Bericht uit Java

Java is een moderne programmeertaal. Moderne programmeertalen zijn zo gemaakt dat je je geen zorgen hoeft te maken over allerlei basale en primitieve zaken. In iets oudere talen zoals C++ moest je vooral niet vergeten om gemaakte objecten weer weg te gooien. Memory leaks waren aan de orde van de dag. Zelfs Ms Word kende memory leaks. Java kent Garbage collection. Garbage collection zorgt er voor dat alle gemaakte objecten vanzelf weer weggegooid worden zodra ze niet meer gekend worden door andere objecten. Veilig, simpel en effectief. Memory leaks bestaan niet meer in Java. Het uitgangspunt is dat alles wat de taal en de omgeving voor je kunnen regelen, ook geregeld moet worden.

Wat gebeurt daar binnen toch allemaal?

Sinds enkele jaren is er J2EE. Ook J2EE is gemaakt om het leven van de programmeur simpel te houden. Systemen moeten 'enterprise waardig' zijn. Systemen moeten failsafe zijn, op meerdere servers tegelijk draaien, beveiligd zijn, webinterfaces hebben, door meerdere gebruikers tegelijk gebruikt kunnen worden ... En dat alles zonder ook maar een centje pijn voor de programmeur. Alles waar de programmeur vroeger heel ingewikkelde toeren voor moest uithalen, wordt nu gegenereerd of anderszins vanzelf tevoorschijn getoverd. We kunnen niet meer om J2EE heen.

Maar dan gaat het mis. Het J2EE systeem gaat stuk. Of het systeem performt voor geen meter. Of er kan toch maar één gebruiker tegelijkertijd inloggen. Ellende! En dan? Waar je vroeger heel zeker wist dat je een programmeerfout had begaan, is dat nu niet meer zo zeker. In Java was het toch niet mogelijk om programmeerfouten maken? Handen in het haar! Waar begin je? Wist je al dat er, speciaal voor J2EE programmeren, een flink aantal regeltjes gelden waar je je als programmeur aan hebt te houden? Geen eigen threads, geen singleton pattern, letten op serialisatie, letten op objecten die je gedupliceerd worden zonder dat je het in de gaten hebt. Maar toch, het gaat dus mis. En dan? In veel gevallen weet zelfs de leverancier van de J2EE omgeving niet precies wat er allemaal mis kan gaan.

We halen er een profiler bij. Profilers zijn programma's die in staat zijn om te kijken naar je Java programma om te analyseren of er iets mis gaat, welke stukken code het meest gebruikt worden, welke stukken code de performance het meest bepalen, of er niet te veel objecten gemaakt worden... Kortom, profilers kijken wat er daar binnen in die Java Virtual Machine allemaal gebeurt. Sommige profilers zijn ook in staat om J2EE programma's te analyseren. Maar ja, wat een J2EE server doet is gestandaardiseerd, hoe een J2EE server dat voor elkaar krijgt is niet gespecificeerd. Dat maakt het niet eenvoudig om zo'n profiler te maken, aan te sluiten op de J2EE omgeving en om de resultaten te interpreteren. Geen garanties dat je te zien krijgt wat er precies gebeurt in je programma.

Het blijft altijd dezelfde afweging; zelf programmeren en daarmee weten welke fouten je maakt, of het systeem gedeeltelijk 'laten programmeren' en maar hopen dat het goed gaat. Wat gebeurt daar binnen in die J2EE omgeving toch allemaal?

Daan Kalmeijer is docent consultant bij CIBIT adviseurs | opleiders (e-mail: daan@cibit.nl)