New & Cool

Charles Nutter en Thomas Enebo werken sinds september 2006 in dienst van Sun aan JRuby. Daarvoor deden ze al jaren hetzelfde, maar dan in hun iets schaarsere eigen tijd. Java Magazine sprak met hen over de voordelen van Java voor Ruby en omgekeerd.

JRuby begon met een directe port

JRuby: Ruby met alle voordelen van Java

Ruby is in praktische zin een concurrent van Java. In die zin is het vreemd dat Sun twee mensen in dienst neemt om Ruby op de JVM beter te ondersteunen.
Nutter: 'Veel ontwikkelaars zijn geïnteresseerd in Ruby on Rails, in Ruby als taal en Ruby-ontwikkeling in het algemeen. Sun is geïnteresseerd in dezelfde dingen waar ontwikkelaars ook in geïnteresseerd zijn, dus vanuit die gedachte was het een goed idee.'
Enebo: 'Sun heeft bovendien de wens om het aantal talen dat de JVM ondersteunt te vergroten.'
Nutter: 'Ons doel is ook ervoor te zorgen dat Ruby niet alleen een andere taal-optie voor Java-ontwikkelaars wordt, maar een andere platform-optie voor bestaande Ruby-ontwikkelaars, zodat ze er ook voor kunnen kiezen JRuby voor hun werk te gebruiken.'
Enebo: 'We proberen het zo goed mogelijk in Java te integreren, op een Ruby-achtige manier.'

En dat betekent?
Enebo: 'Dat we de taal niet willen verpesten, we staan het toe transparante calls naar Java te doen, maar we breiden Ruby niet uit, we bieden alleen het mechanisme aan om vanuit Ruby Java aan te roepen, zodat we alle bestaande Java library's voor Ruby geschikt kunnen maken. Charles deed op Javapolis en demo waarin hij dynamisch een kleine Swing-applicatie bouwde, en alle klassen waarmee hij interacteerde waren Java-klassen, maar hij deed het via een Ruby-interface.'
Nutter: 'En daarna liet Thomas zien hoe hij een kleine Ruby on Rails applicatie liet draaien op de JVM, want dat gaat nu met JRuby. We willen niet zeggen dat er 100% ondersteuning is om Rails *(dus niet Ruby alleen, red.)* op de JVM te laten draaien, maar we zijn er wel heel dicht bij. De standaarddingen van Rails draaien heel erg goed.'

Zijn het niet te veel twee werelden, de grote Java-wereld versus de kleine overzichtelijke Ruby on Rails-wereld?
Nutter: 'Er is geen reden om de andere frameworks niet als deel van je applicatie te gebruiken, wanneer je eenmaal Rails op Java draait. Rails doet zijn kern - webdevelopment - heel goed, maar wanneer je daarbuiten moet gaan en een paar van de Java-library's nodig hebt, zal JRuby je niet in de weg staan. We hebben ook demo's gegeven van Rails-applicaties die EJB's gebruiken of de persistence laag. Het is dus een kwestie van smaak: wanneer mensen liever verder Java gebruiken dan kunnen ze daarmee doorgaan en net zo productief zijn als altijd, maar wanneer ze Ruby willen gebruiken hoeven ze de JVM niet op te geven, of alle Java-library's waaraan ze gewend zijn. Ze krijgen een nieuwe, elegantere en grappiger taal om te gebruiken en alle library's krijg je er gewoon bij.'

Tekst en fotografie:
Dré de Man

Enebo: 'Rails is eigenlijk een RAD development environment, je kunt er heel snel een applicatie mee bouwen en de delen erg snel veranderen die je als onderdeel van een groter bedrijf zou willen inzetten.'
Nutter: 'Nog één ander aspect waarin we voor Rails geïnteresseerd zijn, is dat aangezien we nog steeds aan het ontwikkelen zijn met een web-tier en een server- of service-tier in de back-end, dat er geen reden is om Rails als een webapp als front-end te gebruiken, die al je bestaande Java-infrastructuur aanroept, je services op de back-end, en het beste van twee werelden te hebben. Je kunt de services die je in de loop van de tijd hebt geschreven en getest hebt gebruiken, en je kunt het agile web-framework als front-end gebruiken.'

Het schrijven van documentatie lijkt me moeilijk, omdat je twee verschillende doelgroepen hebt.
Nutter: 'We moeten zeker de boodschap aanpassen aan het publiek. Het Ruby-publiek weet al dat Rails en de taal Ruby geweldig is, we moeten hen vertellen hoe we het beter maken en ze nieuwe opties geven en een nieuw platform, nieuwe stabiliteit, performance, wat dan ook. Wanneer we met Java-developers praten, dan weten die dat ze een solide platform hebben, geweldige server-infrastructuren, een geweldige virtual machine. Wij moeten ze laten zien waarom Ruby en specifiek Rails ze iets geeft wat ze daarvoor niet hadden, om meer gedaan te krijgen in dezelfde tijd, of om hun baan beter te doen. We moeten dus twee boodschappen hebben.'

Klopt het wanneer je zegt dat Ruby veel geschikter is voor een iteratieve werkwijze dan Java?
Nutter: 'Ruby en zeker Rails zijn met die gedachte ontwikkeld. Wanneer je een Java-ontwikkelaar vraagt een applicatie te schrijven, dan beginnen ze met het schrijven van frameworks op basis waarvan ze de applicatie schrijven. Als je daarentegen een Ruby-ontwikkelaar vraagt een applicatie te schrijven, dan beginnen ze met het schrijven van een applicatie en ze maken de frameworks naar behoefte. Het is namelijk veel eenvoudiger om de framework uit de requirements te ontwikkelen, vanuit de applicatie zelf. Anders ga je raden wat je nodig zult hebben van de frameworks en bouwt enorm gecompliceerde frameworks die misschien wel maar misschien ook niet in de behoefte van de applicatie zullen voorzien.'

Enebo: 'Je verliest er ook het zicht op dat je een webapplicatie maakt. Je ziet dat ook: hoeveel webframeworks zijn er niet in de Java-wereld, bijna talloos vele. Java is dus een geweldige infrastructuurtaal maar misschien is het niet zo geweldig aan de domeinkant.'

'Java is een geweldige infrastructuurtaal, maar misschien is het niet zo geweldig aan de domeinkant'

Domeinspecifieke talen proberen ook in die leemte te voorzien.
Nutter: 'DSL's behoren zeker tot de populairste delen van Ruby, omdat het zo simpel is om ze in Ruby te schrijven en ze er leuk uit te laten zien. Ze worden ook voor de moeilijkste applicaties gebruikt. Het is al vaak opgemerkt dat wanneer je een applicatie schrijft in Ruby die er al bijna als een DSL uitziet. We hebben nu zelfs de mogelijkheid om DSL's te schrijven naar Java-library's en deze in frameworks onder te brengen.'

Ruby levert zeker tijdwinst op bij bepaalde projecten. Aan de andere kant kost het ook tijd Ruby te leren. Waar zit het break even point?
Nutter: 'Ik denk dat Ruby één van de gemakkelijkste scripttalen is. De doorsnee persoon die geen ontwikkelaar is, kan naar Ruby-code kijken en in grote lijnen begrijpen wat er gebeurt, want het ligt zoveel dichter bij natuurlijke taal dan Java of Perl of andere talen.'
Enebo: 'Wij waren beide Java-programmeurs voordat we Ruby leerden en ik herinner me niet eens dat er een overgang was. Tim Brady heeft geblogged over het leren van Ruby. Hij beschreef het als een namiddagervaring.'

Wat is er nodig om Java-code binnen Ruby te gebruiken?
Nutter: 'Je hoeft het alleen maar aan te roepen als een Java-applicatie. Als je de Java-library hebt, hoef je alleen te zegen: 'enable Java-support'. Dat is één regel en dan kun je alle klassen gebruiken.'
Enebo: 'Java klassen lijken Ruby-klassen te zijn binnen Ruby; ze gebruiken alleen Ruby-syntax. Het zijn allemaal gewoon objecten, maar we doen een beetje magie in JRuby. Als we een aanroep doen die eruit ziet als Ruby-code, roepen we op de achtergrond Java aan en het werkt zoals je zou verwachten.'

Het wordt dus vertaald?
Enebo: 'Er komt heuristiek bij kijken, want Ruby is een dynamische taal en Java is statisch. Als je een methode aanroept wordt de betreffende methode in Java-code gezocht en aangeroepen.'

En hoe zorg je voor de typing?
Nutter: 'Het typing system in Ruby noem je *Duck typing*. Als het kwaakt als een eend en het waggelt als een eend, dan moet het wel een eend zijn. Het enige waar Ruby op let is of de methode die je aanroept op dat object beschikbaar is op dat moment.'

Voor welke situaties is het bijzonder geschikt?
Nutter: 'Het zou iedere situatie kunnen zijn waarbij je uiteindelijk een hoop dubbele code schrijft, waar de Java syntax je in de weg zit. Een voorbeeld vormen collecties van objecten. String-manipulatie, parsing en strings uit elkaar trekken en weer bij elkaar voegen, Ruby doet dat heel erg goed, DSL's, business rules of business-logica, testen.'

Is er al een verzameling open source DSL's geschreven in Ruby?
Nutter: 'Er zijn er een paar. Ik zou zelfs zeggen dat 90% van de Ruby-applicaties die beschikbaar zijn op de een of andere manier waarschijnlijk een DSL in zich hebben. Of ze nu herbruikbaar in Java zijn is moeilijk te zeggen, een goed voorbeeld is Vista Rig. Dat is een build tool, vergelijkbaar met Ant, zorgt voor een meer scripteable syntax. Dan zijn er mensen die gewoon DSL-wrappers rond dingen als Ant schrijven. In het Groovy-kamp hebben ze bijvoorbeeld Swing-builders die Swing-componenten op de achtergrond gebruiken. Er komen steeds meer van die DSL's voor speciale taken.'
Enebo: 'De metaprogrammeer-aspecten van Ruby maken het erg eenvoudig om je eigen DSL te maken.'
Nutter: 'We zullen meer en meer daarvan zien nu mensen JRuby gaan gebruiken op de JVM.'

Wat is het grootste verschil met het schrijven van een DSL in Java?
Nutter: 'De standaardmanier om een DSL in Java te schijven is het creëren van een XML-dialect. Dat wordt erg wijdlopig en is moeilijk om te onderhouden. Ant is een DSL, de meeste van de configuratiefiles voor diverse configuraties zijn DSL's. Maar het is veel gemakkelijker te zien wat er gebeurt met een DSL in Ruby, dan met een DSL in XML of wanneer je probeert een DSL te schrijven in Java.'

Wat zijn in al die jaren de belangrijkste obstakels geweest?
Nutter: 'Performance is altijd een moeilijk probleem, de originele JRuby-implementatie portte de Ruby-code direct naar Java. Dat maakte het veel moeilijker om een goede performance te krijgen. Misschien wel het moeilijkste daarbij was dat de taal Ruby geen specificatie heeft. Er bestaat geen officiële specificatie over hoe het eruit ziet en hoe het werkt. De C-implementatie is in feite de spec. We moesten dus heel veel van de eigen C-code van Ruby lezen om uit te zoeken wat het doet en het gedrag te dupliceren. We zijn nu zo dicht bij de comptabiliteit met Ruby dat we het gevoel hebben dat we de taal en de implementatie begrijpen en dat we kunnen werken aan performance en het integreren met library's.'

Hoe staat het er nu voor met de performance?
Nutter: 'Sommige dingen zullen beter in pure Java zijn en ander in pure Ruby. Voor echt pure geïnterpreteerde Ruby-code zijn we waarschijnlijk twee of drie keer langzamer dan de C-implementatie. Maar heel weinig applicaties zijn alleen puur Ruby. Meestal roepen ze een socket aan of manipuleren ze documenten, of werken met andere resources. Wanneer je die begint te gebruiken, doen we het een heel stuk beter. Ons doel is dezelfde performance te halen als Ruby.'
Nutter: 'Er is nog een hoop werk te doen maar in de afgelopen release hebben we de performance bijna verdrievoudigd.'
Enebo: 'Snelheid is ook moeilijk te meten, er zijn ook wel benchmarks waar we sneller zijn dan Ruby en zeker wanneer er veel resources in het spel zijn wordt het een stuk minder belangrijk. Correctheid is altijd belangrijker dan performance. Je hebt liever een programma dat werkt dan iets dat niet werkt maar dan wel heel snel.'

Voor meer infomatie over Ruby en Ruby on Rails, zie http://en.wikipedia.org/wiki/Ruby_on_Rails «