Bericht uit 

Voor Java geldt "Write once, run everywhere". Het ontwikkelen van multi-platformprogramma's zou dus een eitje moeten zijn. De praktijk blijkt echter weerbarstig...

Write Once, Run for the Hills...

Er is een aantal zaken op deze wereld waar software engineers stevige hoofdpijn van krijgen. Eentje daarvan is het ontwikkelen van multi-platformprogramma's. Vrijwel ongeacht de taal waarmee wordt gewerkt blijkt dit allerlei problemen met zich mee te brengen. En dat zijn dan ook nog vervelende problemen, die vrijwel niemand leuk vindt om op te lossen. Met Java zou eindelijk het voordeel eens definitief aan de kant van de zwoegende "porting engineers" komen te liggen. Niet meer moeizaam sourcecode aanpassen voor het ene of het andere platform. Zelfs niet eens meer opnieuw vertalen! "Write Once, Run Everywhere" werd het devies. De theorie blijkt echter helaas nog wel eens anders dan de praktijk. Zelfs Java applets blijken met de regelmaat van de klok niet aan de praat te krijgen op andere platformen dan waar ze op zijn ontwikkeld. Waar ligt dit nu aan?

Allereerst kent Java in zijn jonge bestaan al een relatief enorm aantal verschillende versies en patch levels. Er zijn maar liefst drie versies van de Java-standaard (1.0, 1.1 en 2.0), en elke versie heeft ook nog wel een aantal subversies. Van Java 1.1 zijn maar liefst zeven patch levels door Sun uitgebracht! Een Java-applicatie die op de ene release werkt, kan op het andere wel eens heilloos ten onder gaan. Daarnaast heeft menige Java-implementator de euvele moed om uitbreidingen op, en aanpassingen aan de heilige Java-standaard in zijn Java-compiler en/of virtuele machine aan te brengen. Gebruik daarvan is natuurlijk funest voor de portabiliteit van een applicatie. Toegegeven, dit is natuurlijk niet de schuld van de Java-taal, maar van de Java-programmeur. Sneu is echter wel dat de onschuldige Java-programmeur bij gebruikmaking van de "wizards" van zijn interactieve ontwikkelomgeving zonder het te weten dit soort incompatible Java-eigenschappen kan gebruiken. Verder komt het tot overmaat van ramp nog geregeld voor dat Java's virtuele machines bugs bevatten. Met name in gecompliceerde componenten zoals de "Just In Time"-compilers komt dit nog geregeld voor. Java-code die op het ene platform goed draait kan door dergelijke bugs op een ander platform krijsend en gierend tot stilstand komen. Zonde, want het uitschakelen van de JIT-compiler heeft meestal tot gevolg dat de snelheid van het programma aanzienlijk afneemt. Maar, zoals een oud Chinees spreekwoord al zegt: "Je kunt niet alles hebben, waar zou je het laten?". De meeste gemene categorie verschillen tussen Java-implementaties zit hem echter nog wel in dingen die niet nauwkeurig in de standaard zijn gespecificeerd, maar die wel gevolgen kunnen hebben voor het correct werken van een applicatie. Een voorbeeld: Ik schreef eens een Java-programma voor het tonen van enquête-resultaten. In het scherm stonden grafiekjes, met daarbij labels. Als de gebruiker met de muis over die labels bewoog, lichtten de labels rood op en verscheen onder in het window een uitgebreidere beschrijving van het label. Conform de moderne terminologie doopte ik die labels "Active Labels". Om dit gedrag af te dwingen reageerde ik natuurlijk op de "MouseEnter" en "MouseLeave" events die zo'n label ontving. Wat bleek nu echter: als twee ActiveLabel's vlak tegen elkaar aan lagen en de gebruiker de muis bewoog van het ene label naar het andere label dan was de volgorde waarin de events binnenkwamen bij verschillende Java Virtual Machines anders! Bij de ene JVM kreeg ik eerst de "Leave" van het ene label, en daarna de "Enter" van het tweede; bij een andere JVM was dit weer precies andersom: eerst de "Enter" van het nieuwe label, en daarna pas de"Leave" van het oude! Dit soort verschillen had bijvoorbeeld tot gevolg dat het examenprogramma waarmee ik Sun Certified Java Developer ben geworden prima liep onder de Sun JVM, maar er totaal niet uitzag en zich anders gedroeg op een Microsoft JVM! En zo zie je maar weer: 'Compatible is not Identical!'

Jos Visser

is werkzaam als erkend talent bij Open Solution Providers en is bereikbaar via josv@osp.nl