Java auf dem absteigenden Ast?

TIOBE Index July 2015

von Thomas Kestler, 13.07.2015

Einleitung

Jährlich veröffentlicht monatlich eine Statistik der am häufigsten verwendeten Programmiersprachen. Grundlage sind statistische Daten von Suchmaschinen (Google, Bing, etc.). Der aktuelle Index zeigt, wie schon seit längerem, eine sinkende Tendenz für Java. Was ist los mit Java? Stirbt es aus?

Die aktuelle Grafik

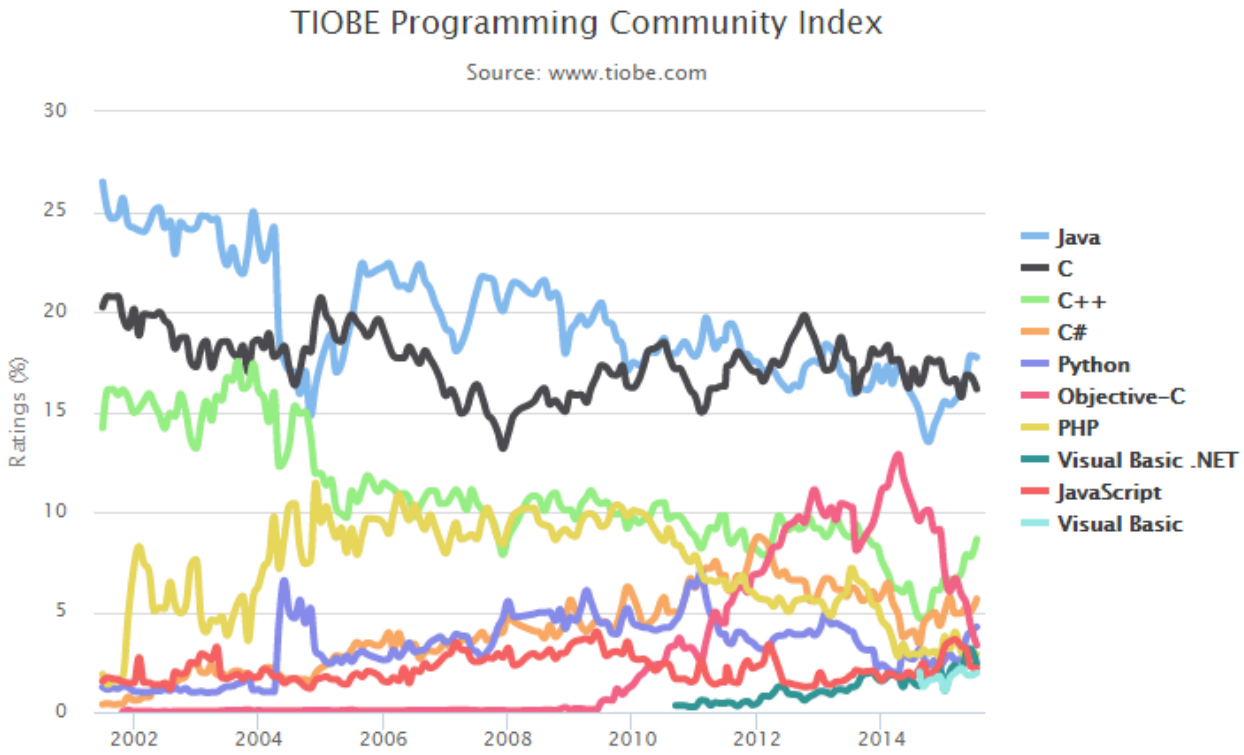

Abbildung 1: Quelle: TIOBE - http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Man sieht eine Abwärtstendenz für Java, Stabilität für C und ansonsten reichlich Dynamik. C++ nimmt wieder Fahrt auf, was an C++11 liegen könnte, was eine sehr gute Basis für Entwickler bietet. Objective-C fährt Achterbahn, es zeigte sich wohl, dass die App-Einsteiger schnell wieder das Interesse verloren.

C++11 das Allheilmittel?

Nein, so etwas gibt es nicht. C++ ist und bleibt eine sehr leistungsfähige aber auch sehr komplexe Programmiersprache. Zwar wurde mit C++11 vieles verbessert und wer jetzt neu einsteigt, tut sich viel leichter als jemand der dies vor Jahren tun musste. Trotzdem bleibe ich bei meiner Aussage, dass nur 1-2 von 100 C++-Programmierer die Sprache wirklich verstanden haben. Immerhin kann man mit C++ 11 nun aber auch bei nicht vollem Durchblick trotzdem sicherer arbeiten als je zuvor, wenn man sich an die neuen Patterns hält.

C im Embedded Bereich

Ein Grund für die starke Verbreitung von C ist die zunehmende Nutzung von Embedded Devices in der Industrie, insbesondere in Automobilen. Denken wir nur an die vielen Assistenzsysteme bis hin zum autonomen Fahren. All diese Systeme werden entweder direkt oder indirekt (z. B. Code-Generator aus SimuLink-Model) in C codiert und kompiliert. Daher wird C auch noch lange Zeit weit verbreitet sein, auch wenn zunehmend C++ Einzug in die Embedded-Welt halten wird, da sich hier Vorteile der templatebasierten Programmierung ergeben.

Ist Java tot?

Oracle hat die Attraktivität von Java zumindest nicht gesteigert. Eher lieblos erfolgt die Weiterentwicklung, weil man eigentlich kein Geld damit verdient. Tot ist Java aber deshalb noch lange nicht, weil sehr viele Unternehmen vor Jahren auf Java und JEE gesetzt haben. Viele mussten schmerzlich erleben, das JEE in den frühen Versionen ziemlich verkopft und umständlich war und bis heute tendiere ich lieber zu leichtgewichtigen Lösungen (Spring, Tomcat, …). Vor allem die Inkompatibilität zwischen den Applikationsservern ist ein Armutszeugnis für JEE.

JSP und viel mehr noch JSF sind tote Technologien und viele Programmierer mühen sich tagtäglich damit herum. Viel Aufwand für wenig Ertrag. Das liegt einfach am falschen Programmierparadigma, nämlich serverseitig HTML-Code zu generieren und diesen dann auch noch mit reichlich Javascript zu verzieren. Abgesehen davon, dass die Browser das CSS-Boxmodel dann eben doch wieder unterschiedlich umsetzen. Insofern liegen für mich auch Ansätze wie Apache Wicket nicht auf der Hand.

Python – coole Skriptsprache, mehr auch nicht

Python ist cool, keine Frage. Man kommt schnell zurecht und es gibt mittlerweile eine breite Unterstützung. Ganz so mächtig wie bei Perl ist die Basis zwar noch nicht, aber nah dran. Und so eignet sich Python auch sehr gut für Admin-Skripte, DevOps, etc. Es gibt wirklich coole Python-Module für statistische und wissenschaftliche Auswertungen und nicht wenige Astronomen nutzen dies täglich mit enormen Datenmengen. Insofern hat Python seinen berechtigen Platz im Index, aber für Anwendungsentwicklung würde ich es nicht sehen.

Hidden Champion: JavaScript

Nur auf Platz 9 kam JavaScript. Schade eigentlich, denn JavaScript ist eine vollwertige Programmiersprache und das Web beweist täglich die Leistungsfähigkeit. Mit dem passenden Framework lassen sich sehr effizient Anwendungen implementieren, die ein klares Programmierparadigma haben: hol' die Daten vom Backend und zeige sie mit Widgets und Controls an. Fast wie in einer Java Swing Anwendung (oder Qt oder …). Ich kann es nur immer wieder wärmstens empfehlen: Qooxdoo.

JSON/REST liefern eine einfach Schnittstelle zum Backend. Letztlich kann man mit Frameworks wie node.js sogar das Backend komplett in JavaScript implementieren. Wer das einmal ausprobiert hat ist begeistert von kurzen Turn-Around-Zeiten (F5-Refresh im Browser), vor allem wenn er quälend lange Minuten beim Deployment auf einen Applikationsserver gewohnt war.

Zusammenfassung

Java bleibt noch eine Zeitlang eine der wichtigen Programmiersprachen. Die abnehmende Beliebtheit dürfte aber vor allem an der Ernüchtung liegen über a) die Produktpolitik von Oracle und b) die Erkenntnis wie aufwändig Entwicklung im JEE/JSF-Universum ist.

Dabei kann man mit Java gerade im Backend sehr effizient entwickeln, wenn man den JEE-Overhead weg lässt und Frameworks wie Spring einsetzt. Es kommt eben immer darauf an, wie man die Werkzeuge einsetzt.

Für Frontend-Entwicklung setze ich entweder auf native Clients oder wenn Web-Client dann pure JavaScript. Die klare Trennung der Zuständigkeiten (was macht das Frontend, was das Backend) macht das Leben viel einfacher und Entwicklungszeiten kürzer.

Weblinks

TIOBE Index July 2015 - http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

node.js - https://nodejs.org/

Qooxdoo - http://qooxdoo.org/