# Introductie ADO.NET Data Services

## Eenvoudig data verwerken in (web)applicaties

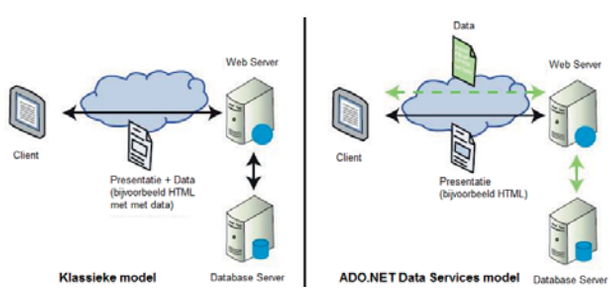
Kevin Timmerman

Microsoft heeft een nieuwe manier ontwikkeld om data te verwerken in applicaties. Dit gebeurt met technologieën als REST, het ADO.NET Entity Framework en LINQ. ADO.NET Data Services is dé techniek van Microsoft om data als een service aan te bieden en te manipuleren in 'the Cloud'. Wat voor mogelijkheden biedt deze nieuwe techniek voor ontwikkelaars? En hoe is het te gebruiken binnen Rich Internet Applications (RIA's) met behulp van C#?

In moderne applicaties willen we op eenvoudige wijze met data kunnen werken. Daarom heeft Microsoft ADO.NET Data Services, voorheen 'Project Astoria', toegevoegd aan Service Pack 1 van .NET Framework 3.5. ADO.NET Data Services is een REST-gebaseerd platform om data als een service beschikbaar te maken via het internet. Bovendien kan het als datalaag dienen achter Windows- en internetapplicaties.
REST staat voor Representational State Transfer. Met dit platform zet je queries om naar een adresseringsschema, in dit geval een URI. Deze URI kan de ontwikkelaar vervolgens gebruiken om de data op te halen. Om dit mogelijk te maken vormt het Entity Data Model (EDM) de basis van ADO.NET Data Services. Het Entity Data Model is een XML-tekstbestand die de databases, tabellen en relaties tussen de tabellen beschrijft. Met REST en het EDM hoeft de ontwikkelaar veel minder aan de datalaag te programmeren.

### Rich Internet Applications

Er komen steeds meer Rich Internet Applications beschikbaar. Bij deze applicaties laadt de website niet telkens opnieuw en krijgt de gebruiker het idee dat het om een gewone Windows-applicatie gaat. ADO.NET Data Services vormt hier een toevoeging op, omdat je de Data Services ook vanuit bijvoorbeeld AJAX en Silverlight kan aanroepen. Natuurlijk is ADO.NET Data Services ook te gebruiken in de traditionele webapplicaties.




AFBEELDING 1. KLASSIEK MODEL EN ADO.NET DATA SERVICES-MODEL

### Architectuur

Maar hoe zit ADO.NET Data Services nu precies in elkaar en wat is er allemaal nodig? ADO.NET Data Services maakt gebruik van een webserver en een datastore. In het klassieke model werd alle data tegelijk naar de client verzonden. Het ADO.NET Data Services-model springt efficiënter met de dataverwerking om door de data- en presentatielaag te scheiden.

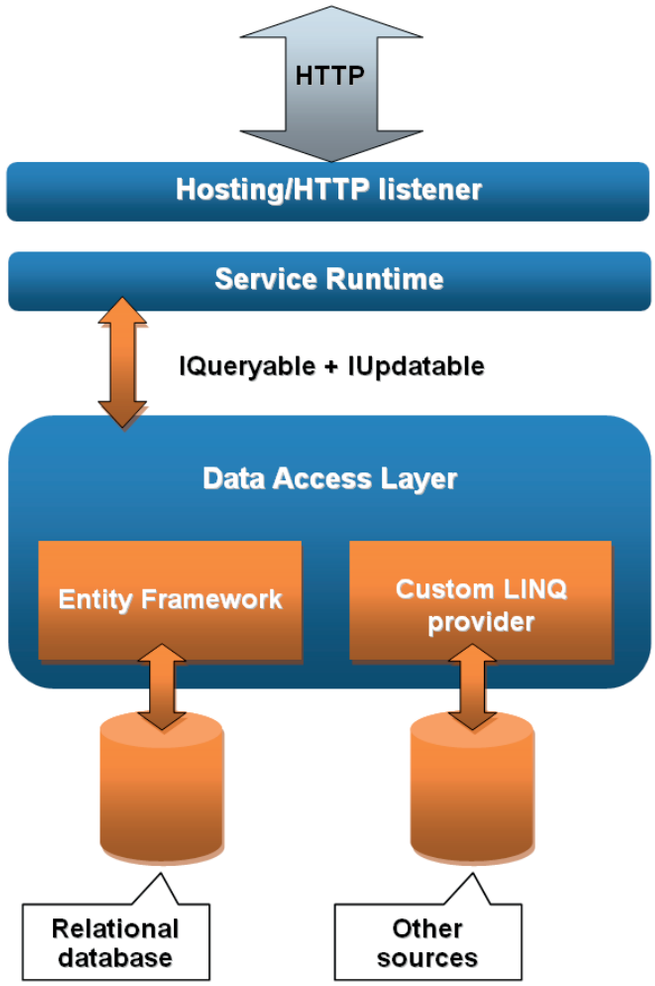
Afbeelding 2 geeft weer op welke wijze ADO.NET Data Services data verwerkt. In de Data Access Layer wordt door middel van het Entity Framework of met LINQ verbinding gemaakt met de database.

### Concurrency

Bij dataverwerking kennen we het klassieke probleem dat als er meer gebruikers dezelfde data gebruiken er wijzigingen verloren kunnen gaan. ADO.NET Data Services lost dit probleem op door ETags (Entity Tags) te gebruiken. Een ETag is een HTTP response header field die de huidige waarde van de entry bevat. Bij het ophalen van de data wordt een ETag aangemaakt. Op het moment dat de data vervolgens wijziging ondergaat, vindt er eerst controle plaats of de ETag hetzelfde is als de data in de database. Is dit niet het geval, dan heeft iemand anders de data tussentijds al gewijzigd en zal ADO.NET Data Services de data niet wijzigen. De ontwikkelaar kan een melding aan de gebruiker weergeven dat de data al is aangepast door iemand anders.
Om een ADO.NET Data Service te maken heb je Visual Studio 2008 SP1 (met .NET Framework 3.5 SP1) nodig en een database. De codevoorbeelden maken gebruik van de 'Northwind' database van SQL Server.

### Een Data Service maken

Om ADO.NET Data Services te kunnen gebruiken, moeten we natuurlijk eerst de Data Service aanmaken.




AFBEELDING 2. DATATRANSPORTMODEL

1. Open een nieuw C# 'ASP.NET Web Application'-project en noem deze "DataService".
2. Voeg nu via 'Add', 'New Item' een nieuw 'ADO.NET Entity Data Model' toe en noem deze 'Northwind.edmx'.
3. Kies nu de de bron. In dit artikel is dat de database 'Northwind'. Doorloop de wizard om het EDM te laten genereren.
4. Voeg nu ook de 'ADO.NET Data Service' toe en noem deze 'Northwind.svc'.

Deze stappen resulteren in het project uit afbeelding 3.

Nu moeten we de ADO.NET Data Service nog configureren.

1. Open het bestand 'Northwind.svc' en wijzig de regel met de code TODO: put your data source class name here in:

```
public class Northwind : DataService<NorthwindEntities>
```

2. Verwijder de commentaar-tags ('//') bij de regel met de config.SetEntitySetAccessRule om toegang tot de database toe te staan:

```
config.SetEntitySetAccessRule("*", EntitySetRights.All);
```

Om rechten toe te kennen aan service operations gebruik je de *config.SetServiceOperationAccessRule*.
De Data Service is nu gereed en zo ingesteld dat iedereen alle rechten heeft. Dit maakt het testen en ontwikkelen eenvoudiger, maar voordat je de applicatie publiceert moet je beslist zorgen voor een betere beveiliging. Als groot voordeel van ADO.NET Data Services geldt dat de beveiliging wordt bepaald in de service in plaats van in de applicatiecode. Alle instellingen voor de beveiliging staan dus op één centrale plaats. Dit vereenvoudigt het beheer van de Data Service aanzienlijk.

## Data Service gebruiken

De Data Service is nu beschikbaar en kan dienen als databron voor applicaties. Je test de Data Service door deze in een browser te openen met de desbetreffende URI. Wanneer bijvoorbeeld alle klanten worden opgehaald (http://localhost:2937/Northwind.svc/Customers), geeft de browser het in afbeelding 4 weergegeven resultaat terug.
Je gebruikt de Data Service in een applicatie door twee nieuwe projecten aan te maken (een Web Application genaamd Northwind-Website en Windows Forms Application, genaamd NorthwindApplication). En door een Service Reference toe te voegen naar het zojuist aangemaakte 'DataService'-project (zie afbeelding 5).
Voeg vervolgens in ieder bestand dat de Data Service gaat gebruiken een using statement toe naar de proxy-klasse en Microsoft.Data.Services.Client, zoals in codevoorbeeld 1.

```
using NorthwindWebsite.NorthwindService; //Proxy voor de Website
using NorthwindApplication.NorthwindService; //Proxy voor de Windows Forms
using System.Data.Services.Client;
```

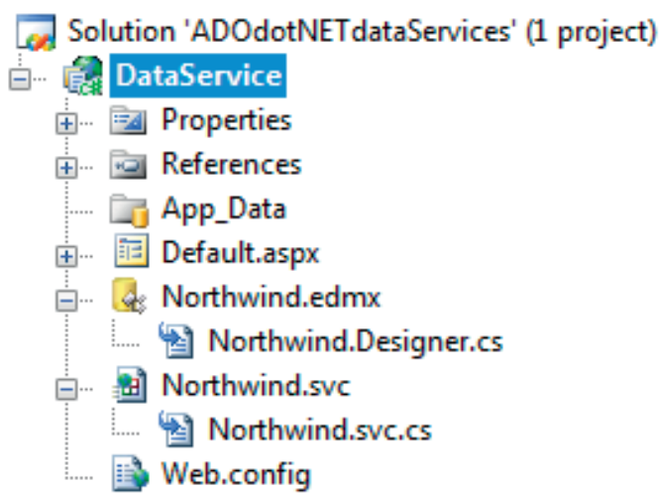
CODEVOORBEELD 1. DE JUISTE ASSEMBLIES TOEVOEGEN

Door middel van NorthwindEntities voer je operaties tegen de Data Service uit (codevoorbeeld 2). MergeOption.AppendOnly betekent alleen aanpassen van nieuwe entries. De bestaande entries ondergaan echter geen wijziging bij een SaveChanges().

```
string serviceUri = "http://localhost:2937/Northwind.svc";
NorthwindEntities ctx = new NorthwindEntities(new
Uri(serviceUri));
ctx.MergeOption = MergeOption.AppendOnly;
```

CODEVOORBEELD 2. DE DATA SERVICE DEFINIËREN

De proxy is gereed, nu moet de ontwikkelaar nog bepalen welke data er wordt opgehaald, bijvoorbeeld door de code in codevoorbeeld 3 toe te voegen. Voor het filteren van de data kun je geen




AFBEELDING 3. HET DATASERVICE-PROJECT

Dit maakt ADO.NET Data Services een goede databron voor onder andere Rich Internet Applications

gebruik maken van de gebruikelijke operatoren (>, <, =) omdat deze anders in de URI worden geformatteerd naar een HTML-code. Daarom zijn de operatoren omgezet in afkortingen: eq (=), ne (<>), gt (>), lt (<), and (&), or (|).

```
Uri dataServiceUri1 = new Uri(serviceUri + "/Customers?$orderby=
CompanyName"); //Sorteer op bedrijfsnaam
IEnumerable<Customers> customers1 = ctx.Execute<Customers>(data-
ServiceUri1);
Uri dataServiceUri2 = new Uri(serviceUri + "/
Customers?$filter=City eq 'London'"); //Alle klanten uit London
IEnumerable<Customers> customers2 = ctx.Execute<Customers>(data-
ServiceUri2);
```

**CODEVOORBEELD 3. DATA OPHALEN EN FILTEREN**

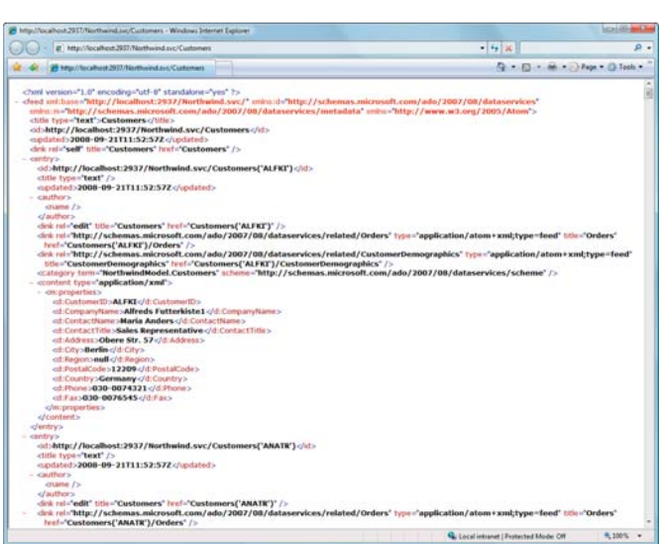
Ook kan je samen met de vergelijkingen een berekening in de URI toevoegen om de data te filteren. In codevoorbeeld 4 staan enkele voorbeelden. De te gebruiken afkortingen voor berekeningen zijn:
Sub: Aftrekken.
Add: Optellen.
Div: Delen door.
Mul: Vermenigvuldigen.
Mod: Modulo.

```
Uri dataServiceUri = new Uri(serviceUri + "/Orders?$filter=Freight
sub 4 eq 18"); //Gewicht-4=18
IEnumerable<Orders> orders = ctx.Execute<Orders>(dataServiceUri);
```

**CODEVOORBEELD 4. FILTEREN MET BEREKENINGEN**

## Gegevens toevoegen

Door een nieuw typesafe object 'Customers' te definiëren (zie codevoorbeeld 5), kan de Data Service de gegevens toevoegen. Ieder 'Customers'-object heeft dezelfde velden als in de database. Na het aanroepen van SaveChanges() wordt de bewerking daadwerkelijk doorgevoerd in de database.



**AFBEELDING 4. DE DATA SERVICE IN EEN WEBBROWSER**

```
Customers c = new Customers();
c.CustomerID = "KEVIN";
c.ContactName = "Kevin Timmerman";
c.ContactTitle = "Junior Consultant";
c.CompanyName = "Avanade";

ctx.AddToCustomers(c);
ctx.SaveChanges();
```

**CODEVOORBEELD 5. EEN ENTRY TOEVOEGEN**

## Gegevens wijzigen

Data wijzigen is mogelijk door eerst een Customers-object aan te maken die de huidige gegevens van de entry bevat. Bijvoorbeeld met een LINQ-query, zoals in codevoorbeeld 6.

```
Customers c1 = (from c in ctx.Customers
          where c.CustomerID == "KEVIN"
          select c).First();

c1.Phone = "+31(0)36-5475100";
c1.Fax = "+31(0)36-5475156";

ctx.UpdateObject(c1);
ctx.SaveChanges();
```

**CODEVOORBEELD 6. EEN ENTRY WIJZIGEN**

## Gegevens verwijderen

Om gegevens uit de database te verwijderen moet je eerst de entry ophalen, zodat deze in het geheugen staat (zie codevoorbeeld 6). Daarna verwijderen door in de plaats van een UpdateObject() gebruik te maken van DeleteObject().

## Service operations

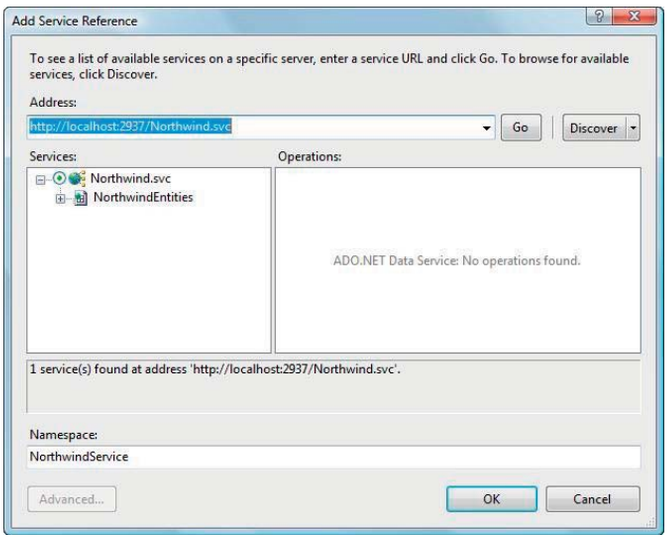
Zelf service operations aanmaken om data op te halen gaat ook met ADO.NET Data Services. Dit is toegevoegd omdat er veel vraag was om op de achtergrond al businesslogica door te voeren. Een in de code aangemaakte methode kan met diezelfde naam in de URI worden aangeroepen om de data op te vragen. Codevoorbeeld 7 geeft via de URI/*CustomersInLondon* alle klanten terug die in Londen wonen. Daarbij zorgt het *WebGet*-attribuut voor het beschikbaar stellen van de de methode als Service.

```
[WebGet]
public IQueryable<Customers> CustomersInLondon()
{
    return from c in this.CurrentDataSource.Customers
           where c.City == "London"
           select c;
}
```

**CODEVOORBEELD 7. EEN SERVICE OPERATION**

## Query interceptors

Nadat je gegevens hebt opgehaald, wil je deze vaak nog bewerken met bedrijfslogica. ADO.NET Data Services biedt deze mogelijkheid door middel van query interceptors. Bij het aanroepen van de URI wordt eerst de query interceptor uitgevoerd, waarin




**AFBEELDING 5. PROXY-KLASSE AANMAKEN VOOR DE DATA SERVICE**

we de query kunnen wijzigen en daarna pas de query. Codevoorbeeld 8 haalt uit de 'Orders'-tabel door middel van de URI */Orders* alleen de orders op die bij de klant met de naam 'Ana Trujillo' horen. Voeg hiervoor het using statement toe naar System.Linq.Expressions.

```
[QueryInterceptor("Orders")]
public Expression<Func<Orders, bool>> OnQueryOrders()
{
    return o => o.Customers.ContactName == "Ana Trujillo";
}
```

**CODEVOORBEELD 8. EEN QUERYINTERCEPTOR**

Naast de query interceptor bestaat ook de mogelijkheid de queries bij het wijzigen of verwijderen van data te onderscheppen met de ChangeInterceptor (codevoorbeeld 9). Acties als toevoegen, wijzigen en verwijderen kun je in een switch uitsplitsen om vervolgens de actie nog aan te passen of de gegevens te valideren.

```
ChangeInterceptor("Orders")]
public void OnChangeOrders(Orders o, UpdateOperations action)
{
    switch (action)
    {
        case UpdateOperations.Add:
            //Data toevoegen
            break;
        case UpdateOperations.Change:
            //Data wijzigen
            break;
        case UpdateOperations.Delete:
            //Data verwijderen
            break;
    }
}
```

**CODEVOORBEELD 9. EEN CHANGEINTERCEPTOR**

## De toekomst

Een belangrijke optie, waarvan Microsoft al vanaf het begin aangeeft deze te willen ontwikkelen, is offline-gebruik van ADO.NET Data Services. Gebruikers zijn namelijk niet altijd verbonden met het internet waardoor het offline gebruiken en verwerken van data in connected-applicaties niet gaat. Het idee is bij Microsoft ontstaan om een lokale kopie van de benodigde database aan te maken door middel van een lokale SQL Server Compact Database. Vervolgens zal het mogelijk zijn deze database met een lokale applicatie te gebruiken en zodra er weer verbinding is deze te synchroniseren met de online database.

## Conclusie

ADO.NET Data Services kunnen we zeer eenvoudig gebruiken in Windows- en webapplicaties. Dit maakt ADO.NET Data Services een goede databron voor onder andere Rich Internet Applications. In tegenstelling tot andere dataproviders zijn vaak maar enkele regels code nodig om de gewenste data door middel van een URI op te halen. Het Data Access Application Block van Enterprise Library heeft bijvoorbeeld verscheidene regels nodig om de connectie met de database te leggen en de query te bepalen. Omdat er steeds meer RIA's worden ontwikkeld, voldoet ADO.NET Data Services aan de vraag om de gebruikerservaring met de applicatie te verbeteren. Deze verbetert doordat we data sneller, eenvoudiger en efficiënter kunnen verwerken. De datalaag blijft gescheiden van de presentatielaag, en op de achtergrond kun je ook nog businesslogica doorvoeren.
Zoals je hebt gelezen, is starten met deze nieuwe techniek erg eenvoudig. De codevoorbeelden geven aan hoe je data met een paar regels code kunt toevoegen, wijzigen en verwijderen. Ook zijn de extra mogelijkheden beschreven, zoals 'Interceptors' om de Data Service zelf aan te passen.
Het Astoria Team publiceert regelmatig op zijn blog de meest recente status en mogelijkheden van ADO.NET Data Services. Zeker dus een blog om regelmatig te bezoeken als je ADO.NET Data Services wil gaan gebruiken.

> Vaak zijn maar enkele regels code nodig om de gewenste data door middel van een URI op te halen

**Links**

1 Project Astoria Team Blog
http://blogs.msdn.com/astoriateam/
2 Entity Data Model
http://msdn2.microsoft.com/en-us/library/aa697428(VS.80).aspx
3 QuickStudy: Representational State Transfer (REST)
http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=297424
4 Expose and Consume Data in a Web Services World
http://msdn.microsoft.com/nl-nl/magazine/cc748663(en-us).aspx

**Interesse?**

Op 20 januari 2009 organiseert Microsoft een Data Access Intrack.
Zie pagina 46 of http://www.msdnintracks.com/msdnintracks voor meer info.

**Kevin Timmerman** is solution developer bij Avanade (www.avanade.com), een samenwerkingsverband tussen Microsoft en Accenture. Voor vragen of opmerkingen is hij te bereiken op kevin.timmerman@avanade.com.