Interview

De Java-wereld zit vol met mensen die een bijzondere bijdrage leveren op het platform. Vaak doen ze dat in de luwte en vallen ze nauwelijks op. Java Magazine gaat in een serie van vijf afleveringen op zoek naar deze professionals en hun activiteiten. In het tweede deel spreken we met Martin Odersky de drijvende kracht achter Scala.

'Scala is wat Java had moeten zijn'

Functionele programmeertaal is sneller

Scala is voor Java het belangrijkste alternatief op de JVM. Het is een taal die vooral professionele programmeurs zal aanspreken omdat zij hiermee een niveau hoger kunnen komen op het gebied van productiviteit en betrouwbaarheid van software. Ze krijgen betere tools in handen om zichzelf uit te drukken. Het is ook niet zo ingewikkeld om van Java op Scala over te stappen. Veel van de syntax is hetzelfde. Er zijn wel wat verschillen, maar die zijn snel aan te leren. Scala is een hybride taal, die zowel object georiënteerd als functioneel programmeren ondersteunt.

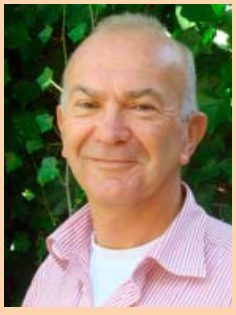
Dat vindt in ieder geval Martin Odersky, professor aan de universiteit van Lausanne en 'uitvinder' van de functionele programmeertaal Scala.

Je kunt, zo stelt hij, met Scala iedere Java library gebruiken. "Als je alleen de Java libraries gebruikt doe je jezelf echter tekort. De Scala libraries zoals die er nu liggen geven – ook al zijn ze niet zo compleet als die van Java – meer mogelijkheden. De Scala libraries zijn nog volop in ontwikkeling. De Scala collection library bestond al voor deze release. De collection API biedt closures en zogenaamde hogere order functies om operaties op collecties uit te voeren. Hogere order functies zijn functies die zelf een functie als invoer hebben, of een functie als uitvoer teruggeven. De collection library biedt hiermee een krachtiger manier om operaties op collecties uit te voeren dan in Java mogelijk is".

Scala wordt nu drie jaar gebruikt voor serieuze industriële ontwikkeling en de belangstelling groeit snel. In 2009 groeide het aantal downloads van de site met een factor 8 van 1.500 naar 12.000. In april van dit jaar werd Scala 15.000 maal gedownload. Het aantal gebruikers ligt nu rond de 100.000, schat Odersky. De meest recente Scala Days conferentie was een maand voor de start al volgeboekt.

Ook voor beginners

Odersky ziet Scala als een heel geschikte taal voor beginnende ontwikkelaars. Het heeft veel features om agile te kunnen werken zoals bijvoorbeeld in een scripting language. Het is dus erg geschikt voor explorative programming. Werken met Scala



Robert de Ruiter
is hoofdredacteur van Java Magazine.

Functionele duizendpoot

Martin Odersky is op het gebied van functioneel programmeren een ware duizendpoot. Hij is professor aan de Ecole Polytechnique Fédérale de Lausanne (EPFL), waar hij de onderzoeksgroep naar nieuwe programmeertalen leidt. Voorheen werkte hij bij IBM Research en de universiteiten van Yale, Karlsruhe en South Australia. Zelf haalde hij zijn doctoraat aan de Eidgenössische Technische Hochschule Zürich. Hij schrijft regelmatig in het Journal of Functional Programming en is lid van een werkgroep van de International Federation for Information Processing (IFIP WG 2.8). Deze werkgroep bestaat uit 40 leden, die iedere tien maanden vergaderen over alle aspecten van functioneel programmeren.

geeft meer voldoening omdat het interactiever is dan Java.
Als je voor het eerst naar de API kijkt kan Scala de Java-ontwikkelaar ook afschrikken vanwege de geavanceerde features van Scala. Volgens Odersky is de API zo gecompliceerd omdat er veel advanced type machinery, waaronder zogenaamde implicit types en 'higher kinder types" in voorkomt om het de gebruiker makkelijker te maken.
"In een aantal gevallen kan het zijn dat de API een beginnend programmeur afschrikt, omdat hier geavanceerde concepten in gebruikt worden. Dit is bijvoorbeeld het geval in de herontworpen collections library. Dit is echter gedaan om het gebruik eenvoudig te maken. Als gebruiker van de Collections library hoef je niet direct te weten hoe dit precies werkt. Je hoeft alleen te weten hoe je een bepaalde methode kan aanroepen, en de library zorgt voor correcte werking".
"Maar je ziet dat niet. Je ziet als ontwikkelaar die implicit types niet. Die kun je dus gewoon negeren als je naar de library kijkt. Je ziet de methodes en de beschrijving en daar kun je als developer goed mee uit de voeten. Alleen als je aan het framework wilt bijdragen zul je dieper in het geavanceerde type systeem moeten duiken, maar dat is werk voor meer ervaren programmeurs. Als gebruiker hoef je niet te weten wat die geavanceerde types zijn. Dan hoef je alleen te weten hoe je de types kunt loslaten op de use case waar je aan werkt".

Leercurve

Als beginner met Scala loop je aan tegen het feit dat er veel verschillende manieren zijn om iets uit te drukken. Hierin schuilt het gevaar dat er slechte gewoonten binnensluipen. Odersky: "De filosofie van Scala is dat we niet willen voorschrijven hoe je programmeert. Je moet je afvragen of je een beginner alle mogelijkheden van de taal moet bieden. We moeten nog uitdenken hoe je de taal aanleert. De docent zou de subsets gedoseerd kunnen aanbieden, zodat deze passen bij het ontwikkelingsniveau van de studenten. Scheme – een dialect van Lisp - is ook zo'n flexibele taal, waarin subsets zijn gedefinieerd. In de VS wordt deze taal ook zo onderwezen dat de leerlingen de subsets gefaseerd krijgen aangeboden. De compiler dwingt je daartoe, omdat die een subset-mode heeft".

Het is waarschijnlijk eenvoudiger om in Scala te leren programmeren dan om van Java op Scala over te stappen. Voor beginnende programmeurs in Java is het geruststellender om met een routinetaal te werken omdat het eenvoudiger is en je er het gevoel bij krijgt dat je daadwerkelijk iets hebt verricht. Een horde die Java ontwikkelaars in Scala moeten nemen is vermoedelijk het functionele karakter. Java zelf is als taal echter ook niet eenvoudig, denk aan bijvoorbeeld generics, enums. Het idioom van Scala kun je geleidelijk aanleren. Als je er eenmaal mee hebt leren werken geeft het veel meer mogelijkheden. Odersky houdt zijn studenten voor dat Scala in het begin wellicht frustrerend kan zijn, maar dat het uiteindelijk zijn vruchten afwerpt. "In Scala telt ieder woord".
Scala is met name populair in de handel. Dat komt wellicht omdat men zich in deze sector niet wil bezighouden met het imperatief opbouwen van de code. Zij zijn vooral gebaat bij het functioneel programmeren in Scala. Het is in Scala ook makkelijk om een Domain Specific Language te ontwikkelen. Zeker in het domein van banken en aandelen/optie handel zijn de modellen vaak min of meer 1 op 1 te vertalen in Scala. Dat moet ook de reden zijn waarom veel ex-Haskell-programmeurs zijn geïnteresseerd in Scala.

Martin Odersky in goed gezelschap op Javapolis 2007 met de 'vader' van Java, James Gosling, Joshua Bloch (Google), Neal Gafter (Microsoft) en Carl Quinn (BASE)

Testen

Een ander voordeel van Scala is dat de code makkelijk kan worden getest. Dat kan in JUnit, maar in de taal zijn ook test specificaties opgenomen. Hiermee kunnen soepeler tests worden geschreven. Daarnaast is er Scala Check, waarin je alleen hoeft aan te geven wat je wilt testen. "De tool schrijft dan automatisch de tests voor je. Als een test mislukt geeft de tool ook aan in welk deel van de code de fout ongeveer kan zitten. ScalaCheck genereert de test gevallen voor je, op basis van de test specificatie die je opgeeft. Je kunt bijvoorbeeld als statement opgeven dat (x + y) = (y + x) voor alle waarden van x en y. Het framework genereert vervolgens 100 testgevallen, inclusief randgevallen, waarbij het dit valideert.

In 1995 kwam Odersky in contact met professor Philip Wadler in Edinburgh, die hem om raad vroeg over Java. "Hij vroeg me wat hij kon doen aan Java, omdat het een goed platform leek en perfect geschikt voor het web, maar dat het functioneel programmeren dreigde onder te sneeuwen. Op dat moment is het idee ontstaan om een functionele taal te bouwen, die compiled in Java. Het eerste experiment was Pizza, een soort generic Java. We noemden dat zo omdat veel hackers op Pizza leven. Pizza had closures, pattern matching en generics. De closures lieten we vallen, omdat we die niet nodig hadden. Pattern matching vonden we iets voor simpele zielen, dus dat viel ook af. Generics vonden we wel interessant. Dat kwam terug in de JCompiler. Op dit idee ben ik bij de IPFL gaan voortborduren ".

De closures en pattern matching ideeën werden niet in standaard Java opgenomen, omdat dit werd tegengehouden door anderen. Closures vanwege het feit dat men anonymous inner classes al genoeg vond, pattern matching omdat men het 'Visitor pattern' beter vond.
Hijzelf wilde beide concepten al wel toevoegen aan de standaard Java taal. Zowel closures als pattern matching zijn nu wel opgenomen in Scala. Closures worden (wellicht) ook in Java 7 opgenomen.
Imperatief programmeren is vooral een

kwestie van denken in tijd, in volgorde. Je moet alles in de juiste volgorde zetten, anders werkt het niet. Functioneel programmeren daarentegen is denken in ruimte. Je bedenkt welke structuur je van welke delen wilt construeren. Dat past ook beter in deze tijd. Tegenwoordig wordt vaker gedacht ‘welke structuur wil ik opbouwen’ dan ‘in welke volgorde ga ik het doen’.

“Ik geef op dit punt geen advies over wat beter is. Natuurlijk kun je nog imperatief ontwikkelen. We proberen wel op een bescheiden manier om delen functioneel te doen. Scala is ook bruikbaar in combinatie met imperatief programmeren. Het heeft bijvoorbeeld een feature, Traits, waarmee je code aan interfaces kunt toevoegen, zolang je een standaard implementatie van de methodes hebt. Traits zijn te vergelijken met Java interfaces. In een Trait is het echter mogelijk om methodes ook een implementatie te geven. Dit zorgt ervoor dat het mogelijk is om een nieuwe methode aan een Trait toe te voegen met een default implementatie, zonder dat je de implementerende klasses hoeft aan te passen.
Je moet dan wel classes recompilen, maar je hoeft deze niet opnieuw te programmeren. Ik verwacht trouwens dat we in een nieuwere versie van Java ook wel zoiets als Traits zullen aantreffen. Een andere feature van Scala is dat je maar een regel nodig hebt om een class met een constructor op te zetten, in plaats van vijf. In het algemeen is hoef je in Scala minder regels code te schrijven dan in Java. Dit komt onder andere door het gebruik van functies, maar ook doordat de taal zelf compacter is dan Java. Scala kan in de meeste gevallen het type van een variabele bepalen, zonder dat je dit expiciet hoeft op te geven, zoals in Java. Ook worden bijvoorbeeld getters en setters in klassen automatisch gegenereerd door de compiler, zonder dat je deze zelf in je code hoeft op te nemen. Kortom: je kunt compacter en nauwkeuriger code schrijven, ook al werk je imperatief”.

Functioneel

Odersky ziet vanwege de effectiviteit volledig functioneel programmeren toch als de toekomst. “Zonder gebruik te maken van closures kun je ook in Java functioneel programmeren, maar je moet dan een hele berg code schrijven. Dat is het doorgaans niet waard. Ik zou het niet doen”.

Tot voor kort was het voor de Scala-ontwikkelaar moeilijk om met bepaalde Java-facetten te communiceren. En in omgevingen waar veel Java-code in zit is het wellicht handiger om maar gewoon in Java verder te gaan. Maar voor nieuwe projecten zie ik niet in waarom je daar nog Java voor zou gebruiken. Dan is Scala zo veel sneller en eenvoudiger”.
Praktijkvoorbeelden van ‘nieuwbouw’ met Scala zijn er inmiddels ook voldoende om te kunnen concluderen dat de taal succesvol wordt ingezet. TomTom bijvoorbeeld is op Scala geprogrammeerd, evenals de backend van Twitter (de website is Ruby on Rails) en LinkedIn. Ook in ‘big business’ wordt Scala in toenemende mate ingezet . General Motors, Warner Brothers en Ericsson gebruiken het. En de Amerikaanse overheid heeft onlangs besloten om de digitale beveiliging van het Capitool met Scala te laten bouwen.
Om als developer met Scala te kunnen werken zul je eerst de klant, of eerder nog je teamleider, moeten zien over te halen om daartoe over te gaan. Dat zal niet altijd meevallen. Odersky ziet in de praktijk vooral voorbeelden waar de teamleider zelf uit de technische hoek komt of de klant zich bezig houdt met technologie. Maar het is geen absolute voorwaarde. Bij Office Depot bijvoorbeeld, een van de grootste leveranciers van kantoorartikelen, wordt ook in Scala geprogrammeerd. Men kampte daar met een ‘mission-critical’ programma van een miljoen regels achterhaalde code, dat samen met verouderde hardware niet meer productief was.

In Scala zijn ook test specificaties opgenomen. Hiermee kunnen soepeler tests worden geschreven

Java 7

Odersky durft geen antwoord te geven op de vraag of het langer uitblijven van Java 7 invloed zal hebben op het succes van Scala. “Dat zal mede afhangen van de vraag of daarin met closures wordt gewerkt. Als dat niet zo is, zal Java 7 maar weinig grote verbeteringen laten zien”.
Op het ogenblik werkt hij aan verschillende projecten. Een van de grootste is om Scala te verbeteren op het gebied van parallellisme en concurrency. We werken aan de integratie van parallel collections in het framework en aan taalvirtualisatie. Dit komt er op neer dat je een programma hebt dat lijkt op Scala, maar dat anders wordt geïnterpreteerd afhankelijk van de manier waarop je het laat lopen. Als je het op een cluster laat lopen dan draait het op de ene manier, maar op een microprocessor of een multicore machine op een andere. Verder is een sponsor gevonden voor het project prolab.NET dat een herimplementatie van Scala op .NET (integratie van Visual Studio) en er loopt nog een project over geavanceerde programma analyse, dat onder meer gaat over event-checking.
Op het ogenblik denkt Odersky na over een business model voor Scala. Binnenkort wordt een supportonderneming opgericht die zich gaat richten op training, consulting, support en de ontwikkeling van tools voor zakelijk gebruik. De eerdergenoemde conferentie heeft hem wat dit betreft vertrouwen gegeven. “Ik denk dat er voldoende steun voor Scala is om er ook een commercieel succes van te maken, maar de toekomst zal het leren. Wat betreft de adoptie heeft ook het feit dat vorig jaar verschillende boeken over Scala zijn uitgegeven, enorm geholpen. Belangstellenden kunnen zich nu beter op de hoogte stellen van wat Scala inhoudt en hoe ze ermee aan de slag kunnen”.

Verkopers gezocht

In totaal werkt een team van dertien mensen bij IPFL aan Scala. Daarnaast dragen nog circa twintig programmeurs regelmatig bijdragen aan. Wat de verspreiding betreft is Odersky altijd op zoek naar consultants, die met Scala werken. Zij moeten het programma uiteindelijk aan de business ‘verkopen’. Programmeurs met kennis van- en toenemende ervaring met Scala zijn in dit stadium belangrijk om de taal onder de aandacht te brengen.
Van een man, die zoals hij zelf zegt al vijf jaar geen Javacode meer heeft geklopt en zich dagelijks bezighoudt met ontwikkeling en verbetering van Scala zou je kunnen verwachten dat hij Java geen warm hart toedraagt. Odersky is in zijn opvattingen opmerkelijk loyaal tegenover Java, maar zijn hart gaat nu eenmaal uit naar functioneel programmeren omdat dit in zijn ogen het klopwerk op een hoger niveau brengt. “Scala is wat Java had moeten zijn”, zegt hij tijdens het gesprek dan ook zonder enige schroom. «