Dennis Doomen

Serviceoriëntatie? Dat is toch kinderspel?

HULP SOFTWARE FACTORY BIJ HET ONTWERPEN VAN SERVICE-INTERFACES

Hoe vaak hoor je niet van een klant dat zijn nieuwe systeem volgens de principes van serviceoriëntatie moet worden gebouwd? De auteur hoort het geregeld in ieder geval. Gelukkig helpt Microsoft's Patterns & Practices-team een handje met de nieuwste generatie van de Web Service Software Factory: Modeling Edition. In dit artikel belicht de auteur enkele facetten van deze nieuwe generatie designers.

Als je op een willekeurig moment je oor te luisteren legt bij een groepje architecten dat discussieert over het wel en wee van SOA, dan zal het niet lang duren voordat de term *contract-first* valt. Contract-first betekent in het algemeen dat je begint met het expliciet vastleggen welke specifieke diensten (*services*) jouw applicatieserver gaat aanbieden. Dit doe je dus al voordat je code aan het schrijven bent. Sommige architecten zijn van mening dat je dit moet doen door al het berichtenverkeer via XSDs te beschrijven. Anderen zeggen dat je dit beter kunt doen door de verschillende attributen die Windows Communication Foundation biedt te gebruiken.

Architecten zijn het niet snel met elkaar eens, maar dat het altijd veel werk is staat buiten kijf. De oplossing die de nieuwste generatie Web Service Software Factory (versie 3, ofwel de Modeling Edition) biedt, heeft iets van beide varianten, maar is een stuk eleganter. Patterns & Practices (P&P) heeft het goede, maar overigens niet geheel nieuwe idee gehad ons de mogelijkheid te geven de services te modelleren. Nu het met de introductie van de Domain Specific Language-technologie relatief gemakkelijk is geworden eigen modellen te ontwikkelen en daarmee met behulp van Visual Studio 2005-code te genereren, heeft P&P drie eigen *designers* gebouwd.

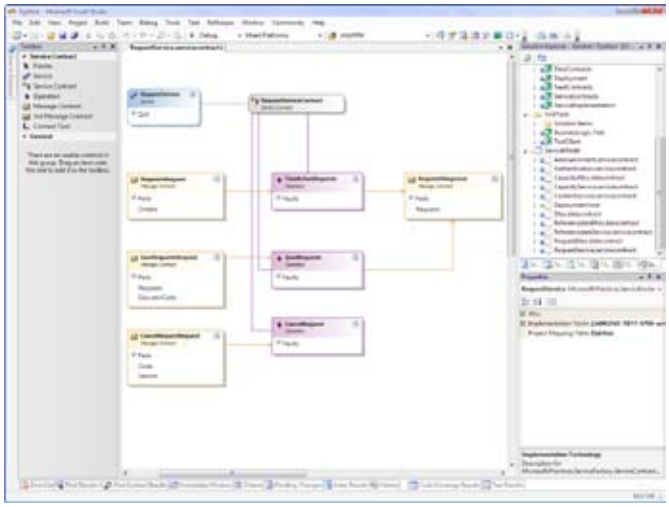
Waar begin ik dan?

Afbeelding 1 toont de Service Contract Designer in actie en laat onder andere de toolbox zien met daarin de elementen voor het modelleren van een service-operation en de bijbehorende inkomende en uitgaande message-contracts. Ook kun je één of meer operaties groeperen in een servicecontract, en zelfs aangeven welke service die contracten gaat implementeren. Hoewel de volgorde waarin je de code moet schrijven normaal gesproken jouw denkproces bepaalt, kun je nu beginnen met de operaties en van daaruit jouw service verder uitwerken. Buiten het message-contract kent de Service Factory ook nog data-contracts.

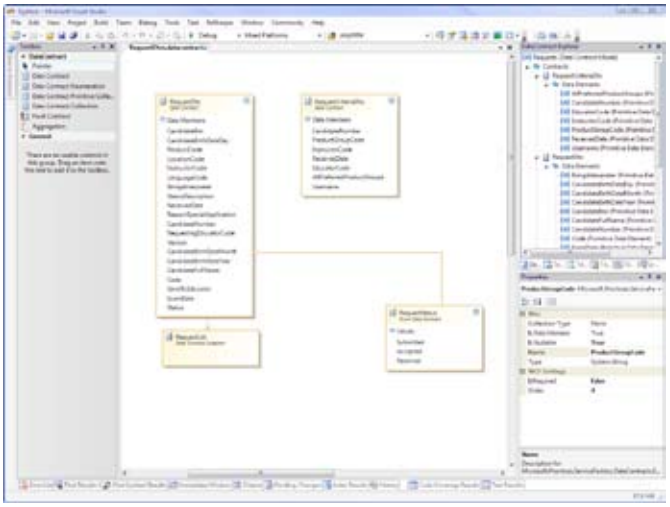
Afbeelding 2 toont de bijbehorende Data Contract Designer. Windows Communication Foundation definieert de rol van beide contracten niet zo heel scherp, maar binnen de Service Factory gebruik je de eerste als de envelop voor inkomende en uitgaande berichten en de tweede alleen als je herbruikbare databerichten kent. In de praktijk werkt dit voor de meeste gevallen prima, alleen is het jammer dat je soms door de beperkingen van een message-contract moet terugvallen op een datacontract. WCF bijvoorbeeld staat toe dat een operatie rechtstreeks een datacontract accepteert of retourneert. Maar binnen de Service Factory ben je altijd verplicht een message-contract als envelop te gebruiken. Omdat de message-contractdesigner (nog) geen ondersteuning biedt voor array's van primitieve types, word je dus soms gedwongen tot enigszins omslachtige oplossingen.

Technology extender

Ga je eenmaal aan de slag met de designers, door bijvoorbeeld de meegeleverde walkthrough uit te voeren, dan valt al snel op dat de



Afbeelding 1. De Service Contract Designer in actie

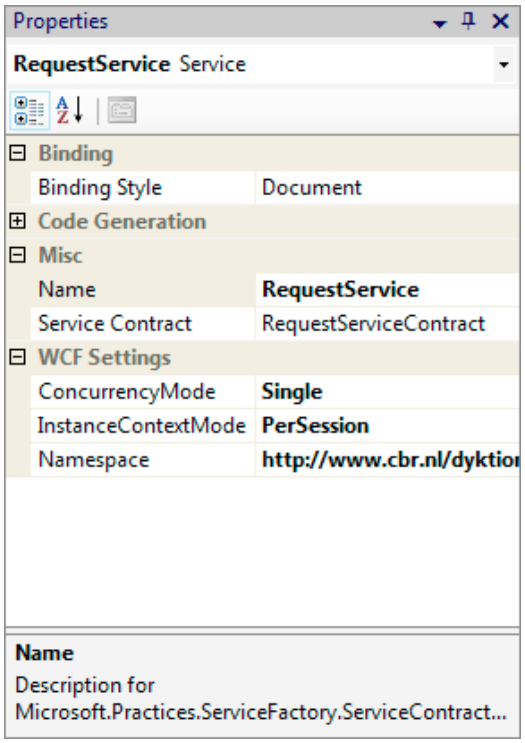


Afbeelding 2. De bijbehorende Data Contract Designer

mogelijkheden van de designers veel beperkter zijn dan datgene wat WCF biedt. De reden hiervan is dat de keuze voor de specifieke technologie van ondergeschikt belang is en zo laat mogelijk gemaakt wordt. Pas als je de servicecontracten in detail hebt uitgewerkt, wordt het interessant of je code genereert op basis van ASMX-webservicetechnologie of WCF. Deze keuze maak je door voor een model een zogenaamde *technology extender* te kiezen. Pas dan komen de attributen die bij die technologie horen beschikbaar. Denk bijvoorbeeld aan de WCF-attributen *ConcurrencyMode* en *InstanceMode* zoals te zien is op afbeelding 3. Het voordeel van deze aanpak is dat ook ontwikkelaars die gebonden zijn aan .NET 2.0 een duidelijk en voorspelbaar groeipad hebben. Het is voor veel bedrijven nog helemaal niet zo vanzelfsprekend dat ze .NET 3.0 meteen al bij de start van jouw project adopteren. Vooral IT-afdelingen willen nog wel eens terugschrikken, omdat ze denken dat de stap van .NET Framework 2.0 naar 3.0 net zo risicovol is als die van 1.x naar 2.0. Je kunt daarom het beste beginnen met standaard ASMX-webservices en later de overstap maken naar WCF, simpelweg door een andere *technology extender* te kiezen.

Green field-scenario

Eén van de veelgehoorde klachten over de recente producten van Patterns & Practices is dat ze allemaal uitgaan van een schone lei, ook wel een *green field scenario* genoemd. Daarom heeft P&P bewust nagedacht over het ondersteunen van al bestaande of in ontwikkeling zijnde systemen. In zo'n geval heeft een architect waarschijnlijk niet de mogelijkheid een en ander zomaar op de schop te gooien. P&P heeft dit, enigszins ironisch, *brown field scenarios* genoemd. Uit eigen ervaring heb ik kunnen ondervinden hoe weinig moeite het kost om een bestaande Visual Studio-oplossing uit te breiden met Service Factory-modellen. In feite is zo'n upgrade niet complexer dan het toevoegen van een nieuw Service Model-project en via het ProjectMapping.xml-bestand aan te geven welke gegenereerde code in welk bestaand project moet worden geplaatst. Heb je al een project waarin bestaande datacontracten zijn geplaatst, dan kun je de Service Factory zo configureren dat alle gegenereerde datacontracten in hetzelfde project terechtkomen. Als jouw projectstructuur al grotendeels overeenkomt met de standaard structuur, zoals te zien is in afbeelding 4, dan kun je zelfs met een simpele menuoptie het eerdergenoemde XML-bestand automatisch laten vullen met de juiste informatie.



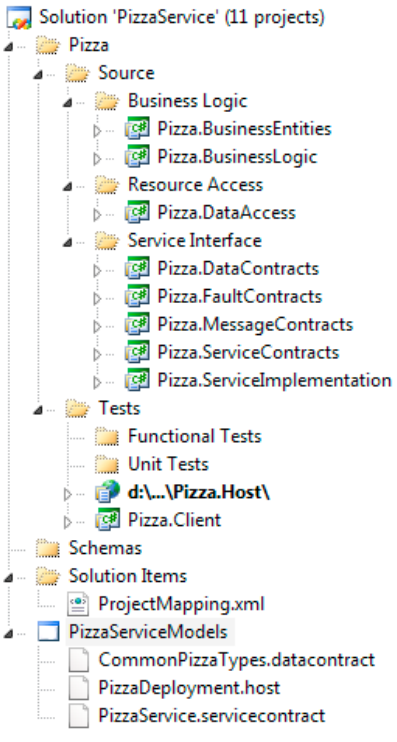
Afbeelding 3. De WCF-attributen ConcurrencyMode en InstanceMode

Maar een contract is toch niet alles?

Dat klopt! Met het modelleren van servicecontracten ben je er natuurlijk nog niet. Het configureren van de WCF-endpoints en -bindings is nog steeds een activiteit die nodig is om de service daadwerkelijk beschikbaar te stellen. De WCF Configuration Editor helpt je daarbij wel wat, maar het blijft foutgevoelig werk. En ook daar heeft P&P aan gedacht. De derde designer, ook wel *Host Designer* genoemd, is namelijk bedoeld om exact aan te geven hoe de eerder gemodelleerde servicecontracten moeten worden aangeboden, en via welk WCF-hostproject. Vanuit zo'n model genereert de factory de svc-bestanden en de bijbehorende configuratiesettings in het <serviceModel>-element van bijvoorbeeld de web.config. Afbeelding 5 toont een preview van de Host Designer. Deze designer was tijdens het schrijven van dit artikel nog volop in ontwikkeling, zodat het nog niet duidelijk was welke WCF-configuratie-instellingen konden worden gegenereerd. Ook de vraag hoe kan worden omgegaan met handmatige WCF Binding-instellingen kan daardoor nog niet worden beantwoord.

One size does not fit all

Rond de periode dat de eerste zogenaamde *alpha drops* werden vrijgegeven (maart 2007), heeft P&P een aantal architecten dat veel ervaring had met de inrichting van ontwikkelstraten uitgenodigd om als *expert advisor* op te treden. De delegatie Nederlanders die hierbij was betrokken, was opvallend groot. Door een Microsoft Groove-workspace werd ons de mogelijkheid geboden feedback te geven, nieuwe ideeën te introduceren, en te discussiëren over de functionaliteit van de Service Factory. Buiten de al eerder genoemde brown field scenario's, was de behoefte aan flexibiliteit en aanpasbaarheid essentieel. Veel architecten willen de factories aanpassen om op die manier de gegenereerde code beter aan te laten sluiten bij de klantomgeving. P&P heeft hier al vanaf het begin aandacht aan gegeven en ons gevraagd hoe wij dat zouden willen zien. Omdat ze de broncode van de DSL's, de Guidance Packages en het bijbehorende framework meeleveren, is het relatief gemakkelijk om een eigen versie van de factory te ontwikkelen. De meegeleverde documentatie legt in detail uit hoe je dit moet aanpakken en welke mogelijkheden er zijn. Zelfs het ontwikkelen van eigen *technology extender* is uitgebreid toegelicht.



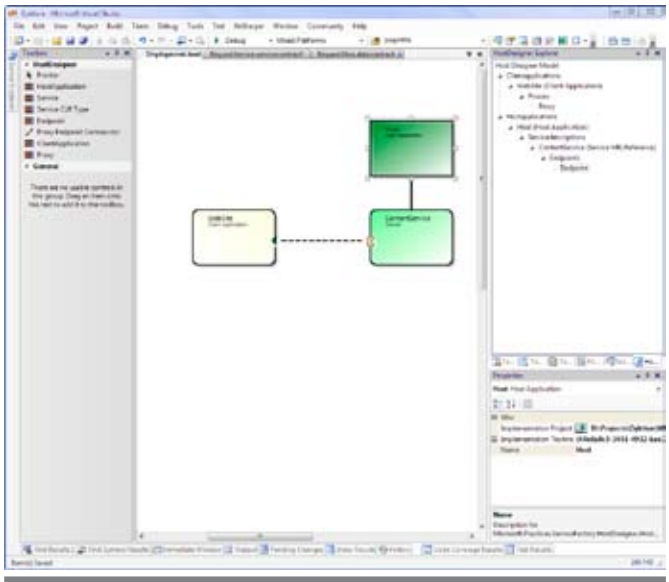
Afbeelding 4. Een projectstructuur die overeenkomt met de standaard structuur

Wat vind ik er zelf van?

In maart ben ik begonnen met het evalueren van de eerste versies. Al snel kreeg ik zo veel vertrouwen in de productiviteitsverbetering die de factory zou kunnen bieden, dat ik samen met de klant heb besloten deze in te voeren als onderdeel van een ASP.NET-ontwikkelstraat. Uiteraard zijn we in de periode tot het schrijven van dit artikel regelmatig tegen bugs en beperkingen aangelopen. En zoals je kunt verwachten van een product in ontwikkeling, zijn er nog belangrijke technische wijzigingen doorgevoerd. Maar met behulp van de vele MSN-chats met P&P hebben we dat elke keer weer kunnen repareren. Ook de CodePlex-projectpagina en de Groove-workspace boden ruim voldoende kans om feedback te geven en bugs te melden. Ik vind het opvallend dat wij op het uiteindelijke product een grote invloed hebben gehad. De belangrijkste problemen zijn inmiddels wel opgelost. Ook het concept van een korte walkthrough die bij de tweewekelijkse releases hoorde, bespaarde je veel tijd. In ongeveer een uur tijd werd je stap voor stap door alle functionaliteit van de factory geloodst, waarbij je soms meteen gevraagd werd om (via een e-mail) feedback te geven. Ook eventuele beperkingen of toekomstige wijzigingen werden meteen in de walkthrough vermeld. Op die manier kon ik in relatief korte tijd bepalen of een nieuwe release interessant voor ons was. Wat ik zelf graag nog zou zien, is ondersteuning voor het Validation Application Block (VAB) dat onderdeel is van de Enterprise Library 3.1. Je zou in het datacontractmodel moeten kunnen aangeven welke regels gelden voor de properties, zoals de reeks waar een getal aan zou moeten voldoen of de maximale lengte van een string. De Data en Service Contract Designers zouden die informatie kunnen gebruiken om de juiste VAB-attributen te genereren (bijvoorbeeld het [StringLengthValidation] attribuut). Ook mis ik de mogelijkheid relaties te leggen tussen datacontractelementen in verschillende modellen. Vaak kent een project een aantal herbruikbare datatypes dat je in meer modellen wilt gebruiken. Uiteraard kun je die wel vanuit verschillende servicecontracten gebruiken, maar niet vanuit andere datacontracten.

En toen?

Bij het inleveren van dit artikel begin oktober stond het versienummer van de meeste recente release nog op build 117 en daarom heb ik nog niet alle mogelijkheden van de Host Designer kunnen bekijken. Volgens de planning van rond die tijd zou P&P eind oktober de definitieve versie gaan opleveren. Tegen de tijd dat dit artikel voor je neus ligt, is die versie van de Web Service Software Factory: Modeling Edition natuurlijk allang vrijgegeven. Voor diegene die al bekend is met de vorige generatie (vrijgegeven in december 2006), zal het opvallen dat de scope van de derde generatie minder architectuurlagen omvat. P&P heeft gemerkt dat architecten vaak afwijken van de standaard invulling van de business- en datalagen zoals versie 2 die kende. Daarom hebben ze besloten om dat deel uit versie 3 te laten. Als fervent gebruiker van *object-relational mappers* zoals NHibernate en de nieuwe Enterprise Library, kan ik die beslissing alleen maar toejuichen. Een belangrijk punt voor de toekomst is ondersteuning voor Visual Studio 2008. In de oorspronkelijke planning was het de bedoeling dat P&P in juli al intern zou overschakelen op bèta 2. Maar als gevolg van de grote hoeveelheid feedback van de expert advisors en de CodePlex-projectpagina heeft Microsoft de energie gebruikt om de factory nog productiever te maken. Wel hebben ze uitgesproken later in dit jaar toch ondersteuning te gaan bieden.



Afbeelding 5. Een preview van de Host Designer

Erg interessant allemaal, maar wat nu?

Download om te beginnen de recentste versie van de CodePlex-projectpagina en ga aan de slag met de uitgebreide walkthrough. Binnen twee uur zie je hiermee alle facetten van de factory en leer je de vele mogelijkheden kennen. Ben je eenmaal zo ver dat je de factory in een bestaand project wilt gaan gebruiken, voeg dan een nieuw modelproject toe en begin met modelleren. Bestaande services kun je simpelweg importeren via de Import WSDL-functie. En heb je al maanden werk gestoken in het beschrijven van jouw services via XSD's, importeer deze dan via de Import XSD-functie. Mocht je nog twijfelen over het risico dat de factory je niet bevalt of toch niet goed past in je werkwijze, wees dan gerust. Je hebt nog altijd de goed onderhoudbare gegenereerde code. Kortom, ga aan de slag!

Dennis Doomen is softwarearchitect bij Aviva Solutions. Hij adviseert en begeleidt klanten bij de invoering van ontwikkelstraten op basis van het Microsoft-platform. Verder is hij gespecialiseerd in het toepassen van ontwerppatronen en is hij als adviseur betrokken bij de ontwikkeling van de Service Factory. Dennis is te bereiken via zijn e-mailadres dennis.doomen@avivasolutions.nl en via het Aviva Solutions Weblog op http://blog.avivasolutions.nl.

Referenties

Het Service Factory-project op CodePlex: http://www.codeplex.com/servicefactory

Guidance Automation Extensions en Toolkit op MSDN: http://msdn2.microsoft.com/en-us/teamsystem/aa718948.aspx

Domain-Specific Language Tools: http://msdn2.microsoft.com/en-us/vstudio/aa718368.aspx