Enterprise

Architecten en ontwikkelaars die .NET webservices bouwen waren al blij met de eerste versies van Web Service Software Factory (WSSF). Bij het maken van services werden ze opeens inhoudelijk geholpen op gebieden als kwaliteit, voorspelbaarheid en productiviteit. De laatste versie van WSSF gaat verder door de toevoeging van designers voor de diverse onderdelen van een moderne service.

Eerste blik op Web Service Software Factory

Modelleerversie

Pieter de Bruin
Avanade
(pieterd@avanade.com)

Een paar jaar geleden kwam de eerste versie van de Web Service Software Factory uit. WSSF was gemaakt door Microsoft Patterns & Practices, destijds een losstaande club die naam verwierf door het uitgeven van richtlijnen in de vorm van dikke boeken, grote pdf-bestanden en later Enterprise Library verbeterde en verspreidde. Patterns & Practices is inmiddels een onmisbaar orgaan geworden dat ontwikkelaars iedere dag de goede richting wijst, onder andere door middel van software factories voor web clients, smart clients en mobile clients. De software factories van Microsoft Patterns & Practices zijn verzamelingen van bewezen documentatie, tools en code en hebben grote voordelen. Ontwikkelaars komen steeds in een bekende omgeving, oplossingen worden gemaakt met best practices en goede patronen, en een ontwikkelaar wordt geholpen door gegenereerde code.

De webservice software factory is gebouwd met Windows Communication Foundation (WCF) in gedachten: het onderdeel van .NET dat de mogelijkheden en voordelen van communicatievormen als ASMX webservices, webservice enhancements, remoting, MSMQ en COM+ combineert en configureerbaar maakt. Om WCF zo configureerbaar te krijgen, worden methoden, parameters en implementatieonafhankelijk van elkaar gebouwd. Binnen de System.ServiceModel namespace leven de ServiceHost class en de attributen voor ServiceContract en DataContract. Deze onderdelen zie je in WSSF terug. Nieuw in de software factories van Patterns & Practices was het gebruik van Guidance Automation: een techniek die Visual Studio uitbreidt met mogelijkheden om informatie te verzamelen en code te genereren. Deze informatie werd verzameld door middel van wizards waar de gebruiker doorheen liep. Guidance Automation genereerde code eenmaal door informatie van wizards toe te passen in recipes (templates). Wilde je een tweede keer iets genereren met een kleine aanpassing op eerdere input, dan ging je weer de hele wizard door.

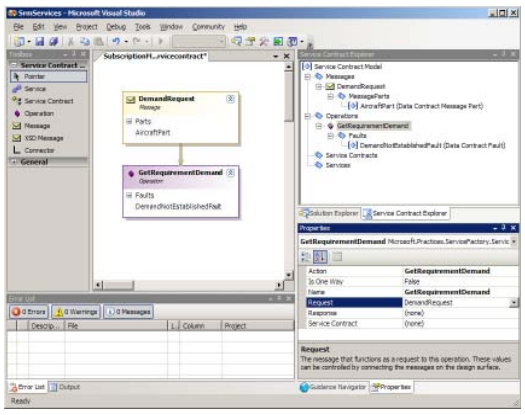
Modelling Edition

Eind 2007 lanceerde Patterns & Practices de Web Service Software Factory: Modelling Edition. De naam geeft al aan dat je kunt modelleren in de laatste versie van WSSF. Deze modellen hebben de wizards grotendeels vervangen en zijn gebouwd met de Domain Specific Languages (DSL) toolkit. Domeinspecifieke talen zijn elementen op een ontwerpblad waarmee allerlei zaken kunnen worden gemodelleerd. Zo kan je een domeinmodel hebben voor vragenlijsten, waarin je vragen, antwoorden, volgorde en conditionele vragen modelleert. Of je hebt een domeinmodel voor iets heel anders als huurauto's, verzekeringen of voor services; zie Software Release Magazine 7 voor meer informatie over horizontale en verticale DSL's. Nadat je een model hebt ontworpen, kies je de implementatietechnologie van de services en ten slotte genereer je het model naar code. De kern van de genereercomponenten is een webservices-domeinmodel. Naast de genereermogelijkheden van Visual Studio, bevat de service factory ook relevante geschreven richtlijnen voor het bouwen

van webservices onder andere op het gebied van architectuur, berichtontwerp, versionering, exception handling.

Het domeinmodel bestaat uit drie domain-specific languages (DSL's) die gebruikt worden om services te modelleren: Service Contract Model, Data Contract Model en het Host Model. Met het Service Contract Model definieer je de servicecontracten, operaties en berichten. Hiermee maak je ook code voor de webservice-interface en de service-implementatie. Het Data Contract Model stelt je in staat herbruikbare fault-contracten te maken en simpele en complexe data types. Het Host Model gebruik je om service endpoints en consumenten te definiëren. Je hebt ondersteuning voor Visual Studio .NET Web project en IIS Web-applicatie, maar je kunt hem ook uitbreiden met andere type bindings en WCF hosting-varianten. Ook zitten hier faciliteiten in om code en configuratie voor de hostapplicatie, service endpoints en client-proxies te genereren.

Services worden niet allemaal hetzelfde gebouwd, je begint bijna altijd met specificaties en assemblies die je al had. Zo kan je bijvoorbeeld al een XSD-file hebben die een bericht definieert of je weet welk gedrag de service moet ondersteunen. De service factory-modellen helpen je te hergebruiken wat je al hebt. Als je gebruikmaakt van het Service Contract Model, kan je services en bijbehorende onderdelen modelleren: service contracts, operaties en berichten. Je kunt ook meer services op een ontwerpblad plaatsen, maar ze kunnen alleen samenwerken als je de broncode van de service factory wijzigt en herbouwt. Zoals bij iedere designer in Visual Studio heb je een oppervlakte om op te ontwerpen en onderdelen in je toolbox om mee te ontwerpen.



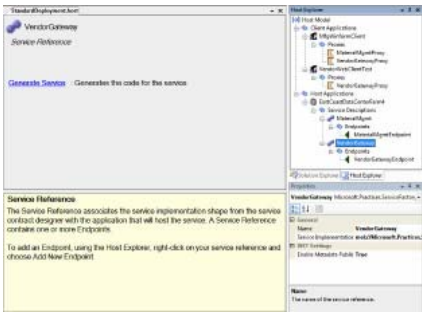
Afbeelding 1: Het servicecontractmodel met DSL-elementen

Een service vertegenwoordigt de implementatie van het contract. Dit element kan alleen verbinden met een enkel servicecontract. Dit element wordt gerefereerd door het Host Model. Het Service Contract is de service-interface die meer operaties kan bevatten, dus hij verbindt een service-element met meerdere operation-elementen. Een operation is een methode die kan worden aangeroepen op de service. Dit element verbindt een servicecontract met een of twee message-contracts. De Message is het request- of response-bericht dat datacontracten bevat. Dit element verbindt met een operatie. Je maakt het een request message door de connector te slepen van de message naar de operation. Een response message maak je precies andersom. Overigens kan je ook een XSD Message maken op basis van een bestaand XSD-bestand.
Met het Data Contract Model modelleer je data- en fault-contracten die herbruikt kunnen worden over meerdere services en operations. Bij dit model heb je de beschikking over een aantal elementen. Een Data Contract vertegenwoordigt een herbruikbaar en serialiseerbaar type. Hij kan primitive types, enumeraties, andere data contracts en collections bevatten. Dit element verbindt ook naar zijn members. Zoals je begrijpt is een Data Contract Collection dan een serialiseerbare collectie, waarvoor je in een eigenschap bepaalt welk data type de collectie bevat. Je kunt ook in de ontwerper vertellen hoe de collectie er uit moet zien in code door het CollectionType te kiezen. Hetzelfde geldt voor de Primitive Data Type Collection, alleen maakt die geen gebruik van een data-contract maar van primitive types. De Data Contract Enumeratie is een serialiseerbare enumeratie en het fault-contract is een serialiseerbare SOAP-fault.

Overigens bepaal je via de Implementation Technology van het Data Contract Model het type serializer. De WCF-implementatie maakt data contracts en de ASMX-implementatie maakt XML-serializable types. Met het Host Model maak je een model van de host-applicatie, service endpoints en client-proxies. Naast de Host Designer en de tool-window gebruik je de Host Explorer-window voor het toevoegen, verwijderen en selecteren van modelelementen. De inhoud van het hoofd- en eigenschappenscherm wordt bepaald door het element dat geselecteerd is in de Host Explorer. In het bovenste deel van het hoofdscherm kan je code genereren als het model gevalideerd is. Het onderste deel beschrijft het geselecteerde element en hoe je andere elementen kunt toevoegen. Je begint met de Host Explorer door een host-applicatie en een client-applicatie toe te voegen.

De Host Application zal request-messages ontvangen en response-messages retourneren. De host heeft een of meer servicereferenties die je toevoegt door te rechtsklikken op de host

Met het Data Contract Model modelleer je data- en fault-contracten die herbruikt kunnen worden over meerdere services en operations



Afbeelding 2: Het host-model

Application in de Host Explorer. Vervolgens verbindt de servicereferentie het service-implementatie-element van de servicecontract-designer met de applicatie die de service zal hosten. Een service-referentie heeft een of meer endpoints. Ook een endpoint voeg je toe in de host-explorer door te rechtsklikken op de service-referentie. Een endpoint beschrijft op een standaard manier waar messages heen moeten worden gestuurd, hoe ze moeten worden verstuurd en hoe ze er uit moeten zien. De Client Application is het consumerende deel van de service. Hij bevat een of meer proxies om met de service te kunnen communiceren. Proxies voeg je ook toe in de Host Explorer. De andere modellen gebruiken de projectmapping-tables, maar het Host Model niet. In plaats daarvan linken de host en client-applicatie direct naar de projecten waarnaar ze gegenereerd worden. Wel hetzelfde met de andere modellen is dat deze verwijzingen pas gemaakt hoeven worden nadat de implementatieprojecten zijn aangemaakt. Validatieregels van het Host Model zijn minder uitgebreid dan die van het Service Contract Model en Data Contract Model. Mogelijk wordt dit door de Service Factory community uitgebreid. Als je er voor zorgt dat externe afhankelijkheden geldig zijn, werken de belangrijkste validaties. Een van de eisen voor deze modellen is dat je webservices gaat ontwerpen op basis van logische modellen die onafhankelijk zijn van technologie of taal. Door deze eis is er een expliciete stap om de implementatie te maken.

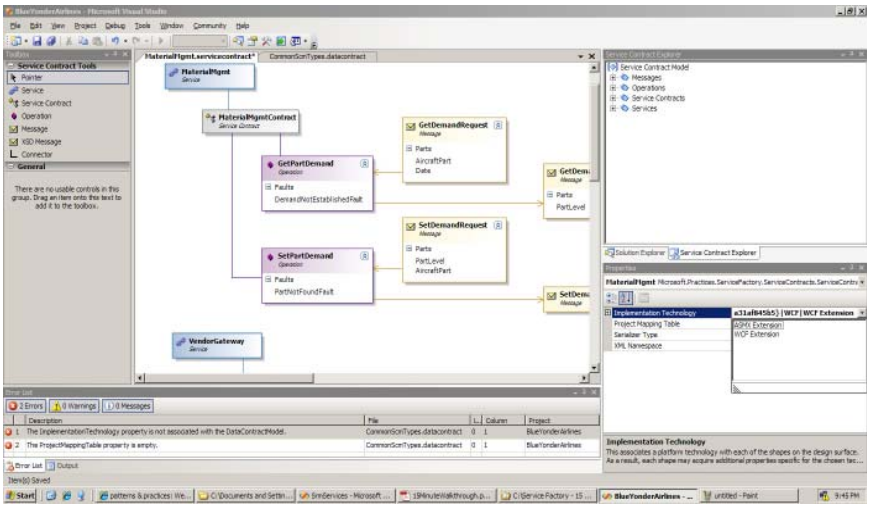
Een stap dieper in de technologie

De service factory ondersteunt ASMX en WCF als uitbreidingen die technologiespecifieke eigenschappen en validatieregels aan elementen van modellen toevoegen. Je kiest de technologie van een model nadat je een aantal ontwerpstappen hebt genomen. Dit is een verplichte handeling voordat je code kunt genereren. Nadat je de technologie hebt gekozen, kan je nog wijzigingen maken in de modellen. Wel is het zo dat als je een technologie gekozen hebt voor een model, het model altijd met een technologie geassocieerd zal zijn. In de service factory kan je extra extenders maken van technologische implementaties door gebruik te maken van de extensibility points in de source-code. Nadat je een technologie hebt gekozen, moet je mogelijk in het Service Contract Model extra waarden ingeven voordat het model valideert. Bijvoorbeeld de IsWrapped-eigenschap van berichten moet True zijn wanneer ze meer dan één message-part bevatten. In het Data Contract Model moet je verscheidene eigenschappen instellen voordat het weer valideert.
De Primitive Data Type Collection en Data Contract Collection hebben een Collection Type-eigenschap waarin het generic collectie-type staat dat gebruikt wordt bij codegeneratie. De data- en fault-contracten hebben een Order-eigenschap. Wanneer je WCF als implementatietechnologie gebruikt, moeten deze waarden in een volgorde worden ingevuld.

Het Host Model is ook afwijkend doordat de technologiekeuze ligt op het niveau van Host en Client en niet in het model zelf. Nadat de keuze voor WCF is gemaakt, worden de volgende eigenschappen toegankelijk: Enable Metadata Publishing is van toepassing op een service reference naar een host-applicatie. Zet je deze eigenschap op True, dan zal het endpoint onzichtbaar zijn op het model, maar getoond worden door de WSDL en policy metadata. Binding Type is een eigenschap van een endpoint in een service reference. Deze waarde komt overeen met de standaard binding types in WCF.

Van model naar echte code

De eerste handeling tijdens het genereren van de implementatie is het aanmaken van de solution en projecten. De standaard structuur is slechts een beginpunt en het is logisch om deze te gaan aanpassen aan gebruikelijke indelingen binnen een organisatie. Projectmapping is een techniek in de Service Factory die zorgt voor ontkoppeling van modellen en de gegenereerde projecten. Je kunt het ProjectMapping.xml-



Afbeelding 3: Implementation technology kiezen voor het service contract

bestand op diverse manieren aanmaken en wijzigen. Wanneer je de template "Add ASMX/WCF Implementation Projects" gebruikt, worden de projectmappings voor je gemaakt als de solution-structuur zichtbaar wordt. Als je liever ieder project individueel op een bepaalde manier genereert, kan je de template "Populate Project Mapping Table" gebruiken. Hiermee komt achter ieder project de naam van het types-element uit het model: "Host", "Client", "BusinessLogic", et cetera. Natuurlijk kan je ook eerst projecten zelf aanmaken en er vervolgens code in laten genereren door de service factory, maar dan moet je handmatige wijzigingen in het ProjectMapping.xml-bestand maken met je favoriete XML-editor. Deze versie van de service factory kan C#-code en XML voor configuratiebestanden maken. Tijdens de ontwikkeling van de Service Factory heeft Microsoft de architectuur getest, zodat later ook andere .NET-talen gemaakt kunnen worden op basis van teksttemplates voor data contracts. Deze teksttemplate zit in de code van Service Factory als voorbeeld voor als je andere Visual Basic .NET teksttemplates wilt gaan maken.

De volgende versie

De service factory is een serieus *versie drie* product van Microsoft Patterns & Practices. WSSF is ontworpen om uitbreidbaar te zijn, zodat het aangepast kan worden aan de behoeften van een ontwikkelteam en -organisatie. Wanneer je de service factory gebruikt, worden implementatietechnologiegerelateerde beslissingen zo lang mogelijk uitgesteld. Deze beslissingen bevatten onder andere het platform (WCF en ASMX) dat gebruikt wordt en welke Visual Studio-projecten de solution voor de service vormen. Wijzigingen aan de service en het ontwerp hebben op deze manier weinig rework nodig.

Het Service Factory-team heeft er aanzienlijke aandacht aan besteed om de Service Factory eenvoudig leerbaar te maken. De Web Service Software Factory: Modeling Edition bevat geschreven documentatie en hands-on oefeningen. Ook zijn er korte videos op de Service Factory Community Site. De WSSF is erg interessant om te gebruiken in jouw project of alleen al om te zien waar Microsoft mee bezig is. Als je WSSF gaat gebruiken, laat dan ook Microsoft weten wat je er van vindt. Wat vind je goed en wat kan er beter? Op HYPERLINK "http://www.codeplex.com/servicefactory" www.codeplex.com/servicefactory kan je alle informatie vinden en feedback leveren.

Op korte termijn verwachten we de officiële versie van WSSF voor Visual Studio 2008, maar daar houdt het modelleren niet op. In de volgende versie van Visual Studio Team System, codenaam Rosario, kan je verregaande integratie verwachten van designers in jouw favoriete ontwikkelomgeving.

Migratieproblemen en oplossingen

Er zijn nogal wat oplossingen voor het migreren van een legacy-applicatie naar Java of .Net. Bluephoenix (PBHX) bijvoorbeeld biedt maar liefst elf migratieoplossingen naar Java aan, onder meer voor PowerBuilder, Natural, COBOL en Cool:GEN. Ondanks de mooie naam weet Bluephoenix zich echter geen raad met Gupta-applicaties. Metex, ook een bedrijf dat oude applicaties uit de as laat herrijzen, wel. Hun oplossing transformeert Powerbuilder, Oracle Forms, Centura, VB en Forte applicaties naar Java óf .Net. In SRM 2005/4 hebben we software besproken om COBOL naar .Net te migreren. Sybase zelf biedt een migratietool van Powerbuilder naar .Net. De huidige eigenaar van Gupta, Unify, heeft Active Data Corp overgenomen.

Dit bedrijf heeft een tool dat nu door het leven gaat onder de naam Composer for Team Developer, voorheen heette het Sabertooth. Bij de IceTeaGroup heeft men overigens ook gekeken naar mogelijkheden om SAL-code naar Java te porten, maar dat bleek door de idiosyncratieën van Java (aldus XLSGlobal) veel moeilijkheden op te leveren.

Meer informatie over het Porting Project Framework: www.xlsglobal.com