Interface

We schrijven het jaar 2009. Iedereen is druk bezig om zijn bestaande Java-applicaties te herschrijven met Closures en XML literals, want die zijn met het verschijnen van Java SE 7 ineens populair geworden. Als je echt hip wilt zijn, dan programmeer je in je zelf geschreven DSL. Conservatievelingen schrijven nog in Ruby. Maar, wacht even, Java EE 6 zal worden gebaseerd op Java SE 6. Dus wacht nog maar even met die Closures, als ze er überhaupt al gaan komen.

Vooruitblik op Java EE 6

Ing. Bert Ertman
is werkzaam als Technology Manager bij het Competence Center Java van Info Support BV.
E-mail: berte@infosupport.com

In mei 2008 viert Java EE 5 alweer zijn tweede verjaardag. De vraag die niemand durft te stellen is: Waarom wordt Java EE 5 nog maar zo weinig gebruikt? Alles is toch juist zo veel gemakkelijker geworden? De complexiteit is stukken verminderd, XML descriptors zijn niet meer verplicht, er is veel minder boilerplate noodzakelijk en afhankelijkheden kunnen gemakkelijk worden opgelost met dependency injection. Hoe kan het dan dat niemand het gebruikt? Het antwoord op deze vraag vereist wat nadere analyse.

Complexiteit

Allereerst is het zo dat ondanks de op het eerste oog significante versimpelingen Java EE 5 uiteindelijk nog steeds complex is. Ik moet bekennen dat ik zelf ook af en toe het gevoel krijg dat het hele ease-of-use thema in feite niet meer is dan ease-of-demoing. Daarmee bedoel ik dat het voor evangelisten allemaal prachtig is, aangezien zij maar zeer oppervlakkig iets laten zien van nieuwe technologie en deze oppervlakkige zaken allemaal erg simpel zijn te gebruiken. Wil je echter meer – en voor een beetje enterprise applicatie is dat al snel noodzakelijk – dan zul je wel degelijk van de hoed en de rand moeten weten. Ontwikkelaars die aan de slag gaan met dergelijke versimpelde technologie zullen dus na een tijdje opnieuw tegen complexiteit aanlopen en daarmee wellicht een déjà vu of het ‘ zie je nou wel’-gevoel krijgen. Mocht iemand nog de illusie hebben dat je als volledige dummy in staat kunt zijn enterprise applicaties te maken dan wil ik hem bij deze uit de droom helpen.

Complexiteit is niet alleen een probleem van ontwikkelaars, maar ook van application server-bouwers. Om in aanmerking te komen voor het Java EE 5 compliant stickertje moeten maar liefst drieentwintig specificaties worden geïmplementeerd. Geen wonder dat het lang heeft geduurd voordat de eerste application servers gecertificeerd waren voor Java EE 5. Sommige, waaronder de veelgebruikte Websphere application server van IBM – zijn dat nog steeds niet. Maar, stellen dat het allemaal de schuld van Websphere is, is denk ik toch te gemakkelijk.
Een andere reden dat Java EE 5 nog niet veel wordt gebruikt, is het feit dat in veel productieomgevingen Java SE 1.4 nog steeds het uitgangspunt is. Technisch gezien zou het wel mogelijk zijn om met behulp van veel XML-geknutsel de technologie van Java EE 5 te gebruiken, maar juist die annotaties bieden veel meerwaarde. Dus stellen dat zonder Java SE 5 run-time geen Java EE 5 gebruikt zal worden, is denk niet al te ver naast de waarheid.

Veel gefrustreerde J2EE gebruikers waren al weggejaagd in de armen van zogenaamde rebel frameworks of netter gezegd: de facto standaarden. Bijvoorbeeld naar één van de vele web frameworks, web services stacks, of alternatieve ORM-tools. Zij hebben waarschijnlijk zo’n nare bijsmaak overgehouden aan sommige J2EE-technologie dat ze niet heel gemakkelijk zullen geloven dat het nu allemaal beter is. Het feit dat sommige terminologie van nieuwe oplossingen nog sterk lijkt op de oude, controversiële, oplos-

sing, helpt daarbij ook niet echt. Denk bijvoorbeeld aan entity (EJB 3) en Entity Bean (EJB 2.x). Kortom, werk aan de winkel voor de evangelisten om over de bühne te krijgen waarom Java EE 6 wel het upgraden waard is. Voordat we al te hard gaan oordelen, is het de hoogste tijd om eens te kijken naar wat Java EE 6 eigenlijk allemaal voor ons in petto heeft.

JSR 316

De overkoepelende zogenaamde umbrella-specificatie voor Java EE 6 is Java Specification Request 316. De Expert Group wordt geleid door Bill Shannon en Roberto Chinnici, beiden werkzaam voor Sun. De Expert Group bestaat verder uit veel bekende namen in enterprise Java-land, waaronder afvaardigingen van alle bekende application server vendors.
In tegenstelling tot wat veel mensen denken zijn de specification requests over het algemeen goed leesbaar. Dat geldt niet alleen voor een umbrella-specification als JSR 316, maar ook voor de meer technische JSR's die bijvoorbeeld een nieuwe API beschrijven. Het lezen van de JSR 316 specificatie leert ons dat er voor de komende release van het enterprise Java-platform feitelijk twee hoofddoelen zijn, namelijk extensibility en profiles. Naast deze hoofddoelen worden nog een aantal andere zaken beschreven, zoals pruning, ondersteuning voor SOA, en een aantal belangrijke keuzes met betrekking tot welke set van API's onderdeel zullen vormen van Java EE 6 en welke zijn afgewezen.

Interessant is de verandering in de opstelling van de JCP ten aanzien van de one-size-fits-all standaards

Referentie-implementatie

Ondertussen wordt er ook al gewerkt aan de referentie-implementatie van Java EE 6, in de vorm van de Open Source GlassFish application server. Bij de term referentie-implementatie denken veel mensen misschien nog met ongenoegen terug aan de misbaksels die met J2EE 1.2 en 1.3 meekwamen. Deze waren dan ook niet anders bedoeld dan om te kunnen spelen met de ook al ongelukkig ontvangen Pet Store-applicatie. Met Project GlassFish heeft Sun het fenomeen referentie-implementatie inmiddels radicaal over een andere boeg gegooid. Feitelijk is GlassFish nu een applicatieserver met productieaspiraties. GlassFish versie drie staat inmiddels volop in de steigers en geeft nu al een eerste preview van de nieuwe functionaliteit die onderdeel zal worden van Java EE 6.

Tijdpad

In het oorspronkelijke proposal wordt gesproken over een final release in het vierde kwartaal van 2008, maar er is nog enige onduidelijkheid over het precieze tijdpad. Tijdens JavaPolis sprak Roberto Chinicci over het tweede kwartaal van 2009. Dit laatste lijkt op dit moment het meest realistische scenario en sluit mooi aan op de rond dat tijdstip geplande JavaOne conferentie. Releases van dit soort omvang worden meestal op een dergelijk evenement wereldkundig gemaakt. De eerste publieke review van de specificatie kan rond de tijd van het verschijnen van dit artikel worden verwacht.

Wat kost dat?

Voor het geval je je nog af mocht vragen waarmee Sun geld verdient aan Java? De Java EE 6 specificatie beschrijft in het kort de pricing van het Java EE compatible stickertje. Per gemarket product moet een commerciële vendor tussen de 350.000 en 700.000 dollar betalen. In het geval van enkele tientallen vendors met ieder een aantal verschillende distributies van hun product in de prijslijst levert dit toch nog steeds een aardig zakcentje op.

Uitbreidbaarheid

Interessant om waar te nemen is de verandering in de opstelling van de JCP ten aanzien van de one-size-fits-all standaards. Tot voor kort moest een standaard als Java EE worden gezien als de enige gezaghebbende implementatie voor alle enterprise computing op het Java-platform. Het was ondenkbaar dat je iets anders zou willen gebruiken. Ondertussen ging de rest van de wereld vrolijk aan de slag met alternatieven als Struts, Hibernate en Spring, omdat die op de een of andere manier toch beter aansloten bij de gekozen oplossing. Voor sommigen een kwestie van 'the right tool for the right job', anderen verafschuwden alles wat geassocieerd kon worden met Java EE en kozen uit overtuiging voor een rebel framework.
In eerste instantie koos de JCP voor de tactiek van het doodzwijgen en negeerde ze het bestaan van alternatieven gewoon, maar al gauw bleek dat dit niet het gewenste resultaat opleverde. Plan B zou je kunnen karakteriseren als 'steal from the best or die like the rest'. Eerst voorzichtig en later meer openlijk werden de nodige features gekopieerd van succesvolle open source-alternatieven. Deze ommekeer is deels te verklaren door de toetreding van een aantal vooraanstaande open source afgevaardigden tot de Expert Groups van verschillende technologieën. Het resultaat van deze verschuiving is dat Java EE niet meer moet worden gezien als de complete stack voor het bouwen van alle enterprise applicaties, maar meer als een stuk enterprise infrastructuur op basis waarvan commerciële leveranciers en open source-ontwikkelaars hun eigen oplossingen kunnen baseren.

Uitbreidbaarheid van Java EE is daarmee een heel belangrijk doel geworden voor deze en toekomstige versies van de specificatie. De Java EE 6 specificatie onderkent dit fenomeen en spreekt zelfs van een te grote en complexe specificatie en stelt daarbij twee oplossingen voor. De eerste oplossing is het introduceren van het pruning-proces (daarover verderop meer) en de tweede oplossing moet worden gevonden in het introduceren van profiles.

Profiles

Hoe profiles er exact uit komen te zien is op dit moment nog onduidelijk. Vooralsnog is het voornamelijk een theoretisch verhaal, maar in ieder geval heeft dat al redelijk wat stof doen opwaaien in tientallen blogs wereldwijd. Voor sommigen is het een verademing dat open source-alternatieven eindelijk de enterprise erkenning krijgen die ze verdienen. Anderen staan er juist heel negatief tegenover en vrezen dat de profiles een herhaling worden van de niet zo heel succesvolle profiles die bijvoorbeeld in de Java ME wereld worden gebruikt. Zeker is dat er in ieder geval één profile door de Expert Group wordt opgeleverd, namelijk het Java EE Web Profile. Daarnaast wordt een set van regels en richtlijnen vastgesteld voor de totstandkoming van nieuwe profiles.

Web Profile

De bedoeling van profiles is om een soort van selectieve implementatie van Java EE technologie te bundelen. Deze vormt vervolgens een basis om op voort te bouwen. In het geval van het Web Profile zal dit dus een laagdrempelige Java EE omgeving zijn, waarin gebruik kan worden gemaakt van Java EE webtechnologie (JSP, Servlets, JSF, enzovoort), maar waarin niet de twintig andere API's zitten die doorgaans ook onderdeel uitmaken van het Java EE platform. In plaats daarvan kan bijvoorbeeld gebruik gemaakt worden van een open source framework en een lichtgewicht runtime voor dat betreffende profile. Bijvoorbeeld Tomcat als run-time voor het Java EE 6 Web Profile in plaats van een volwaardige applicatieserver waarvan misschien maar 10% van de functionaliteit wordt gebruikt.

Een ander heel belangrijk voordeel dat profiles bieden is dat ze zich los kunnen evolueren van de Java EE specificatie. De totstandkoming van een nieuwe versie van Java EE neemt al gauw enkele jaren in beslag. Als zich tussentijds een interessante technologische ontwikkeling voordoet is het vaak lang wachten totdat deze zich ook weet binnen te worstelen in de Java EE specificatie. Met profiles biedt de Expert Group een mogelijkheid om bijvoorbeeld een onderhoudsrelease van een specifieke technologie al eerder uit te laten komen zonder dat gewacht moet worden op de volledige release van het hele platform.

Pruning

Tijdens de ontwikkeling van Java SE 6 werd het pruning-systeem geïntroduceerd. De reden voor het ontstaan was dat de omvang van het Java-platform door de tijd heen alleen maar is toegenomen door het toevoegen van nieuwe features, zonder dat er ooit iets uit is verwijderd. Feitelijk stond de toenmalige specificatie het verwijderen van oude features gewoon niet toe. Dit had uiteraard zo zijn voordelen, met name in de hoek van (binaire) compatibiliteit. Door deze regel strak toe te passen kon de garantie worden gegeven dat een applicatie geschreven voor release N altijd zou blijven werken op release N+1. Met het verstrijken van de tijd bracht deze regel uiteraard ook nadelen met zich mee, waarvan de toenemende complexiteit en de steeds maar groeiende omvang van de Java Runtime Environment de voornaamste zijn. Hier moest dus een oplossing voor komen en daarom werd het pruning-systeem bedacht. Kort gezegd komt het er op neer dat er een gedegen proces is bedacht voor het verwijderen van onderdelen van het Java-platform.

Pruning betreft het verwijderen van bijvoorbeeld een complete API en dus niet van deprecated methodes binnen een API. Het proces werkt als volgt. De Expert Group voor release N besluit een bepaald onderdeel voor te dragen voor pruning. Dit besluit wordt vervolgens gedocumenteerd in de specificatie behorende bij release N. De Expert Group voor release N+1 besluit om op basis van het voorstel van de

Hoe kun je zelf invloed hebben op Java EE 6?

Het is nog niet te laat om je eigen persoonlijke invloed uit te oefenen op de totstandkoming van de Java EE 6 specificatie of samenstelling van de API's. Commentaar kan via de homepage van de JSR op de JCP-website worden gemaild aan de spec leads. Wellicht is het beter om te reageren op specifieke zaken uit de binnenkort te verschijnen early draft van de specificatie. Op technisch inhoudelijk vlak kun je je wellicht beter richten tot de specifieke specificatie van de API zelf. Heb je bijvoorbeeld commentaar op een nieuwe feature van JPA, dan is het beter om dat bij de betreffende Expert Group te melden. Indien opmerkingen op een zinnige manier worden onderbouwd, wordt je bijdrage absoluut serieus genomen. Of het de gevraagde verandering tot gevolg heeft is een ander verhaal, maar de feedback wordt zeker gewaardeerd en het geeft de Expert Group een beeld van hoe hun werk wordt ontvangen.

Tijdens de ontwikkeling van Java SE 6 werd het pruning-systeem geïntroduceerd

voorgaande Expert Group ofwel de verwijdering uit te voeren, of om de beslissing door te schuiven naar de volgende Expert Group. In dit laatste geval zou de Expert Group wel kunnen besluiten om er een optionele feature van te maken. De enterprise kandidaten die voor pruning in aanmerking komen zijn: JAX-RPC (JSR 101), EJB Container Managed Persistence en JAXR (JSR 93). Op zichzelf zijn dit geen schokkende zaken. Aangezien ze op dit moment slechts zijn aangemerkt om in een volgende release (Java EE 7) verwijderd te worden, zijn ze dus nog gewoon onderdeel van Java EE 6.

Onderdelen van Java EE 6

De volgende nieuwe specificaties zullen onderdeel uitmaken van Java EE 6:

JSR 196:	Java Authentication SPI for Containers
JSR 236:	Timer for Application Servers
SR 237:	Work Manager for Application Servers
JSR 299:	Web Beans
JSR 311:	JAX-RS - Java API for RESTful Web Services

Daarnaast krijgt een aantal bestaande API's een update:

- Enterprise JavaBeans 3.1
- Java Persistence API 2.0
- Servlet API 3.0
- JavaServer Faces 2.0
- JAX-WS
- Java EE Connector API

Ten slotte is er besloten dat de volgende JSR's geen onderdeel zullen vormen van Java EE 6, maar dat ze in plaats daarvan in aanmerking komen om onderdeel te worden van één of meer nieuw te creëren profiles:

JSR 168	Portlet Specification
JSR 170	Java Content Repository
JSR 207	Process Definition for Java
JSR 208	Java Business Integration (JBI)
JSR 225	XQuery API for Java (XQJ)
JSR 235	Service Data Objects
JSR 286	Portlet Specification 2.0
JSR 289	SIP Servlet 1.1
JSR 301	Portlet Bridge Specification for JavaServer Faces

Conclusie

De grote vraag zal zijn of de Expert Group er in zal slagen om enterprise Java volgens de Java EE 6 standaard weer 'hot' te krijgen. Met het nieuwe profiles-systeem heeft het mogelijk de oplossing in handen. Door een uitgeklede, op maat gemaakte, stack van Java EE technologie te combineren met een framework zoals Spring of Hibernate, kunnen aansprekende alternatieven ontstaan die een mix van twee werelden geven. Innovatie en standaardisatie gaan dan weer hand in hand. Het heeft echter ook volop potentie om zichzelf in de voet te schieten. Of Java EE dat overleeft? We zullen het zien! «

Referentie

Java EE 6 – JSR 316 Specificatie
http://jcp.org/en/jsr/detail?id=316