Danny Lagrouw

Ruby on Rails

Wat is Ruby on Rails

Wat is ervoor nodig om een dertien jaar oude, in Japan ontwikkelde programmeertaal die in vergetelheid dreigt te geraken, in korte tijd te laten uitgroeien tot een wereldwijde hype? Het antwoord: een framework waarmee webapplicaties ongekend snel kunnen worden ontwikkeld. In 2004 besloot de jonge Deen David Heinemeier Hansson zijn ideeën over het bouwen van webapplicaties vast te leggen in een framework. Hij koos daarvoor de programmeertaal *Ruby* en doopte zijn framework Ruby on Rails. Nog geen twee jaar later veroveren het framework *Rails* en in het kielzog de programmeertaal Ruby de wereld van de webprogrammeurs.

Ruby

Het was geen toeval dat Heinemeier Hansson voor Ruby koos. Ruby, ontwikkeld in 1993 door Yukihiro Matsumoto ('Matz'), biedt als programmeertaal unieke mogelijkheden om snel en efficiënt te programmeren, zonder afbreuk te doen aan de leesbaarheid. Ruby is een objectgeoriënteerde, geïnterpreteerde, dynamische taal. Matz nam in Ruby concepten over uit talen als Lisp en Smalltalk. Met name het gebruik van zogenaamde *lambda's* of *closures* (blokken code die via een variabele kunnen worden gerefereerd en uitgevoerd) zal kenners van die talen bekend voorkomen. Ruby wordt geïnterpreteerd, wat zijn weerslag heeft op de performance. De verwachting is echter dat Ruby een soortgelijke ontwikkeling van performanceverbetering zal doormaken als Java – feitelijk ook een geïnterpreteerde taal. Ruby en Rails zijn overigens volledig open source. Wat betekent het dat Ruby een dynamische programmeertaal is? Een Ruby-programma kan zichzelf, uiteraard geïnstrueerd door de programmeur, tijdens het uitvoeren veranderen. Zo kan een enkele instructie leiden tot het toevoegen en uitvoeren van een geheel nieuw stuk programmacode. Dit gebeurt tijdens het uitvoeren van het programma in het geheugen, dus zonder de broncode zelf aan te passen. Daarom is dit iets anders dan codegeneratie, een concept dat als nadeel heeft dat eenmaal gegenereerde code niet straffeloos handmatig kan worden gewijzigd. Het resultaat in Rails, dat veelvuldig gebruikmaakt van dit principe, is dat de webprogrammeur aanzienlijk minder programmacode schrijft om hetzelfde resultaat te bereiken. Dit alleen al betekent dat applicaties niet alleen sneller kunnen worden ontwikkeld, maar ook beter onderhouden.

Rails

De filosofie achter Rails is 'convention over configuration'. De heersende gedachte in applicatieontwikkeling is

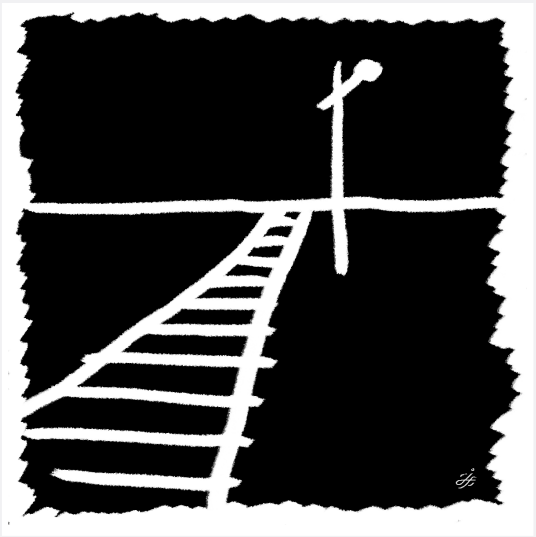
Domain specific languages

Het dynamische karakter maakt Ruby bijzonder goed geschikt voor het creëren van *domain specific languages* (dsl's). Een dsl is een gelegenheidsprogrammeertaal die bedacht wordt voor het oplossen van problemen in een specifiek domein. Bovendien kan een dsl eenvoudig genoeg zijn om te worden gebruikt door mensen met verstand van het probleemdomein maar zonder programmeerkennis. Door een programmeertaal te creëren die lijkt op een natuurlijke taal maar geïnterpreteerd kan worden als een programmeertaal, stel je mensen met de domeinkennis in staat hun kennis in een programma vast te leggen. Zo konden in een recent project testers van een Java-webapplicatie hun geautomatiseerde testcases beschrijven in een simpel ogende taal (zie kader). Vervolgens kon met Ruby en de library Watir Internet Explorer worden aangestuurd om de webapplicatie automatisch te onderwerpen aan de gedefinieerde testcases.

Voorbeeld van een met Ruby gemaakte domain specific language

```
click "Toevoegen"
vul_in "naam", "Jansen"
vul_in "straat", "Hoogstraat"
vul_in "plaats", "Rotterdam"
selecteer "leeftijdsgroep", "35-50 jaar"
click "Bewaren"

vul_in "zoeken", "Jansen"
click "Zoeken"
check "straat", "Hoogstraat"
check "plaats", "Rotterdam"
…
```



dat een applicatie zoveel mogelijk configureerbaar moet zijn. Op die manier kan de functionaliteit van de applicatie worden gewijzigd zonder de programmacode te hoeven aanpassen. Dat laatste vereist immers een programmeur en is daardoor duurder. Dat was althans het idee. Het resultaat, in ieder geval op het gebied van webapplicaties, is dat programmeurs tijdens het ontwikkelen een toenemende hoeveelheid configuratiebestanden moeten schrijven en onderhouden. Rails werkt andersom. Datgene wat hetzelfde is in het merendeel van de webapplicaties, wordt verheven tot standaard (*convention*) en vereist geen werk van de programmeur. Alleen als van de standaard moet worden afgeweken, wordt dat expliciet geprogrammeerd. Het resultaat is dat de programmeur zich vooral kan richten op datgene wat de applicatie moet doen, in plaats van op hoe de applicatie het moet gaan doen. Bovendien is hij veel minder tijd kwijt aan het inrichten en configureren van de ontwikkelomgeving.
Rails biedt, dankzij deze filosofie, nog andere voordelen. Zo wordt het mogelijk om op een prototypingachtige manier te ontwikkelen. Als je zo snel kunt ontwikkelen als met Rails, waarom zou je dan niet samen met een gebruiker achter de pc plaatsnemen en op diens aanwijzingen de grote lijnen van de applicatie ter plekke in elkaar zetten? Dit is overigens ook de werkwijze die wordt beschreven in het boek over Rails *Agile Web Development with Rails*. Als het resultaat niet voldoet, doe je het overnieuw; je hebt er immers geen weken aan hoeven werken, hooguit een paar uur of een paar dagen. Verder sluit het op conventie gebaseerde ontwikkelen natuurlijk mooi aan bij het streven van veel organisaties naar een standaardmanier van applicatieontwikkeling, bijvoorbeeld in een softwarefactory. Ten slotte biedt Rails in alle lagen van de applicatie goede faciliteiten voor *test driven development* (tdd). Tdd houdt in dat consequent vooraf testprogrammatuur wordt geschreven voor alle 'echte' programmacode. Daardoor worden veel bugs al in een vroeg stadium de kop ingedrukt – en niet pas achteraf, als de programmeur waarschijnlijk ook al niet meer weet hoe de code precies werkt.

JRuby

Hoe fraai en snel Rails ook is, de inzetbaarheid is bewust beperkt tot het ontwikkelen van webapplicaties en dan liefst ook nog volgens een bepaald stramien. Daar ligt ook het voordeel van Rails boven omgevingen als Java/J2EE en .Net. In een ideale situatie zou Rails kunnen worden gebruikt voor het zeer snel ontwikkelen van webapplicaties boven op bestaande Java/.Net-omgevingen. Daarom wordt momenteel hard gewerkt aan koppelingen tussen Ruby en Java en tussen Ruby en .Net. Zo is er het project JRuby, waarmee Ruby-programma's via een interface kunnen worden uitgevoerd in de Java Virtual Machine. Dit maakt het mogelijk dat bedrijfslogica wordt geschreven in Java en beschikbaar wordt gemaakt als webservices of Enterprise Java Beans (EJB's) – waarna webapplicaties, geschreven in Rails, deze bedrijfsservices kunnen aanroepen en combineren. Het voordeel van deze integratie tussen Ruby en Java is niet alleen het sneller kunnen ontwikkelen van webapplicaties voor de enterpriseomgeving, het zal ook een stuk eenvoudiger worden om Ruby te introduceren in de it-omgeving van die enterprises. Niet voor niets heeft Sun, eigenaar van Java, onlangs de twee hoofdontwikkelaars van JRuby in dienst genomen. Het uiteindelijke doel is om Ruby-programma's te kunnen 'compileren' naar Java-bytecode, zodat de snelheid van Ruby niet meer onderdoet voor die van Java.

De hype voorbij

De belangstelling voor Ruby en Rails lijkt in de eerste plaats te zijn gedreven door ontwikkelaars. De elegante eenvoud van Rails, waarin je meteen aan de slag kunt met wat de applicatie moet doen, kwam voor veel ontwikkelaars als een verademing. Zodra duidelijk werd welke voordelen dit kon hebben voor applicatieontwikkeling, volgden management en klanten. Architectuurgoeroe Martin Fowler (2006) schreef eerder dit jaar, in een evaluatie van Ruby: "It's still early days yet, but I now have a handful of project experiences to draw on. So far the results are firmly in favor of Ruby. When I ask the question 'do you think you're significantly more productive in Ruby rather than Java/C#,' each time I've got a strong 'yes'. This is enough for me to start saying that for a suitable project, you should give Ruby a spin."

Ook in Nederland begint belangstelling te ontstaan voor het toepassen van Ruby en Rails. Enkele kleinere, in Rails gespecialiseerde bedrijven voeren inmiddels diverse projecten uit bij uiteenlopende klanten zoals ministeries, mediaorganisaties en het mkb. Grotere organisaties, die vaak veel hebben geïnvesteerd in Java/J2EE of .Net, nemen nog een afwachtende houding aan. Zij zullen vooral profiteren van de koppeling tussen Ruby en Java en .Net. Aan de ontwikkelaars zal het niet liggen, getuige de grote belangstelling voor de eerste Ruby en Rails-gebruikersdag die in mei dit jaar werd georganiseerd. De vraag blijft hoe je de wereldwijde hype over Ruby en Rails in korte tijd terugbrengt tot realistische proporties. Het paradoxale antwoord: ga er zelf mee aan de slag!

Literatuur

Fowler, M. (2006). EvaluatingRuby, martinfowler.com/bliki/EvaluatingRuby.html.

Tate, B. (2006). *From Java to Ruby: Things Every Manager Should Know*. Pragmatic Programmers.

Thomas, D. (2005). *Agile Web Development with Rails: A Pragmatic Guide*. Pragmatic Programmers.

Links

ruby.pagina.nl
rubyenrails.nl

Danny Lagrouw
is ict-consultant bij Profict b.v. E-mail: danny.lagrouw@gmail.com.