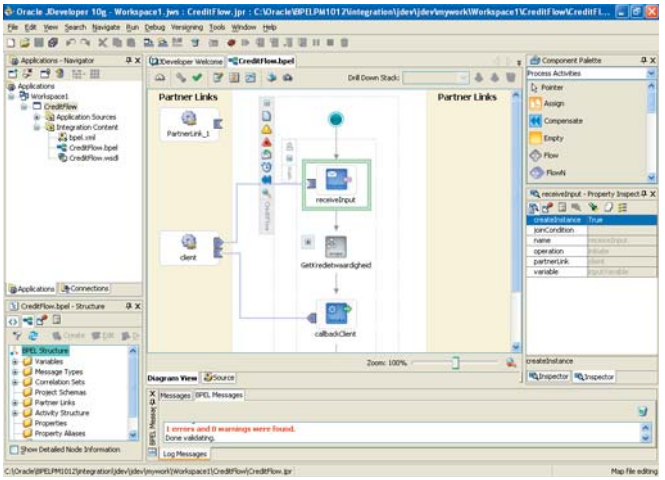
BPEL Process Manager 10.1

ROI op traditionele applicaties

De BPEL Process Manager kan goed toegepast worden om Designer- en Developer-applicaties om te vormen naar een service georiënteerde architectuur. Reden genoeg hier eens uitgebreid bij stil te staan. In een tweetal artikelen legt Harold Gerritsen de achtergronden uit van BPEL, geeft hij aan met welke eenvoudige stappen Java- en traditioneel ontwikkelde applicaties geïntegreerd kunnen worden met behulp van BPEL en geeft hij een tutorial van BPEL en de BPEL Process Manager.

Het is alweer ruim een jaar geleden dat Oracle het kleine Amerikaanse softwarebedrijf Collaxa en hun core product, BPEL4WS Orchestration Server, inlijfde. Met die overname werd Oracle in één klap een serieuze speler op het vlak van procesintegratie op basis van webservices. In Nederland heeft op dit moment het merendeel van de Oracle-klanten nog steeds applicaties die voornamelijk zijn gebouwd met de traditionele Oracle-tooling: Oracle Designer en Oracle Developer. Daarom staat de Orchestration Server, of zoals Oracle het product heeft gedoopt, de BPEL Process Manager, in de praktijk nog tamelijk ver af van de belevingswereld van de gemiddelde Oracle-ontwikkelaar. Niettemin kan de Process Manager ook goed toegepast worden om juist deze bestaande applicaties om te vormen naar een service georiënteerde architectuur, of SOA, om het buzzword te gebruiken.



Afbeelding 1. BPEL Designer voor JDeveloper.

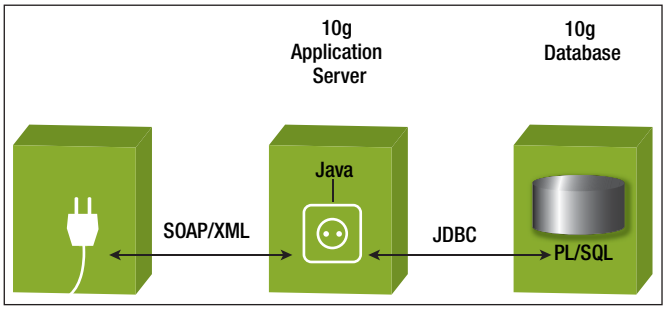
Historie

Voor we dieper ingaan op de functionaliteit van BPEL en de Process Manager duiken we eerst kort in de historie van BPEL en de overname van Collaxa. Begin 2002 vatten in Amerika Edwin Khodabakchian en twee andere briljante software-ontwikkelaars, samenwerkend onder de naam Collaxa, het plan op om een tool te ontwikkelen om op webservices gebaseerde bedrijfsprocessen te modelleren èn een engine te bouwen die deze processen kan uitvoeren. In dezelfde periode is er een standaard bezig te ontstaan: Business Process Execution Language for Web Services, kortweg BPEL4WS of, nog korter: BPEL.

Collaxa kiest er daarom voor om de tool de gemodelleerde processen intern op te laten slaan in het formaat van die BPEL-standaard. Omdat ze hiervoor kiezen zijn er al gauw grote bedrijven geïnteresseerd in de alfa versies van hun product. Zo zijn bedrijven als Boeing bereid om betaalde Proof Of Concepts te laten uitvoeren. Ondertussen wordt - we schrijven augustus 2002 - de eerste versie van de BPEL specificatie bevroren. Op dat moment is Oracle overigens nog niet betrokken in dat traject. Maar al snel zien veel grote softwarebedrijven (waaronder Oracle) het belang in van de zich snel ontwikkelende standaard en nemen zij actief deel aan het specificatietraject. In maart 2003 resulteert dat in versie 1.1 van BPEL. Een maand later wordt de specificatie ter standaardisering ingediend bij OASIS[1]. Precies op dat moment brengt Collaxa 's werelds eerste implementatie van een BPEL4WS-gebaseerde server uit onder de naam Collaxa BPEL4WS Orchestration Server 2.0.

Een jaar later heeft Collaxa nog immer het technologisch meest geavanceerde BPEL-tool, maar het besef groeit bij hen dat het een kwestie van tijd zal zijn voor de grote spelers met hun bijbehorende marketing-machines hen zullen wegvagen. Al

Advertentie



Afbeelding 2. Webservice voor het ontsluiten van PL/SQL- of Java-procedure.

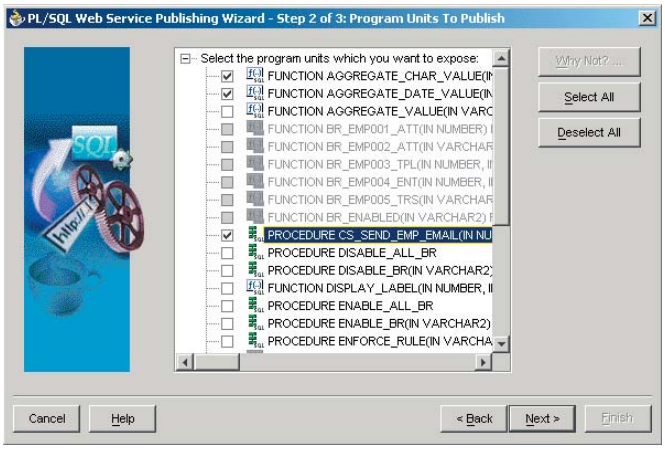
gauw voert Collaxa oriënterende gesprekken met Oracle, Microsoft en IBM. De Orchestration Server blijkt het meest complementair te zijn aan het Oracle-productportfolio en de toekomstplannen van Oracle. Eind juni 2004 maakt Oracle op JavaOne bekend dat het Collaxa heeft overgenomen en de Orchestration Server opgenomen zal worden in de Oracle internet Application Server. Inmiddels maakt het product onder de naam BPEL Process Manager 10.1 volwaardig onderdeel uit van de iAS10g.

Webservices en standaarden

Om te begrijpen wat het concept is van BPEL en afgeleide BPEL-tools en -servers is het goed nog even stil te staan bij de basis: webservices. Zoals ik inmiddels alweer ruim twee jaar geleden beschreef in het juni nummer van Optimize onder de titel 'Traditioneel of J2EE, de keuze in 2003'[2], is het kinderlijk eenvoudig om functionaliteit van een applicatie te ontsluiten en aan te bieden in de vorm van een webservice. De webservice

Met die overname van Collaxa werd Oracle een serieuze speler op het vlak van procesintegratie op basis van webservices

vormt als het ware een gestandaardiseerde interface voor de aangeboden functionaliteit, vergelijkbaar met een stopcontact (zie afbeelding 2). De taal waarin de functionaliteit achter het stopcontact geïmplementeerd is kan van alles zijn: C# of andere .NET-varianten, Java, maar bijvoorbeeld ook PL/SQL. Het stopcontact zelf wordt 'gebruikt' door het een XML-document aan te bieden met daarin de naam van de aan te roepen procedure en de naam en waarden van de parameters. De webservice ontleedt vervolgens het XML-document, roept de gespecificeerde procedure aan met de meegegeven parameters en antwoordt vervolgens door het teruggeven van een XML-document met daarin de returnwaarde(n) van de aangeroepen procedure. De structuur van de XML-documenten is geformaliseerd op basis van de SOAP-standaard: Simple Object Access Protocol. Het XML-document wordt daarmee vaak de 'SOAP envelop' genoemd van de Remote Procedure Call (RPC). Vrijwel alle ontwikkeltools bevatten tegenwoordig wizards waarmee procedures en functions (of equivalenten in een andere programmeertaal) ontsloten kunnen worden in de vorm van een webservice. Met deze wizards kan door het selecteren van de betreffende program unit en het specificeren van enkele aanvullende gegevens de programmacode voor het genoemde stopcontact gegenereerd worden. In afbeelding 3 is te zien hoe met behulp van een wizard in JDeveloper een webservice voor een PL/SQL-function kan worden gegenereerd. De gegenereerde code kan vervolgens gedeployed worden in een Application Server. Hiermee hebben we met de opkomst van de webservice-standaard - inmiddels alweer enkele jaren geleden - een manier gevonden om de vroegere concepten van Component Based Development en Distributed Computing inhoud te geven. Om echter effectief en efficiënt systemen te ontwikkelen gebaseerd op hergebruik van 'ergens' kant en klaar aangeboden services is meer nodig. Allereerst moet je weten wààr de services zijn aan te roepen (op welke server ze zijn gedeployed) en hòe ze zijn aan te roepen (met welke naam en parameters). Hiervoor zijn achtereenvolgens de UDDI- en WSDL-standaarden in het leven geroepen. UDDI[3], hetgeen staat voor Universal Description, Discovery, and Integration, is evenals de BPEL-standaard ondergebracht bij OASIS. Het beschrijft een protocol waarmee als het ware een registry is te implementeren voor SOA, zeg maar een gouden gids voor webservices. Door informatie over aangeboden webservices te registreren in een UDDI-server is deze info voor UDDI-clients beschikbaar. Alle .NET- en J2EE-ontwikkeltools (zoals JDeveloper) hebben een UDDI-client in zich. Hierdoor zijn er wizard beschikbaar waarmee ontwikkelaars kunnen browsen door de catalogus van elders beschikbare webservices. Het enige dat ze hiervoor hoeven te doen is de URL van de UDDI-server toe te voegen aan de lijst van UDDI-servers van het ontwikkeltool. De UDDI-info is opgebouwd uit vier niveaus[4], te beginnen met de algemene business-informatie over de bedrijven die webservices aanbieden. Deze informatie betreft eigenschappen zoals de naam, industrie of productcategorie en geografische ligging. De tweede laag betreft de feitelijke business services die de bedrijven verlenen. De twee onderste niveaus slaan de brug naar de techniek. Op basis van een geselecteerde aanbieder van webservices kan hier de bijbehorende informatie gevonden worden om de specificatie van de aanroep naar deze webservices op te halen. Deze informatie is vastgelegd in de vorm van een WSDL-beschrijving van de service. WSDL, hetgeen staat voor Web Service Description Language, beschrijft de webservice in een formaat waarmee ontwikkeltools uit de



Afbeelding 3. Wizard voor publiceren van PL/SQL procedure als webservice.

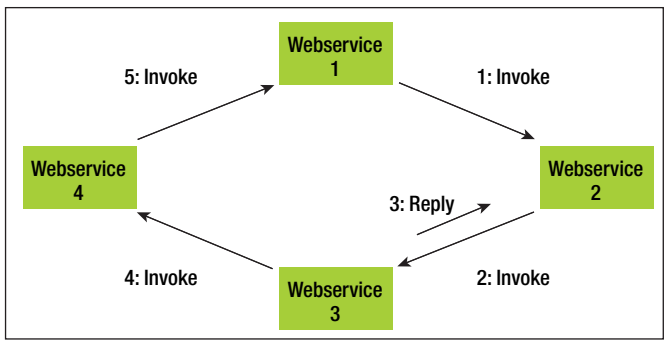
voeten kunnen. Zo kan een ontwikkeltool op basis van een WSDL-beschrijving bijvoorbeeld een zogenaamde clientstub genereren; zeg maar de stekker in de eerder genoemde metafoor van het stopcontact. Runtime kunnen lokale procedures dan de lokale clientstub - waarvoor de programmacode is gegenereerd in dezelfde programmeertaal - aanroepen. De stekker en het stopcontact communiceren zoals toegelicht op basis van uitwisseling van SOAP-documenten in XML. WSIL tot slot, staat voor Web Service Inspection Language en is ontstaan zo'n half jaar na de UDDI-standaard en is daar eigenlijk complementair aan. Evenals UDDI is het bedoeld om het 'ontdekken' van webservices te faciliteren. Het wijkt echter af in de opzet: UDDI is een soort directory-service, WSIL wordt gebruikt als standaard om een XML-document te maken met daarin een opsomming van verwijzingen naar webservices. UDDI-directory's zijn meer geschikt als openbare, gemeenschappelijke directory's (vergelijkbaar met een discussion forum voor het delen van opinies), WSIL-documenten zijn vooral bedoeld binnen een bepaalde gesloten gemeenschap (zoals een bedrijf) waarbij iemand verantwoordelijk is voor het beheren van het WSIL-document.

Integratiescenario's

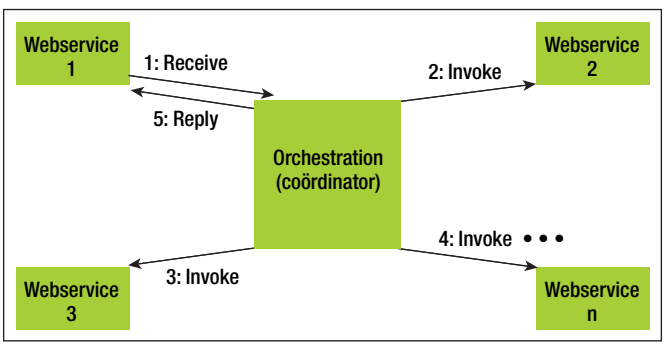
Met het ontstaan van de webintegratiestandaarden (SOAP, UDDI, WSDL, WSIL etc.) werd het mogelijk met weinig inspanning applicaties service-gerichter te laten werken. Moderne systeemontwikkeling leidde niet meer per definitie tot monolieten (applicaties uit één stuk). Applicaties konden voortaan elementaire functionaliteiten aanbieden aan hetzij de front-end laag van de 'eigen applicatie', hetzij aan de buitenwereld, dus aan andere applicaties dan die waarvoor de functionaliteit primair werd ontwikkeld. In de traditionele Oracle-wereld werd deze trend zichtbaar met de komst van het CDM RuleFrame. De standard en custom services in de CAPI's hoefden in feite alleen nog maar ontsloten te worden als webservice om het SOA concept invulling te geven.

De toepassing van webintegratie was aanvankelijk redelijk beperkt. De meest toegepaste vorm van integratie was de vorm die in de literatuur wordt aangeduid met choreografie (zie afbeelding 4). Dit is een samenwerking van een aantal services die kennis van elkaar hebben en weten in welke volgorde ze elkaar onderling aanroepen en welke informatie ze wanneer dienen uit te wisselen. Eigenlijk werd de gewenste functionaliteit verdeeld over verschillende services die elkaar vervolgens onderling aanriepen. Het nadeel moge duidelijk zijn. Bij de geringste wijziging heeft dit meteen impact op potentieel een aanzienlijke hoeveelheid services. Dat organisch gegroeide integratie toch vaak gebaseerd is op de samenwerkingsvorm 'choreografie' is wel te verklaren. Ontwikkelaars werkten nog vanuit de vroegere aanpak van modulair ontwikkelen. De integratiemogelijkheid op basis van SOAP werd alleen gezien als middel om op eenvoudige wijze software welke op andere machines draaide aan te roepen. De eerder genoemde stekkers en stopcontacten vormden daarbij het sluitstuk van de implementatie. De nadruk bleef liggen op de veelal 3GL-implementatie van de services. De XML-communicatie in de integratie was voor de ontwikkelaars veelal een *black box.* Dit werd nog eens versterkt door de gebruikersvriendelijke hulpmiddelen van de ontwikkeltools om de 'stubs en hubs' (de stekkers en stopcontacten) te genereren.

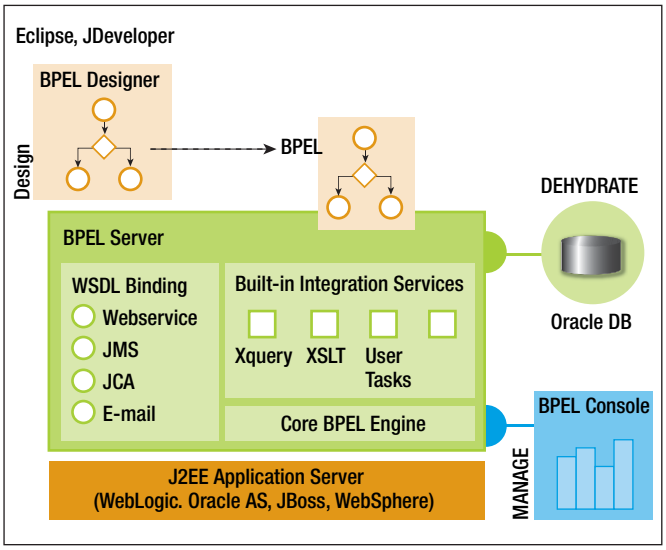
Toen webservices gemeengoed waren geworden werd er al snel een kantelpunt bereikt. De nadruk kwam niet te liggen op ontwikkelen van nieuwe services, maar slim combineren van bestaande. Door die trend maakte samenwerking van webservi-



Afbeelding 4. Integratiescenario op basis van choreografie.



Afbeelding 5. Integratiescenario op basis van orchestration.



Afbeelding 6. BPEL Process Manager architectuur.

ces op basis van choreografie als snel plaats voor samenwerking op basis van orkestratie (zie afbeelding 5).

Bij orkestratie hebben webservices nadrukkelijk géén kennis over andere webservices. In plaats daarvan is die kennis bijeengebracht in één coördinator (ook wel: orchestrator). Deze coördinator is in feite de implementatie van het bedrijfsproces dat geïmplementeerd dient te worden. Hij roept op de juiste wijze in de juiste volgorde alle noodzakelijke webservices aan. Wijzigingen hebben op deze manier veel minder invloed dan bij choreografie: vaak blijven wijzigingen beperkt tot één webservice en de orchestrator. De coördinator kan uiteraard zelf ook weer als webservice worden uitgevoerd. Door het aanroepen van deze webservice wordt dan de complete orchestration gestart.
Naarmate er meer en meer functionaliteit beschikbaar is waar de coördinator gebruik van kan maken, neemt de noodzaak voor functionaliteit ín de coördinator meer en meer af. Althans, er kunnen best complexe activiteiten in de coördinator nodig zijn, maar dan is dat uitsluitend ondersteunend en aanvullend op het aanroepen van andere webservices. De coördinator moet onder andere goed zijn in de volgende kerntaken:

- het aanroepen van webservices;
- het wachten op antwoord en opvangen van retourinformatie van webservices;
- het manipuleren van variabelen (waarden van parameters) voor bijvoorbeeld het omzetten van het ene coderingsstelsel naar het andere, vgl. 'I' en '0' versus 'J' en 'N';
- het kunnen definiëren en afhandelen van foutsituaties.

Daarnaast dient de coördinator bij voorkeur ook scenario's aan te kunnen waarbij complexere logica kan worden gedefinieerd rondom het aanroepen van webservices. Hierbij valt te denken aan constructies die in de meeste 3GL's voorkomen. Voorbeelden zijn de CASE-constructie, de WHILE-constructie voor loops en de FLOW-constructie. Laatstgenoemde is iets minder gemeengoed en wordt gebruikt om vast te leggen dat een aantal acties parallel uitgevoerd dient te worden.

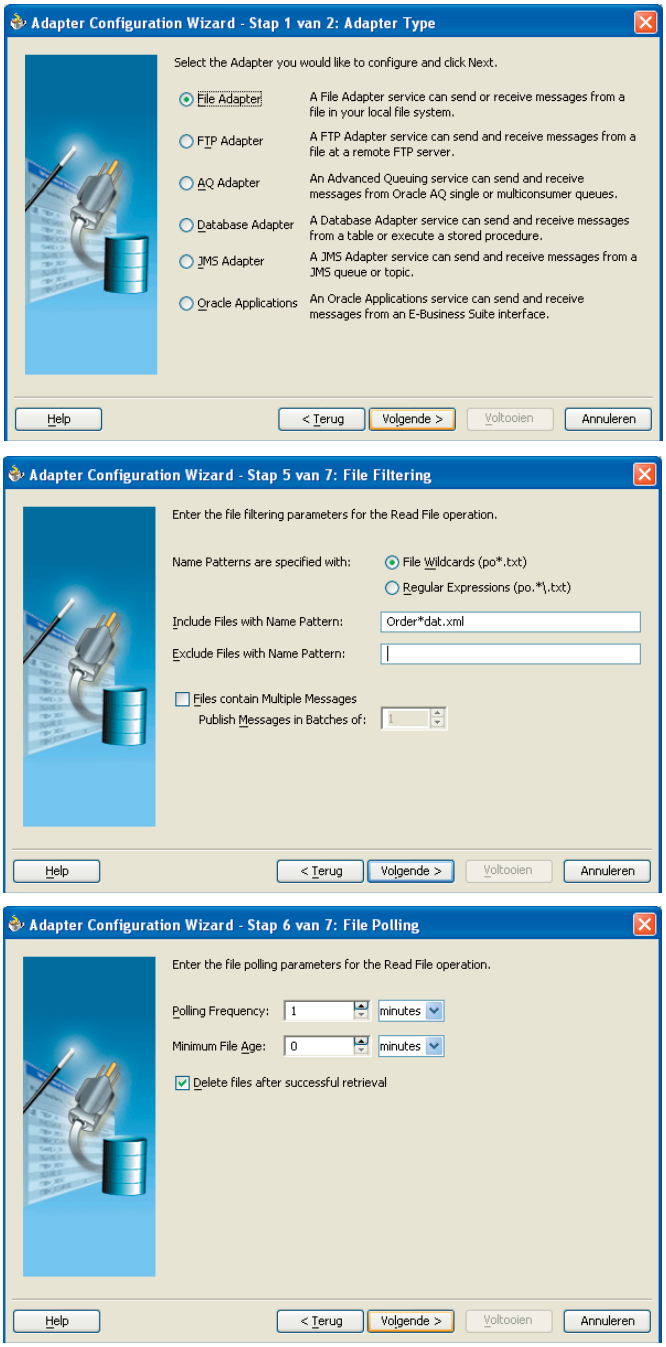
Coördineren in XML

Kort na het ontstaan van integratiestandaarden lag de nadruk op de 3GL-implementatie van de services. Toen systeemontwikkeling meer en meer op basis van servicegeoriënteerde architectuur werd uitgevoerd, kwam de nadruk al snel op het integratie-aspect te liggen. Scenario's op basis van choreografie maakten plaats voor scenario's op basis van orchestration. Hierbij werd de coördinator aanvankelijk net als de aangeroepen webservices in een 3GL geïmplementeerd. Er was immers geen alternatief beschikbaar. Maar op zich was dit wel vreemd. Ga maar na: eigenlijk was de coördinator een centrale component die variabelen in een intern 3GL-formaat versleutelde tot een XML-document, dat vervolgens communiceerde met een andere component, het antwoord weer ontcijferde van XML naar het interne 3GL-formaat, bewerkte en hetzelfde ritueel uitvoerde met de volgende aan te roepen component. Zuiver beschouwd was de coördinator eigenlijk een component die intelligent om diende te gaan met XML-bewerkingen, want er ging XML in en XML uit. Dat aspect, gecombineerd met de eerder genoemde kerntaken van een coördinator, maakten de tijd rijp voor de definitie van BPEL. BPEL is een in XML gespecificeerde taal waarmee orchestrations kunnen worden vastgelegd. Het beschrijft welke webservices moeten worden aangeroepen en in welke volgorde, met welke parameters ze moeten worden aangeroepen en wat ze teruggeven, het biedt mogelijkheden voor programmeertaal- constructies, parallelle verwerking, foutafhandeling. Natuurlijk is BPEL niets meer dan een gestructureerde taal waarin kan worden vastgelegd hoe de feitelijke orkestratie luidt. Of waarmee, enigszins denigrerend gezegd, een stukje formele proza kan worden geschreven. Om BPEL 'tot leven te laten komen' is meer nodig: bij voorkeur een specifiek ontwikkeltool met grafische editors en wizards waarmee al browsend en assemblerend een orchestration bij elkaar wordt geklikt. Daarnaast is een bijbehorende BPEL-server onmisbaar om aldus geschreven BPEL-orchestrations in te kunnen deployen. En voilà: dat is precies de functionaliteit van de Oracle BPEL Process Manager. De BPEL Process Manager bestaat uit drie componenten: BPEL Designer, BPEL Server en de BPEL Console. In afbeelding 6 is de totale architectuur te zien van de Process Manager.

Oracle BPEL Designer

Omdat Collaxa bewust mikte op open standaarden heeft het haar ontwikkeltool medio 2002 uitgebracht op Eclipse. Hoewel Eclipse een grote schare aanhangers heeft, is het waarschijnlijk

Advertentie



Afbeelding 7. Enkele stappen van de File Adapter Wizard

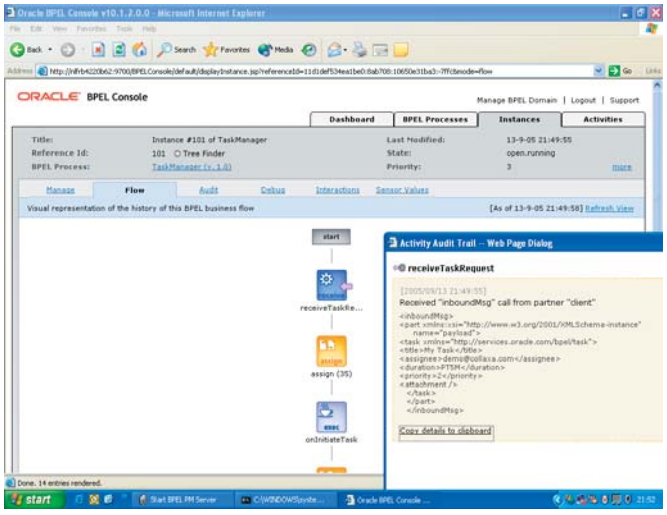
een onbekende naam voor de meeste Oracle-ontwikkelaars. Eclipse is een platform dat aanvankelijk door IBM is ontwikkeld voor eigen gebruik, maar later aan de open source community is gegeven. Met dit platform kan door middel van plug-ins een eigen IDE worden ontwikkeld. Toen Oracle Collaxa overnam erfde het dus een ontwikkeltool dat niet was gebaseerd op haar eigen IDE (JDeveloper) maar op Eclipse. Inmiddels heeft Oracle een vrijwel gelijkwaardige versie van BPEL Designer als plug-in voor JDeveloper uitgebracht. Oracle heeft aangegeven de Eclipse plug-in te blijven ondersteunen om zo de community die zich reeds gevormd had rond Collaxa's BPEL Designer voor Eclipse te blijven bedienen. In afbeelding 1 is een illustratie te zien van BPEL Designer voor JDeveloper. In het vervolgartikel over BPEL zullen we dieper ingaan op BPEL Designer.

Oracle BPEL Server

Geheel in lijn met haar open source beleid heeft Collaxa haar BPEL Server in Java geschreven. Dit betekent dat de server niet alleen in de applicatieserver (of preciezer: in OC4J, de J2EE application server) van Oracle kan worden gedeployed, maar ook in andere applicatieservers, zoals de open source JBoss, BEA WebLogic en IBM WebSphere. BPEL Server bevat verschillende componenten.

Allereerst is er de component 'Integration Services': zo zijn er voor het manipuleren van XML data diverse mogelijkheden. Er kunnen XPath en XQuery operaties en XSLT transformaties worden uitgevoerd met behulp van de in de server aanwezige transformation services. Deze features illustreren het efficiënt gebruik van XML het best. Daar waar een coördinator-webservice die is geïmplementeerd in een 3GL string-manipulaties en andere bewerkingen doet met de mogelijkheden van de 3GL, gebeurt dit in een BPEL-proces met behulp van bewerkingen uit het XML-kennisdomein. In de categorie 'Integration Services' valt ook de ingebouwde takenlijst applicatie. Ingebouwd is hierbij eigenlijk niet het juiste woord, het is eerder een meegeleverde applicatie die zelf rondom BPEL is gebouwd. De applicatie is analoog aan de Worklist applicatie van Oracle Workflow.

De tweede component van de BPEL Server wordt gevormd door de categorie 'WSDL Bindings'. Deze categorie behelst eigenlijk de mogelijke manieren van aanroepen van en aangeroepen worden door de buitenwereld. Dit is niet uitsluitend beperkt tot het aanroepen van webservices op basis van SOAP. Het is ook mogelijk gebruik te maken van bijvoorbeeld - voor Javanen welbekende mechanismen - J2EE Connector Architecture (JCA) en Java Message Service (JMS) en het communiceren via mail. Kennis van deze mechanismen is echter niet echt nodig om er toch - onder water - gebruik van te maken. BPEL Designer heeft verschillende wizards waarmee adapters kunnen worden gedefinieerd, bijvoorbeeld voor het pollen en inlezen van files uit een directory (zie afbeelding 7). Deze wizards maken op basis van de opgegeven gegevens een WSDL document met daarin zogenaamde JCA bindings. Ten slotte kan er ook nog gebruik worden gemaakt van zogenaamde Web Service Invocation Framework (WSIF)-bindings. Als vanuit een BPEL-proces Java-code moet worden aangeroepen (omdat bepaalde functionaliteit bijvoorbeeld reeds in Java is geïmplementeerd) dan is de WSIF-binding de meest gebruikte manier daarvoor. Deze heeft de beste performance maar is minder uitwisselbaar (voor elke applicatieserver dient een specifieke binding te worden gedefinieerd). Een andere keerzijde is



Afbeelding 8. BPEL Console met drill-down op processtap 'receiveTaskRequest'.

dat de huidige versie van BPEL Designer (10.1.2) geen ondersteuning biedt voor WSIF. Dat betekent dat deze bindings met de hand moeten worden geschreven. Voor gevorderden dus.
De derde en tevens laatste component van de BPEL Server is de Core Engine. Als in het ontwikkelproces BPEL-files gecompileerd zijn en verpakt in een .jar file (de zogenaamde 'BPEL suitcase') kan deze worden gedeployed in de BPEL Server. De processen in de suitcase worden uiteindelijk uitgevoerd door de Core Engine, daarbij gefaciliteerd door de eerder beschreven componenten.
Voor langlopende processen is het niet verstandig de procesinformatie uitsluitend in het (vluchtige) geheugen van een server opgeslagen te hebben. Daarom biedt BPEL-server de mogelijkheid de sessie informatie ('state') van deze processen op te slaan in een database die daarmee als zogenaamde 'dehydration'-database wordt gebruikt.

Oracle BPEL Console

De BPEL Console biedt de mogelijkheid om BPEL-processen te starten, te monitoren en te beheren. BPEL-processen zullen veelal via de API van de Process Manager worden gestart. Voor testdoeleinden kan het soms handig zijn om een proces te starten via de Console. Omdat processen soms langlopend zijn is het prettig de status daarvan inzichtelijk te kunnen maken.
De Console is in staat om het in uitvoering zijnde proces bijzonder fraai grafisch weer te geven, vergelijkbaar met de layout van de processen ten tijde van het modelleren in BPEL Designer. Hiervoor wordt gebruik gemaakt van de 'Partial Rendering Facility' van Microsoft Internet Explorer. Dat betekent dat heel selectief bepaalde delen van een browserpagina kunnen worden ververst met andere informatie. Vandaar dat momenteel alleen MS IE wordt ondersteund.
De Console kent veel mogelijkheden voor drill-down op processtappen en kan dan bijvoorbeeld inzicht verschaffen in de feitelijke inhoud van de XML-stroom die het proces doorloopt (zie afbeelding 8).

Resumerend

Veel Oracle-klanten die in het verleden applicaties hebben ontwikkeld met Oracle Designer en Oracle Developer laten de ontwikkelingen rond Java, XML en BPEL nog vaak gelaten over zich heenkomen. BPEL lijkt de zoveelste hype die vanuit de Verenigde Staten is overgewaaid. Het valt echter niet meer te ontkennen dat er inmiddels een meer dan volwassen 'technologiestack' beschikbaar is om systemen te ontwikkelen gebaseerd op een service georiënteerde architectuur (SOA). Dat betekent niet dat de bestaande applicaties moeten worden afgedankt om plaats te maken voor nieuw te ontwikkelen service gebaseerde applicaties. Integendeel! Met een kleine inspanning kan de functionaliteit van bestaande applicaties worden ontsloten in de vorm van webservices. Het voordeel hiervan is dat de applicaties daarmee voorbereid zijn op integratie met de .NET- en J2EE-platforms. Maar met de BPEL Process Manager wordt de investering voor webservices nog extra te gelde gemaakt. Al modellerend en specificerend kan bestaande functionaliteit (verspreid over verschillende applicaties) worden gecombineerd tot compleet nieuwe functionaliteit zonder dat daar aanvullende programmatuur voor dient te worden geschreven.
Met de achtergrondinformatie in dit artikel en het inzoomen op BPEL in het vervolg ervan, hebben ontwikkelaars alle informatie voorhanden om er mee aan de slag te gaan!

Referenties

1 *OASIS Homepage*: http://www.oasis-open.org/
2 *Traditioneel of J2EE, de keuze in 2003*: http://www.anewlink.nl/publicaties.htm
3 *UDDI OASIS Homepage*: http://www.uddi.org/
4 *Developer UDDI sectie op OTN*: http://www.oracle.com/technology/tech/webservices/htdocs/wsvsm/uddiover.html

Harold Gerritsen (e-mail: h.gerritsen@anewlink.nl) is Principle Consultant bij A New Link bv. Hij heeft meer dan vijftien jaar ervaring in het adviseren over effectief inzetten van Oracle technologie in projecten en bedrijven (www.anewlink.nl).