Java 8 vs. Scala

Einschätzung und Ratschläge für eine Technologie-Entscheidung
(Stand: Dezember 2014)

.consulting .solutions .partnership

.msg

Ralf S. Engelschall, msg Applied Technology Research (XT)

Ein paar Fakten

- Sowohl Java als auch Scala sind Programmiersprachen, die beide auf Bytecode zur Ausführung auf der Java Virtual Machine (JVM) kompiliert werden.
- Die beiden Sprachen sind auf der JVM interoperabel, d.h. man kann von Scala (Anwendungen) auf Java (Bibliotheken) zugreifen und (mit gewissen Einschränkungen) auch umgekehrt.
- Scala nutzt somit das gleiche gesamte „Ökosystem“ von Java, einschließlich Laufzeitumgebung (JVM), Entwicklungsumgebungen (z.B. Eclipse), Build-Tools (z.B. Gradle), Continuous Integration Systems (z.B. Jenkins), etc.

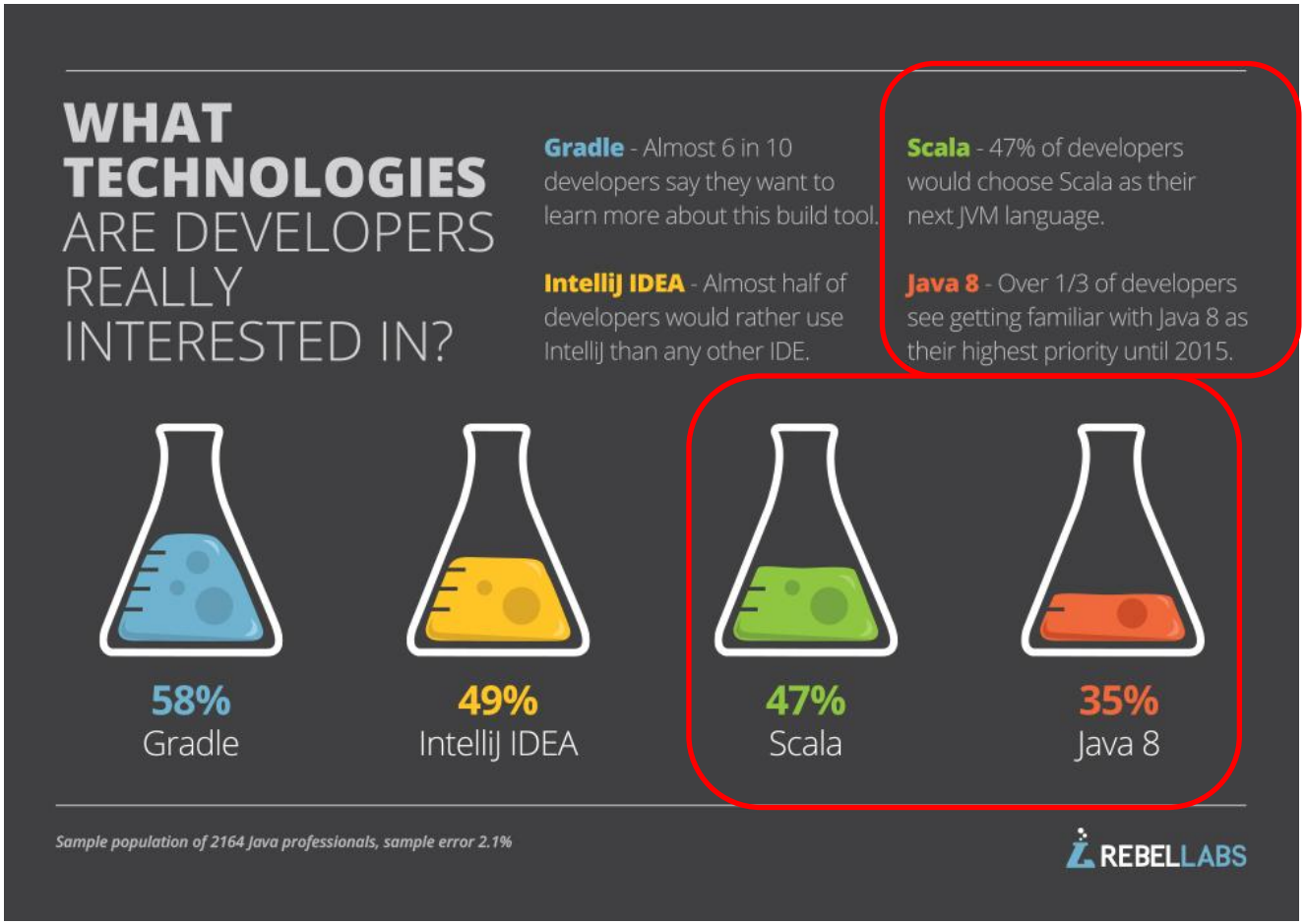
- Java wurde 1995 in USA bei Sun von James Josling entwickelt und gehört heute ORACLE. Fast alle Teile (Compiler, Standard-Bibliothek, Laufzeitumgebung etc) sind als Open Source Software verfügbar.
- Scala wurde 2003 in der Schweiz beim EPFL von Prof. Martin Odersky entwickelt. Alle Teile (Compiler, Standard-Bibliothek, IDE Plugin, etc) sind als Open Source Software verfügbar.

Ein kleiner Vergleich

- Der Fokus von Java war ursprünglich das Paradigma Objekt-Orientierung (OO) gepaart mit Portabilität („runs everywhere“), hat sich aber im Laufe der Jahre aber auf besonders die Entwicklung von großen Anwendungen im OO Paradigma verlagert. Die gefühlte Komplexität kommt bei Java vorallem durch Frameworks.
- Der Fokus von Scala war und ist die Entwicklung großer Anwendungen unter Einsatz der Paradigma-Kombination von Objekt-Orientierung (OO) und Funktionaler Programmierung (FP). Die gefühlte Komplexität kommt bei Scala vorallem durch die Sprache.

- Java ist deutlich weiter verbreitet und hat eine viel größere Community als Scala.
  (d.h. Vorteil Java: leichteres Recruiting und höherer Support)
- Java benötigt für gleiche Funktionalität etwa 2-3 Mal mehr(!) Code als Scala, Scala hat also eine viel höhere Ausdrucksstärke.
  (d.h. Vorteil Scala: höhere Entwicklerproduktivität und leichtere Wartbarkeit)
- Java hat eine eher geringe, Scala eine eher hohe Einstiegshürde für Entwickler.
  (d.h. Vorteil Java: weniger Weiterbildungsnotwendigkeit)
- Java kann fast jeder Entwickler, Scala nicht. Dennoch würden viele gerne in Scala programmieren.
  (d.h. Scala-Projekte benötigen motivierte oder zumindest genügend seniore Entwickler)

Schnappschuss: Entwickler bevorzugen Scala



ZeroTurnAround: „Java Tools and Technologies Landscape for 2014” (Sommer 2014)
http://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-for-2014/

Schnappschuss: Manager bevorzugen Java 8

- Wenn man von Java 8 maximal profitieren will, benötigen die Entwickler gewisse Weiterbildungsmaßnahmen. Die Lernkurve ist dennoch flach und die Aufwände sind überschaubar.
- Der „Upgrade Path“ von Java 5/6/7 zu Java 8 ist gegeben, sowohl technologisch als auch beim Know-How der Entwicklern. Der Schwenk von Java 5/6/7 zu Scala kann dagegen aufgrund neuer Programmierparadigmen (z.B. Funktionale Programmierung) für Entwickler durchaus heftig sein.
- Die Werkzeugunterstützung für Java 8 ist bereits hervorragend und übertrifft die von Scala.
- Auch jüngere Entwickler (Job-Level 1-3) haben bereits Java-Know-How und sind daher sehr schnell produktiv. Scala bedingt entweder motivierte seniore Entwickler (Job-Level 4-5) oder zumindest geeignete Weiterbildungsmaßnahmen.

Transparente Einschätzung über gewichtete Entscheidungsmatrix

- Java 5/6/7 und auch Java 8 sind Scala organisatorisch überlegen.
- Java 8 ist Java 5/6/7 technologisch deutlich überlegen, Java 8 ist Scala technologisch teilweise überlegen.
- Scala ist sowohl Java 8 als auch Java 5/6/7 sprachlich deutlich überlegen.
- Insgesamt ist Java 8 überlegen, aber dicht gefolgt von Scala und mit großem Abstand zu Java 5/6/7.

Gewichtete Entscheidungsmatrix		Java 5/6/7	Java 8	Scala
	Punkte:	25,0	53,0	50,0
Organisatorisch	Punkte:	6,0	6,0	0,0
Projektrisiken sind maximal minimiert	2,0	2	1	-2
Kein Umdenken bei den Entwicklern notwendig (Programmierparadigmen)	1,0	2	1	-2
Existierendes Know-How bei den Entwicklern weiterverwenden	1,0	2	1	-2
Initiale Weiterbildungsaufwände sind minimiert	1,0	2	1	-2
Initiale Einarbeitungsaufwände sind minimiert	1,0	2	1	-2
Hohe Produktivität der Entwickler	2,0	0	1	2
Hohe Motivation für Entwickler	2,0	-1	0	2
Den Kunden und Mitarbeitern gegenüber Innovationskraft beweisen können	2,0	-2	-1	2
Technologisch	Punkte:	11,0	36,0	27,0
Zukunftssicherheit der Technologie	4,0	-2	2	1
Hohe Nutzung der Technologie-Möglichkeiten	2,0	-1	1	2
Gute Werkzeug-Unterstützung (Integrated Developer Environment)	2,0	2	2	0
Gute Werkzeug-Unterstützung (Build Automation Tools)	2,0	2	2	1
Unterstützung der existierenden Technologien in der Scala-Welt	1,0	-1	0	2
Unterstützung der existierenden Technologien in der Java-Welt	4,0	2	2	2
Compile-Time Performance	1,0	2	2	-1
Laufzeit-Performance	4,0	1	2	2
Sprachlich	Punkte:	8,0	11,0	23,0
Ausdrucksstärke der Programmiersprache (für Produktivität und Wartbarkeit)	2,0	-2	-1	2
Flexibilität der Programmiersprache (z.B. für "Internal DSLs", etc)	1,0	-2	-2	2
Skalierbarkeit der Programmiersprache (von kleinen bis großen Anwendungen)	2,0	1	1	2
Unterstützung des Programmierparadigmas Objekt-Orientierung (OO)	2,0	2	2	2
Unterstützung des Programmierparadigmas Funktionale Programmierung (FP)	1,0	-2	-1	2
Leichtigkeit der Entwicklung von Anwendungs-Code	4,0	2	2	2
Leichtigkeit der Entwicklung von Bibliotheks-Code	1,0	2	2	-1

Transparente Einschätzung über gewichtete Entscheidungsmatrix

Gewichtete Entscheidungsmatrix		Java 5/6/7	Java 8	Scala
	Punkte:	25,0	53,0	50,0
Organisatorisch	Punkte:	6,0	6,0	0,0
Projektrisiken sind maximal minimiert	2,0	2	1	-2
Kein Umdenken bei den Entwicklern notwendig (Programmierparadigmen)	1,0	2	1	-2
Existierendes Know-How bei den Entwicklern weiterverwenden	1,0	2	1	-2
Initiale Weiterbildungsaufwände sind minimiert	1,0	2	1	-2
Initiale Einarbeitungsaufwände sind minimiert	1,0	2	1	-2
Hohe Produktivität der Entwickler	2,0	0	1	2
Hohe Motivation für Entwickler	2,0	-1	0	2
Den Kunden und Mitarbeitern gegenüber Innovationskraft beweisen können	2,0	-2	-1	2
Technologisch	Punkte:	11,0	36,0	27,0
Zukunftssicherheit der Technologie	4,0	-2	2	1
Hohe Nutzung der Technologie-Möglichkeiten	2,0	-1	1	2
Gute Werkzeug-Unterstützung (Integrated Developer Environment)	2,0	2	2	0
Gute Werkzeug-Unterstützung (Build Automation Tools)	2,0	2	2	1
Unterstützung der existierenden Technologien in der Scala-Welt	1,0	-1	0	2
Unterstützung der existierenden Technologien in der Java-Welt	4,0	2	2	2
Compile-Time Performance	1,0	2	2	-1
Laufzeit-Performance	4,0	1	2	2
Sprachlich	Punkte:	8,0	11,0	23,0
Ausdrucksstärke der Programmiersprache (für Produktivität und Wartbarkeit)	2,0	-2	-1	2
Flexibilität der Programmiersprache (z.B. für "Internal DSLs", etc)	1,0	-2	-2	2
Skalierbarkeit der Programmiersprache (von kleinen bis großen Anwendungen)	2,0	1	1	2
Unterstützung des Programmierparadigmas Objekt-Orientierung (OO)	2,0	2	2	2
Unterstützung des Programmierparadigmas Funktionale Programmierung (FP)	1,0	-2	-1	2
Leichtigkeit der Entwicklung von Anwendungs-Code	4,0	2	2	2
Leichtigkeit der Entwicklung von Bibliotheks-Code	1,0	2	2	-1

Unser Ratschlag allgemein

Java 8 im Projektgeschäft:
Uneingeschränkt empfehlenswert!

Gegenüber den Kundenvorgaben durchaus dafür kämpfen, daß Java 8 statt Java 5/6/7 genutzt werden kann, um von den Vorteilen von Java 8 profitieren zu können (Vorteile überwiegen die notwendige Überzeugungskraft).

Java 8 im Produktgeschäft:
Uneingeschränkt empfehlenswert!

Auf keinen Fall mehr Java 5/6/7, sondern sofort und ausschließlich Java 8 verwenden, aber Entwickler vorher weiterbilden, um von Java 8 sinnvoll zu profitieren. (Vorteile existieren, müssen aber leider noch teilweise aktiviert werden)

Scala im Projektgeschäft:
Nur sehr eingeschränkt empfehlenswert!

Wenn der Kunde selbst Scala bevorzugt, Scala einsetzen, aber auf Risiken hinweisen. Ansonsten nicht für Scala kämpfen (Organisatorische Handycaps überwiegen noch zu stark und können in diesem Kontext weniger leicht ausgeglichen werden)

Scala im Produktgeschäft:
Leicht eingeschränkt empfehlenswert!

Wenn wir den Weiterbildungsaufwand, den Einarbeitungsaufwand und die Einschränkungen bei der Werkzeugunterstützung akzeptieren können, kann Scala durchaus als sehr innovative Alternative herangezogen werden. (Organisatorischen Handycaps müssen aber vorher ausgeglichen werden)

Fazit für unser Produktgeschäft

Wir sollten Java 8 nutzen…

- **RISIKEN**
  Wenn wir das Risiko bei der Entwicklung minimieren müssen, da wir straffe Zeitvorgaben haben und bereits erfahrende Java 5/6/7 Entwickler haben.
- **AUFWÄNDE**
  Wenn wir initiale organisatorische Aufwände (Weiterbildung und Einarbeitung der Entwickler) vermeiden wollen, da wir keinen Bedarf an Scala Know-How in der Zukunft sehen.
- **INNOVATIONSKRAFT**
  Wenn es nicht darauf ankommt, daß wir bei der Entwicklung als besonders innovativ gelten.

Wir sollten Scala nutzen…

- **RISIKEN**
  Wenn wir ein erhöhtes Risiko bei der Entwicklung akzeptieren können, welchem wir aber durch Weiterbildung, Einarbeitung und nicht zu straffe Zeitvorgaben entgegenwirken.
- **AUFWÄNDE**
  Wenn wir die initialen organisatorischen Aufwände (Weiterbildung und Einarbeitung der Entwickler) investieren wollen, weil dies auch für zukünftige Entwicklungen sowieso ein unvermeidliches Muß darstellt.
- **INNOVATIONSKRAFT**
  Wenn wir den Mut zur Entwicklung mit Scala bewußt marketingtechnisch dazu nutzen, um gleichzeitig unsere Innovationskraft bei unseren Kunden explizit zu unterstreichen.

Unser Ratschlag für das Produktgeschäft

- Java 8 ist Scala in der Gesamtbetrachtung meist noch überlegen, es hängt aber ganz stark davon ab, wie die organisatorischen Rahmenparameter der Einheit sind.
- Wir raten deshalb nachdrücklich dazu, die Entscheidung weniger auf technologischer oder gar sprachlicher Ebene zu bestreiten, sondern primär die organisatorischen Rahmenparameter im Detail zu überprüfen.
- Wenn diese Rahmenparameter geeignet sind, kann eine Entscheidung auch durchaus für Scala anstatt Java 8 getroffen werden. Wenn sie nicht oder nur teilweise geeignet sind, dann sollte man sich auf keinen Fall für Scala entscheiden, da die Risiken zu hoch sind.

Thank you for your attention.

Dipl.-Inf. Univ. Ralf S. Engelschall

Principal IT Consultant, Department Manager
msg Applied Technology Research

+49 89 96101-1913
ralf.engelschall@msg-systems.com

www.msg-systems.com

.consulting .solutions .partnership

.msg