelen die gebruikmaken van EntLib?

De eerste versie van EntLib zal zeven application blocks bevatten, te weten Data Access, Exception Handling, Configuration, Caching, Logging & Instrumentation, Security en Cryptography. In afbeelding 1 is de architectuur van EntLib te zien. Nu zal je zeggen, de meeste hadden we toch al? En waar zijn smart client, UIP, updater, aggregation, asynchronous invocation, enzovoort? Uiteraard zal Microsoft in toekomstige versies uitbreidingen leveren met mogelijk enkele toevoegingen van de oude application blocks. Maar om zo snel mogelijk te komen tot een eerste versie is besloten alleen de genoemde blocks op te nemen. Ook heb ik al beloftes van Microsoft gezien voor de levering van nieuwe updates van EntLib specifiek voor Service Oriented Architecture (SOA) en Aspect Oriented Programming (AOP), maar ook ter ondersteuning van het .NET Framework 2.0 in verband met de komst van Visual Studio 2005 (Whidbey). De eerste versie van EntLib kunnen we verwachten in januari 2005. De download bevat naast de volledige broncode ook 1900 unittests die het ontwikkelteam gebruikt heeft bij het ontwikkelen van EntLib. Tijdens het ontwikkelen van EntLib is uitgebreid gebruik gemaakt van de principes van 'agile development' en 'test driven development'. Mocht je aanpassingen willen doen, dan kun je met NUnit controleren of je geen code van Microsoft breekt met je wijzigingen. De uitgebreide documentatie is ook de moeite waard. Bovendien is het de bedoeling om voor EntLib een actieve community te creëren, zodat er mogelijk meer blocks of uitbreidingen op blocks beschikbaar komen.



Afbeelding 1. Architectuur van EntLib versie 1.0

ACA.NET

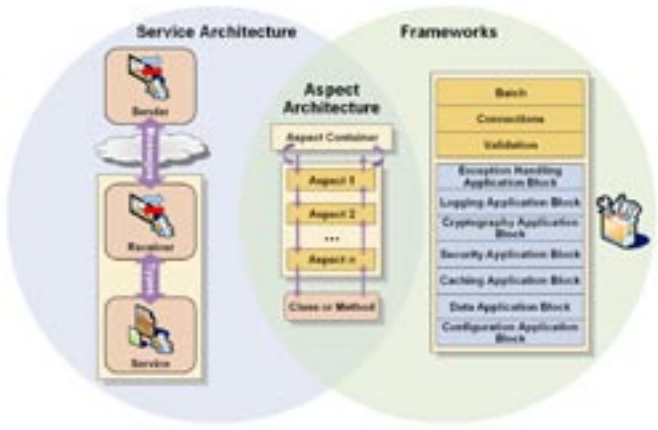
EntLib is niet iets nieuws. Veel van de ideeën die in EntLib zijn toegepast, komen uit een intern door Avanade ontwikkelde library genaamd ACA.NET. De nieuwe versie van ACA.NET voegt functionaliteit toe aan EntLib. De blocks in ACA.NET zijn vervangen door EntLib en daar waar Microsoft nog niet ver genoeg is met bijvoorbeeld Service- en Aspect-oriëntatie, vult Avanade dit aan. In afbeelding 2 is een architectuurplaatje te zien van ACA.NET.

De blocks

Zoals we gewend zijn van de application blocks is ook in EntLib een sterke afhankelijkheid te zien van het data-access block. Dat is logisch want de security-, logging- en caching-blocks kunnen niet zonder data-access. De blocks kunnen ook niet zonder het configuration-block. Met dit block zijn de diverse blocks beter met elkaar te combineren.

Data-access

Dit vernieuwde block is nog steeds eenvoudig in gebruik, maar heeft toch enkele krachtige features erbij gekregen. Zo wordt door slim gebruik te maken van het factory- en provider-pattern standaard ondersteuning geleverd voor SQL Server, Oracle en DB2, en is het toevoegen van een nieuwe database relatief eenvoudig. Ondersteuning voor databasetransacties is nu ook aanwezig en goed gedocumenteerd. In codevoorbeeld 1 staat een simpel stukje code waarmee in de Northwind Sample-database van SQL Server alle klanten worden opgehaald. Het is niet meer te zien dat de data uit SQL Server komt en ook de connection-string wordt onder de



Afbeelding 2. Architectuur van ACA.NET versie 4.0

motorkap opgehaald uit de configuratie. Vanzelfsprekend hoort er nog exception handling-code bij, maar dat is voor het gemak achterwege gelaten.

Exception management

Het Exception Handling Management Block maakt exception handling aanzienlijk eenvoudiger en maakt het mogelijk exceptions eenduidig in applicaties af te handelen. Zo kan het een eis aan de applicatie zijn dat alle exceptions in de data-accesslaag worden gelogd en in een generieke exception worden verpakt voordat ze worden doorgestuurd naar de businesslaag. Dit was altijd al mogelijk, maar het was zeer bewerkelijk om het gedrag hiervan in een later stadium te veranderen. Met het Exception Handling application block van EntLib wordt alles een stuk simpeler doordat het mogelijk is in de configuratie diverse policies op te geven. In codevoorbeeld 2 is te zien hoe een exception wordt ingepakt in een andere exception op basis van de 'wrap policy' die in de configuratie is gedefinieerd. In deze wrap policy kun je alle typen exceptions opgeven die je wilt afhandelen, en voor ieder type hoe je deze wilt afhandelen. Er zijn drie typen afhandeling mogelijk: *None*, *NotifyRethrow* en *ThrowNewException*. Bij None gebeurt niets, bij NotifyRethrow wordt de aanroepende code geadviseerd de exception door te geven en bij ThrowNewException wordt een nieuwe exception doorgegeven. Vervolgens is het mogelijk aan elke exception een actie te koppelen, zoals loggen, inpakken of vervangen van exceptions. Let wel, dit is configuratie. Je ziet immers in de code niet dat er ook gelogd wordt, maar toch is dit het geval als je de wrap policy op de juiste manier configureert. Deze configuratie is altijd naderhand aan te passen, bijvoorbeeld als blijkt dat in een stuk code, waar je een policy gebruikt die normaal gesproken warnings logt, veelvuldig een exception optreedt. Als je van dit type error voortaan liever errors in plaats van warnings in je log wilt hebben, hoef je alleen de configuratie van je applicatie aan te passen. Je zult zien dat het Exception Management Block anders is dan we gewend waren van het oude block. In combinatie met het Logging & Instrumentation Block en de LoggingException Provider is echter hetzelfde te bereiken.

Configuratie

Het Configuration Application Block is het hart van EntLib. Niet alleen omdat het gebruikt wordt door alle andere blocks, maar ook omdat je het kunt inzetten voor eigen uitbreidingen. Het block bestaat uit een runtime library en een designtime editor die te zien is in afbeelding 3. Het screenshot toont de configuratie van de voorbeelden uit codevoorbeeld 1 en 2. Met de runtime library is het, net als met het oude Configuration Management Block, mogelijk om configuratiebestanden te lezen en te bewerken. Met de designtime editor, waar overigens ook een in Visual Studio geïntegreerde versie van beschikbaar komt, kun je tijdens ontwikkeling alle benodigde configuraties instellen. Ontwikkelaars kunnen in een intuïtieve Explorer-achtige interface nieuwe nodes toevoegen aan de configuratie. Door slim gebruik te maken van validatieregels worden fouten direct getoond. Dit geldt zowel voor databaseconnecties die niet werken als voor ongeldige waarden in diverse dropdown-menu's. Ook dit zal een gunstig effect hebben op de productiviteit. Hiermee voorkom je namelijk dat je een applicatie start om er vervolgens achter te komen dat je een typefout in de configuratie hebt gemaakt!

Caching

Het Caching Application Block biedt veelzijdige en uitbreidbare caching. Het block bevat in eerste instantie een memory-cache, maar kan ook geconfigureerd worden om caches op te slaan in bijvoorbeeld een database via het Data Access Application Block. Het block kan direct gebruikt worden om elementen toe te voegen, te verwijderen of te veranderen in de cache. Expiratie en het opschonen van de cache is via configuratie in te stellen. Via helder

```
Database database = DatabaseFactory.CreateDatabase("Northwind");
DbCommandWrapper command = database.GetSqlStringCommandWrapper
   ("SELECT * FROM Customers");
DataSet ds = database.ExecuteDataSet(command);
```

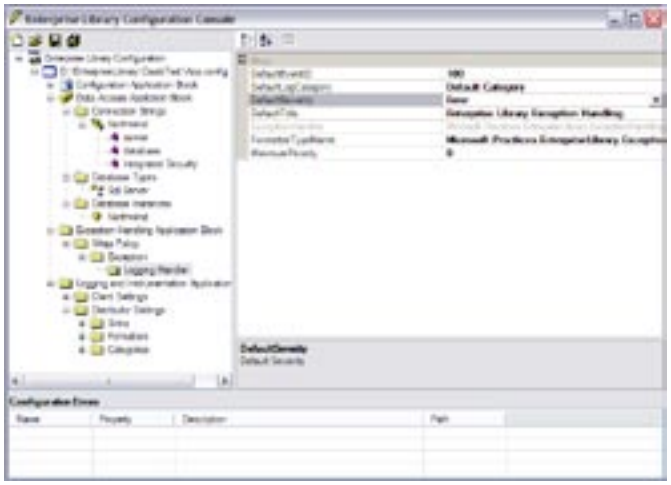
Codevoorbeeld 1.
Data-access block

```
public void ProcessB()
{
   throw new System.Data.DBConcurrencyException("Original Exception:
      Concurrency problem in business layer");
}

public bool ProcessWithWrap()
{
   try
   {
      this.ProcessB();
   }
   catch(Exception ex)
   {
      bool rethrow = ExceptionPolicy.HandleException(ex, "Wrap Policy");

      if (rethrow)
      {
         throw;
      }
   }

   return true;
}
```

Codevoorbeeld 2.
Exception Management Block

```
CacheManager cachemanager = CacheFactory.GetCacheManager("MyCache");
DataSet dsfromcache = cachemanager.GetData("MyDataSet") as DataSet;
if(dsfromcache == null)
{
   Database database = DatabaseFactory.CreateDatabase("Northwind");

   DbCommandWrapper command = database.GetSqlStringCommandWrapper
      ("SELECT * FROM Customers");
   DataSet ds = database.ExecuteDataSet(command);
   cachemanager.Add("MyDataSet", ds);
   return ds;
}
return dsfromcache;
```

Codevoorbeeld 3.
Caching Application Block



Afbeelding 3. Configuration console

```
LogEntry log = new LogEntry();
log.EventId = this.eventForm.EventId;
log.Message = this.eventForm.Message;
log.Category = this.eventForm.Category;
log.Priority = this.eventForm.Priority;
Logger.Write(log);
```

Codevoorbeeld 4.

Logging & Instrumentatie Application Block

```
IIdentity identity;
string username = "denni";
string password = "mulder";

byte[] passwordByteArray = Convert.FromBase64String(password);

NamePasswordCredential credentials = new NamePasswordCredential
   (username, passwordByteArray);

AuthenticationProviderFactory factory;
factory = new AuthenticationProviderFactory
   (ConfigurationManager.GetCurrentContext());

// The authentication provider "Authentication Provider"
   is defined in configuration
IAuthenticationProvider authProvider = factory.GetAuthenticationProvider(
"Authentication Provider");

bool authenticated = authProvider.Authenticate(credentials,
   out this.identity);

// Token for valid identity
IToken token;

// Security cache to hold cached identities
ISecurityCacheProvider cache;

SecurityCacheProviderFactory factory = new SecurityCacheProviderFactory(
ConfigurationManager.GetCurrentContext());
this.cache = factory.GetSecurityCacheProvider();

// Cache the identity.
// The SecurityCache will generate a token which is then returned to us.
this.token = this.cache.SaveIdentity(this.identity);

// Retrieves the identity previously saved by using the
   corresponding code
IIdentity savedIdentity = this.cache.GetIdentity(this.token);
```

Codevoorbeeld 5.

Security Application Block

```
string secretText = "Mijn supergeheime pincode!";
string base64encryptedText = Cryptographer.EncryptSymmetric
("MyCryptoProvider", secretText));

secretText = Cryptographer.DecryptSymmetric("MyProvider",
base64encyptedText);
```

Codevoorbeeld 6.

Cryptography Block

gedefinieerde interfaces is het mogelijk om zelf de manier van herladen van objecten te implementeren als deze uit de cache worden verwijderd. Ook is het mogelijk om zelf de regels te implementeren die bepalen wanneer objecten uit de cache mogen worden verwijderd.
Zoals te zien is in codevoorbeeld 3 is het mogelijk om via een factory een cache uit de configuratie op te halen. Je kunt dus meer caches in je applicatie gebruiken om zo de soorten data te scheiden en bijvoorbeeld de grootte of de expiratie-interval apart in te stellen.

Logging & instrumentatie

Logging is een eis van vrijwel alle enterprise-applicaties. Altijd nauw verweven met exception management heeft Microsoft met het Logging Block van EntLib een nieuwe manier neergezet voor logging, die het vorige Application Block en het Enterprise Instrumentation Framework (EIF) volledig vervangt. Tegelijk maakt het block gebruik van de faciliteiten die de configuratiemanager van EntLib biedt om logging configureerbaar te maken. De kracht is dat het loggen is losgekoppeld van de logging- en instrumentatie-providers. Of je nu wilt loggen in een file, MSMQ, e-mail, databases, WMI, eventlog of zelf een locatie (sink) wilt maken: alles kan ingesteld worden met de configuratiemanager. Op basis van de ernst van een bericht dat wordt gelogd, is het zelfs mogelijk om bijvoorbeeld een ernstig bericht naar een emailontvanger te versturen en tevens een bericht dat dient als waarschuwing te loggen in een bestand.
De formattering van berichten kan worden ingesteld door gebruik te maken van tokens als {message}, {severity}, {timestamp} enzovoort. De formatter-engine is ook uitbreidbaar, zodat je je eigen custom formatter kunt schrijven. In codevoorbeeld 4 zien we een simpel bericht dat wordt gelogd. Op basis van de configuratie zal dit bericht gestuurd worden naar een of meer van de geconfigureerde locaties. Als ontwikkelaar hoef je je dus niet tijdens het ontwikkelen af te vragen waar het gelogd moet worden. Dit bepaalt de beheerder in de EntLib-configuratie.

Security

Met het Security Application Block heeft Microsoft een architectuur neergezet die, net als alle andere blocks, uitbreidbaar en goed geïntegreerd is met de andere blocks. Zo zijn het Security Block en het Cryptography Block een prima combinatie om eenvoudig wachtwoorden te encrypten. Verder biedt het block mogelijkheden voor authenticatie, (op rollen gebaseerde) autorisatie, profielen en caching van security credentials. Authenticatie is standaard mogelijk op basis van ActiveDirectory, Azman en een EntLib-database. Het is relatief eenvoudig om zelf een provider te schrijven voor een eigen authenticatie-database. Op dezelfde manier zijn de overige mogelijkheden uitbreidbaar. In codevoorbeeld 5 authenticeer ik een gebruiker, vraag het token op en cache deze, zodat voor een volgend verzoek de gebruiker op basis van het token weer opgehaald kan worden.

Cryptografie

Het beveiligen van data wordt een steeds belangrijkere eis van enterprise-applicaties. Logisch, want er zijn talloze gevallen bekend waarbij gevoelige data in handen kwamen van de verkeerde mensen. Hoewel cryptografie een onderdeel is van de .NET System.Net.Cryptography namespace wordt het niet genoeg ingezet.
Het Cryptography Application Block stelt je in staat eenvoudi-

ger gebruik te maken van algemene functionaliteit om data in je applicatie te beveiligen. Er zijn diverse encryptie-mechanismen die worden ondersteund. Het gaat te ver om in dit artikel een uitleg te geven over de diverse encryptie-mechanismes. Via de configuratiemanager, hoe kan het ook anders, kunnen cryptografische sleutels worden beheerd. In codevoorbeeld 6 is te zien hoe op basis van de MyProvider, die is geconfigureerd in de configuratiemanager, een tekst wordt versleuteld en ontsleuteld met een symmetrische sleutel. De sleutel wordt met deze provider gebruikt voor zowel het versleutelen als het ontsleutelen.

Minder werk

Mijn advies is nieuwe applicaties te baseren op EntLib. Het neemt een heleboel werk uit handen en is een stuk krachtiger dan de diverse application blocks en open sourceprojecten afzonderlijk. Zeker als je op dit moment in de ontwerpfase van een nieuw project verkeert, kun je met de documentatie op de GotDotNet-workspace (zie referenties) je ontwerpdocumentatie completeren om vanaf januari te starten met EntLib. Vooral het gebruik van de configuratiemanager is even wennen, maar na enkele dagen zul je snel vertrouwd raken en de voordelen ervan inzien.

Dennis Mulder

is senior consultant bij Avanade (www.avanade.com), een samenwerkingsverband van Microsoft en Accenture. Hij is al sinds het allereerste begin van .NET bezig met het ontwerpen en ontwikkelen van oplossingen voor klanten. Het e-mailadres van Dennis is dennism@avanade.com en zijn blog is te vinden op http://www.dennismulder.net/blog.

Nuttige internetadressen

http://www.gotdotnet.com/entlib
http://msdn.microsoft.com/architecture/
http://www.microsoft.com/resources/practices/default.mspx
http://www.microsoft.com/resources/practices/comingsoon.mspx