Wir wollen es wissen: **C# oder Java?**

Welcher Programmiersprache gehört die Zukunft? Zwei IT-Experten steigen in den Ring zum Schlagabtausch C# vs. Java. In der linken Ecke: Gustav Paulus, Senior C# Fullstack DevOp bei InfraSoft Datenservice GmbH als »C#-Herausforderer«. In der Ecke rechts ist InfraSoft-Kollege und »Java-Verteidiger« Erich Stadler, Senior Java Fullstack DevOp.

Das Duell

Gustav Paulus – C#-Herausforderer	Erich Stadler – Java-Verteidiger
Als 2003 die erste Version des .Net-Framework released wurde, konnte niemand ahnen, welchen Grundstein Microsoft damit legen würde. Heute ist es eine der modernsten Möglichkeiten für alle Gelegenheiten. Kann dein Dinosaurier mit dieser Entwicklung mithalten?	Java hält sich seit mehr als 26 Jahren aktiv am Markt! Und ist trotz seines Alters die zweitbeliebteste Programmiersprache (die beliebteste ist übrigens nicht C# sondern Python). Mit modernen Frameworks wie zum Beispiel Spring, beschränkt sich die Implementierung auf die wesentlichen Teile. Stichwort: Convention over Configuration!
Das ist quasi in ASP.Net inkludiert. Klassen, die im Controller-Folder sind, werden automatisch zu Endpunkten für den Host und die Klassen im View-Folder werden per Convention gefunden. Auch Domainlogic mit Validierungsregeln sind mittlerweile Standard. Einfacher geht es nicht.	Doch – Java benötigt mit Spring Data JPA nur ein Interface, in dem Methodennamen so geschrieben werden, wie man sie braucht. Die Implementierung findet im Hintergrund statt und zum Zeitpunkt des Starts wird geprüft, ob der Methodenname auch dem Datenbank-Modell entspricht. Das beschleunigt die Entwicklung ungemein. Und wenn man keine SQL-Datenbank verwenden möchte, ist die Verwendung einer NoSQL-Datenbank sehr einfach. Gibt es in C# fertige Frameworks?

Foto: Raphaela Ch. Müller

Gustav Paulus – C#-Herausforderer	Erich Stadler – Java-Verteidiger
In .Net gibt es unzählige Frameworks und Datenbanken oder Alternativen. Und für die meisten geteilte Interfaces. Daher ist ein schneller Wechsel der Implementierung oft ohne Codechange möglich. Diese Frameworks sind im Gegensatz zu Java klein und fokussiert. Man stellt sich genau die Pakete zusammen, die man braucht. Was macht man in Java, wenn man das Framework wechseln möchte?	Die zwei Frameworks Spring und Quarkus sind leider nur teilweise bis gar nicht kompatibel. Aber auch Application-Server hatten bisweilen ihre Inkompatibilitäten. Fertige Funktionalitäten ohne Code schreiben zu müssen, gibt's in den beiden Frameworks aber schon lange. Aber zu etwas anderem: wie steht es denn um den Ressourcen-Verbrauch bei C#? Speicher, CPU?
Auf die Frage habe ich mich gefreut! Mit jedem neuen Release von .Net wird das Framework nicht nur mit Features erweitert, sondern auch effizienter. Eine Web-API-Applikation mit mehreren Domains, Datenbankanbindungen und Eventbus benötigt weniger als 250 MiB Hauptspeicher und eine zeitgemäße Verwendung von asynchroner Programmierung entblockt Threads auf Methodenebene. Das kann Java nicht bieten, oder?	Java kommt mit wenig Speicher aus, dafür ist der CPU-Verbrauch leider (noch) höher. Pro geöffnetem Netzwerk-Kanal bleibt ein Thread aktiv. Somit sind bei laufendem Java-Server viele Threads offen. Aber Spring Reactive bietet hier schon wesentliche Verbesserungen. Und, weil wir bei Servern sind: Java kann sowohl mit Spring als auch mit Quarkus ausgezeichnete REST-Schnittstellen zur Verfügung stellen, die mittels Open API sehr gut dokumentierbar sind. Kann C# das auch?
Hier helfen NuGet-Pakete, um REST-Schnittstellen zu dokumentieren. Besonders gefällt mir hier die einfache Testbarkeit: Man modifiziert den implementierten Webserver, ändert die Datenbanken auf InMemory und schon kann man die Applikation bequem testen. Unit-Tests sind so performant, dass sie im VisualStudio ausgeführt werden und so direkt Feedback geben, während man den Code schreibt. Die Specflow-Integration ermöglicht zudem Behaviour-driven-Tests. Um das Paket abzurunden, gibt's Coded-UI-Tests, ein Framework, das hilft, den kompletten Integration-Scope zu testen. Automatisches Testing, kann das Java auch?	Selbstverständlich! Das Spring-Framework stellt zum Beispiel eine Test-Plattform zur Verfügung, mit deren Hilfe REST-APIs hervorragend automatisiert getestet werden können. Dabei wird das JUnit-Framework als Grundlage benutzt und durch die Verwendung einer H2 In-Memory Datenbank ergänzt. Durch die Kombination verschiedener Test-Techniken kann somit eine aussagekräftige Code-Coverage erreicht werden. Und statische Code-Analysen werden durch Build-Frameworks einfach eingebunden.
So wie in C# auch. Mit der Einführung des Roslyn-Compilers 2015 stellt der Compiler APIs zur Verfügung – als Folge gibt es nicht nur eine Menge Codeanalyse-Pakete, es ist sogar sehr einfach, eigene Codeanalysen zu schreiben, um etwa Enterprise-Rules zu erzwingen. Außerdem wirkt die DotNet-Truppe aktiv an OpenTelemetry mit: Einen User-Call vom Frontend bis zur Datenbank über eine distributed Trace-ID zu verfolgen, die entsprechenden Logs dieses Calls aus dem Elastic abzufragen, weil Grafana die Entwickler alarmiert, nachdem Prometheus-Metriken eine Schwelle überschreiten, *bevor* die User über langsam reagierende Software klagen, das entlastet die Problemsuche enorm.	Das ist heutzutage ja schon fast Standard und funktioniert selbstverständlich auch in Java. Tracing findet zum Beipiel über Spring Cloud statt, was dann auch mittels Jaeger oder Zipkin ausgewertet werden kann. Metriken werden in Spring über eine Prometheus-Schnittstelle zur Verfügung gestellt, die dann auch über Grafana ausgewertet werden können. Übrigens: man kann in Spring aus mehreren Server-Plattformen wählen. Bei C# ist man wohl immer noch auf den IIS angewiesen, nicht wahr?
Schon lange nicht mehr! Es funktioniert, ist aber nicht mehr notwendig. Mit Kestrel ist jede Applikation selbst gehostet und läuft überall, wo .Net (Core) läuft, also auch auf Linux, iOS oder Android. Ohne Änderung des Frameworks kann man aber auch gleich in die Cloud deployen. Apropos deployen – Du hast vorhin Build-Systeme erwähnt. Was können die denn?	Maven und Gradle stellen in der Java-Landschaft zwei etablierte Build-Systeme dar. Mit beiden ist man in der Lage, Java-Applikationen in Docker-Images zu verpacken, damit diese in Kubernetes-Clustern verwendet werden können. Und beide Build-Systeme können durch Build-Pipelines verwendet werden. Aber, nachdem wir jetzt so intensiv über die Unterschiede debattiert haben, lass uns doch mal festhalten, was Java und C# in Bezug auf die Applikationsentwicklung können (siehe Tabelle unten).

Funktion	C#	Java
REST-API bzw. SOAP-API, GraphQL	👍	👍
Anbindung an unterschiedliche SQL- und NoSQL-Datenbanken	👍	👍
CI/CD	👍	👍
Verwendung von Open Source Libraries	👍	👍
Automatisierte Tests & Code Coverage	👍	👍
Logging, Metriken & Tracing	👍	👍
Security mittels openID-connect	👍	👍
Cloud-native fähige Apps erstellen	👍	👍
Healthchecks, Scalability	👍	👍

Fazit des Schlagabtausches

Innerhalb von Unternehmen können unterschiedliche Sprachen verwendet werden. Auf die Interoperabilität hat es keinen Einfluss und auch die Verwendung gemeinsamer Datenbanken ist möglich. Es gibt demnach kein »ausschließlich, entweder, oder«: Mittlerweile ist Programmierung keine Frage der Sprache allein, mit der richtigen Technik arbeiten moderne Frameworks problemlos zusammen.