Back to the Future

Erinnern Sie sich noch an den 23. Januar 1996? An diesem Tag ist Mia Farrow in der Late Night Show von David Letterman aufgetreten und am gleichen Tag Kevin Eubanks in der Sendung von Jay Leno. Der australische Programmierer John Mackin verstarb tragisch mit lediglich 36 Jahren an einer Lungenembolie, als gleichzeitig Justin Bieber seinen zweiten Geburtstag feiern konnte. Die Welt beging den siebten Todestag von Salvator Dali und den zehnten von Joseph Beuys. Während diese Ereignisse nur für wenige von uns allzu große Bedeutung haben, dürfte sich dies für ein weiteres historisches Ereignis gänzlich anders darstellen. Genau am besagten Tag erblickte nämlich die Version 1.0 des Java Development Kit das Licht der Welt, ohne die es die vorliegende Zeitschrift gar nicht geben würde.

Zur damaligen Zeit stellte sich die Java-Plattform noch sehr übersichtlich dar und ließ sich problemlos in wenigen Tagen erlernen, zumindest was ihre Grundzüge betrifft – David Flanagans Buch „Java in a Nutshell" umfasste dementsprechend nur eine vergleichsweise bescheidene Zahl an Seiten. Ein heutiger Chefredakteur, der es bevorzugt, an dieser Stelle anonym zu bleiben, hielt die neue Technologie daher lediglich für ein nettes Spielzeug, was sich für ihn am deutlichsten in der verspielten Applet-Technologie manifestierte. So kann man sich also irren! Allerdings sind Prognosen, speziell hinsichtlich der Zukunft, immer von diffiziler Natur, wie wir wissen.

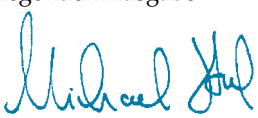
Seitdem hat die Java-Technologie einen unglaublichen Siegeszug angetreten und der weitgehenden C/C++-Monokultur aus den Neunziger Jahren eine exzellente Alternative gegenüber gestellt. Auch Microsoft profitierte letztendlich von der Java-Technologie, da das Unternehmen aus Redmond sich dank Java gezwungen sah, die etwas komplexe COM-Technologie durch die schlankere Kombination aus .NET und Java-Cousin C# zu revolutionieren.

Java ist inzwischen auf dem ganzen Planeten und vielleicht sogar darüber hinaus verbreitet. Somit könnte man annehmen, es herrsche Friede, Freude, Eierkuchen. Doch weit gefehlt! Die Java-Technologie hat inzwischen mehr als sechzehn Jahre auf dem Buckel, weshalb sich die Frage nach der Zukunft von Java unweigerlich stellen muss. Und manche (selbst ernannten) Experten sehen kein Licht am Ende des Tunnels, sondern prophezeien stattdessen der Java-Technologie ein ähnliches Schicksal, wie es Cobol in der IT-Bronzezeit widerfuhr. Tatsächlich zeigen sich bei näherer Betrachtung einige Risse im Java-Zeit-Raum-Kontinuum. Da wären die vielen missglückten Versuche zu nennen, eine vernünftige Technologie für Präsentationslogik zu kreieren, was zuletzt in der JavaFX-Strategie mündete. Oder die selbst verschuldete Komplexität und Mannigfaltigkeit in Bezug auf Java-Enterprise-Technologien, die das Aufblühen zahlreicher Open-Source-Frameworks provoziert hat – exemplarisch sei an dieser Stelle Spring genannt. Das alles erinnert an den guten und weisen Spruch von Betriebssystemguru Andrew Tanenbaum „das Gute an Standards ist, dass es so viele davon zur Auswahl gibt." Im stark wachsenden Marktsegment mobiler Endgeräte vermag Java bislang „nur" auf der Android-Plattform Erfolg aufzuweisen, weshalb sich Oracles Teilerfolg im Rechtsstreit mit Google als Pyrrhussieg erweisen könnte. Die Vormachtstellung von Oracle und IBM, was die Standardisierung von Java betrifft, sehen nicht wenige eher mit weinendem als mit lachendem Auge. Und nicht zuletzt sehnen sich viele Java-Jünger nach dem Propheten OpenJDK, der sich auf der anderen Seite aber weniger offen gibt, als viele hoffen.

Warum sollten wir in Anbetracht dieser Fakten trotzdem auch nur einen Cent auf Java setzen? Zu den schwächeren, wenngleich richtigen Argumenten gehört, dass es zu Java keine wirklich ernsthafte Alternative gibt. So erweist sich in Entwicklerhabitaten die Java-Plattform eindeutig als führend. Das ist nicht nur einem Bauchgefühl geschuldet, sondern lässt sich sehr gut belegen. Zum einen gehören Java-Kenntnisse in Stellenausschreibungen zu den wichtigsten Anforderungen und zum anderen setzen Universitäten sehr stark auf Java als Lehrsprache. Die weltweit verfügbare Java-Codebasis und der damit verbundene Markt an Werkzeugen dürften schon allein wegen ihres schieren Umfangs ein starkes Argument darstellen. Auffallend ist des Weiteren die Bedeutung der Java-Plattform bei innovativen Ansätzen wie dem Cloud Computing oder Big Data. Sogar Microsoft bemüht sich verstärkt um Java-Entwickler für Azure-basierte Cloud-Lösungen. Allen Unkenrufen zum Trotz repräsentiert die JVM die wichtigste Basis polyglotter Programmierung; an dieser Stelle seien die repräsentativen Vertreter ihres Genres, Clojure, Scala, Jython, JRuby und Groovy, genannt. Als letztendlich stärkstes Argument überzeugt die Zukunftssicherheit von Java. Kommende Versionen integrieren unter anderem Sprachmerkmale für effizienteres Programmieren wie Closures, die bessere Unterstützung großer Systeme (Stichwort Jigsaw) sowie zahlreiche Effizienzsteigerungen. Damit sieht sich die Plattform einer kontinuierlichen Evolution gegenüber. Wie heißt es in der Autowerbung so schön, „Vorsprung durch Technik".

Natürlich ist es wichtig und richtig, die jüngsten Entwicklungen im Java-Ökosystem kritisch zu hinterfragen. Konstruktives Feedback sowie Wettbewerb fördern die Innovation. Es wirkt aber übertrieben und zuweilen eher marktpolitisch motiviert, die Totenglocken in der Ferne vernehmen zu wollen. Wie ausführlich erläutert, stellt sich die Realität bei genauerer Betrachtung anders dar und lässt auf eine lange und erfolgreiche Zukunft von Java hoffen. Wie für vieles andere gilt also auch für Java die alte Binsenweisheit „Totgeglaubte leben länger".

In diesem Sinne viel Spaß mit der vorliegenden Ausgabe



Ihr Prof. Dr. Michael Stal