Ausgabe 01/2020

Java aktuell

Java aktuell

iJUG Verbund
www.ijug.eu

Sicherheit
Schützen Sie Ihre Anwendung vor unbekannten Bedrohungen

DevOps
Worauf es bei einer erfolgreichen DevOps-Kultur ankommt

GraalVM
Der heilige Gral der Compiler?

Immer auf der sicheren Seite

Praxis. Wissen. Networking. Das Magazin für Entwickler

D: 4,90 EUR | CH: 9,80 CHF | A: 5,60 EUR | Benelux: 5,80 EUR

4 191978 304903 01


ORAWORLD
Das e-Magazine für alle Oracle-Anwender!
EOUC
EMEA ORACLE USERGROUP COMMUNITY
• Spannende Geschichten aus der Oracle-Welt
• Technologische Hintergrundartikel
• Leben und Arbeiten heute und morgen
• Einblicke in andere User Groups weltweit
• Neues (und Altes) aus der Welt der Nerds
• Comics, Fun Facts und Infografiken
Jetzt Artikel einreichen oder Thema vorschlagen!
Jetzt e-Magazine herunterladen
www.oraworld.org

Sicher? Sicher!

Mit dieser ersten Ausgabe des Jahres 2020 beginnt ein neues Kapitel der Java aktuell; denn ab sofort und erstmals wird jede Zeitschrift einen Themenschwerpunkt haben. Diesem widmen sich die Artikel im vorderen Bereich der Zeitschrift. Danach folgt wie üblich eine bunte Mischung aus spannenden Artikeln aus dem gesamten Java-Themenkosmos. Die kommenden Ausgaben werden folgende Schwerpunkte haben: 02/20 Core Java, 03/20 Mobile, 04/20 Testing, 05/20 Microservices, 06/20 Werkzeuge, Tools und Frameworks, 01/21 Cloud und Container.

Der Schwerpunkt dieser Ausgabe 01/20 ist "Sicherheit". Ein relevantes Thema, das leider bei Entwicklern oftmals völlig zu kurz kommt. Im ersten Artikel, der auf Seite 12 beginnt, präsentiert Frank Ully, Oneconsult, nicht alltägliche Sicherheitslücken, mit denen potenzielle Angreifer großen Schaden anrichten können. Der Experte gibt Tipps, wie Sie diese bestmöglich schließen. Weiterhin haben wir ein Interview zum Thema Datensicherheit mit Tobias Schrödel geführt. Der "IT-Comedian" gibt uns darin eine Einschätzung, welche Sicherheitsrisiken in Zukunft immer mehr an Bedeutungen gewinnen werden und welche Rolle der Mensch im Bezug auf Sicherheit spielt.

Jatumba! Sie wissen mit diesem Wort (noch) nichts anzufangen? Dann wird es aber allerhöchste Zeit! Denn es ist das Leitwort zur großen Community-Konferenz JavaLand, die jedes Jahr mehr als 2.000 Java-Fans aus aller Welt ins Phantasialand nach Brühl lockt. Ab Seite 28 geben wir Ihnen eine Vorschau dessen, was Sie auf der kommenden JavaLand 2020 erwartet.

Außerdem erwarten Sie in dieser Ausgabe gleich zwei Artikel zum Thema DevOps. Maximilian Braun, virtual7, zeigt in "DevOps?! Gerne, aber richtig!", was erfolgreichen DevOps-Unternehmen zu ihrem Erfolg verhilft und welche Fakoren Sie bei der Umsetzung dieses Ansatzes beachten sollten.

Wir wünschen Ihnen viel Spaß beim Lesen!

Ihre

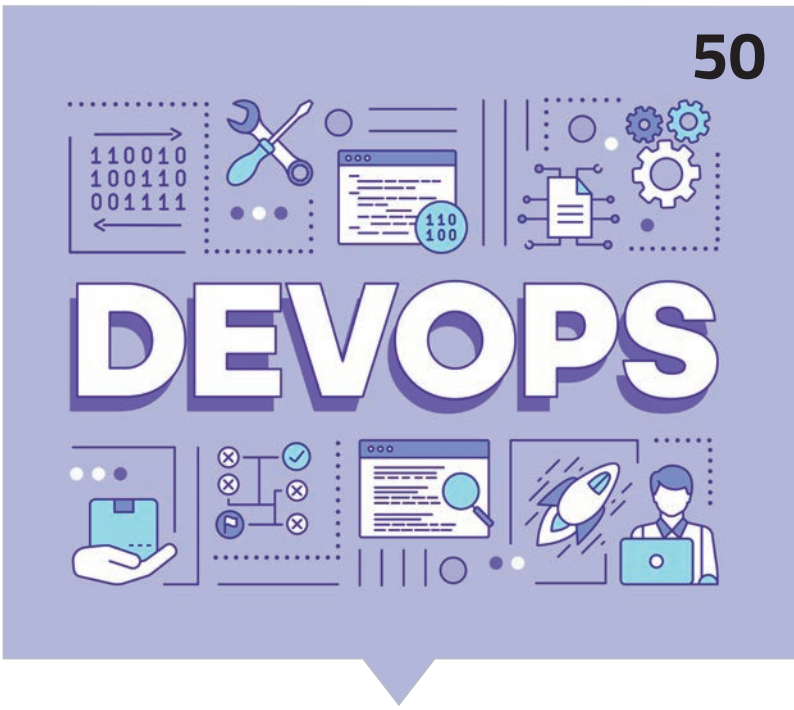
Lisa Damerow
Redaktionsleitung Java aktuell

Inhalt

Ungeläufige Sicherheitsrisiken bei Anwendungen und APIs

App-Entwicklung mit Kotlin

Finden Sie Ihre ganz persönliche Motivation



Möglichkeiten und Werkzeuge für DevOps mit Jenkins

Das Java-Tagebuch

Andreas Badelt, stellv. Leiter der DOAG Java Community

Das Java-Tagebuch gibt einen Überblick über die wichtigsten Geschehnisse rund um Java.

14. August 2019

Jelastic und der Zoo

Der Zoo an JDKs und JVMs wächst, wie man beispielsweise an der Ankündigung von Jelastic sehen kann: Der PaaS- und CaaS-Betreiber unterstützt jetzt auch die Varianten von AdoptOpenJDK, Liberica, Zulu, Corretto, Eclipse OpenJ9 sowie die GraalVM. Letztlich basieren sie ja alle auf dem OpenJDK und drei VMs – Hotspot, J9 und GraalVM, das macht die Sache für den Zoowärter wieder etwas einfacher. Grundsätzlich tut Vielfalt ja auch dem Java-Ökosystem gut.

20. August 2019

Microsoft schnappt sich jClarity

Microsoft kauft den Java-Spezialisten jClarity. Das von führenden Mitgliedern der London Java Community gegründete Unternehmen hat sich mit Software und Beratung zum Java Performance Tuning einen Namen gemacht, aber auch mit dem Einsatz für kostenlose OpenJDK Binaries im Rahmen der AdoptOpenJDK-Initiative und darauf aufbauendem kommerziellen Support. Bei Microsoft sollen sie sich natürlich um die Kunden der Azure-Plattform kümmern, aber auch um konzerneigene Services wie den Analysedienst Azure HDInsight und – Bonus! – Minecraft. Schon enorm, wer sich inzwischen zumindest virtuell in Redmond tummelt. Erst im Juni war ja auch Sun- und Oracle-Urgestein Ed Burns zu Microsoft gewechselt (das hatte ich im alten Tagebuch tatsächlich unterschlagen).

20. August 2019

Nägel mit Köpfen

Ivar Grimstad, bislang hauptberuflicher IT Consultant bei Cybercom und ehrenamtlicher Kopf des Jakarta-EE-Projekts, wechselt den Job. In Zukunft ist er als „Jakarta EE Developer Advocate" bei der Eclipse Foundation angestellt. Das könnte dem Projekt noch mal einen wichtigen Schub geben – und Ivar vielleicht mal ein paar Stunden Ruhe zwischen Job und dem Einsatz für Jakarta EE von zuhause aus und auf Konferenzen weltweit.

22. August 2019

Auch VMware ist auf Shoppingtour

Pivotal ist im Jahr 2012 als Ausgründung von VMware und dem Mutterkonzern EMC entstanden. Jetzt kauft sich der Hersteller von Virtualisierungssoftware das Unternehmen zurück, das Spring und die Cloud-Foundry-Plattform im Portfolio hat. Gleichzeitig wurde auch die Übernahme von Carbon Black bekannt gegeben, das Sicherheitssoftware für den „Cloud-native"-Einsatz entwickelt. Das sind allerdings nur die letzten Einkäufe. In den vergangenen Monaten wurde zum Beispiel auch Bitnami, Spezialist für „vorkonfektionierte" VM- und Container-Images, erworben.

26. August 2019

VMware und Kubernetes

Nur um das noch klarzustellen: An der Dominanz von Kubernetes wird der Kaufrausch von VMware auch nichts mehr ändern: Cloud Foundry setzt eh seit Längerem statt Konkurrenzkampf auf die Integration von Kubernetes (Pivotal Container Services oder kurz PKS). Der alte und neue Mutterkonzern hat heute die Projekte „Pacific" und „Tanzu Mission Control" vorgestellt: Das eine ist die Integration von Kubernetes in vSphere, um Container und virtuelle Maschinen gleichermaßen auf einer einzelnen Plattform zu steuern und konvergieren zu lassen. Das andere ist eine universelle Steuerungsebene für Kubernetes-Cluster, unabhängig davon, wo diese laufen.

10. September 2019

JakartaOne mit Release von EE 8

Das Warten hat ein Ende: In einem Livestream während der virtuellen JakartaOne-Konferenz ist heute Jakarta EE 8 freigegeben worden! Die Anwälte haben das Tauziehen verloren… Spaß! 1.350 Teilnehmer aus aller Welt waren laut Eclipse Foundation angemeldet. Das sogenannte „Base Level Release" ist der eineiige Zwilling von Java EE 8, mit identischen APIs und dem noch unangetasteten „javax.*"-Namespace – aber als vollständiges Open-Source-Projekt bei der Eclipse Foundation. Damit haben für Java EE 8 zertifizierte Produkte den denkbar einfachsten Migrationspfad Richtung Jakarta. Mit GlassFish 5.1 und Open Liberty 19.0.0.6 sind auch bereits zwei Anwendungsserver („lightweight frameworks for cloud-native Java microservices" heißen die ja jetzt) offiziell als Jakarta-EE-kompatibel zertifiziert. Ab jetzt können neue Features hinzugefügt werden, in Jakarta EE 9 und hoffentlich noch vielen weiteren Releases. Neben der Release-Feier und Vorträgen zu Details von Jakarta bot die Konferenz auch Raum für benachbarte Themen wie natürlich MicroProfile, aber auch zu den Micro-Frameworks Quarkus.io und Helidon gab es Sessions. Videoaufzeichnungen der Vorträge gibt es auf der Veranstaltungswebsite *[1]*.

10. September 2019

OpenJFX 13

Wieder einmal einige Tage vor dem neuen Java-Release ist das getrennte JavaFX-Release OpenJFX 13 erschienen. Es setzt Java 11 oder höher voraus und bietet unter anderem Unterstützung für e-Paper Displays und natives Rendering. Bei Letzterem kann Speicher über ein Public-API mit nativen Anwendungen geteilt werden. Damit sollen grafikintensive Anwendungen wie CAD oder 3D-Editoren mit JavaFX möglich werden.

13. September 2019

WildFly 17.0.1 für Jakarta

Red Hat hat eine Patch-Version für WildFly herausgebracht. In Version 17.0.1 ist er der dritte Anwendungsserver mit offizieller Jakarta-EE-8-Kompatibilität. WildFly 18 kommt ja auch demnächst heraus, aber darauf wollte man wohl nicht warten.

14. September 2019

Helidon: MicroProfile 3.0

In schneller Folge veröffentlicht Oracle neue Versionen des Microservice-Frameworks Helidon. Nach 1.2 mit Unterstützung für MicroProfile 2.2 implementiert Version 1.3 jetzt zusätzlich den vollen Umfang von MicroProfile 3.0. Da MicroProfile 3.0 jedoch nicht vollständig rückwärtskompatibel zu 2.2 ist, muss der Entwickler beim Bauen über eine Bundle Dependency entscheiden, welche der beiden Versionen unterstützt werden soll.

16. September 2019

Start der Oracle CodeOne

Die zweite Auflage der CodeOne ist gestartet. Der Nachfolger der legendären JavaOne, dem anfangs viele Skepsis entgegenschlug, hat sich doch schnell etabliert und muss sich in Bezug auf Inhalt, Beteiligung prominenter Speaker und „Community Feeling" nicht verstecken. Die Java-Keynote steht natürlich insbesondere im Zeichen von Java 13, dazu später mehr. Den üblichen „Warm-up"-Vortrag hält diesmal eine Doktorandin der Uni Stanford zum aktuellen Thema Quantum Computing. „Post-Quanten-Kryptographie" ist mein neuer Lieblingsbegriff, den ich daraus mitnehme. Und dass das Internet daher nicht spontan explodieren wird (oder war es „implodieren"?) – sehr beruhigend. Danach geht es mit Java weiter. Zunächst wird der Java Community Process (JCP) gefeiert, der seit 20 Jahren besteht. Er ist zwar auf SE-Themen geschrumpft, spielt dort aber weiter eine wichtige Rolle. Bekannte Namen aus der Community auf der Bühne und per Video bezeugen, wie gut das Zusammenspiel zwischen Innovation im OpenJDK und der Sicherung von Kompatibilität und Stabilität durch den JCP funktioniert. Gil Tene von Azul erzählt, dass der schnellste JSR vor der Umstellung auf den „Release Train" zehn Monate gebraucht hat. Jetzt läuft alles annähernd reibungslos im Sechs-Monats-Rhythmus.

16. September 2019

Java 13 Release

Den wichtigsten Teil der Java-Keynote auf der CodeOne nimmt die Vorstellung von Java 13 ein, mit über 2.000 Fixes und kleineren Verbesserungen. Brian Goetz, Java Language Architect, und Mikael Vidstedt, Director of JVM Development, stellen die neuen Features vor und lassen sich unter anderem über Performance-Verbesserungen bei Startup Times und dem Zero Garbage Collector aus. Die fünf wesentlichen Features habe ich schon im letzten Tagebuch aufgelistet, die Liste ist ja auch im Web verfügbar *[2]*. Zum Schluss gibt es noch einen Ausblick auf die umfangreiche „Pipeline": Project Amber, das uns in den vergangenen Releases bereits Local Type Inference, Switch Expressions und Text Blocks gebracht hat, wird vermutlich im nächsten Jahr Records und Sealed Types liefern; auch Pattern Matching für `instanceof` und Switch Expressions steht an. Project Panama („Interconnecting JVM and native code.") soll im Jahr 2020 eine Preview-Version liefern. Außerdem – ohne Konkretes zum Zeitplan – Project Loom, das der JVM „Fibers" bringen soll („lightweight user-mode threads"), und Project Valhalla, das bereits seit fünf Jahren an Value Types und der Anpassung an moderne Hardware arbeitet, in der arithmetische Berechnungen viel schneller sind als Speicherzugriffe.

17. September 2019

Java 13: Wer ist beteiligt?

Oracles Sharat Chander listet im Blog auf, dass 70 Prozent der JIRA Issues zu Java 13 von Oracle-Mitarbeitern geschlossen wurden. Die Quote hat sich über die letzten Releases reduziert (80 Prozent bei Java 11). Das zeigt jedoch in erster Linie, dass andere Hersteller sich stärker engagieren. Fünf Prozent der Beiträge kommen von unabhängigen Entwicklerinnen und Entwicklern (zwei beziehungsweise drei Prozent in den letzten Releases) – das ist auch ein Zeichen für ein lebendiges Ökosystem!

17. September 2019

Java 13 – alle Binaries sind schon da

Gestern wurde Java 13 vorgestellt, heute ist der Zoo an Varianten der verschiedenen Hersteller verfügbar. Neben Hotspot ist auch OpenJ9 als Laufzeitumgebung dabei, die Binaries können zum Beispiel von der AdoptOpenJDK-Seite geladen werden *[3]*. Andererseits – welche Entwicklerin und welcher Entwickler nutzen schon einen Browser, wenn es auch ein API gibt *[4]*?

18. September 2019

New Relic unterstützt AdoptOpenJDK

Das AdoptOpenJDK-Projekt wächst und wächst. Mit New Relic (Application Performance Management) ist jetzt ein weiteres Unternehmen als offizieller Unterstützer hinzugekommen – was zum einen reines Sponsoring beinhaltet, aber auch mehr (Wo-)Manpower für das Projekt.

19. September 2019

Ende der CodeOne

Die CodeOne geht schon wieder zu Ende. Etwas weniger Besucher als letztes Jahr hatte sie wohl, aber der „Community Spirit" war da.

Es gab eine große Vielfalt an Themen: von Sprach-Features über Tooling, Tipps und Tricks rund um Debugging und Performance Tuning oder nicht-technische Themen wie Support bis hin zu den Hype-Themen wie „Cloud Native". Was jedoch auch wichtig war: Die Zukunft von Enterprise Java (also insbesondere Jakarta EE und MicroProfile), die im Vergleich zu Java SE sicher deutlich weniger klar ist, wurde in Podiumsdiskussionen und auf den Gängen lebhaft diskutiert.

23. September 2019

Jakarta EE, MicroProfile, Namespaces

Jakarta EE 8 ist da, allerdings sind einige große Fragen rund um Jakarta weiterhin nicht geklärt. Zwei wesentliche Punkte sind: Erstens die Diskussion um den javax-Namespace, die im Mai nach erfolglosen juristischen Verhandlungen richtig losging, aber bislang nicht zu einer Einigung geführt hat. Und zweitens das Verhältnis zu MicroProfile, das weiterhin nicht geklärt ist. Soll MicroProfile der „Experimentierraum" für anschließende Standardisierung in Jakarta EE sein – wie es schon mal vorgeschlagen wurde, als wir noch von Java EE sprachen? Eine große Mehrheit ist laut der letzten Umfrage der Java EE Guardians dafür. Sollen die beiden Projekte sogar vereinigt werden – logisch oder physisch? In den Mailinglisten und Google Groups beider Projekte gibt es dazu lange, häufig emotionale Threads. Eine Zusammenfassung der jüngeren Diskussionen ist im Web zu finden *[5]*.

Der jüngste Vorschlag: Phillip Krüger, MicroProfile-Committer aus Südafrika, auch für Jakarta EE aktiv, hat die beiden wesentlichen Punkte miteinander verbunden *[6]*: Jakarta soll einen neutralen Package-Namen erhalten, unabhängig von seinem Markennamen, aber auch unabhängig vom „Projektort" (also weder jakarta.* noch org.eclipse.enterprise.*), und MicroProfile soll denselben Package-Namen erhalten. Damit wäre es deutlich einfacher, MicroProfile-APIs zur Standardisierung an Jakarta EE zu übergeben; MicroProfile würde dann weiter die „stabilen" EE-APIs verwenden und sich den nächsten Innovationen zuwenden. Der Vorschlag ruft jedoch zum Teil deutlich ablehnende Reaktionen hervor und es wird sichtbar, dass es nicht so einfach ist, zwei große, unabhängige Projekte mit vielen hochmotivierten (und meinungsstarken) Leuten in eine gemeinsame Richtung zu lenken.

3. Oktober 2019

WildFly 18

Die nächste Major-Version von WildFly ist da; sie deckt neben dem Umfang von Jakarta EE 8 aus 17.0.1 auch alle MicroProfile-3.0-APIs ab, die nicht in EE 8 enthalten sind. Auch das gerade erschienene JDK 13 wird – mehr oder weniger – bereits unterstützt (laut eigener Aussage laufen die wesentlichen Testsuiten bis auf wenige Fehler in „vermutlich nicht häufig genutzten Bereichen" durch); empfohlen wird aber weiterhin das aktuelle „Long Term Support" Release 11

Referenzen

[1] *jakartaone.org*
[2] *http://openjdk.java.net/projects/jdk/13/*
[3] *https://adoptopenjdk.net/releases.html*
[4] *https://api.adoptopenjdk.net*
[5] *https://jaxenter.com/jakarta-ee-eclipse-microprofile-ongoing-discussions-161031.html*
[6] *https://www.phillip-kruger.com/post/proposed_namespace_jakarta_ee/*

Andreas Badelt
stellv. Leiter der DOAG Java Community
andreas.badelt@doag.org

Andreas Badelt ist stellvertretender Leiter der DOAG Java Community. Er ist seit dem Jahr 2001 ehrenamtlich im DOAG e.V. aktiv, zunächst als Co-Leiter der SIG Development und später der SIG Java. Seit 2015 ist er stellvertretender Leiter der neugegründeten Java Community innerhalb der DOAG. Beruflich hat er seit dem Jahr 1999 als Entwickler und Architekt für deutsche und globale IT-Beratungsunternehmen gearbeitet und ist seit dem Jahr 2016 als freiberuflicher Software-Architekt unterwegs („www.badelt.it").

Markus' eclipse-Corner

Hurra, hurra! Jakarta EE 8 ist da! Und wieder mal ein Grund zum Feiern für die Eclipse Foundation!

Die Eclipse Foundation ist schon genial. Sie hat doch tatsächlich das Unmögliche geschafft und eine Punktlandung hingelegt und damit alle Zweifler (wie mich) Lügen gestraft! Wie Mike Milinkovic in seinem Blog *[1]* formuliert, hat die Eclipse Foundation mit der „Veröffentlichung von Jakarta EE 8 ein neues Zeitalter der Java-Innovation eingeläutet!". Dank der immensen Anstrengungen aller Beteiligten, hat es – oh Wunder! – wie im Juli angekündigt *[3]* auf den Tag genau funktioniert: Exakt am 10. September, dem Tag des *[2]* JakartaOne-Livestreams, das lang ersehnte, erste echte und rein nach den Regeln des kürzlich gegründeten Jakarta EE Specification Committee zertifizierte Release von Ex-Java-EE zu veröffentlichen. Kurz darauf sind auch gleich fertige Produkte zu haben, da bereits die ersten Application-Server offiziell als Jakarta-kompatibel *[4]* „selbst-zertifiziert" sind! Yippie! Nun ist doch hoffentlich endlich, endlich alles gut!

Also gleich mal alle neuen Spezifikationsdokumente *[5]* heruntergeladen und studiert und… Moment mal… wo ist denn der Text? Ja, genau, die sind fast alle leer, da stehen ja nur die Lizenz und der obligatorische Urheberrechtsvermerkt! Nein, das ist kein Fehler, sondern Absicht – nur wir Normalsterblichen halten das für einen Schildbürgerstreich. Denn das war ja auch schon immer so geplant, weil man sonst den Termin ja nicht halten kann. Aha. Na gut. Stimmt also – Punktlandung!

Aber wie steht es denn um den Rest? Nehmen wir uns doch mal die TCKs vor. Leider sind sie entgegen der allgemeinen Erwartungshaltung noch nicht in einzelne Test Kits zerlegt. Auch diese hat sich die Stiftung für später aufgespart. Nun hat Oracle vorgeschlagen *[6]*, dass man dies durchaus gerne mal irgendwann beginnen könne, es aber nicht zwingend in Jakarta EE 9, sondern eher noch später (lies: oder auch nie) fertig machen könne. Hm, vielleicht für Application-Server-Hersteller angenehm, doch für die echten Cloud Natives unter uns, die eigentlich gar keinen solchen Hinkelstein mehr benötigen und nur einen kleinen Teil der Spezifikationen nutzen wollen (wie zum Beispiel die beliebte Kombination CDI, JAX-RS und JSON-B), ist das ein echter Schlag in den Magen. Die TCKs sind nämlich, auch das wird gerne verschwiegen, keineswegs von herausragender Güte. Aus aktueller Projekterfahrung kann ich hierzu sagen, dass bereits in einem kleineren Projekt von nur acht Wochen Laufzeit bereits mehrere offenkundige Testschwächen im Bereich JSON-B und JAX-RS aufgefallen sind: Tauscht man beispielsweise die ehemaligen Referenz-Implementierungen Jersey und Yasson gegen (ebenfalls das TCK bestehende) Alternativen aus, läuft plötzlich die (ebenfalls die Spezifikation einhaltende) Anwendung nicht mehr! WORA sieht anders aus! Im konkreten Fall retteten mehrere selbst entwickelte und an die Community gespendete Bug Fixes den Projekterfolg. Doch eigentlich soll ein Standard und das damit verbundene Test Kit diesen Zustand verhindern. Das Ergebnis: Vier Wochen Projektverzug, dafür neue Freunde gewonnen, die sich über die Bug Fixes freuten.

Nun wäre das alles ja keiner Kritik wert, wenn es zumindest irgendeinen Nutzen hätte, dieses Jakarta EE 8 Release. Doch wo soll der stecken, im Vergleich zu der Zeit vor dem 10. September? Trotz intensiver Nutzung der Standards und Produkte und trotz eigener Mitarbeiter an selbigen Open-Source-Projekten konnte ich keinen ausmachen – denn neue Features sind ja keine drin. Also habe ich einfach mal öffentlich gefragt und eine überraschende Antwort erhalten. TL;DR: Keinen. Natürlich sehen das die Eclipse Foundation und deren (erhebliche Summen) zahlende Mitglieder anders. Unisono sprechen sie davon, dass „wir alle" die Nutznießer sind. Leider konnten sie mir nicht sagen, wer „wir alle" denn sein soll und welcher Nutzen exakt das ist. Behauptet wurde, der Spezifikationsprozess sei nun validiert. Nun ja, leider stimmt das nicht. Denn unabhängige Entwickler haben sich weitgehend rausgehalten, sodass das ganze Thema eigentlich, bis auf geringe Ausnahmen, in den Händen von Oracle lag. Und eigentlich wollte man ja zeigen, dass man selbst schon groß ist und das auch alleine kann. Außerdem war ja das Ergebnis, dass eben keinerlei Features, Spezifikationen oder TCKs unter diesem Prozess entwickelt wurden. Von daher ist das ein eher zweifelhafter Beweis für funktionierende Prozesse. Es sei denn, und nun schließt sich der Kreis, man wollte sowieso nur die Fähigkeit zur Lieferung leerer PDFs, weiterhin lückenhafter und monolithischer TCKs und Null-Feature-Sprints beweisen. Das hat ja dann tatsächlich perfekt und 100 Prozent planungskonform funktioniert. Bravo!

Insofern haken wir das Thema Jakarta EE 8 jetzt einfach ab und konzentrieren uns auf Jakarta EE 9. Die Planung dazu hat Oracles Bill

Shannon ja bereits vorgeschlagen *[6]*: Einfach mal was wegwerfen und sonst eher mal langsam machen. Na prima! „Wir machen es wie immer nur noch doller", wie ein Kollege von mir immer zu sagen pflegt. In diesem Sinne daher mein üblicher Aufruf: Wenn ihr wollt, dass sich was ändert, dann wirkt bitte aktiv an den Open-Source-Projekten mit! Alles, was ich bislang gespendet habe, ist nach kurzer Zeit angenommen worden. Wenn also jeder Leser nur ein einziges Feature liefert oder nur einen einzigen Bug fixt, haben wir in zwei Wochen mehr geändert als die EF in zwei Jahren...

[1] *https://blogs.eclipse.org/post/mike-milinkovich/welcome-future-cloud-native-java* - „Welcome to the Future of Cloud Native Java" / (Mike MILINKOVIC, Eclipse Fondation)

[2] *https://jakartaone.org/* - JakartaOne Livestream (Eclipse Foundation)

[3] *https://blogs.eclipse.org/post/tanja-obradovic/update-jakarta-ee-community-july-2019* - „Update for Jakarta EE community: July 2019" (Tanja OBRADOVIC, Eclipse Foundation)

[4] *https://jakarta.ee/compatibility/* - Jakarta EE Compatible Products (Eclipse Foundation)

[5] *https://jakarta.ee/specifications/* - Jakarta EE Specifications (Eclipse Foundation)

[6] *https://www.eclipse.org/lists/jakartaee-platform-dev/msg00648.html* - Oracle's position on Jakarta EE 9 (Bill SHANNON, Oracle)

Markus Karg

markus@headcrashing.eu

Markus Karg ist Entwicklungsleiter eines mittelständischen Softwarehauses sowie Autor, Konferenzsprecher und Consultant. JAX-RS hat der Sprecher der Java User Group Goldstadt von Anfang an mitgestaltet, zunächst als freier Contributor, seit JAX-RS 2.0 als Mitglied der Expert Groups JSR 339 und JSR 370.

Unbekannte Kostbarkeiten des SDK Heute: try-with-resource-Statement mit „effectively final" Ressourcen

Bernd Müller, Ostfalia

Das Java SDK enthält eine Reihe von Features, die wenig bekannt sind. Wären sie bekannt und würden sie verwendet, könnten Entwickler viel Arbeit und manchmal sogar zusätzliche Frameworks einsparen. Wir wollen in dieser Reihe derartige Features des SDK vorstellen: die unbekannten Kostbarkeiten.

Das Ziel dieser Ausgabe unserer unbekannten Kostbarkeiten ist nicht das Reduzieren von Schreibaufwand, sondern die Verbesserung von Code-Lesbarkeit. Mit Java 9 wurde das try-with-resource-Statement überarbeitet und lässt nun Ressourcen zu, die nicht innerhalb des Statements definiert sein müssen, wenn sie „final" oder „effectively final" sind.

Automatic Resource Management (ARM) mit Java 7

Java 7 verbesserte das automatisierte Verwalten von Ressourcen mit dem sogenannten try-with-resource-Statement, dessen konkrete syntaktische Form weder „*with*" noch „*resource*" enthält. Es dürfte allen Java-Entwicklern bekannt sein und breite Verwendung finden, da es verwendeten Ressourcen automatisch schließt und uns so Arbeit abnimmt. Der in *Listing 1* beschriebene Code-Ausschnitt verdeutlicht dies.

Die Klassen `Scanner` und `PrintWriter` implementieren beide das Interface `AutoClosable`, sodass ihre Instanzen automatisch beim Verlassen des Blocks geschlossen werden. Vor Java 7 musste im „finally"-Teil des `try`-Statements zunächst geprüft werden, ob die beiden Ressourcen tatsächlich geöffnet sind, um sie dann zu schließen. Dies kann nun komplett entfallen.

Das Beispiel kopiert lediglich eine Datei, wofür deutlich besserer Alternativen existieren. Es soll hier nur dazu dienen, die Verwendung der beiden Variablen `scanner` und `writer` zu verdeutlichen. Sie müssen innerhalb der runden Klammern des Statements erzeugt und mit einem Semikolon getrennt werden. Der Autor ist der Meinung, dass dies der Lesbarkeit des Codes nicht guttut.

Finale Ressourcen

Seit Java 9 dürfen die Ressourcen des try-with-resource-Statements außerhalb des Statements definiert werden, wenn sie „final" oder „effectively final" sind. Wenn man das obige Beispiel derart überarbeitet, ändert sich der Code wie in *Listing 2* abgebildet.

Der Autor findet, dass diese Version besser lesbar ist und damit eine höhere Code-Qualität aufweist und hofft, dass die Leser diese Meinung teilen.

Zusammenfassung

Seit Java 9 verlangt das try-with-resource-Statement nicht mehr, dass die verwendeten Ressourcen innerhalb der runden Klammern deklariert werden, sondern erlaubt deren Deklaration auch außerhalb des Statements, wenn die Variablen final oder effectively final deklariert werden. Das kommt der Lesbarkeit des Codes zugute.

Bernd Müller
Ostfalia
bernd.mueller@ostfalia.de

Nach seinem Studium der Informatik und der Promotion arbeitete Bernd Müller für die IBM und die HDI Informationssysteme. Er ist Professor, Geschäftsführer, Autor mehrerer Bücher zu den Themen JSF und JPA, sowie Speaker auf nationalen und internationalen Konferenzen. Er engagiert sich im iJUG und speziell in der JUG Ostfalen.

```
try (Scanner scanner = new Scanner(new File(FILE_TO_READ));
  PrintWriter writer = new PrintWriter(new File(FILE_TO_WRITE))) {
  while (scanner.hasNext()) {
    writer.print(scanner.nextLine());
  }
}
```

Listing 1

```
Scanner scanner = new Scanner(new File(FILE_TO_READ));
PrintWriter writer = new PrintWriter(new File(FILE_TO_WRITE));
try (scanner; writer) {
  while (scanner.hasNext()) {
    writer.print(scanner.nextLine());
  }
}
```

Listing 2

Beyond OWASP Top 10 – Unbekanntere Arten von Schwachstellen in Webanwendungen und APIs

Frank Ully, Oneconsult Deutschland GmbH

Auch wenn bei der Anwendungsentwicklung grundlegende Sicherheitslücken wie Cross-Site-Scripting (XSS) oder SQL-Injections vermieden werden, sind Webapplikationen und Schnittstellen anfällig für teilweise kuriose Schwachstellen. Dieser Artikel stellt ausgewählte, unbekanntere Schwachstellenarten vor, die ein Penetration Tester in der Berufspraxis findet, und gibt Hinweise auf gute Quellen zur Vertiefung der Inhalte.

Sicherheit im Web muss beim Entwerfen und Entwickeln von Webanwendungen und Schnittstellen von Anfang an berücksichtigt werden – dieses Bewusstsein und das notwendige Wissen werden jedoch in Studium und Kursen unzureichend vermittelt, wovon zahlreiche Schwachstellen in verbreiteten Anwendungen zeugen.

Im Dienste der Websicherheit – Open Web Application Security Project (OWASP)

Das Open Web Application Security Project (OWASP) *[1]* ist eine Non-Profit-Organisation, die sich auf die Fahnen geschrieben hat, die Sicherheit von Webanwendungen zu verbessern. Dazu veröffentlichen die zahlreichen Freiwilligen, die zum Projekt beitragen, unter anderem technische Dokumentationen, Softwarebibliotheken und Testwerkzeuge – allesamt kostenfrei nutzbar, unter freien Lizenzen wie Creative Commons, und sowohl bei Dokumenten als auch bei Software in bearbeitbaren Formaten beziehungsweise im Quelltext verfügbar.

Ihre wohl bekannteste Publikation ist die „OWASP Top 10", eine Aufzählung der zehn kritischsten Sicherheitsrisiken in Webanwendungen *[2]*. Die Liste wurde erstmals im Jahr 2003 veröffentlicht, zuletzt 2017 angepasst und ist vielen Entwicklern ein Begriff. Die nächste Aktualisierung der Top 10 steht im Herbst 2020 an *[3]*.

Die „OWASP Top 10" stellt zehn Angriffsarten auf Webanwendungen vor, ihre Ursachen und welche Maßnahmen bei der Entwicklung dagegen helfen. Die Liste beantwortet unter anderem folgende Fragen: Was muss ein Entwickler beachten, wenn er Mechanismen zur Authentifizierung und Autorisierung implementiert? Warum sind Eingabevalidierung und Ausgabekodierung wichtige Grundlagen von sicheren Webanwendungen? Inwiefern kann eine Anwendung von Injektionsschwachstellen betroffen sein? Warum sollte man beim Entwickeln auf verlässliche Komponenten von Dritten achten und darauf, dass die Anwendung möglicherweise sicherheitsrelevante Aktionen loggt?

Manche Anbieter gehen sogar so weit, ihre Software als „sicher" zu bezeichnen, weil sie „gemäß den OWASP Top 10" entwickelt worden sei; nicht immer seriöse Sicherheitsdienstleister bieten Tests nur „gegen die OWASP Top 10" an und stellen danach der Anwendung ein „Sicherheitszertifikat" aus.

Das ist jedoch unaufrichtig, denn die Top 10 sind nach eigener Zielsetzung kein ausführlicher Sicherheitsstandard *[4]*. Die Top-10-Verantwortlichen selbst sehen ihre Liste als Awareness-Dokument, das einem breiten Publikum von Informationssicherheitsverantwortlichen über Softwareprojektmanager und Entwickler bis zu Testern grundlegende Schwachstellen zugänglich machen soll. Deren Ursachen sollten auf jeden Fall in Webanwendungen vermieden werden – denn das Fehlen von Sicherheitsmaßnahmen dagegen grenzt an Fahrlässigkeit.

Wie festgestellt, können die Top 10 also nicht alle Schwachstellenarten abdecken, die bei modernen Webanwendungen und Schnittstellen bei einem technischen Sicherheitsaudit wie einem Web Application Penetration Test aufgespürt werden.

Gute Vorbereitung ist die halbe Miete – Interception Proxy

Die Untersuchung einer Webanwendung oder Schnittstelle beginnt sowohl für wohlmeinende Tester als auch für böswillige Angreifer in der Regel damit, einen Interception Proxy einzurichten. Der Proxy fungiert als Man-in-the-Middle zwischen dem Browser und der Anwendung, die angegriffen wird. Er zeichnet den gesamten HTTP(S)-Verkehr auf und erlaubt es, Anfragen beliebig zu wiederholen oder zu manipulieren. Der bekannteste Vertreter ist der Burp-Proxy *[5]*, wie in *Abbildung 1* gezeigt.

Umgehung von Rate Limiting

In der Regel sollte jede Anwendung Mechanismen gegen Brute-Force-Angriffe enthalten, mindestens an besonders sensiblen Stellen. Andernfalls kann ein Angreifer zum Beispiel versuchen, sich an einer Anmeldemaske mit einem ihm bekannten individuellen Benutzernamen – oder auch naheliegenden Standard-Accountnamen wie „admin" – und Passwörtern aus einer Passwortliste so lange anzumelden, bis er die richtige Kombination gefunden hat.

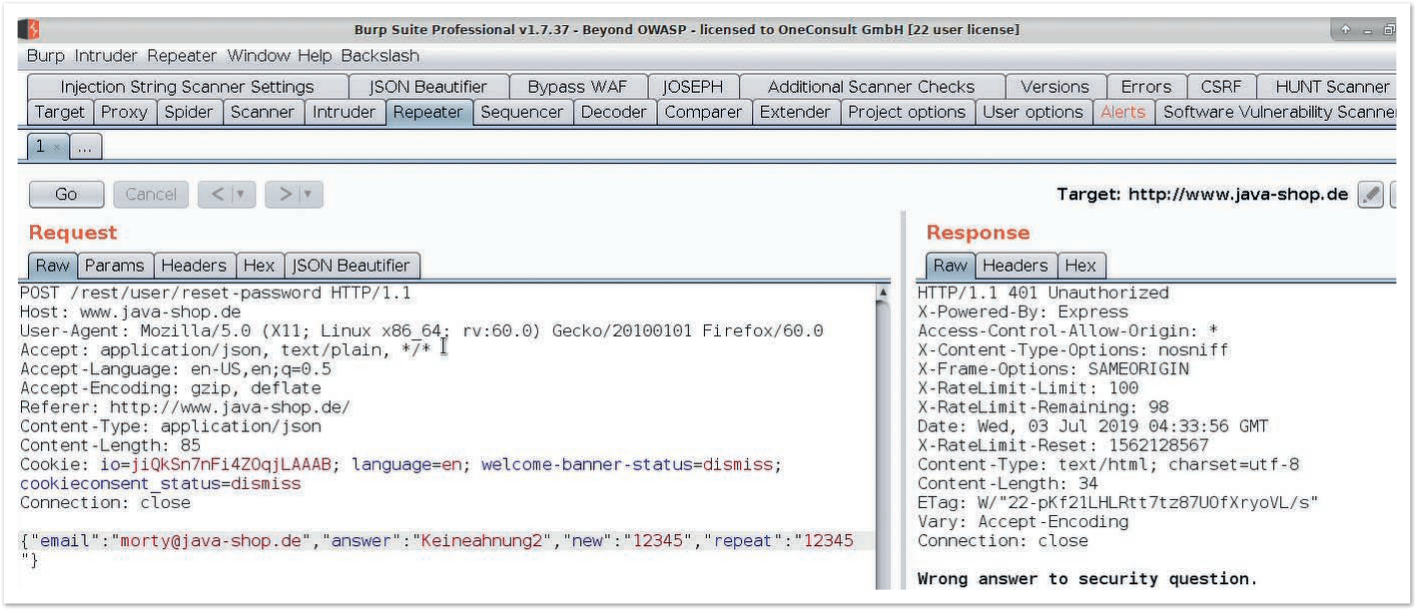


Abbildung 1: Burp-Proxy mit Anfrage (Request) links und Antwort (Response) rechts (Quelle: Frank Ully)

Eine verbreitete Schutzmaßnahme neben CAPTCHAs ist das Rate Limiting *[6]* – auf Deutsch Ratenbegrenzung: Wenn von einer einzelnen IP-Adresse, im Kontext eines Accounts oder innerhalb einer Session zu viele Anfragen in einer bestimmten Zeit gestellt werden, ignoriert die Anwendung weitere Versuche und antwortet mit dem Statuscode `429: Too Many Requests`. Meist senden Endpunkte mit Ratenbegrenzung auch Antwort-Header wie `X-RateLimit-Limit` und `X-RateLimit-Remaining`, die über konfigurierte Grenzen und verbleibende Versuche informieren, wie in *Abbildung 2* dargestellt.

Doch auch wenn ein Rate-Limiting-Mechanismus implementiert ist, etwa weil dafür eine Security-Drittbibliothek eingebunden wurde, kann dieser öfter als gedacht umgangen werden. Header wie `X-Forwarded-For` *[7]* werden normalerweise von Infrastrukturkomponenten wie Reverse Proxys gesetzt, die damit der Anwendung die eigentliche IP-Adresse einer Anfrage mitteilen. Aber nichts hindert einen Angreifer daran, einen solchen Header in einer Anfrage selbst einzufügen, zum Beispiel `X-Forwarded-For: 127.0.0.1` in einem Interception Proxy. Vertraut die Anwendung diesen Angaben, lässt sich die Ratenbegrenzung einfach umgehen *(siehe Abbildung 3)*.

Dadurch ist die Anwendung wieder anfällig für Denial-of-Service- (DoS) und Brute-Force-Angriffe. Schlimmer noch: Ein Angreifer kann den Rate-Limiting-Mechanismus selbst für DoS missbrauchen, wenn er im Header IP-Adressen einträgt, durch die Organisationen oder Einzelpersonen auf das Internet und damit die Anwendung zugreifen – und diese damit aussperren.

Anfällig für Manipulationen dieser Art sind nicht nur weitere Header in Bezug auf die *Quelle* einer Anfrage wie `True-Client-IP`, sondern auch Header bezüglich des *Ziels* einer Anfrage, etwa `X-Forwarded-Host`. Durch blindes Vertrauen in Ziel-bezogene Header entstehen Schwachstellen wie Web Cache Poisoning *[8]*.

Entdeckt werden können solche Lücken durch automatisierte und manuelle Tests, in denen diese Header gezielt eingefügt werden und beobachtet wird, ob sich das Antwortverhalten der Anwendung ändert. Solche anfälligen Konfigurationen sind auch in verbreiteten Rate-Limiting-Bibliotheken zu finden. Ursächlich ist hier also sowohl der Grundsatz „traue niemals Benutzereingaben" missachtet worden wie auch „kenne und prüfe deine Abhängigkeiten".

Server-Side Template Injection (SSTI)

Injektionen sind der erste Punkt der regulären Top-10-Liste. Solche Schwachstellen entstehen dann, wenn ein Angreifer ungehindert bösartige Daten an die Anwendung senden kann, die unverändert von einem Interpreter verarbeitet werden. Die bekannteste Art sind SQL-Injections, mit denen Datenbankabfragen manipuliert werden können; die Top 10 erwähnt aber auch Injektionen für NoSQL, Betriebssystemkommandos oder XML-Technologien wie XPath *[9]*.

Weniger bekannt hingegen ist, dass eine Anwendung auch von Injektionen in anderen und in der modernen Webentwicklung verbreiteten Komponenten wie Templates betroffen sein kann. Templates dienen dazu, HTML-Seiten einfacher zu gestalten, indem statische Vorlagendateien mit Platzhaltern zur Laufzeit mit den eigentlichen Werten befüllt werden; dadurch sind Programmlogik und Präsentationsschicht einer Webanwendung besser getrennt.

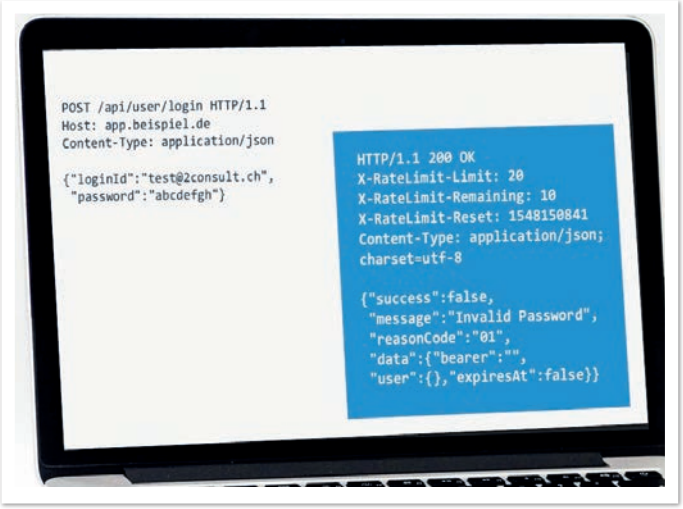


Abbildung 2: Anfrage und Antwort mit Rate-Limiting-Headern (Quelle: Frank Ully)

Verbreitete Template-Sprachen sind FreeMarker oder Velocity unter Java; Jade unter Node.js und Jinja2 unter Python.

Werden jedoch Benutzereingaben ungeprüft direkt in ein serverseitiges Template übernommen, kann ein Angreifer unter Umständen selbst Ausdrücke in der jeweiligen Template-Sprache formulieren – es entsteht eine Server-Side Template Injection (SSTI) *[10]*. Durch sie kann der Angreifer den internen Zustand der Anwendung auslesen und manipulieren oder seine Privilegien erweitern – im schlimmsten Fall direkt Code auf dem Server im Kontext der Anwendung ausführen.

Gefunden werden können SSTIs zum Beispiel, indem in Eingabefelder, die in Templates wieder ausgegeben werden, Ausdrücke mit den jeweiligen Metazeichen eingefügt werden, beispielsweise `{{3+3}}`. Gibt der Server als Antwort „6" zurück, wurde der Ausdruck von der Anwendung interpretiert, wie in *Abbildung 4* gezeigt.

In manchen Fällen kann eine ursprünglich als Cross-Site Scripting (XSS) entdeckte und klassifizierte Schwachstelle eigentlich eine Form von SSTI sein. Auch Werkzeuge wie Tplmap *[11]* – benannt in

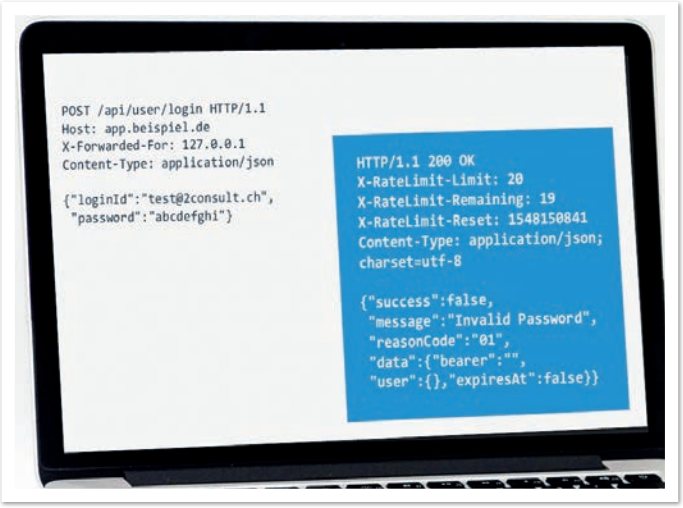


Abbildung 3: Anfrage mit eingefügtem X-Forwarded-For Header (Quelle: Frank Ully)

Anlehnung an das bekannte SQLmap für SQL-Injections – helfen beim Finden und Ausnutzen von Schwachstellen in Templates.

Entwickler verhindern Injektionen in Templates, indem sie die Korrektheit von Benutzereingaben über eine strikte Eingabevalidierung prüfen und Metazeichen entfernen sowie Funktionen zur Interpretation von Template-Ausdrücken niemals mehrfach aufrufen. Bei besonders hohen Sicherheitsanforderungen empfiehlt es sich, die Template-Umgebung in einem gut gehärteten Docker Container *[12]* vom Rest der Anwendung zu trennen.

Ebenfalls von Injections betroffen sein kann Client-seitiges Templating *[13]* in Frameworks wie React oder Angular. Hier wird jedoch im schlimmsten Fall nicht Code auf dem betroffenen Server ausgeführt, sondern JavaScript (XSS) im Browser des Opfers.

Server-Side Request Forgery (SSRF)

Was hat es mit Server-Side Request Forgery (SSRF) *[14]* auf sich? Ursache dieser Schwachstelle ist, dass eine Anwendung einen Request ausführt, der als Folge eine interne oder externe Ressource aufruft und dessen Ziel der Benutzer ganz oder teilweise vorgeben kann.

Das ist heutzutage nichts Ungewöhnliches – viele Anwendungen oder Schnittstellen nehmen in der Anfrage direkt eine URL entgegen und tun etwas damit, überprüfen beispielsweise den angegebenen Link oder erzeugen aus einem Bildpfad ein Vorschaubild. In anderen Fällen leiten sie Teile einer Anfrage an andere Systeme weiter, etwa zur Authentifizierung oder zum Monitoring *(siehe Abbildung 5)* – oder an ein Zahlungs-Gateway oder ein anderes externes API im Hintergrund.

Oft bestehen dabei Vertrauensbeziehungen, weil Server sich selbst und anderen Systemen im selben Netzwerk vertrauen. Nutzen Angreifer dies aus, können sie somit: Firewalls umgehen und aus dem Internet Dienste erreichen, die eigentlich nur auf dem lokalen Host oder im internen Netz verfügbar sind; das entfernte Netzwerk scannen; sensible interne Daten auslesen. Gegebenenfalls sind sogar reflektierte XSS-Angriffe oder das Ausführen von Code auf einem Server möglich.

In einer Cloud-Umgebung kann SSRF sehr kritisch sein. Wie in *Abbildung 6* gezeigt, können zum Beispiel über Metadaten-URLs wie `http://169.254.169.254/latest/user-data` unter Umständen direkt Zugangsdaten abgefragt werden *[15]*.

Entdeckt werden kann SSRF durch automatisierte Werkzeuge wie den Collaborator *[16]* des kostenpflichtigen Burp-Proxy. Manuell kann es gefunden werden, indem überprüft wird, ob ein Wert eines Request ganz oder teilweise an andere Systeme weitergeleitet wird und dabei manipuliert werden kann. Ebenfalls hilft ein Blick in die *ausgehenden* Firewall-Logs des Servers, um ungewöhnliche Ziele von Anfragen zu entdecken.

Eine unwirksame Schutzmaßnahme gegen SSRF ist Blacklisting, also das Blockieren von mutmaßlich bösartigen Eingaben, die zum Beispiel `127.0.0.1` oder `localhost` enthalten. Jedoch lassen sich IP-Adressen auf unterschiedliche Weise darstellen *[17]*, beispielsweise abgekürzt als `127.1`, in Dezimalnotation `2130706433` oder

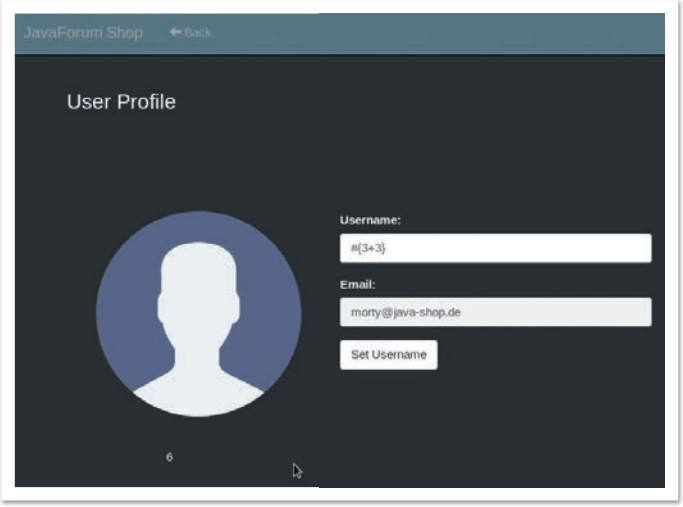


Abbildung 4: Server-Side Template Injection (Quelle: Frank Ully)

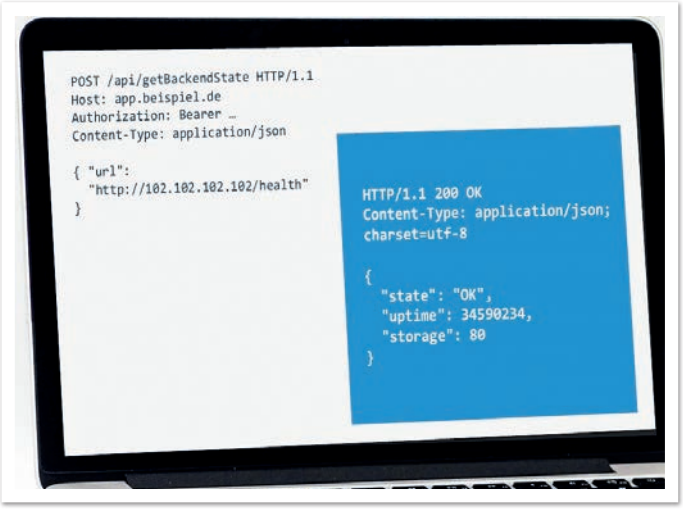


Abbildung 5: Anfrage mit externer URL, potenziell anfällig für Server-Side Request Forgery (Quelle: Frank Ully)

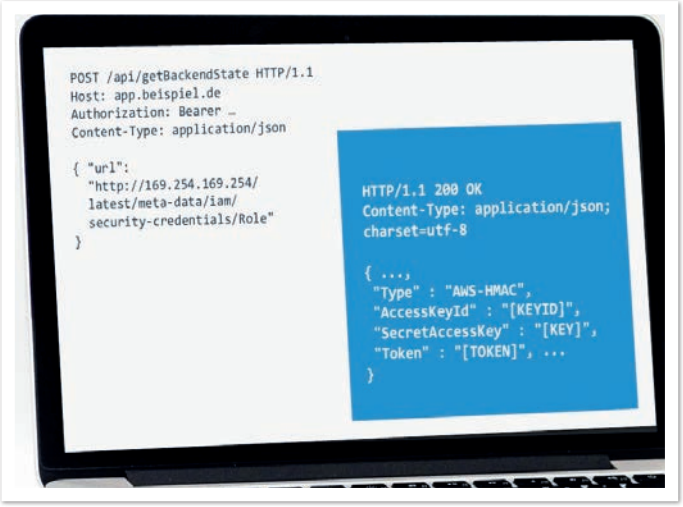


Abbildung 6: Anfrage mit veränderter URL, Server-Side Request Forgery wurde ausgenutzt (Quelle: Frank Ully)

hexadezimal `0x7F000001`. Darüber hinaus kann auch ein entsprechend konfigurierter Domainname wie `9zlhb.xip.io` als aufgelöste IP-Adresse `127.0.0.1` zurückmelden.

Zur Prävention von SSRF ist es wie immer wichtig, die vom Benutzer – oder Angreifer – gelieferten Daten so zu validieren, dass nur erwartete Werte akzeptiert werden, zum Beispiel über entsprechende Maßnahmen für Eingabevalidierung *[18]*. Idealerweise findet jedoch ein Whitelisting statt: Explizit erlaubt ist nur der Zugriff auf eine Liste mit bekannten Ressourcen – nicht nur bezogen auf Hostnamen, sondern auch auf Protokoll und Port. Als weiterer Schutz können Firewall-Regeln für ausgehenden Verkehr dienen, die einschränken, mit welchen entfernten Servern und auch mit welchen Ports auf dem lokalen System(!) die Anwendung kommunizieren kann. Schließlich sollten alle Dienste in einem Netzwerk, wie Monitoring-Tools oder Admin-Oberflächen, immer mit einer Anmeldemaske gesichert sein.

Cross-Origin Resource Sharing (CORS)

Normalerweise sind Webanwendungen von der Same Origin Policy (SOP) *[19]* geschützt, die vom Browser durchgesetzt wird.

Die SOP erlaubt Client-seitigen Skriptsprachen wie JavaScript nur dann uneingeschränkten Zugriff auf Daten in einer zweiten Webseite, wenn beide vom selben Ursprung (Englisch: *Origin*) stammen. Zwei Webseiten oder Ressourcen haben dann denselben Ursprung, wenn Protokoll, Domain und Port übereinstimmen. Beispielsweise kann ein unter `https://beispiel.de/verzeichnis1` eingebettetes JavaScript auf Elemente unter `https://beispiel.de/verzeichnis2` zugreifen, nicht aber auf `https://beispiel.de/`, weil hier eine andere Domain steht.

Cross-Origin Resource Sharing (CORS) *[20]* ist ein Mechanismus, um die Same Origin Policy gezielt aufzuweichen, und erlaubt über den Browser Kommunikation zwischen Webanwendungen, die auf unterschiedlichen Origins laufen. CORS als neuerer Standard wurde notwendig, weil mit moderner Microservice-Architektur und externen APIs Zugriffe auf entfernte Ressourcen zunehmen und andernfalls nur mit eher unschönen Abhilfen möglich sind.

Dazu verwendet CORS Header. Der `Origin`-Header wird bei Cross-Origin Requests vom Browser oder Client selbst gesetzt. Die Webanwendung kann nun mit `Access-Control-Allow-`-Headern antworten und damit dem Client signalisieren, woher (`Access-Control-Allow-Origin`) welche HTTP-Methoden (`Access-Control-Allow-Methods`) erlaubt sind, welche Header in der Anfrage enthalten sein dürfen (`Access-Control-Allow-Headers`) und ob der Client vorhandene Credentials über Cookies oder HTTP-Authentifizierung mitsenden soll (`Access-Control-Allow-Credentials`).

Anfällig *[21]* wird eine Webanwendung nun, wenn sie blindlings jeden empfangenen `Origin`-Anfrage-Header im Antwort-Header `Access-Control-Allow-Origin` reflektiert und zudem in der Antwort den Header `Access-Control-Allow-Credentials: True` setzt. In diesem Fall kann ein bösartiges JavaScript von der Domain eines Angreifers, die ein Opfer ansurft, auf einer verwundbaren Webanwendung oder API im Namen des dort angemeldeten Opfers Aktionen durchführen. Aber auch in subtileren Fällen bestehen Gefahren: Wenn das `null`-Origin erlaubt ist oder das Origin nur in Bruchstücken validiert wird und etwa – statt der Domain `beispiel.de` - `keinbeispiel.de` als erlaubter Ursprung angesehen wird *(siehe Abbildung 7)*.

Auch Anwendungen ohne `Access-Control-Allow-Credentials` sind möglicherweise anfällig, wenn als Origin die Wildcard `*` oder

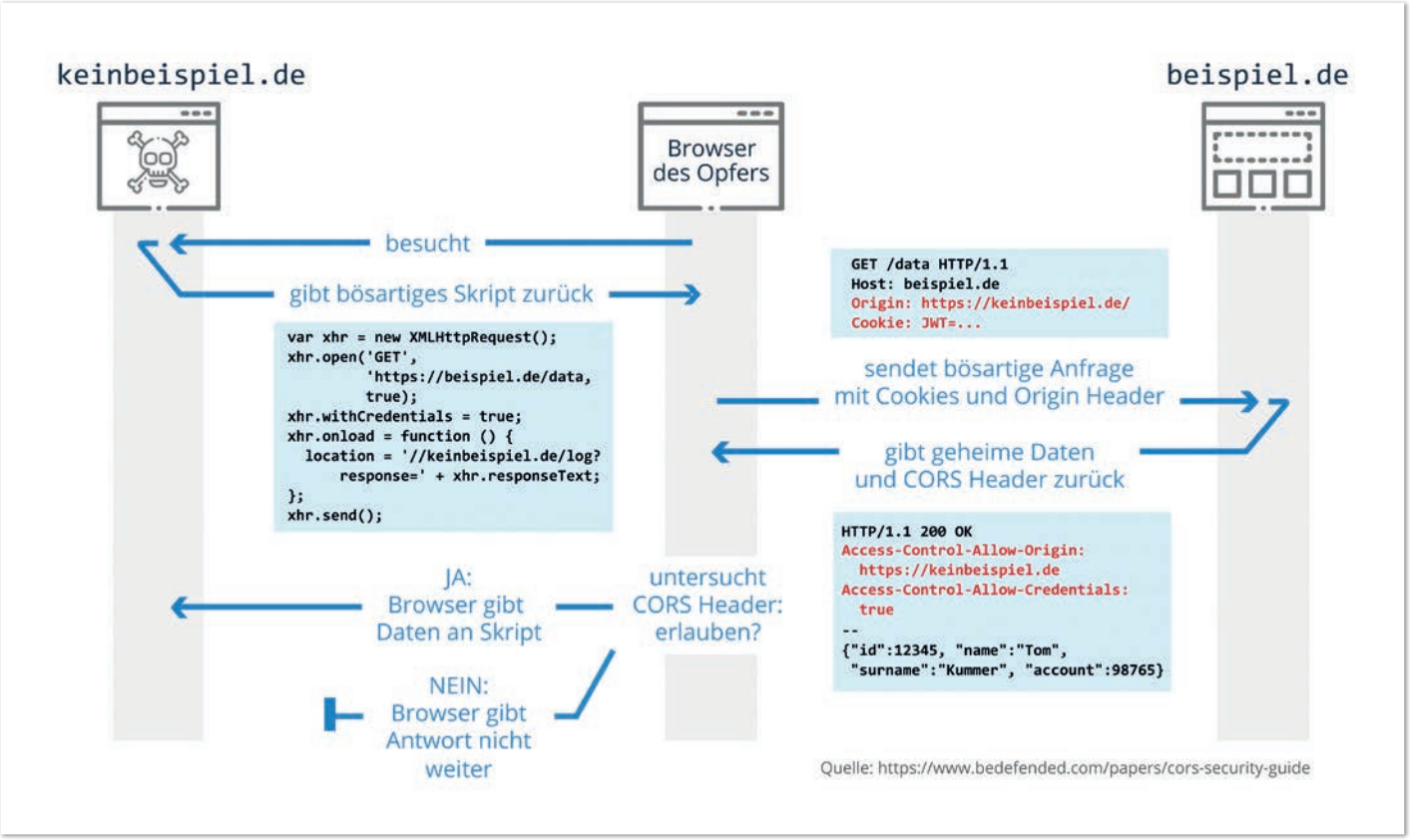


Abbildung 7: Schematischer Ablauf der Ausnutzung einer Schwachstelle im Cross-Origin Resource Sharing (in Anlehnung an [21])

`null` erlaubt sind oder der Origin nur unzureichend überprüft und dann reflektiert wird – zum Beispiel für das Umgehen von IP-Beschränkungen oder Client-Side Cache Poisoning.

Man kann solche Schwachstellen finden, indem man mit einem Proxy in Anfragen einen `Origin`-Header einfügt und wie zuvor beschrieben verändert. Auch automatisierte Werkzeuge wie CORScanner *[22]* helfen dabei, Fehlkonfigurationen aufzudecken.

Häufig sind CORS-Einstellungen in Frameworks fehlerhaft. Verhindern kann man solche Schwachstellen oft schon, indem man die Standard-Konfigurationen überprüft und CORS nur aktiviert wird, wenn es wirklich benötigt wird. Dann sollte das Origin gegen eine Whitelist strikt geprüft werden – und nicht mit regulären Ausdrücken. Weitere Maßnahmen wie das Einschränken der erlaubten Methoden in `Access-Control-Allow-Methods` oder das Setzen des Headers `Vary: Origin` erhöhen die Sicherheit zusätzlich.

OAuth und JWT – weitere Quellen unbekannterer Schwachstellen

Es gibt noch zahlreiche weitere unbekanntere und durchaus kritische Schwachstellenarten. Besonders hervorzuheben sind Schwachstellen in neueren, als besonders sicher angesehenen Authentifizierungsmechanismen wie OAuth 2.0 *[23]* und beim Einsatz von JSON Web Token (JWT) *[24]* statt normaler Sitzungstoken. Sind JWT unzureichend kryptographisch gesichert oder die Implementierung oder Konfiguration ihrer Prüfung fehlerhaft, können Angreifer Daten für Authentifizierung und Autorisierung manipulieren und so ihre Privilegien erweitern. Dagegen schützen können sich Entwickler beispielsweise, indem sie die „Best Current Practices" zu OAuth *[25]* und JWT *[26]* berücksichtigen. Diese sind derzeit allerdings noch im Entwurfsstadium.

Websicherheit grundsätzlich – weiterführende Quellen

Wie oben beschrieben, beleuchten die OWASP Top 10 – und auch dieser Artikel – nur schlaglichtartig einzelne Schwachstellen. Eine Veröffentlichung, die sich systematisch dem Entwickeln von sicheren Webanwendungen widmet, ist der „Application Security Verification Standard" (ASVS) *[27]*. Er gibt in verschiedenen Kategorien detaillierte Sicherheitsanforderungen für Anwendungen vor, die von Stakeholdern genutzt werden können, um zu bestimmen, was eine sicher entwickelte Anwendung ist und wie das erreicht und getestet werden kann.

Der „OWASP Testing Guide" *[28]* beschreibt ein umfassendes Framework für Penetrationstests, das Organisationen an ihre Bedürfnisse anpassen können, und enthält einen detaillierten und sehr technischen Leitfaden, anhand dessen Tester einzelne Sicherheitsprobleme aufdecken können. Während der Testing Guide die Überprüfung von Webanwendungen in *Grey-Box*-Szenarien beschreibt, also mit nur teilweise verfügbaren Informationen, gibt der „Code Review Guide" *[29]* Hinweise, wie bei *White-Box*-Analysen mit vorliegendem Quelltext Lücken gefunden werden können.

Analog zu den Top 10 mit deren Blick auf Schwachstellen aus Sicht der Angreifer beschreiben die „Top 10 Proactive Controls" *[30]* aus Sicht der Verteidiger, welche Sicherheitsmaßnahmen Anwendungen in jedem Fall enthalten sollten. Zu jedem Punkt – unter anderem Eingabevalidierung und Ausgabekodierung, Verschlüsselung und Zugriffskontrolle – wird dargestellt, was darunter zu verstehen ist, wie er umgesetzt werden könnte, welche Schwachstellen dadurch verhindert werden und welche Tools und Quellen dabei helfen.

Selbst ausprobieren – Werkzeuge und Anwendungen

Am einfachsten lernt man durchs Machen. Um sich eine Übungsumgebung für einfache bis fortgeschrittene Webapp-Lücken einzurichten, egal ob man Projektleiter, Entwickler oder Tester ist, bietet sich eine Kombination von Interception Proxy und absichtlich verwundbarer Software an.

OWASP bietet den kostenfreien Zed Attack Proxy (ZAP) an *[31]*, der teilweise auch automatisiert Schwachstellen findet. Als Alternative kann der Burp-Proxy in der kostenlosen Community Edition *[5]* dienen, mit allerdings deutlich eingeschränktem Funktionsumfang.

Ziel der Übungsangriffe ist am besten eine Webanwendung, in die mit Vorsatz Lücken eingebaut wurden. Davon gibt es inzwischen eine unübersehbare Menge – entweder lokal als Installationspaket, virtuelle Maschine oder Docker-Image *[32]* eingesetzt oder direkt online im Internet *[33]*.

Hervorzuheben ist der „OWASP Juice Shop" *[34]*, der lokal wie auch in Cloud-Umgebungen vielfältig installiert werden kann. Dieser sprichwörtliche Saftladen ist nach modernem Ansatz als JavaScript-lastige Frontend-Anwendung (Angular) entwickelt, die auf REST-APIs (Node.js mit Express) zugreift. Der Juice Shop wird regelmäßig mit aktuellen Schwachstellen bestückt, etwa im Zuge des „Google Summer of Code". Er bietet eine sehr umfangreiche Dokumentation *[35]*, die hilft, das Anzapfen des Saftladens vorzubereiten und Schwachstellen systematisch zu finden, und enthält zudem einen ausführlichen Lösungsleitfaden.

Quellen

[1] *https://www.owasp.org/index.php/Main_Page*
[2] *https://www.owasp.org/images/9/90/OWASP_Top_10-2017_de_V1.0.pdf*
[3] *https://github.com/OWASP/Top10/milestones*
[4] *https://www.owasp.org/index.php/Top_10-2017_Foreword*
[5] *https://portswigger.net/burp*
[6] *https://nordicapis.com/everything-you-need-to-know-about-api-rate-limiting/*
[7] *https://portswigger.net/kb/issues/00400110_spoofable-client-ip-address*
[8] *https://portswigger.net/blog/practical-web-cache-poisoning*
[9] *https://www.owasp.org/index.php/Top_10-2017_A1-Injection*
[10] *https://portswigger.net/blog/server-side-template-injection*
[11] *https://github.com/epinna/tplmap*
[12] *https://dev.to/petermbenjamin/docker-security-best-practices-45ih*
[13] *https://portswigger.net/kb/issues/00200308_client-side-template-injection*
[14] *https://portswigger.net/web-security/ssrf*
[15] *https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Request%20Forgery*

[16] *https://portswigger.net/burp/documentation/collaborator*
[17] *https://www.psyon.org/tools/ip_address_converter.php?ip=127.0.0.1*
[18] *https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Input_Validation_Cheat_Sheet.md*
[19] *https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy*
[20] *https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS*
[21] *https://www.bedefended.com/papers/cors-security-guide*
[22] *https://github.com/chenjj/CORScanner*
[23] *https://github.com/koenbuyens/Vulnerable-OAuth-2.0-Applications/blob/master/authorizationcode_tester.md*
[24] *https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_Cheat_Sheet_for_Java.html*
[25] *https://tools.ietf.org/html/draft-ietf-oauth-security-topics-13*
[26] *https://tools.ietf.org/html/draft-ietf-oauth-jwt-bcp-04*
[27] *https://github.com/OWASP/ASVS*
[28] *https://www.owasp.org/index.php/OWASP_Testing_Project*
[29] *https://www.owasp.org/index.php/Category:OWASP_Code_Review_Project*
[30] *https://www.owasp.org/index.php/OWASP_Proactive_Controls*
[31] *https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project*
[32] *https://www.owasp.org/index.php/OWASP_Vulnerable_Web_Applications_Directory_Project/Pages/Offline*
[33] *https://github.com/geeksonsecurity/vuln-web-apps*
[34] *https://www.owasp.org/index.php/OWASP_Juice_Shop_Project*
[35] *https://bkimminich.gitbooks.io/pwning-owasp-juice-shop/*

Frank Ully
Oneconsult Deutschland
frank.ully@oneconsult.com

Frank Ully schloss sein Studium an der Hochschule Karlsruhe als Diplom-Technikredakteur ab. Er baute die Dokumentationsabteilung eines Softwareherstellers auf und leitete sie einige Jahre. Später war er dort für IT-Sicherheit verantwortlich. Berufsbegleitend absolvierte er den Masterstudiengang Security Management an der TH Brandenburg und zertifizierte sich zum Offensive Security Certified Expert (OSCE) sowie Professional (OSCP); er erwarb zudem weitere Zertifikate. Im November 2017 begann er seine Arbeit bei Oneconsult Deutschland, seit April 2018 ist er Senior Penetration Tester und Security Consultant.

„Unternehmen werden noch abhängiger von Daten sein“

Der Fachinformatiker Tobias Schrödel ist Entertainer, Buchautor und gefragter TV-Experte. In einer Mischung aus Informationsvortrag und Unterhaltung demonstriert er auf Konferenzen, wie sicher unsere Daten und Geräte im Internet und in Clouds wirklich sind. In unserem Interview mit Schrödel geht es um den Sicherheitsfaktor Mensch und die Frage, warum die Investitionen in IT-Sicherheit für Unternehmen zukünftig nicht geringer werden.

Herr Schrödel, welche ist denn Ihre Lieblings-Sicherheitslücke?

Tobias Schrödel: Ich habe sogar zwei. Es gibt einen kleinen Trick, wie man zumindest bei nicht IP-basierten Hotel-Videosystemen kostenlos den Pay-TV-Kanal gucken kann. Und wenn ich das in meinen Vorträgen zeige, ist die Stimmung eigentlich immer sehr gut und die Leute fangen an, mitzuschreiben, wie das funktioniert. Leider gibt es heutzutage kaum noch solche Systeme. Die zweite Lieblingslücke ist das Caller-ID-Spoofing. Dabei lasse ich bei einem Anruf eine beliebige Absendernummer im Display anzeigen. Das kommt selbst abends beim Bier noch gut an, wenn ich jemanden mit seiner Büronummer anrufe, wenn da garantiert niemand mehr sitzt.

Welche sind denn die am häufigsten genutzten Sicherheitslücken?

Tobias Schrödel: Es ist schwierig, das allgemein zu sagen, weil es natürlich darauf ankommt, um welche Art von Angriff es sich handelt. Ist es ein gezielter Angriff, dann wird der Angreifer sicher andere Wege als bei einem Zufallsopfer beschreiten. Beobachtet man Presse- und Polizeimeldungen, ist es aber tatsächlich so, dass Ransomware – also Erpresser-Viren – immer noch mit Abstand am meisten Schaden anrichten. Dafür zahlen die Leute am meisten.

Google hat einmal verkündet, das Passwort abschaffen zu wollen. Über eine Schnittstelle sollen Geräte zu jeder Zeit Daten speichern und auswerten, zum Beispiel über den Standort und das Stimmmuster oder die Gesichtserkennung. Passwörter sind häufig ein Schwachpunkt, wenn es um IT-Sicherheit geht. Müssten daher nicht alle Sicherheitsexperten über Googles Pläne froh sein?

Tobias Schrödel: Sie haben natürlich völlig recht. Man wäre froh, wenn es eine akzeptierte und verbreitete Alternative zum Passwort gäbe. Das Passwort führt gerade bei nicht-IT-affinen Menschen immer wieder zu Problemen. Entweder weil sie es vergessen oder einfach ein schwaches Passwort wählen. Ich denke, dass der geplante Google-Login-Mechanismus technisch wunderbar funktionieren wird. Offensichtlich kombiniert er viele körpereigene Merkmale, sodass es statistisch gesehen sehr unwahrscheinlich ist, dass diese auf einen Angreifer zutreffen. Das ist mit der Sicherheit eines Passworts gleichzusetzen. Aber auf der anderen Seite heißt das, dass Google so viele personenbezogene Merkmale meines Körpers speichern und auch auswerten kann, dass auch ganz andere Rückschlüsse gezogen werden können. Zum Beispiel, sofern diese Daten es hergeben, ob ich einen Herzfehler habe. Und da bin ich dann schon wieder kritisch, ob Google der richtige Ansprechpartner ist. Ich persönlich setze lieber auf FIDO2, das ohne biometrische Merkmale auskommt. Wenn sich das System durchsetzt, brauchen wir bei vorher definierten Seiten gar kein Passwort mehr, wenn wir zum Beispiel unser Handy oder einen Token dabeihaben.

Immer mehr Menschen und Unternehmen nutzen Cloud-Speicher und vertrauen ihre Daten Datenbanken und Administratoren an. Warum ist es aber praktisch unmöglich, für absolute Datensicherheit zu sorgen?

Tobias Schrödel: Grundsätzlich ist es so: Wo Technik im Einsatz ist, wird es immer wieder Lücken geben, weil kein System fehlerfrei sein kann. Und wo Menschen beteiligt sind, werden auch Fehler gemacht. Das heißt, wir haben potenziell immer eine Lücke oder ein Problem, das theoretisch ausgenutzt werden kann. Die Frage ist, ob man sich nicht anders schützt, indem man Angriffe sofort erkennt oder die Mauer so hoch setzt, dass es nicht einfach wird, reinzukommen.

Sie erwähnten gerade den Menschen als Faktor, wenn es um IT-Sicherheit geht. Ist der Mensch denn die Schwachstelle?

Tobias Schrödel: Ehrlich gesagt: ja. Und zwar, weil unsere Systeme immer besser werden, ausgeklügelter arbeiten und Anomalien erkennen. Deswegen ist es für einen Angreifer auch durchaus interessant, einfach auf Personen zu zielen. Man darf eines nicht vergessen: Es gibt ja nicht nur die Administratoren, die sich auskennen und sehr bewusst arbeiten. Um mit Informationen und Daten arbeiten zu können, brauchen auch normale Anwender Zugriff auf die Systeme und die sind oft anders ausgebildet, buchhalterisch zum Beispiel. Diese Leute auf Glatteis zu führen ist deutlich einfacher und – da sie ja auch Zugriff auf Daten haben – sehr effektiv. Ja, ich glaube, dass deswegen der Mensch eine der interessantesten Angriffsflächen ist und die Technik immer weiter in den Hintergrund rückt.

Wird denn in Unternehmen zu wenig Weiterbildung angeboten, zum Beispiel in Awareness-Schulungen?

Tobias Schrödel: Es wäre ja blöd, wenn ich sagen würde, dass dies nicht der Fall wäre. Ich lebe ja schließlich von Awareness-Trainings (lacht). Also natürlich glaube ich, dass ein entsprechendes Training von Personen sehr wichtig ist, denn es ist vergleichsweise günstig. Eine neue Technik, zum Beispiel ein komplett neues Firewall-System, ist meines Erachtens viel teurer, als wenn man seine Mitarbeiter einmal zu einem zweistündigen Awareness-Training schickt. Es ist schon sehr viel gewonnen, wenn die Mitarbeiter selbst in der Lage sind, Angriffe zu erkennen. Sie sollen ja nicht lernen, sie zu verhindern, sondern die IT informieren, wenn ihnen etwas komisch vorkommt oder sie irgendetwas nicht verstehen. Die IT-Abteilungen müssen dann prüfen, ob es sich um einen Angriff handelt.

Auf der anderen Seite wird bereits immer mehr Arbeitszeit in den Aufbau und Betrieb von IT-Sicherheit investiert. Ist das ein Trend, an den sich große Unternehmen gewöhnen müssen?

Tobias Schrödel: Ja, ich denke schon. Das lässt sich nicht ändern, und zwar aus einem einfachen Grund: Unternehmen werden viel abhängiger von Daten und deren Verarbeitung sein. Es gibt fast kein Unternehmen mehr, das ohne IT auskäme. Und wenn die IT ausfallen würde, würde erst einmal zwei Tage alles stillstehen, bis man mit einem Notfallplan wieder die Arbeit aufnehmen kann. Das hat ja der Verschlüsselungstrojaner Locki gezeigt, der zum Beispiel die IT des Lukas-Krankenhauses in Neuss befallen hat. Oder ganz aktuell EMOTET, der Virus ist schon länger bekannt, legt aber trotzdem reihenweise und fast täglich Behörden und Firmen auf der ganzen Welt lahm.

Heutzutage ist ja praktisch jeder online und durch soziale Netzwerke vernetzt. Sei es nun über Smartphones, Tablets oder den PC. Hinzu kommen SmartHome-Technologien wie Licht- und Heizungssteuerung per App. Werden kriminelle Hacker es in Zukunft einfacher haben, Schaden anzurichten und an sensible Informationen zu gelangen?

Tobias Schrödel: Schaden anrichten ja, an sensible Informationen gelangen, vielleicht. Wenn Sie heute die Suchmaschine für smarte Geräte – Shodan – nehmen und sich dort mal durchklicken, dann wird Ihnen schwindelig, was da schon alles vernetzt ist und was jetzt schon offen im Internet erreichbar ist. Ja, das wird ein großes Problem werden – nicht nur weil die Leute unbedacht sind und irgendwelche Sachen ohne Passwort ins Netz stellen und erreichbar machen, sondern auch, weil wir einfach keine Update-Philosophie haben. Wer denkt denn daran, seinen Lichtschalter oder Kühlschrank upzudaten? Da kommen wir ja irgendwann dahin, dass ich 80 bis 90 Geräte updaten muss. Das macht ja kein Mensch. Problematisch wird das Internet der Dinge aber auch deshalb, weil wir uns noch gar nicht vorstellen können, wo dann überall Risiken lauern. Eine smarte Glühbirne, die ein Hacker in einer Leselampe ausschalten kann, ist sicherlich nur ärgerlich. Ist diese Glühbirne aber das Rotlicht einer Ampel, dann ist sie plötzlich potenziell lebensgefährlich.

Welche Trends im Sicherheitsbereich sehen Sie denn für die nächsten Jahre?

Tobias Schrödel: Ich glaube, dass sich Erpresser-Software und das Verschlüsseln von Daten auf Rechnern noch ganz massiv bei uns einnisten werden. Es fängt beim Smartphone an und wird über das Internet der Dinge weitergehen. Wenn Sie zum Beispiel in der Zukunft ein Auto haben, das per Software gesteuert fährt, dann wird auch dort irgendwann ein Erpresser mit Ransomware kommen. Und wenn man dann nicht 500 Euro überweist, dann fährt das Auto nur noch 60 km/h und nicht schneller. Das ist ja das Problem an dieser ganzen Ransomware: Die verlangen in der Regel nur kleinere Beträge, weil die Leute in der Lage sind, genau das zu bezahlen, um ihre Daten wiederzubekommen.

Ganz anders der bereits erwähnte EMOTET. Da langen die Erpresser richtig zu, fordern teilweise hohe fünf- oder gar sechsstellige Beträge, wenn sie große Firmen oder Landkreise erwischen und lahmlegen. Die Stadt Baltimore hat durch einen solchen Angriff einen Schaden von 16 Millionen Dollar gehabt. Sie haben auf Anraten des FBI lieber neue Systeme gekauft, anstatt den Verbrechern knapp 100.000 US-Dollar zu zahlen. Es sei nämlich nicht sicher, ob die Erpresser die Systeme wirklich wieder freigegeben hätten. Ich unterstütze das natürlich auch nicht, aber ich persönlich glaube, dass die Stadt Baltimore da schlecht beraten wurde. Wahrscheinlich hat ihnen keiner gesagt hat, dass eine Polizeibehörde niemals dazu raten wird, Erpressern Geld zu zahlen. Ich gehe aber jede Wette ein, dass die Erpresser die Daten freigegeben hätten. Schon aus wirtschaftlicher Sicht. Wenn sich das nämlich herumspricht, dass man seine Daten nach der Zahlung wiederbekommt, dann zahlen zukünftige Opfer deutlich schneller. Andersrum aber niemand mehr.

Vielen Dank für das Interview, Tobias!

Sharing is caring oder geteilte Freude ist doppelte Freude

Michal Harakal

Laut aktueller Zahlen von Google sind über eine Millionen Apps für Android-Geräte verfügbar. Die Zahlen im App Store von Apple sind mit über zwei Millionen Apps noch höher. Auch die Anzahl der Entwickler ist in der letzten Zeit enorm gestiegen. Aber nicht nur Plattformen und Entwickler konkurrieren untereinander, sondern auch die Art und Weise, wie Apps entwickelt werden. Angefangen mit unterschiedlichen Bibliotheken (mit Picasso, Fresco, Glide und Coil gibt es bereits vier für Android, die sich nur um Image Loading kümmern), über Design Patterns (MVP, MVVM oder MVI) bis hin zu kompletten Frameworks wie React Nativ oder Flutter – alle haben nur ein Ziel: Schnell und einfach qualitativ hochwertige Apps an die Nutzer bringen.

Im Bereich mobiler Apps ist man von Anfang an auf der Suche nach dem perfekten Weg. Angefangen mit HTML5-basierten Web-Apps (Cordova, Ionic), mit schwergewichtigen Cross Platform Frameworks wie Xamarin, bis bin zu hybriden Ansätzen wie React Native, Vue Native oder Flutter. Als eine ernstzunehmende Alternative zeichnet sich Codesharing mit Kotlin in Multiplattform-Projekten ab, wo in Kotlin geschriebener Code auf unterschiedlichen Plattformen wiederverwendet werden kann. Der Fokus liegt auf dem Code, der keine oder geringe Abhängigkeiten auf der Plattform hat. Üblicherweise handelt es sich um Fachlogik, Daten-Modelle und sogar Netzwerkkommunikation oder lokale Datenhaltung in Dateien oder in einer Datenbank.

Kotlin

Im Mai 2019 hat Google auf einer hauseigenen Entwickler-Konferenz Google I/O Kotlin als die bevorzugte Sprache für Android deklariert – und das nur ein Jahr, nachdem Kotlin als unterstützte Sprache ins Portfolio genommen wurde. Inzwischen nutzt man die volle Kraft der Sprache, was sich zum Beispiel in der Unterstützung von Koroutinen in Jetpack Libraries zeigt. Es werden auch neue Wege gegangen. Bei der Implementierung einer neuen UI-Bibliothek Jetpack Compose kommt ein neues Kotlin Compiler Plug-in zum Einsatz, das Sprachkonstrukte erlaubt, die mit Java und Java-Annotationen nur begrenzt möglich sind.

Das wachsende Ökosystem von Kotlin „first" Libraries, Blog-Beiträgen und Code-Beispielen weist nicht nur auf eine steigende Popularität bei Entwicklern hin, sondern auch auf eine lebendige Community. Wenn es um die native Entwicklung für Android geht, hat Kotlin bereits einen festen Platz eingenommen. Die Programmiersprache spielt aber auch beim Codesharing eine bedeutende Rolle.

Die Vorteile von Kotlin wie Lesbarkeit, starke statische Typisierung, Nullsicherheit und die hervorragende Interoperabilität mit Java werden bereits an anderen Stellen sehr gut beschrieben. In diesem Artikel werden wir uns auf das Thema Codesharing auf den mobilen Plattformen fokussieren.

Codesharing mit Kotlin

Was muss man also machen, um möglichst viel Code zwischen den mobilen Plattformen zu teilen? Wir schauen uns zuerst an, welche Möglichkeiten Kotlin, Kotlin/Native und Kotlin Multiplatform bieten. Danach schauen wir uns eine existierende Kotlin-Multiplatform-App an.

Kotlin/Native und Kotlin Multiplatform

Kotlin/Native ist eine Technologie zum Kompilieren vom Kotlin-Quellcode in einen nativen binären Code, der ohne eine virtuelle Maschine laufen kann. Es beinhaltet ein Backend für LLVM und native Implementierungen der Kotlin-Standard-Bibliothek für die jeweilige Zielplattform. Auch Kotlin/Native verfolgt das Ziel, eine exzellente Interoperabilität mit den existierenden Plattformen anzubieten. Am wichtigsten ist die Möglichkeit, die bestehenden C-Bibliotheken aus Kotlin zu nutzen. Dies kann man mit dem Werkzeug „C Interop" erreichen, das alle notwendigen Dateien für Kotlin schnell generieren kann.

Kompiliert als ein Apple Framework, bietet Kotlin/Native bidirektionale Interoperabilität mit Objective-C/Swift. Die bereits in Objective-C geschriebenen Frameworks und Bibliotheken können im Kotlin-Code benutzt werden. Die Kotlin-Module können umgekehrt in Swift/Objective-C Code benutzt werden.

Mit Kotlin/Native können wir Kotlin-Code für unterschiedliche Plattformen kompilieren. Wir können sogar native Bibliotheken aus Kotlin aufrufen. Wie erreichen wir aber, dass man in Common Code die Plattform-spezifischen Aufrufe Plattform-neutral gestaltet? Es bietet sich eine Lösung mit Interfaces in Common Code und deren Implementierungen in Plattform-spezifischen Modulen an. Kotlin geht einen Schritt weiter und bringt mit den neuen reservierten Schlüsselworten „expected" und „actual" einen Mechanismus mit, der im Common Code die gewünschten Methoden definiert und auf den Zielplattformen die aktuelle Implementierung nutzt. Zusätzlich ist es mit dem Schlüsselwort „alias" möglich, schon bestehende Bibliotheken mit dem „actual"-Schlüsselwort direkt einzubinden. Und das alles ohne zusätzliche Wrappers oder Brücken. Genauso freut sich der Entwickler, der eventuell Stacktraces lesen muss.

Als Beispiel zeigen wir, wie man mit dem „expected/actual"-Konstrukt einfache Key/Value-Paare speichern kann. Als Erstes definieren wir eine einfache Klasse *(siehe Listing 1)*, die Methoden zum Lesen und Schreiben von String- und Boolean-Werten zur Verfügung stellt. Die Android-spezifische Implementierung benutzt „SharedPreferences" und ist in *Listing 2* dargestellt. *Listing 3* zeigt die iOS-Variante.

Um mit Kotlin Multiplatform zu starten, ist es am einfachsten, mit dem Assistenten für neue Projekte in der aktuellen Version der IntelliJ IDEA von JetBrains anzufangen. In der Sektion „Kotlin" wird ein Mobile-Android- beziehungsweise iOS-Projekt-Template angeboten, das nicht nur entsprechende Gradle-Module und „build.gradle"-Dateien erzeugt, sondern auch die dazu notwendige Ordnerstruktur und Kotlin-Dateien mit Beispielen für die Nutzung von „expected" und „actual". Die Ordnernamen folgen einer verständlichen Namenskonvention, anhand derer man darauf schließen kann, dass es sich bei dem Ordner „commonMain" beispielsweise um den Teil mit Code zum Teilen handelt und bei „iosMain" iOS-Geräte die Zielplattform sind. Genauso sind die „Codelabs" auf der offiziellen Kotlin-Website mit den Schritt-für-Schritt-Anleitungen zu empfehlen *[1]*. Als Beispiel kann man sich die offizielle App zur Kotlin-Konferenz von JetBrains „Kotlinkof-app" *[2]* anschauen, genauso wie die „DroidconKotlin"-App von Touchlab *[3]*, die SQLDelight *[4]* als Multiplatform-Bibliothek zum Speichern von Daten in SQlite-Datenbank benutzt.

Konferenz-iOS- und -Android-App mit Kotlin Multiplatform

Um die Fähigkeiten von Kotlin Multiplatform zu zeigen, schauen wir uns ein Open-Source-Projekt an: eine existierende native Konferenz-App aus dem DukeCon-Projekt *[5]*. Anhand dieses Beispiels können wir die Domain-Fachbegriffe noch verständlicher machen.

Die App zeigt eine Liste mit Vorträgen, gegliedert nach Konferenztagen. Nach dem Klick/Touch auf einen Vortrag werden Details zu diesem angezeigt. Die App sollte auch ohne Internetverbindung (was bei einer Konferenz ein nicht seltener Umstand ist) gespeicherte Daten anzeigen. Weiterhin werden die Referenten, ihre Biografien

und weitere Vorträge angezeigt. Damit haben wir die funktionalen Anforderungen vereinfacht dargestellt. Um die gewünschte Funktionalität zu erreichen, muss schon einiges unter der Haube passieren. Die Konferenz-Daten werden aus dem Netzwerk geholt, in Domain-Modelle umgewandelt, nach Tagen gefiltert und sortiert. In dem Projekt werden bereits zwei unterschiedliche Backends unterstützt. Für den Offline-Einsatz werden die Daten zusätzlich lokal abgespeichert.

App-Architektur

Die Softwarearchitektur einer App kann maßgeblich dazu beitragen, wie viel von dem in Kotlin geschriebenen Code wiederverwendet werden kann. Beim Entwurf einer Softwarearchitektur für ein Kotlin-Multiplatform-Projekt steht neben den üblichen nicht-funktionalen Anforderungen wie Lesbarkeit oder Wartbarkeit nämlich der hohe Grad an Wiederverwendbarkeit von Code im Vordergrund. Eine einmal programmierte Lösung spart nicht nur Aufwand nach dem DRY-Prinzip („Don't Repeat Yourself"), sondern kann so auch eine höhere Qualität und fachliche Korrektheit durch mehrfaches Anwenden erreichen.

Dafür bieten sich Softwarearchitekturen an, die Abhängigkeiten von dem Betriebssystem, dem App-Framework und den UI-Libraries minimieren. Dieser Ansatz empfiehlt sich auch unabhängig vom Sharing, gerade bei Android-Apps, wo die JVM Unit Tests ohne Android-Emulator deutlich schneller laufen.

Zusätzlich ist bei Kotlin-Multiplatform-Klassen darauf zu achten, dass keine Abhängigkeiten aus dem java*-Paketbereich importiert und genutzt werden. Die ausgezeichnete „Kotlin Standard Library" und die „kotlinx"-Bibliotheken (Serialisierung und Koroutinen) machen es auch viel einfacher, da man keinen Multiplattform-fähigen Ersatz für zum Beispiel RxJava oder Guava suchen muss. Viele hilfreiche Klassen und Methoden werden somit schon von Hause aus mitgeliefert (Collections, Sequences oder Koroutinen).

Die Architektur, die wir für die App gewählt haben, ist stark an „Clean Architecture" von Uncle Bob *[6]* angelehnt. Die Tatsache, dass aus Sicht der Clean Architecture das Android Framework nur am Rande steht und der Fokus auf Entities und Use Cases in der Domain liegt, macht es einfach, eine Umsetzung unabhängig von der Plattform zu gestalten. Die Koroutinen wiederum vereinfachen deutlich, wie man mit dem Multithreading auf unterschiedlichen Plattformen umgeht. Eine vereinfachte Struktur unserer App ist in *Abbildung 1* zu sehen.

Das Herzstück ist die CommonLib, die den Teil des Codes beinhaltet, den man auf Android und iOS wiederverwenden kann. Koroutinen und die „kotlinx"-Serialisierung werden zusätzlich zur Kotlin Standard Library benutzt und in *Abbildung 1* als ein wichtiger Baustein dargestellt.

Durch eine weitere Trennung der Implementierung, die sich um die Netzwerkkommunikation kümmert und den Code, der für das

```
expect fun ApplicationStorage(): ApplicationStorage

interface ApplicationStorage {
    fun putBoolean(key: String, value: Boolean)
    fun getBoolean(key: String, default: Boolean): Boolean
    fun putString(key: String, value: String)
    fun getString(key: String): String?
}
```

Listing 1: Deklaration in Common Code

```
actual fun ApplicationStorage(): ApplicationStorage = AndroidStorage

object AndroidStorage : ApplicationStorage {
    private val sharedPreferences by lazy {
        PreferenceManager
           .getDefaultSharedPreferences(appAndroidContext)
    }

    override fun putBoolean(key: String, value: Boolean) {
        sharedPreferences.edit()
                         .putBoolean(key, value)
                         .apply()
    }

    override fun getBoolean(key: String, default: Boolean): Boolean =
          sharedPreferences.getBoolean(key, default)

    override fun putString(key: String, value: String) {
        sharedPreferences.edit()
                         .putString(key, value)
                         .apply()
    }

    override fun getString(key: String): String? = sharedPreferences.getString(key, null)
}
```

Listing 2: Android-Implementierung mit SharedPreferences

```
actual fun ApplicationStorage(): ApplicationStorage = IosStorage()

internal class IosStorage : ApplicationStorage {
    private val delegate: NSUserDefaults = NSUserDefaults.standardUserDefaults()

    override fun putBoolean(key: String, value: Boolean) {
        delegate.setBool(value, key)
    }

    override fun getBoolean(key: String, defaultValue: Boolean): Boolean =
        if (hasKey(key)) delegate.boolForKey(key) else defaultValue

    override fun putString(key: String, value: String) {
        delegate.setObject(value, key)
    }

    override fun getString(key: String): String? = delegate.stringForKey(key)

    private fun hasKey(key: String): Boolean = delegate.objectForKey(key) != null
}
```

Listing 3: iOS-Implementierung mit NSUserDefaults

Caching und die lokale Datenspeicherung zuständig ist, erzeugen wir zwar einen kleinen Mehraufwand (zum Beispiel bei den notwendigen ModelMappers), erreichen dafür allerdings die Freiheit, unterschiedliche Datenklassen für Netzwerk und Speicher zu nutzen. Dies wiederum ermöglicht eine Austauschbarkeit von Kommunikationsmodulen für unterschiedliche Konferenz-Backend-Systeme mit unterschiedlichen Datentransfer-Objekten und Endpunkten. Bei größeren Datenmengen kann man die Daten im lokalen Cache, zum Beispiel mit einer SQLite-Multiplattform-Datenbank-Bibliothek wie SQLDelight, speichern. In unserem einfachen Fall und bei den Datenmengen, die bei einer üblichen Konferenz anfallen, speichern wir direkt die deserialisierten Daten als „String" in einem einfachen persistenten Key/Value Store. Zum Absetzen von Netzwerk-Aufrufen hat sich die Multiplatform-Bibliothek „ktor.io" von JetBrains bewährt, die den Nutzer wählen lässt, welchen „http client" er auf der Zielplattform benutzen möchte. So kommt bei Android „Okhttp" zum Einsatz. Zusätzlich bringt „ktor.io" weitere nette Funktionen (Features) wie JSON-Deserialisierung oder Logging mit.

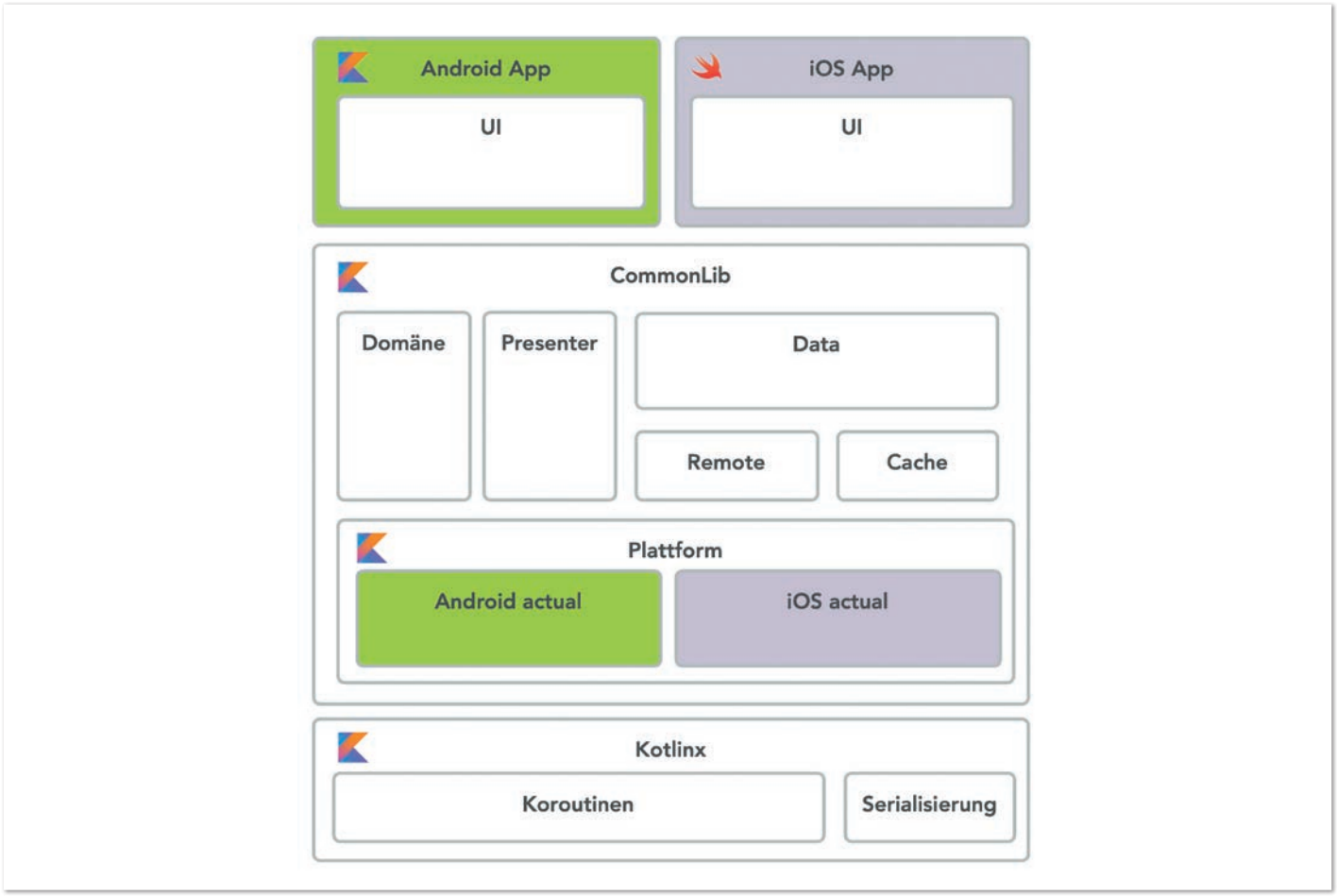


Abbildung 1: App-Module (© Michal Harakal)

Um die Verarbeitung von Datum und Zeit, was bei einer Konferenz-App einen wichtigen Teil der Logik darstellt, muss man sich noch selbst kümmern. Das gewohnte Java-Date-API wurde mit dem ThreeTen Backport zwar auch an ältere Android-Geräte gebracht, läuft aber weiterhin nur auf JVM und kann somit nicht Teil der CommonLib werden. Als Lösung kommt wieder „ktor.io" zum Einsatz, das im „utils"-Paket ein einfaches API zur Verfügung stellt, das mit ein paar Extension-Methoden für das Projekt ausreichend ist.

Die Präsentationsschicht ist mit einem „Model View Presenter" (MVP) und einer passiven View als Design Pattern umgesetzt. Hierbei sind nicht nur die Modelle, sondern sogar die Presenter als Multiplattform-Code implementiert. Somit sind nur die Views samt Layout an der Zielplattform mit dazugehörigem UI-Framework und der hauseigenen Sprache zu implementieren, wo man dann die volle Kraft der nativen Plattform, wie zum Beispiel Animationen, ohne Kompromisse und ohne zusätzliche Nachteile an der Performance nutzen kann. Von Google wird auf Android eine Nutzung von „Model View ViewModel" (MVVM) als Design Pattern empfohlen, bei dem die „LiveData"- und „LiveModel"-Klassen aus der Jetpack-Architektur-Komponenten-Bibliothek stammen und die sich exzellent in den Lebenszyklus einer mobilen App einbinden lassen. MVVM ist auch für iOS mit kleinen Anpassungen nutzbar.

Wir bauen die App

Nachdem wir gelernt haben, was die App macht und wie die App-Architektur aussieht, untersuchen wir die Ordnerstruktur. Um ein Kotlin-Multiplatform-Projekt zu bauen, eignet sich Gradle mit entsprechenden Plug-ins am besten. Bei komplexeren Apps empfiehlt es sich, natürlich abhängig von der angewendeten Software-Architektur, auch die Schichten und Features in separate Gradle-Module zu unterteilen. Gradle 5.x bietet mit den Funktionen „METADATA", „composite build" und „Bill Of Materials" (BOM) weitere wichtige Instrumente, die die Entwicklung von Kotlin-Multiplatform-Projekten unterstützen.

CommonLib ist ein separates Gradle-Modul und kann noch als eigenständiges Artefakt ins Artefakte-Management-System publiziert werden.

Für Android und Java bieten sich Maven Dependencies an. Bei iOS könnte man mit „cocoapods" arbeiten, die inzwischen bereits in dem Kotlin Multiplatform Gradle Plug-in direkt unterstützt werden. Die Anbindung der CommonLib in die Android- oder iOS-App ist direkt möglich, genauso wie man es mit einer beliebigen nativen Bibliothek machen würde. Mit Maven Dependencies bei Android oder mit „cocoapods" bei iOS. Dadurch, dass alle Gradle-Module in einem Projekt liegen, kann man sie mit relativen Pfaden adressieren.

Freude teilen geht auch mit Kotlin Multiplatform

Die oben genannte Open-Source-Konferenz-App war bereits bei mehreren Konferenzen im Einsatz und kann die Frage beantworten, ob es möglich und überhaupt sinnvoll ist, mit Kotlin Multiplatfrom mobile Apps zu entwickeln. Wie muss man vorgehen, um die höchste Sharing-Quote zu erreichen? Wir haben folgende Antworten gefunden:

1. Mit Kotlin Multiplatform und dem Android Studio oder IntelliJ ist es jetzt schon möglich, native wiederverwendbare Komponenten zu schreiben, die man in Android-Projekten entweder als Android oder Java-Bibliotheken einbinden kann und als Objective C Xcode Framework in einem XCode-Projekt.
2. Kotlin als eine moderne Sprache weist viele Ähnlichkeiten mit Apples eigener Sprache Swift auf, was das Erlernen für iOS-Entwickler deutlich vereinfacht und bei diesen auf große Akzeptanz stößt.
3. Es ist nicht das Ziel, 100 Prozent des Codes zwischen den Plattformen zu teilen. Die UI- und Plattform-spezifischen Teile müssen weiterhin nativ an der jeweiligen Plattform entwickelt werden. Hier können die deklarativen UI-Frameworks von Google und Apple (Swift UI und Jetpack Compose) in baldiger Zukunft Abhilfe schaffen.
4. Mit Kotlin Multiplatform und einer pragmatischen Software-Architektur ist es möglich, eine Wiederholung von Teilen des Codes zu vermeiden.

Referenzen

[1] *https://play.kotlinlang.org/hands-on/overview*
[2] *https://github.com/JetBrains/kotlinconf-app*
[3] *https://github.com/touchlab/DroidconKotlin*
[4] *https://github.com/cashapp/sqldelight*
[5] *https://github.com/dukecon/dukecon_mobile*
[6] *https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html*

Michal Harakal
michal@harakal.de

Michal Harakal arbeitet seit 2010 bei der Deutschen Telekom AG. In die spannende Welt der mobilen Geräte ist er nach einer langjährigen Berufserfahrung als Entwickler von Mess- und Datenauswertungssystemen und später auch Steuerungen für industrielle Maschinen gewechselt. Sein Werkzeugkasten, in dem Java und seit Neuestem auch Kotlin einen besonderen Platz haben, ist vielseitig gefüllt. Bei Android ist er seit der ersten Stunde mit dabei. Michal teilt gerne seine Erfahrung und ist in Open-Source-Projekten aktiv.

Mitmachen und Autor werden!

Sie kennen sich in einem bestimmten Gebiet aus dem Java-Themenbereich bestens aus und möchten als Autor Ihr Wissen mit der Community teilen?

Nehmen Sie Kontakt zu uns auf und senden Sie Ihren Artikelvorschlag zur Abstimmung an **redaktion@ijug.eu.**

Wir freuen uns, von Ihnen zu hören!

iJUG Verbund

Jatumba und das große Wiedersehen der Community

Lisa Damerow, DOAG Dienstleistungen GmbH

Das Phantasialand in Brühl wird auch in diesem Jahr wieder einmal für drei Tage zum Java-Land – oder besser gesagt zur JavaLand, die diesmal vom 17. bis 19. März 2020 zur bereits siebten Ausgabe ihre Tore öffnet. Mehr als 2.000 wissbegierige Java-Begeisterte aus aller Welt versammeln sich hier jedes Jahr, um sowohl alte Bekannte als auch neue Gesichter zu treffen, Wissen auszutauschen und gemeinsam Spaß und Lernen zu verbinden. Neben Bewährtem gibt es natürlich auch Neues zu entdecken, wie beispielsweise neue Community-Aktivitäten, Keynotes und den neuen Thementag „Microservices". Dieser Artikel gibt einen Einblick in die JavaLand 2020 und erläutert, was die Teilnehmer erwartet.

Das Programm

Auch in diesem Jahr ist das Herzstück der JavaLand ein attraktives Vortragsprogramm aus rund 120 Vorträgen jeden Levels. International bekannte Top-Speaker sowie hochmotivierte Newcomer geben in 13 Streams ihr Wissen aus dem Java-Themenkosmos zum Besten. Sowohl beliebte Dauerbrenner als auch neue Technologien und Ansätze werden von ihnen näher unter die Lupe genommen und beleuchtet. In der Eröffnungs-Keynote am Dienstag, die direkt im Anschluss an die Begrüßung stattfindet, berichtet Arun Gupta, Principal Technologist bei Amazon Web Services (AWS), mehr zum Thema Chaos Engineering und darüber, welche Vorteile es bringt, Dinge mit dieser neuen Testdisziplin absichtlich kaputt zu machen. Auch die beliebte Community-Keynote hat wieder ihren festen Platz im Programm eingenommen. Ein Blick in den Konferenzplaner auf der JavaLand-Website *[1]* lohnt sich, denn es gibt viel zu entdecken!

Community-Aktivitäten und Ausstellung

Langeweile? Nicht auf der JavaLand! Die eigens von der Community organisierten Aktivitäten außerhalb der Vortragsräume laden zum gemeinsamen Lernen, Entdecken und Hacken ein. Zahlreiche Workshops bieten selbst Anfängern genug Basis-Wissen, um mit ihrer zukünftigen Lieblingstechnologie so richtig durchzustarten. Wie wäre es zum Beispiel mit einem Android-Programmiercrashkurs oder einer Einführung in Property Based Resilience Testing? Alle, die neben ihrem Geist auch ihren Körper fit halten möchten, können sich beim JavaLand-Jogging, Kung-Fu oder #Freeletics ordentlich auspowern. Das Beste daran: Diese Angebote sind im Rahmen der Konferenzteilnahme kostenfrei.

Wer sich für Technologie-Ansätze und -Lösungen großer und mittelständischer Firmen interessiert oder bei Gewinnspielen das ein oder andere Goodie abstauben möchte, sollte einen Blick in die Ausstellung werfen. Auf zwei Stockwerken freuen sich rund 50 Aussteller aus aller Welt auf spannende Gespräche.

Kleine Entwickler ganz groß

Beim JavaLand4Kids, das traditionell am Montag vor der JavaLand stattfindet, dürfen sich die Entwickler von morgen wieder einmal so richtig austoben. In eigens für Kinder und Jugendliche entwickelten Workshops werden diese unter Aufsicht von Mentoren spielerisch an das Programmieren herangeführt.

Work hard, play hard: Nervenkitzel und Spaß bei der Abendveranstaltung

Actionliebhaber und Adrenalinjunkies kommen bei der „Open Park"-Abendveranstaltung am Ende des ersten Konferenztages voll auf ihre Kosten, denn der Freizeitpark wird zum Leben erweckt! Hier stehen abenteuerlustigen Besuchern für einige Stunden ausgewählte Attraktionen und Achterbahnen des Phantasialands zur Verfügung. Da ist Spaß vorprogrammiert! Leckere Gratisspeisen und -getränke sowie Live-Musik, die zum Tanzen und Feiern anregt, machen diesen Ausklang zu einem Highlight, das man sich auf keinen Fall entgehen lassen sollte.

Schulungs- und Thementag

Besonders wissensdurstige Teilnehmer, denen zwei Tage Konferenzgeschehen noch nicht genug sind, haben während des Schulungstags am Donnerstag die Möglichkeit, in einem von acht Hands-on-Workshops ihr Wissen themenspezifisch zu vertiefen. In kleineren Gruppen von maximal 30 Teilnehmern wird in einer Kombination aus Theorie und reichlich Praxis ein bestimmtes Themengebiet fokussiert und vermittelt.

Parallel zum Schulungstag findet erstmals ein Thementag rund um Microservices statt. Hier berichten Anwender und Endkunden, unter anderem die Deutsche Bahn und die Otto Group, über ihre bisherigen Erfahrungen mit dieser Technologie. Danach wird bei den Deep-Dive-Sessions der jeweiligen Firmen tiefer in die Thematik eingetaucht. Zum Abschluss beantworten die Microservice-Experten in einer Paneldiskussion alle Fragen, die den Teilnehmern unter den Fingernägeln brennen.

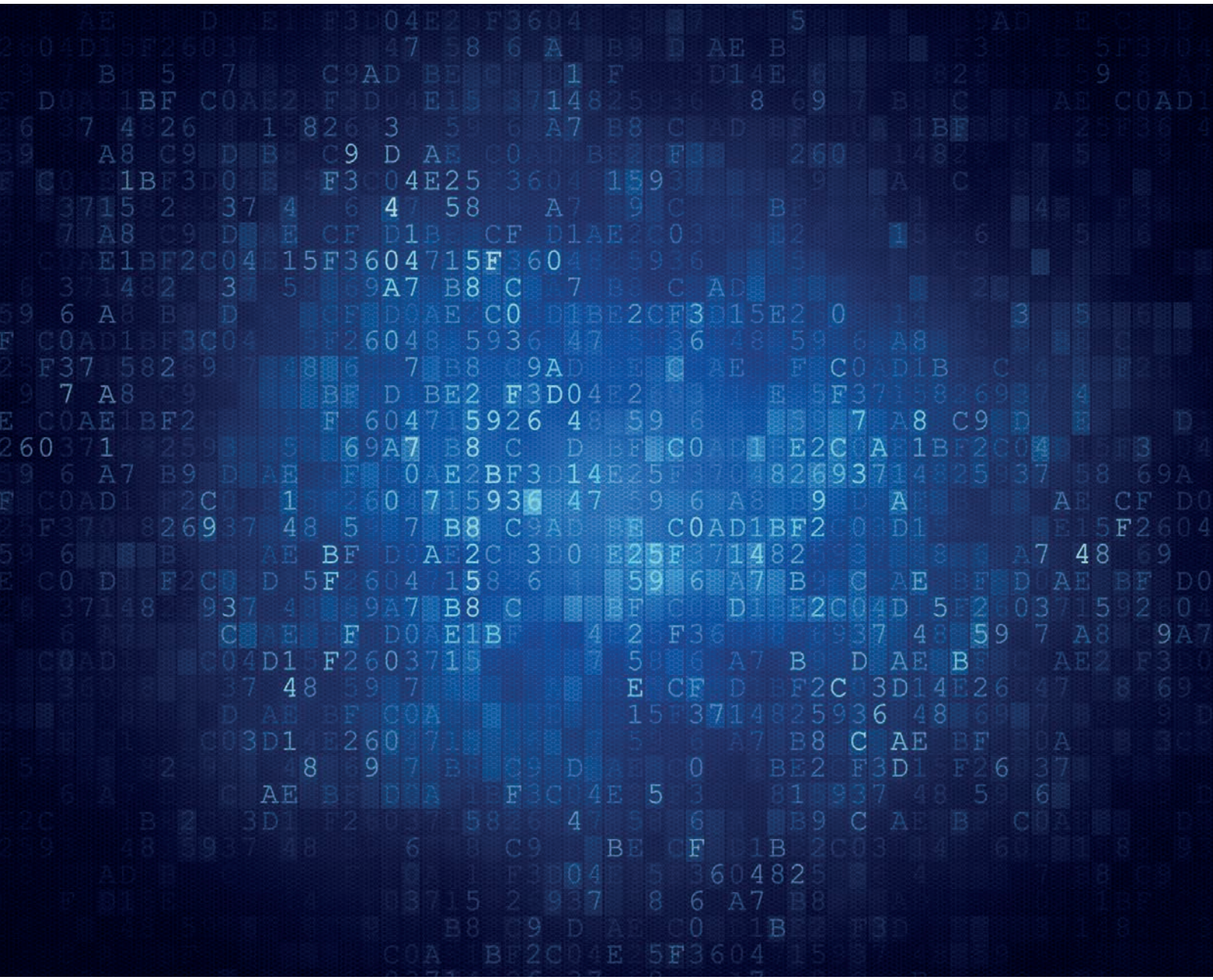
Praktische Informationen

Wer schon einmal die JavaLand besucht hat, weiß, dass die Parkhotels Matamba und Ling Bao sehr beliebt und darum schnell ausgebucht sind. Eine rechtzeitige Buchung direkt über die Ticketseite lohnt sich, um direkt in fußläufiger Entfernung vom Eingang zu übernachten. Um den Javaianern die Anreise zu erleichtern, gibt es einen Shuttleservice, der die Teilnehmer ab Montag vom Hauptbahnhof Brühl und vom Flughafen Köln/Bonn abholt und zum Phantasialand bringt. Weitere wertvolle Informationen, die Anreise und Aufenthalt erleichtern, sind auf der Veranstaltungswebsite zu finden.

Wir freuen uns, euch auf der kommenden JavaLand begrüßen zu dürfen! Jatumba – hier wird die Community großgeschrieben.

Referenzen

[1] *https://www.javaland.eu/de/home/*



Eine JVM für die Cloud: die GraalVM

Karine Vardanyan und Stephan Rauh, OPITZ CONSULTING Deutschland

Seit Anfang 2019 sorgt Oracles neue Virtual Machine, die GraalVM für Aufsehen. Für Java-Entwickler ist sie ein Ahead-of-Time-Compiler und verspricht, polyglott und hochperformant zu sein. Doch kann sie sich wirklich gegen Javas Just-in-Time-Compiler durchsetzen? Welche Vor- und Nachteile bietet sie und wie schlägt sie sich in der Cloud? In diesem Artikel werfen wir einen detaillierten Blick auf diese universelle Virtual Machine und was man mit ihr so alles machen kann. Kann sie ihrem Namensbezug, dem heiligen Gral, gerecht werden?

Die Geschichte des Java-Compilers reicht über zwanzig Jahre zurück. Die ersten Versionen von Java waren bemerkenswert langsam. Mit der Veröffentlichung des HotSpot-Compilers im April 1999 änderte sich die Situation. Beispielsweise zeigt eine Performance-Messung *[1]*, dass Java zwischen den Versionen 1.1.7 und 1.2 um den Faktor 40 schneller wurde. Wie so oft hängen die genauen Zahlen stark vom jeweiligen Anwendungsfall ab. Fest steht, dass Java seinerzeit dramatisch schneller wurde. Es wurde erstmals möglich, Performance-intensive Anwendungen, wie zum Beispiel Text-Editoren, in Java zu schreiben. Beispielsweise hat das Software-Unternehmen Borland seine IDE JBuilder im Jahr 1999 von Delphi auf Java migriert.

Eine kurze Geschichte der Compiler

Der Compiler wurde auf eine damals sehr ungewöhnliche Weise implementiert. Um den Unterschied zu erklären, machen wir einen kurzen Ausflug in die Geschichte.

Traditionelle Compiler übersetzen Quelltext in Maschinencode. Erst dadurch kann der Computer das Programm ausführen. Solch ein Ahead-of-Time-Compiler (kurz: AOT-Compiler) braucht üblicherweise sehr lang zum Kompilieren, dafür läuft das Programm anschließend sehr schnell. Programmiersprachen wie C und C++ verwenden diesen Ansatz. Das macht sie prädestiniert für die Entwicklung von Betriebssystemen, Computerspielen und „embedded Systems“: Wenn ein Autofahrer bremst, hilft ihm mit großer Wahrscheinlichkeit ein C-Programm dabei.

Eine andere Strategie ist es, ein Programm zu schreiben, das den Quelltext des Entwicklers direkt lesen und ausführen kann. Lange Zeit wurde JavaScript so implementiert. Der Vorteil eines solchen Interpreters ist, dass der zeitfressende Kompilierschritt entfällt. Dafür läuft das Programm sehr viel langsamer. In den 90er Jahren galt als Faustregel, dass ein (interpretiertes) BASIC-Programm circa hundert- bis tausendmal langsamer ist, als wenn es mit einem guten C-Compiler geschrieben worden wäre.

Die frühen Java-Versionen verwendeten eine dritte Strategie. Sie übersetzten das Programm in einen „Bytecode“. Das ist eine Programmiersprache, die von einem Interpreter ausgeführt wird, aber sehr viel besser für die Arbeitsweise einer CPU optimiert ist als der ursprüngliche Quelltext. Der Bytecode-Interpreter ist erheblich schneller als ein normaler Interpreter und hat noch weitere Vorteile. SUN hatte Bytecode-Interpreter für viele verschiedene CPUs und Betriebssysteme geschrieben. Java-Programme laufen heutzutage auf einer Vielzahl von Plattformen – vom Großrechner über PC bis zum Android-Handy. Sie werden für eine „virtuelle Maschine“ geschrieben; diese virtuelle Maschine wird für die jeweilige Zielhardware angepasst und erlaubt es, ein und dasselbe Programm quasi überall auszuführen. Detailliertere Informationen über Bytecode sind im Blog des Autors Rauh *[3]* zu finden.

Der Just-in-Time-Compiler

Die virtuelle Maschine und das „write once, run anywhere“-Versprechen waren das Killer-Argument für den Einsatz von Java. SUN konnte den Bytecode also nicht durch einen traditionellen Compiler ersetzen. Stattdessen wurde die virtuelle Maschine um den „Just-in-Time“-Compiler ergänzt. Die Idee ist dabei, dass das Programm erst während der Programmausführung kompiliert wird. Das kann man sich wie einen Cache vorstellen: Am Ende des Tages führt der Interpreter ebenfalls Maschinencode aus. Diesen Maschinencode kann man cachen. Damit entfällt der zeitaufwendige Übersetzungsschritt. Wenn ein Algorithmus zum ersten Mal ausgeführt wird, ist er wie gewohnt langsam, bei der nächsten Wiederholung nimmt er jedoch Fahrt auf.

Dieser Effekt ist sehr eindrucksvoll in der Performancemessung sichtbar, die einer der Autoren (Stephan Rauh) vor einigen Jahren auf seinem Blog *[4]* durchgeführt hat *(siehe Abbildung 1)*. Die Grafik zeigt auf der Y-Achse die Zeit, die der Testalgorithmus braucht. Auf der X-Achse ist die Anzahl der Wiederholungen dargestellt. Bei jeder Wiederholung wird der Algorithmus etwas schneller. Erst ab zehntausend Wiederholungen ändert sich nicht mehr viel.

Zwanzig Jahre Optimierungen

Abbildung 1 zeigt eines ganz deutlich: Es ist nicht bei dem einfachen Cache geblieben. Der Compiler optimiert den Maschinencode permanent. Er konzentriert sich dabei auf die Stellen, die oft aufgerufen werden, die quasi „heißgelaufen“ sind. Er wurde konsequenterweise „HotSpot-Compiler“ getauft. Die Tipps und Tricks, die der Hot-Spot-Compiler anwendet, sind äußerst umfangreich. Zu diesem Thema empfehlen die Autoren die Vorträge von Charles Nutter *[5]*, die unter anderem Loop Unrolling, Escape Analysis und viele weitere Ideen umfassen.

Viele Getter und Setter in Programmen überleben nicht lange. Sie sind die ersten Opfer des „Inlining“. Der HotSpot-Compiler merkt, dass die Getter und Setter nur triviale Logik enthalten, und ersetzt sie kurzerhand durch den direkten Zugriff auf das Attribut. Allein durch diesen kleinen Trick wird das Programm erheblich effizienter.

Den Rest können wir uns denken: Im Laufe der letzten zwanzig Jahre wurde ununterbrochen am HotSpot-Compiler entwickelt. Java-Programme laufen heute erheblich schneller als früher. Auch das sehen wir an der Grafik, die die Entwicklung zwischen Java 1.2 und Java 1.7 zeigt. In der Zeit danach ist die Entwicklung keineswegs stehengeblieben. Laut Open Hub hat der Compiler von Java 7 ganze 734.000 Zeilen, der Compiler von Java 9 bereits 1,34 Millionen. So beeindruckend der Erfolg der HotSpot-JVM ist, sie ist vermutlich nicht der richtige Ort, um radikal neue Ideen auszuprobieren. Das Risiko ist einfach zu groß, und es dürfte auch gar nicht so einfach sein. Cliff Click – einer der Entwickler des HotSpot-Compilers C2 – sagt [11, Folie 40], dass er heutzutage keinen Compiler mehr in C oder C++ schreiben würde. C++-Programme neigen dazu, komplex und schwer wartbar zu werden. Laut Chris Seaton *[9]* ist der C2-Compiler – wie nach 20 Jahren nicht anders zu erwarten – schwer zu warten und zu erweitern.

Polyglot Programming

Dabei gibt es durchaus wichtige Entwicklungen, die regelrecht nach einem neuen Compiler rufen. Vor einigen Jahren wurde beispielsweise die Idee des „isomorphen JavaScript“ populär. Fast alle Benutzeroberflächen werden heutzutage in JavaScript implementiert. Auf dem Server hat sich dagegen Java etabliert. Für Entwickler von Eingabeformularen ergibt sich damit ein Problem. Sie müssen Validierungen und Plausibilitäten doppelt implementieren. Einmal auf dem Client – um dem Anwender sofortiges Feedback zu geben –

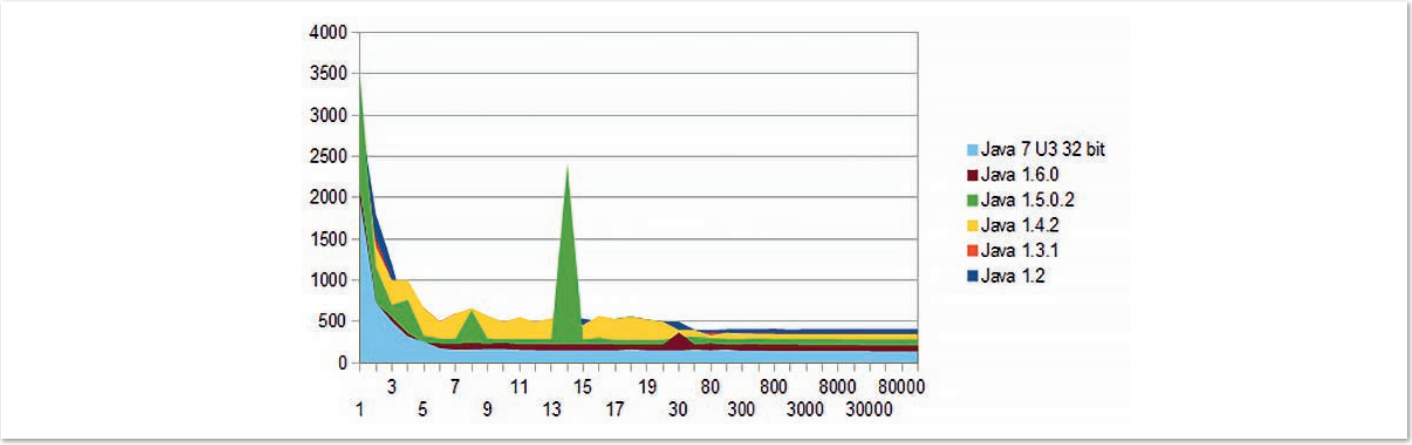


Abbildung 1: Performancegewinne durch den JIT-Compiler im Laufe der Zeit (© Stephan Rauh)

und zum anderen aus Sicherheitsgründen auf dem Server. Es wäre also schön, auch auf dem Server JavaScript ausführen zu können. Dann würde eine Implementierung reichen.

Mit den Projekten Nashorn und Rhino gibt es diese Möglichkeit in aktuellen Java-Projekten. JavaScript fühlt sich in diesen Umgebungen aber immer noch wie ein Fremdkörper an und die Performance reicht nicht an die im Browser heran.

Polyglot Compiler

Hier kommt die GraalVM ins Spiel. Streng genommen ist der Name „GraalVM" irreführend: Aus Sicht eines Java-Entwickler handelt es sich um einen neuen Compiler. Der Rest der JVM wird unverändert weiterverwendet. Man kann mit der GraalVM jedoch auch JavaScript-Anwendungen laufen lassen. In diesem Fall fühlt sich die GraalVM so an, als wäre sie ein Ersatz für Node.js und für npm. Bei Experimenten haben die Autoren es sogar geschafft, die Angular-CLI zu installieren. Sie konnten mithilfe der CLI ein Angular-Programm bauen und kompilieren. Das lässt hoffen, dass die Warnungen bezüglich der Node.js-Kompatibilität auf der GraalVM-Homepage *[8]* übertrieben sind.

Als wäre das noch nicht spannend genug, bietet die GraalVM auch Support für Python, R, Ruby, Scala und den LLVM-Bitcode. Das wiederum bedeutet, dass auch Sprachen wie C/C++ und Rust auf der GraalVM laufen, wenn auch über den Umweg eines LLVM-Compilers.

Language-Agnostic Compiler

Diese Programmiersprachen kommen aus denkbar unterschiedlichen Traditionen. Einige dieser Sprachen sind schon aus der JVM-Welt bekannt. Scala und JRuby generieren Bytecode und können damit auf der JVM laufen und – innerhalb gewisser Grenzen – sogar mit Programmteilen, die in anderen Sprachen geschrieben wurden, zusammenarbeiten. Ganz reibungslos war das allerdings nie: Die JVM ist und bleibt für Java optimiert und mit Ausnahme des Bytecodes invokedynamic wurde die JVM nie für andere Programmiersprachen erweitert. Das zwingt die Sprachentwickler zu Klimmzügen, die sich typischerweise in der Performance und/oder dem Speicherverbrauch niederschlagen.

Die GraalVM geht einen anderen Weg. Auch hier gibt es wieder einen zweistufigen Prozess: Der Quelltext wird in einen Graphen übersetzt und aus diesem wird der Maschinencode generiert. Den Graphen kennen einige Entwickler bereits aus anderen Kontexten unter dem Namen „Abstract Syntax Tree", kurz „AST". Wer das zum ersten Mal sieht, mag vielleicht ein wenig verblüfft sein: Jeder Programmcode kann als Graph dargestellt werden. Tatsächlich ist der AST das Kernstück praktisch jeden Compilers. Auf die Einzelheiten einzugehen, würde den Rahmen dieses Artikels sprengen; für einen Einstieg empfehlen die Autoren einen Blick in die englischsprachige Wikipedia-Seite *[13]* und in die ersten paar Dutzend Seiten des berühmten „Drachenbuches" *[14]*.

Die Besonderheit des Graal-Graphen ist, dass er nicht auf eine einzige Programmiersprache beschränkt ist. Bei Graal verwenden alle Programmiersprachen denselben AST. Sie haben damit ein einheitliches Framework, um neue Sprachen zu entwickeln. Für Sprachentwickler ist das ein sehr attraktives Feature.

Es ist auch relativ einfach, eigene Optimierungen zu implementieren. Der Compiler ist in Java geschrieben und durch den Graphen ist es ziemlich einfach, Optimierungen zu entwerfen. Chris Seatons Vortrag *[9]* zeigt die Idee anhand einiger Beispiele.

```
@CoreMethod(names = "+", required = 1)
public abstract static class AddNode
   extends BignumCoreMethodNode {

   @Specialization
    protected long addWithOverflow(int a, int b) {
        return (long) a + (long) b;
    }
  }
```

Listing 1

Truffle

Will man selbst eine Sprache entwickeln, ist das Framework Truffle sogar noch einfacher. In diesem Fall hinterlegt man einfach für jeden Knoten der AST eine in Java geschriebene Implementation. Die GraalVM kompiliert daraus nativen Maschinencode. Durch Optimierungen wie die Partial Evaluation ist die Performance sehr gut. Laut Chris Seaton war beispielsweise TruffleRuby durch diesen Ansatz 2017 die schnellste Implementation von Ruby, nach seinen Worten „oft zehn Mal schneller als andere Implementationen" *[9]*.

Auf GitHub ist ein Quelltext zu finden, der eindrucksvoll zeigt, wie einfach es ist, mit Truffle einen Compiler zu schreiben *[10]*. Auf das Wesentliche reduziert, sieht die Implementation der Addition zweier Zahlen wie in *Listing 1* aus. Auf den ersten Blick ist das nicht beeindruckend. Klar, so sieht eine Addition in Java aus, ja und? Der Clou an der Sache ist, dass es eben gerade keine Addition in Java ist. Es zeigt, wie man beim Entwerfen einer Programmiersprache die Addition implementiert. Es könnte PASCAL sein oder Ruby oder LUA oder eine Programmiersprache, die man entwirft, um bestimmte Ideen auszuprobieren.

Man braucht also nicht mehr Assembler zu lernen, um eine Sprache zu entwickeln oder bei der Weiterentwicklung einer Programmiersprache zu unterstützen. Es reicht, eine weitverbreitete Hochsprache – Java – zu kennen.

Warum ist der Graal-Compiler besser?

Nach Einschätzung der Autoren ist das der große Vorteil des Graal-Compilers. Er macht es Entwicklern leicht, sich bei der Sprachentwicklung einzubringen. Damit ist der Pool an Leuten, die gute Ideen einbringen können, größer als beim HotSpot-Projekt.

Das gilt auf allen Ebenen. Der Java-Compiler von GraalVM erzeugt durchaus nativen Assemblercode, und das kann man auch für andere Programmiersprache tun. Der Entwickler hat die Wahl. Der größte Teil des Compilers wird in allen Fällen mit Java geschrieben, mit allen Vorteilen, die das mit sich bringt: einem leistungsfähigen Debugger, ausgereiften IDEs und vielem mehr.

Als die Autoren sich im Frühjahr 2018 mit der GraalVM zu beschäftigen begannen *[12]*, war es noch so, dass die Performance der GraalVM tendenziell schlechter als die des HotSpot-Compilers war. Inzwischen scheint die GraalVM aufgeholt und zum Überholen angesetzt zu haben.

Herausforderung Cloud

Zurück zum Anfang. Bisher haben die Autoren beschrieben, warum sie die GraalVM faszinierend finden, und eine Menge über Interpreter, Compiler und Sprachimplementierung berichtet. Unter anderem auch, dass Java-Programme nicht sofort, sondern erst nach einiger Zeit sehr schnell sind.

Genau das ist in Cloud-Umgebungen allerdings unerwünscht. Die Idee der Cloud ist, nur dann für einen Service zu bezahlen, wenn dieser auch benutzt wird. Die übrige Zeit kann der Service vom Cloud-Provider abgeschaltet werden. Der Speicherplatz und die CPU können sinnvoller verwendet werden, als auf einen Service-Aufruf zu warten, der vielleicht nie kommt. Besonders konsequent ist dies bei Amazon Lambda umgesetzt. Im Prinzip besteht der Service hier nur aus einer – potenziell kleinen – Funktion. Wenn der Service in Java implementiert wird, passiert jedoch etwas ganz anderes. Jedes Mal, wenn die Funktion aufgerufen wird, wird eine JVM gestartet. Das bedeutet, dass rund 2.000 Klassen geladen werden. Die meisten dieser Klassen benötigt der Entwickler gar nicht. Sie werden aber zum Starten der JVM benötigt. Erst nach ein bis zwei Sekunden ist die JVM verfügbar und kann mit dem Ausführen des Service beginnen.

Mit einigen Tricks kann man erreichen, dass diese JVM nicht wieder abgeschaltet wird. Beim nächsten Mal entfällt dadurch die Aufwärmzeit. Allerdings widerspricht das der Idee der Cloud. Anstatt Ressourcen möglichst effizient zu nutzen, wird mit diesem Ansatz das Recycling von CPU und Speicher verhindert.

Ahead-of-Time-Compiler

Die GraalVM bringt einen Ahead-of-Time-Compiler mit (die „SubstrateVM"). Der Entwickler kann damit wie in den guten, alten C++-Zeiten ein Programm in nativen Maschinencode übersetzen. Unter Windows ist das dann ein .EXE-File. Diese Datei läuft auf jedem Rechner, auch wenn dort kein Java installiert ist. Lediglich die Plattformunabhängigkeit geht verloren. Wenn man in die Cloud gehen möchte, muss man darauf achten, für das richtige Betriebssystem zu kompilieren – in der Regel also Linux.

Der springende Punkt ist, dass das Executable sehr schnell startet. Bei den Messungen der Autoren lag die Startzeit im Millisekundenbereich, eigentlich sogar noch unterhalb ihrer Messgenauigkeit.

Nachteile und Einschränkungen

Es gibt ein paar Einschränkungen. Die offensichtlichste ist, dass das Programm nicht mehr zur Laufzeit optimiert werden kann. Das Executable wird immer etwas langsamer sein als das gleiche Programm mit dem JIT-Compiler – zumindest, wenn die Zeit reicht, um den HotSpot-Compiler warmlaufen zu lassen.

Der andere Nachteil ist, dass das Reflection-API nicht funktionieren kann. Die Idee des Reflection-API ist es, zur Laufzeit herauszufinden, welche Klassen im Speicher sind. Der springende Punkt ist „zur Laufzeit". Der Ahead-of-Time-Compiler läuft ins Leere. Das würde in der Java-Community auf wenig Gegenliebe stoßen, und das weiß auch das GraalVM-Team. Im Prinzip ist praktisch jedes moderne Java-Framework davon betroffen: Hibernate, Spring und viele weitere. Darauf möchte niemand verzichten.

Die SubstrateVM löst das Problem durch einen Kompromiss. Die Idee dahinter beruht auf der Beobachtung, dass so gut wie niemand Änderungen an einer laufenden Softwareinstallation vornimmt. Wenn ein Programm erst einmal an den Kunden ausgeliefert wurde, bleibt das Programm so. Veränderungen gibt es erst bei der nächsten Lieferung.

Die Flexibilität des Reflection-API ist also überflüssig. Es ist leicht möglich, bereits zur Kompilierzeit zu erkennen, welche Klassen geladen werden. Im Prinzip kann man jeden Reflection-API-Aufruf auflösen und direkt die angefragte Klasse oder Methode liefern. Das funktioniert auch in der Praxis, wenn auch mit einer beeindruckenden Liste von Einschränkungen *[15]*. Wenn

man nach „SubstrateVM" und „Hibernate" oder „Spring Boot" googelt, finden man in erster Linie Artikel, die sagen, dass das nicht funktioniert. Soweit die Autoren das erkennen können, wird intensiv daran gearbeitet.

In der Zwischenzeit sind andere Frameworks populär geworden, die von vornherein für die SubstrateVM entwickelt wurden. Insbesondere ist hier Quarkus zu nennen.

State of the Art

Die aktuellen Versionen von GraalVM werden als produktionsreif bezeichnet. Im Großen und Ganzen deckt sich das mit den Erfahrungen der Autoren. Nichtsdestotrotz empfehlen sie, einen Umstieg vorsichtig zu planen und erst einmal gründlich zu testen. In einigen Bereichen sind ihnen auch noch offene Baustellen aufgefallen. Beispielsweise fehlt die Node.js-Umgebung in der Windows-Version von GraalVM. Außerdem sind Entwickler mit der GraalVM 19.2.0 noch auf Java 8 festgelegt. Für Java 11 gibt es aktuell (Stand: September 2019) noch keine GraalVM.

Interessiert man sich für den Ahead-of-Time-Compiler, sollte man sich mit Frameworks wie zum Beispiel Quarkus vertraut machen. Das bedeutet möglicherweise den Abschied von liebgewonnenen Frameworks wie Spring Boot oder Hibernate. Der Vorteil ist, dass Quarkus speziell für die GraalVM entwickelt wurde.

Fazit

Nach zwanzig Jahren wagt Oracle mit der GraalVM einen radikalen Neustart. Die Community kann gespannt sein, ob die GraalVM irgendwann Teil der Standard-Java-Installation oder ob es auch weiterhin zwei verschiedene Installationen geben wird. Der aktuelle Stand der Entwicklung stimmt die Autoren allerdings optimistisch!

Quellen

[1] *http://www.commercialventvac.com/java_performance.html*
Jeff Silverman (2008)

[2] *https://en.wikipedia.org/wiki/JBuilder*
Wikipedia (2019)

[3] *https://www.beyondjava.net/java-programmers-guide-java-byte-code*
Stephan Rauh (2019)

[4] *https://www.beyondjava.net/java-performance-through-the-ages*
Stephan Rauh (2013)

[5] *https://www.youtube.com/watch?v=MFvDLg-qKuU*
Charles Nutter (2017), „From Java to Assembly: Down the Rabbit Hole"

[6] *https://www.openhub.net/p/jdk7u-hotspot*
OpenHub (2019), Projektkennzahlen jdk7u-hotspot

[7] *https://www.openhub.net/p/jdk9-hotspot*
OpenHub (2019), Projektkennzahlen jdk9-hotspot

[8] *https://www.graalvm.org/docs/reference-manual/languages/js/#graalvm-javascript-compatibility*
GraalVM.org (2019), GraalVM JavaScript Compatibility

[9] *https://chrisseaton.com/truffleruby/jokerconf17/*
Chris Seaton (2017), Understanding How Graal Works - a Java JIT Compiler Written in Java

[10] *https://github.com/oracle/truffleruby/blob/master/src/main/java/org/truffleruby/core/numeric/IntegerNodes.java*
GitHub (2019), Implementation of the integer operations in TruffleRuby

[11] *https://ia601208.us.archive.org/16/items/vmss16/click.pdf*
Cliff Click (2016), Bit of Advice for the VM Writer

[12] *https://www.beyondjava.net/graal-towards-the-holy-grail-of-polyglot-programming*
Stephan Rauh (2018), Graal - Towards the Holy Grail of Polyglot Programming

[13] *https://en.wikipedia.org/wiki/Abstract_syntax_tree*
Wikipedia (2019), Abstract Syntax Tree

[14] Compiler: Prinzipien, Techniken und Werkzeuge; Aho, Sethi und Ullman (2008 – 3. Auflage)

[15] *https://github.com/oracle/graal/blob/master/substratevm/LIMITATIONS.md*
GraalVM-Projekt (2019): Native Image Java Limitations

Karine Vardanyan
OPITZ CONSULTING Deutschland
Karine.Vardanyan@opitz-consulting.com

Parallel zu ihrem Masterstudium an der TU Darmstadt arbeitet Karine Vardanyan seit sieben Monaten als Werkstudentin im Bereich Software Engineering bei der OPITZ CONSULTING Deutschland GmbH. Sowohl an der Universität als auch bei der Arbeit befasst sie sich mit spannenden Themen und Software-Entwicklungstechnologien wie künstliche Intelligenz, Java und vielem mehr.

Stephan Rauh
OPITZ CONSULTING Deutschland
Stephan.Rauh@opitz-consulting.com

Stephan Rauh arbeitet als Solution Architect bei der OPITZ CONSULTING Deutschland GmbH und befasst sich seit Jahren mit Angular, JSF und generell mit UI-Technologien. Er ist durch seinen Blog http://www.beyondjava.net und sein Open-Source-Framework BootsFaces bekannt geworden. Stephans weitere Steckenpferde sind das „Tech-Scouting", Programmiersprachen und Compiler.

Wie Sie morgens ohne Wecker aus dem Bett springen

Vincent Schwarzmeier, DCD Training GmbH

In diesem Artikel werden Sie die mächtigsten Geheimnisse der Motivation kennenlernen. Sie werden anschließend in der Lage sein, zu jeder Zeit eine natürliche, intrinsische Motivation in Ihnen zu wecken – besonders für Aufgaben und Tätigkeiten, bei denen es Ihnen am schwersten fällt.

„Chef, ich komme am Montag etwas später zur Arbeit." „Ok, wann kommen Sie?" „Am Dienstag." Zugegeben, als ich diesen Spruch las, musste ich laut losprusten, während ich fast meinen Kaffee verschüttete. Auch für einen Motivationstrainer gibt es Tage, an denen es an Motivation mangelt. Wir sind auch gerne mal faul und bequem unterwegs, keine Frage. Das ist selbstverständlich menschlich. Gleichzeitig sollten die Alarmglocken schrillen und Maßnahmen dagegen ergriffen werden, sobald sich ein solcher Dauerzustand breitmacht.

Welche Gedanken kommen in Ihnen hoch, wenn es um das Thema Motivation geht? Fühlen Sie sich regelmäßig so sehr voller Tatendrang, dass Sie am liebsten alle Ihre Pläne gleichzeitig in die Tat umsetzen möchten und dabei die Sorge haben, dass Sie definitiv noch ein weiteres Leben bräuchten, um alles unter einen Hut zu bekommen?

Dieser Artikel soll Ihre persönliche „Mona Lisa" der Motivationsschriftstücke sein, auf die Sie sich als Erstes zurückbeziehen können, falls es Ihnen wiederholt an Motivation mangelt. Ich möchte es Ihnen einfach machen und für Sie das absolut Wichtigste zum Thema Motivation enthüllen. Sie werden am Ende dieses Artikels in der Lage sein, etwas unschätzbar Kostbares in Ihnen zu entdecken. Etwas, das Sie von ganz allein so sehr antreibt, dass Sie tatsächlich morgens ohne einen Wecker aus dem Bett springen, weil Sie so begierig darauf sind, Ihre Pläne in die Tat umzusetzen, und plötzlich deutlich mehr Lebensfreude verspüren. Besonders auch an Tagen, an denen Sie mit Tätigkeiten und Aufgaben konfrontiert sind, bei denen der bloße Gedanke daran Ihnen den Magen umdreht. Sie werden sich auch nie wieder motivieren müssen, sondern werden es von selbst sein und auch bleiben, spätestens ab dem Zeitpunkt, an dem Sie fündig geworden sind.

Was ist Ihr persönlicher, größter Antrieb?

Kommen wir gleich zum Punkt. Der dominanteste Grund für chronische Antriebslosigkeit ist das ausbleibende „Warum" im Leben. Stellen Sie sich selbst folgende Fragen: Warum tun Sie jeden Tag das, was sie tun? Wofür sind Sie angetreten in diesem Leben? Wie würde der Buchtitel Ihres Lebens heißen? Ja, das sind tiefe Fragen. Sie bekommen von mir auch nicht die üblichen 0815-„Express-Motivationsmaßnahmen", um schnell die Symptome zu betäuben, damit Sie bis zum Wochenende „durchhalten" können. Sie bekommen hier komprimiert die reine Essenz der Motivation, damit Sie das Thema, falls nötig, ein für alle Mal in den Griff bekommen.

Lassen Sie mich Ihnen ein Beispiel geben, wie ein starkes „Warum" aussehen und welche Auswirkungen es auf Ihr Leben haben könnte. Vor einigen Jahren, bevor ich hauptberuflich als Seminarleiter in den Bereichen Persönlichkeitsentwicklung und Präsentationen für große Firmen arbeitete, saß ich in einer Bahn auf dem Weg zu meiner alten Arbeit. An einer Station angekommen, beobachtete ich die Menschen beim Aussteigen der Bahn und sah plötzlich bewusster als sonst in ihre Gesichter. Dabei lief mir ein kalter Schauer über den Rücken. Ich blickte in graue, traurige und gestresste Gesichter. Erdrückt vom Alltag, mangelnder Gesundheit aufgrund meist ungesunder Ernährung, Zigaretten und falscher Lebensweise. Sie liefen in einen Job, den sie hassten, und lebten ein Leben, das sie nicht erfüllte. Die nackte Realität sprang mir ins Gesicht. Zumindest war das mein Film, der sich in meinem Kopf abspielte.

Gleichzeitig wurde mir bewusst, dass meine junge Tochter (ein kleiner, durchaus frecher, rothaariger Engel), die zu diesem Zeitpunkt zwei Jahre alt war, einmal in genau dieser Welt aufwachsen würde. Sie darf später ebenfalls in einen städtischen, grauen Kasten gehen, an dem ihr ihr freies Denken und ihre Kreativität abtrainiert und sie für eine bevorstehende Infrastruktur vorbereitet wird. Sie wächst ohne das nötige Wissen auf, wie man erfolgreiche Beziehungen aufbaut, selbstsicher auftritt, sich richtig ernährt, richtig lebt, emotionale Themen behandelt und so weiter. Das komplette 360-Grad-, Rundum-sorglos- „So baust du dir 'ne Hammer Zukunft auf"-Paket wird ihr komplett vorenthalten und endet später in einem nur teilweise erfüllten Leben.

Allein der Gedanke daran, dass ich später auf mein Leben zurückblicken und der Tatsache ins Auge sehen müsste, dass ich nichts dagegen unternommen habe, nur weil ich ständig meinem Türsteher der Komfortzone (auch bekannt als „Schweinehund") nachgiebig war, ist Motivation genug. Ab diesem Zeitpunkt wusste ich eines ganz genau: Ich werde alles dafür geben, um in dieser Gesellschaft einen maximalen, wertvollen Beitrag zu leisten, der nachhaltig etwas verändert. Ich werde als Persönlichkeitstrainer anderen Menschen dabei helfen, in ihre Kraft zu kommen und aus ihrem Leben ein fantastisches Meisterwerk zu formen. Solange mein Herz in meiner Brust schlägt, kämpfe ich jeden Tag dafür, dass diese Welt zu einem Ort wird, in dem meine Tochter sich optimal entfalten und später sagen kann: „Ich bin stolz darauf, dass es meinen Papa gibt!"

Das ist mein „Warum" in meinem Leben, das mich jeden Tag antreibt, warum ich jedes Mal 100 Prozent in meinen Trainings für meine Teilnehmer gebe, egal, ob ich schlecht geschlafen habe oder mit dem falschen Fuß aufgestanden bin.

Was ist Ihr „Warum"? Was treibt Sie wirklich an? Sie müssen sich diese Frage stellen und so lange dranbleiben, bis Sie eine klare Antwort darauf haben. Geben Sie sich Zeit. Das ist mitunter die wichtigste Frage, die Sie sich in Ihrem Leben beantworten müssen, auch wenn es sich vielleicht für Sie nach etwas Tschakka-Motivation anhört. Auch diese Art Seminare haben meiner Meinung nach ihre Daseinsberechtigung, wenn wir schon beim Thema sind. Es holt viele Menschen ab, führt sie zurück in ihre Kraft, liefert nötige, erste Impulse. Viele Wege führen nach Rom.

Ich empfehle Ihnen vor allem, neben Ihren eigennützigen Zielen (zum Beispiel Materielles, Wohlstand, Sicherheit etc.), auch altruistische zu verfolgen. Der Antrieb ist dadurch noch größer, es bietet Ihnen mehr Erfüllung und die Auswirkungen sind weitläufiger, da deutlich mehr Leidenschaft hinter Ihren Handlungen steckt. Sehen Sie sich das Beispiel von Dale Carnegie an:

Dale Carnegie hielt Vorträge und Seminare in Abendschulen. Als ihm irgendwann der Gesprächsstoff ausging, rief er spontan Teilnehmer nach vorne und bat sie darum, etwas von sich zu erzählen. Bei deren Reden fiel ihm auf, dass man die ein oder andere rhetorische Stellschraube noch nachziehen könnte, und fing an, sie zu coachen. Die Leute entwickelten sich und die Seminare wurden immer voller. Ich bezweifle übrigens, dass Dale Carnegie morgens aufgestanden ist und vor dem Spiegel griesgrämig murmelte: „Montag… das f' steht für Freude'", oder sich motivieren musste. Er wusste, dass ihn zig Menschen heute wieder erwarten, die allesamt durch seine Tätigkeit

als Trainer in ihrer Persönlichkeit wuchsen und deren Lebensqualität sich dadurch erhöhte. Es gab ihm Sinn und Erfüllung. Natürlich trieb ihn das auf natürliche Weise an, täglich sein Bestes zu geben und weiterzumachen. Heute ist Dale Carnegie Training ein internationales Trainingsunternehmen, das das Leben von Millionen Menschen bereichert.

So verstärken Sie Ihre Motivation

Schauen Sie sich doch bitte für einen kurzen Moment nochmal das Wort „Motivation" an. Welches kleine Wort können Sie darin noch entdecken? Richtig, es ist natürlich das Wort „Motiv". Die Qualität Ihrer Bilder, die Sie vor Ihrem geistigen Auge sehen, bestimmt den Grad Ihrer Motivation, die Sie fühlen. Werfen wir doch mal einen Blick auf Arnold Schwarzeneggers Lebenslauf, um es genauer zu erklären.

Klein Arni wuchs in einem Dorf in Österreich auf. Als Teenager entwickelte er plötzlich den Traum, der weltgrößte Bodybuilder zu werden, den es je gegeben hat, nachdem er sich voller Begeisterung einige Bodybuilding-Zeitschriften ansah. Noch heute spricht man von der besten physischen Form, die Arnold hatte und an die bis heute keiner rankommt. Da ihm das noch nicht genug war, setzte er noch eins oben drauf und wurde berühmt für Hollywood-Filme, in denen er erfolgreich mitspielte. Das alles nur, weil er sich selbst vorher als ein Hollywood-Star gesehen hat.

Ich könnte Ihnen noch weitere Beispiele und besonders auch Theorien erklären, warum es wichtig ist, welche Art von Kopfkinos Sie sich regelmäßig visualisieren sollten. Lassen Sie uns doch gleichzeitig ein Experiment auf der praktischen Ebene machen, um Ihnen einen kurzen Beweis dazu zu liefern.

Stellen Sie sich bitte einmal bildlich vor, Sie liegen in der Sonne auf einem Liegestuhl. Sie spüren die warmen Sonnenstrahlen auf der Haut, hören das Plätschern eines Wassers in Ihrer Nähe. Neben Ihnen steht ein kleiner Tisch, auf dem eine halbe Zitrone auf einem kleinen Teller liegt. Sie verspüren plötzlich einen starken Durst nach diesem sauren Zitronensaft. Sie greifen nach der Zitrone, werfen Ihren Kopf in den Nacken, halten die Zitrone über Ihren offenen Mund und pressen die gesamte Zitrone aus. Der ganze Saft tröpfelt in Ihren Mund, bis er komplett voll mit Zitronensaft ist und Sie ihn genüsslich herunterschlucken.

Was genau ist gerade in Ihrem Mund passiert? Die meisten Teilnehmer berichten hinterher, nachdem sie es sich bildhaft vorgestellt haben, dass ihr Gaumen zu kribbeln angefangen und der Mund mehr Speichel ausgestoßen hat. Doch warum reagiert unser Körper so? Ich meine, hatten Sie wirklich Zitronensaft im Mund oder reichte die bloße Vorstellung daran, das Gesicht zu verziehen?

Die Theorie dahinter ist einfach, in sich jedoch komplex. Unser Gehirn hat keine Ahnung, ob das, was gerade passiert, Realität oder Einbildung ist. Es erkennt den Unterschied nicht. Das ist auch der Grund, warum unsere Träume uns so real vorkommen. Wir wachen teilweise währenddessen auf und fragen uns für ein paar Sekunden, ob wir gerade wirklich in unserem Bett liegen oder noch in dem Traum gefangen sind.

Wenn also unser Körper auf geistige Bilder ebenso reagiert, als würden sie wirklich gerade passieren, wäre es doch schlau, genau diese zu beeinflussen, um unsere Gefühlswelt zu steuern. Diese ruft ja schließlich körperliche Reaktionen hervor, die wiederum unsere Ergebnisse in unserem Leben kreieren. Statt sich morgens die heftigen Schlagzeilen der Zeitung anzusehen, könnte man sich auch einige Bilder von Zielen visualisieren, die man in seinem Leben erreichen möchte. Die Wirkung wird eine andere sein – garantiert. Auch der morgendliche Gruß an die Kollegen wird enthusiastischer sein als sonst, weil Sie nun –Überraschung – motivierter sind!

Warum Sie den höchsten Preis zahlen, wenn Sie zögern

Kommen wir zuletzt zu einem der mächtigsten Werkzeuge, wenn es um Motivation geht. Wenn hier der Groschen fällt, geht Ihre Motivation durch die Decke. Lassen Sie mich Ihnen ein Beispiel eines prägenden Erlebnisses geben:

Vor einigen Monaten saß ich abends mit einem alten Freund zusammen bei einem schönen kühlen Bier. Nach einer Weile Plauderei setzte wieder einmal meine Berufskrankheit ein und ich fing an, ihn ungefragt zu coachen: „Du Berti, lass uns mal ein kleines Experiment machen. Stell dir vor, ich würde dir jeden Tag 86.400 Euro geben. Du dürftest das gesamte Geld ausgeben, für alles, was du möchtest. Es gibt allerdings zwei Dinge, die du beachten musst: Du darfst das Geld nicht sparen. Wenn etwas davon auf deinem Konto übrigbleibt, wird es wie von Zauberhand gelöscht. Außerdem kann es sein, dass es jeden Tag vorbei ist und ich dir den Hahn zudrehe." Er blickte mich gespannt an und sagte: „Hört sich bis jetzt gut an!" „Ok, nun meine Frage: Was würdest du mit dem Geld jeden Tag machen?" „Na... ich würde es ausgeben. Für alles Mögliche. Und ich würde es natürlich auch investieren." „Wirklich alles? Jeden Tag? Oder würdest du, wenn was übrig bleibt am Ende des Tages, das Geld verschwinden lassen?" „Nein, sicher nicht. Dann würde ich es noch auf den letzten Drücker verschenken."

Ist doch wirklich interessant oder, wie wir Menschen ticken? Wenn es um Geld geht, würden wir keinen Cent verschwenden. Wir würden es alle, in einem solchen Fall, mit vollen Händen ausgeben. Bis auf den letzten Penny. Nur, wie sieht es mit einer Sache aus, die noch viel wertvoller als alles Geld dieser Welt ist: unsere Zeit?

Jeden Tag werden Ihnen 86.400 Sekunden gutgeschrieben. Einsparen geht nicht. Der „Hahn" kann auch jederzeit abgedreht werden... nutzen Sie Ihre Zeit genauso sinnvoll, wie Sie das Geld im anderen Fall ausgeben würden? Leben Sie jeden Tag zu 100 Prozent bewusst und genießen Sie jede einzelne Sekunde? Mag sein, dass das zu viel verlangt sein mag, aber es lohnt sich, sich dem anzunähern. Wir haben nur das eine Leben. Wie viel Zeit verschwenden wir ständig mit unnötigen TV-Sendungen, E-Mails checken, Social-Media-Kanälen, wo wir uns das Leben anderer ansehen, bei denen immer alles perfekt ist?

Mein Freund Berti war schockiert, aber gleichzeitig auch erleichtert über einige Erkenntnisse, die er in so kurzer Zeit gewonnen hatte: „Darüber habe ich mir so noch nie Gedanken gemacht..." „Ja, weißt du, was auch ziemlich augenöffnend sein kann? Wenn man mal die Tage ausrechnet, die man noch zu leben hat. Die durchschnittliche Lebenserwartung eines Mannes beträgt 85 Jahre. Das wären dann rund 31.000 Tage, **insgesamt**. 31.000... wirkt plötzlich ziemlich real, oder? Nehmen wir jetzt mal einen 35-jährigen, der ja gesellschaftlich noch als recht jung betitelt wird. Wenn er jetzt theoretisch noch 50 Jahre zu leben hätte, dann sind das gerade mal 18.250 Tage! Spätestens ab dem Zeitpunkt, an dem einem das bewusst wird, muss man doch jeden Tag so leben, als wäre es der letzte, alle Träume und Pläne in die Realität umsetzen, oder etwa nicht?"

In diesem Moment riss Berti seine Augen weit auf, sprang vom Stuhl und stand zappelnd vor mir: „Wow! Hey, ich muss was tun! Vince, jetzt ehrlich! Ich muss irgendwas ändern in meinem Leben, das ist ja ein Witz, wie wenig Zeit wir eigentlich haben!"

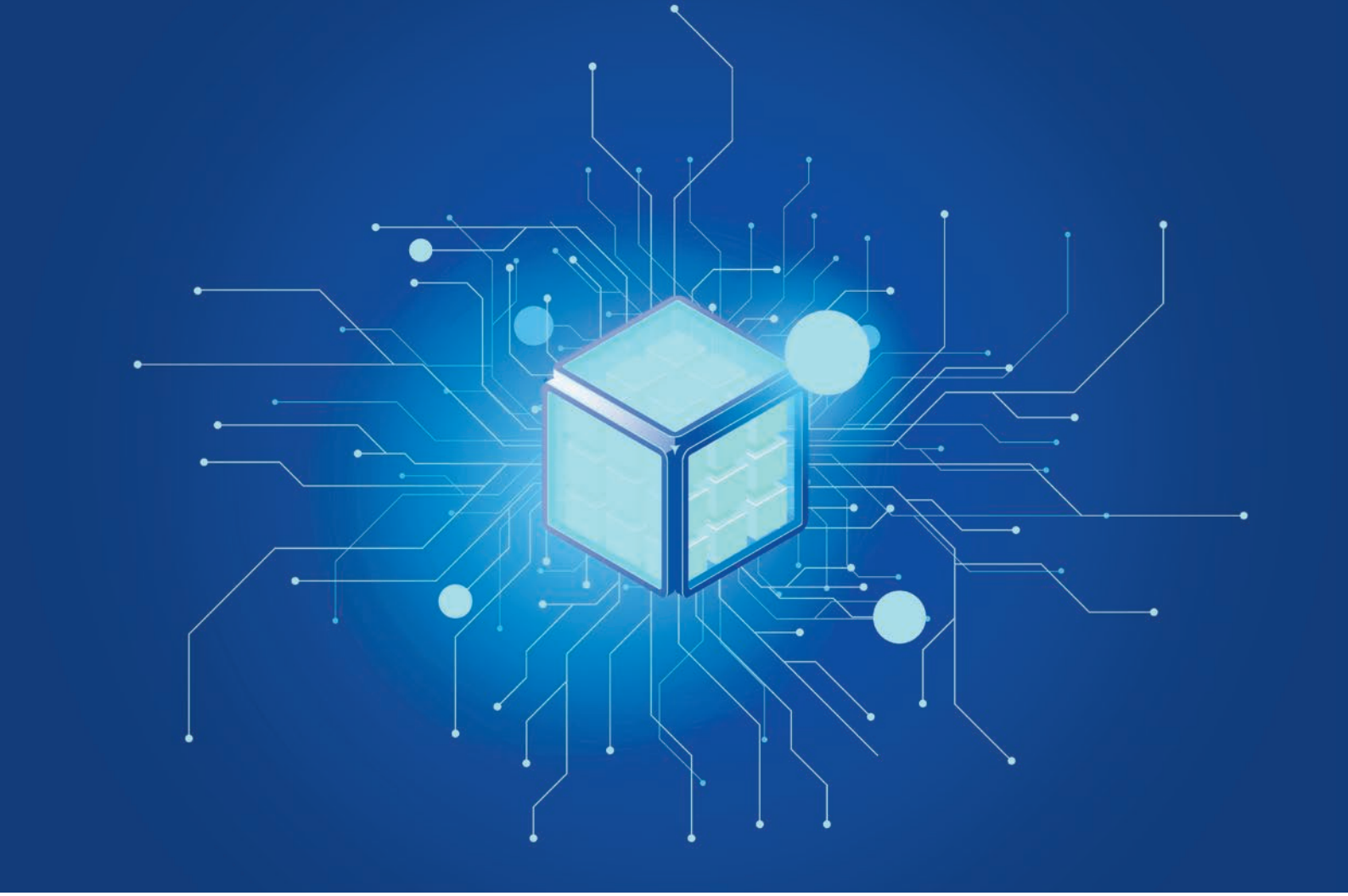
Wir amüsierten uns prächtig über seine Reaktion und die Erkenntnisse. Interessant war vor allem, da wir ja hier über das Thema Motivation sprechen, wie unfassbar schnell diese in ihm plötzlich aufkam, nachdem er seine Uhr lauter ticken hörte.

Machen Sie den Test mit sich. Rechnen Sie sich aus, wie viele Tage Sie noch erleben dürfen. Wie viel wertvolle Zeit Ihnen noch geschenkt wird. Zumindest hypothetisch. Auch wenn ihr Ergebnis unangenehm für Sie erscheinen mag, blicken Sie der Tatsache mutig ins Auge. Ich meine, wann lernen wir mehr? Wenn wir fröhlich auf der Schaukel schwingen und es uns dabei gut geht – oder wenn ein Familienmitglied gestorben ist und wir feststellen, dass wir viel zu wenig Zeit mit diesem verbracht haben? Sorgen Sie sich nicht darum, zu wenig Zeit übrig zu haben. Bereuen Sie es nicht, sie teilweise verschwendet zu haben.

Akzeptieren Sie es und fassen Sie den eisernen Entschluss, ab sofort einen Ihrer kostbarsten Vermögenswerte dauerhaft sinnvoll zu investieren. Entdecken Sie Ihr „Warum" und richten Sie Ihr ganzes Leben danach aus. Investieren Sie einen Großteil Ihrer Zeit in die Beantwortung dieser Frage, indem Sie langfristig danach streben, nur noch Tätigkeiten und Dienstleistungen auszuführen, die Sie glücklich machen. Führen Sie ab jetzt ein Leben, auf das es sich später lohnt zurückzublicken. Tschakka.

Vincent Schwarzmeier
DCD Training
Vincent.schwarzmeier@dalecarnegie.de

Vincent Schwarzmeier war damals der international jüngste, lizenzierte Seminarleiter von Dale Carnegie Training, den es seit über 100 Jahren je gegeben hat. Als der deutsche Franchisenehmer im Erstgespräch Vincent den Einwand entgegenbrachte, er sei mit 22 leider noch zu jung, diese Trainings zu geben, überzeugte dieser erst seinen Chef von sich und setzte sich anschließend gegen Hunderte andere Bewerber durch. Heute ist er im Auftrag für große Unternehmen als Trainer, Coach und Vortragsredner tätig und begeistert Tausende Menschen deutschlandweit. Zu seinen Stärken gehören seine Leidenschaft für das Entwickeln seiner Teilnehmer, seine Begeisterungsfähigkeit und seine Inspirationskraft.



Entwickeln mit der Autonomous Database

Marcel Amende und Kersten Mebus, ORACLE Deutschland B.V. & Co. KG

Hardware anschaffen, Betriebssysteme aufsetzen, Datenbanken installieren und dabei immer hoffen, dass der DBA einen guten Tag erwischt. In Zeiten schneller Zielerreichung, agiler Entwicklung und Rapid Prototyping kann der Entwickler solche langwierigen Prozesse nicht mehr hinnehmen. Mit der Oracle Autonomous Database [1] ist eine hochverfügbare, hochskalierbare, sichere, administrations- und wartungsfreie Datenbank in der Cloud nur wenige Klicks und Minuten entfernt. Der Artikel beleuchtet die Besonderheiten bei der Entwicklung von Java-, SpringBoot- und JavaScript-Anwendungen mit der Oracle Autonomous Database.

Die Autonomität der Datenbank bezieht sich auf drei Kernbereiche: Sie wird vollautomatisch provisioniert, unterbrechungs- und administrationsfrei gesichert, gewartet, aufgerüstet und optimiert. Ihre Daten sind immer verschlüsselt und die Sicherheitsfunktionalität wird zum Schutz vor internen und externen Bedrohungen laufend gepflegt. Fehlerzustände werden automatisch erkannt und behoben, um auf Grundlage der Clustertechnologie der Oracle-Datenbank („Real Application Cluster, RAC") und einer regionsübergreifenden Replikation einen unterbrechungsfreien Zugang zu den Daten zu gewährleisten. Eine Verfügbarkeit von 99,995 Prozent ist so garantiert.

Grundlagen der Autonomous Database

Die gute Nachricht vorweg: Die Oracle Autonomous Database lässt sich kostenfrei testen und sogar dauerhaft kostenfrei nutzen. Erste Gehversuche kann man im Rahmen eines kostenlosen Trials *[2]* (30 Tage, plus Credits im Wert von 300 US-Dollar) der Oracle Cloud unternehmen. Darüber hinaus sind mit jedem Oracle-Cloud-Zugang

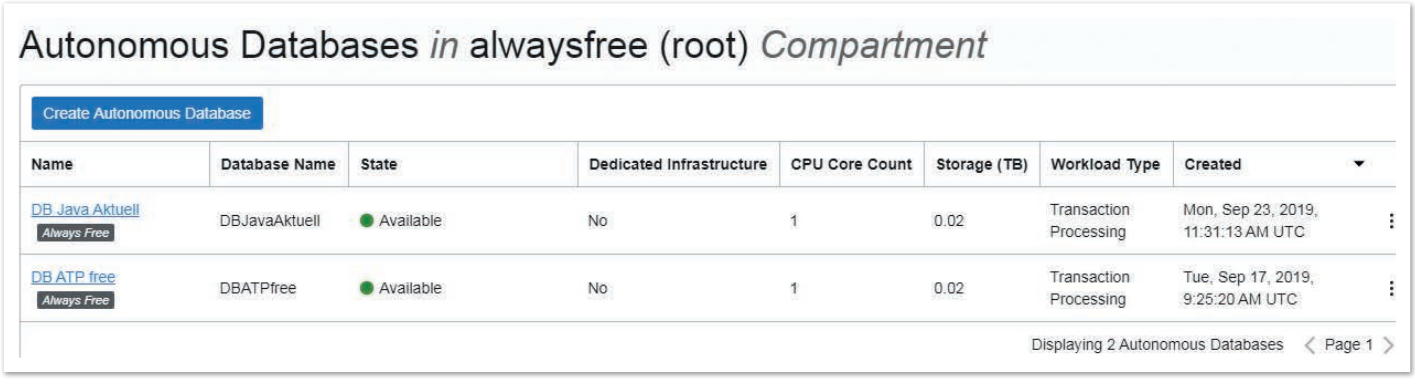


Abbildung 1: Autonome "Always Free" Datenbanken für verschiedene Workloads in der Oracle Cloud Konsole (Quelle: Marcel Amende)

zwei autonome Datenbanken mit jeweils einer OCPU (entspricht der CPU-Kapazität eines physischen Cores eines Intel-Xeon-Prozessors mit aktiviertem Hyperthreading) und 20 GB Datenspeicher als „Always Free"-Dienste *[3]* lebenslang kostenfrei nutzbar *(siehe Abbildung 1)*. Gleiches gilt übrigens auch für Compute (zwei VMs mit 1/8 OCPUs und 1 GB RAM), einen Load Balancer mit 10 MBit/s Bandbreite, 100 GB Block Storage, 10 GB Object Storage, 10 GB Archive Storage, 10 GB Outbound Traffic, eine Million Notifications und 1.000 E-Mails im Monat.

Die autonome Datenbank ist in zwei Produktvarianten und auf zwei verschiedenen Infrastrukturplattformen verfügbar. Die Varianten sind:

- Autonomous Transaction Processing (ATP), für transaktionale Verarbeitungen
- Autonomous Data Warehouse (ADW), für Data-Warehouse-Anwendungen

Beide Varianten lassen sich wiederum „Serverless" oder „Dedicated" betreiben. Die Grundlage für die schnelle Provisionierung ist in beiden Fällen die Mandantenfähigkeit der Oracle-Datenbank („Multitenancy"). Eine „Serverless"-Datenbank ist aus technischer Sicht ein isolierter und elastischer Cluster von Datenbankcontainern („Pluggable Database, PDB"), die auf einer extrem leistungsfähigen Oracle-Exadata-Datenbankmaschine in der Oracle Cloud laufen. Bei den „Dedicated"-Datenbanken nutzen Kunden exklusiv zugeordnete Exadata-Datenbankmaschinen in ihrem privaten Subnetz der Oracle Cloud für den Betrieb eigener Containerdatenbanken.

Provisionierung einer „Serverless"-Datenbank

Eine Autonomous Database kann über die Oracle Cloud Infrastructure (OCI) Konsole, die OCI-Kommandozeilenschnittstelle (OCI CLI) oder automatisiert per Terraform bereitgestellt werden. *Abbildung 2* zeigt die Provisionierung über die Konsole in der Rechenzentrumsregion Frankfurt. Es müssen lediglich ein Compartment, der Datenbankname, die Art der Arbeitslast (Data Warehouse oder Transaction Processing), der Infrastrukturtyp (Serverless oder Dedicated), die Anzahl der CPUs (bei Serverless 1 bis 128), die Speichergröße (1 – 128 TB beziehungsweise 20 GB bei „Always Free"), die Art der Datenbanklizenz (bereits vorhanden oder inkludiert) gewählt und ein Passwort vergeben werden. Nicht einmal drei Minuten später steht die neue Datenbank inklusive Oracle Application Express (APEX), Oracle Rest Data Services (ORDS), Oracle SQL Developer Web und Oracle Machine Learning SQL Notebooks zur Verfügung *(siehe Abbildung 3)*.

JDBC

Der Zugriff auf die Datenbanken in der Cloud erfolgt immer über eine sichere SSL-Verbindung (TLSv1.2) und mit einem Zertifikat. Java-Applikationen benötigen daher entweder einen Java Key Store (JKS) oder das sogenannte Oracle Wallet, um sich zu verbinden. Nach Provisionierung einer Autonomous Database kann die Wallet-Datei (Wallet_<Datenbankname>.zip) über die Administrationskonsole des Datenbankdienstes heruntergeladen werden. Dabei wird ein Passwort für einige Komponenten des Wallet vergeben. Das Wallet ist ein Zip-Archiv mit folgendem Inhalt:

- tnsnames.ora – Verbindungsdeskriptoren der Datenbank
- sqlnet.ora – Client-seitige SQL*Net-Konfiguration
- ewallet.p12 – passwortgeschütztes Wallet mit dem Zugriffszertifikat
- cwallet.sso – Single-Sign-on-Datei für die automatisierte Arbeit mit dem Wallet
- keystore.jks – passwortgeschützer Java Keystore mit dem öffentlichen RSA-2048-Schlüssel
- truststore.jks – passwortgeschützer Java Truststore mit den Zertifikaten der vertrauenswürdigen Server
- ojdbc.properties – JDBC-Verbindungsparameter, verweist unter Nutzung der Umgebungsvariable TNS_ADMIN auf den Speicherort des Wallet

Da das Wallet zusammen mit einem Benutzer und Passwort den Zugang zu einer Datenbank ermöglicht, sollte es immer an einem sicheren Ort und nur mit eingeschränkten Zugriffsrechten (zum Beispiel Dateiberechtigung 600 unter Linux) abgelegt werden. Natürlich sollte es auch niemals auf unsicherem Wege (beispielsweise per öffentlicher E-Mail) und immer getrennt von seinem Zugriffspasswort versendet werden. Erweiterungen in den JDBC-Treibern der Version 19c (19.3) und 18c (18.3) vereinfachen die Benutzung der Wallet-Dateien. Für die weitere Verwendung packt man das Wallet-Archiv in einem Verzeichnis nach Wahl aus und setzt die Umgebungsvariable TNS_ADMIN auf das Wallet-Verzeichnis: `$ export TNS_ADMIN=<WALLET_VERZEICHNIS>`.

Die neuesten JDBC-Treiber sind auf der Oracle-Website *[4]* zu finden. Passend zum verwendeten JDK lädt man hier die neuesten Treiber herunter. Wenn eine JDK-Version kleiner als JDK8u162 (JDK 8, Update 162) verwendet wird, muss man zusätzlich die „Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8"-Dateien herunterladen *[5]*. Mit JDK11, JDK10 und JDK9 ist das nicht nötig. Folgende Dateien werden benötigt:

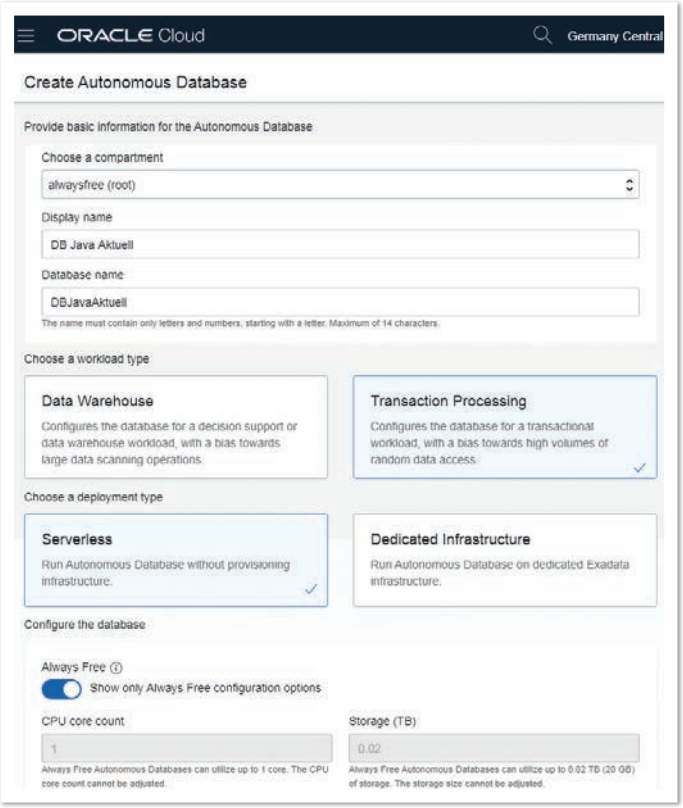


Abbildung 2: Einfache Provisionierung einer Autonomous Database über eine einzige Parametrisierungsseite (Quelle: Marcel Amende)

- ojdbc10.jar (zertifiziert mit JDK10) bzw. ojdbc8.jar (JDK8) – die JDBC-Treiber
- ucp.jar – der Universal Connection Pool
- oraclepki.jar, osdt_core.jar und osdt_cert.jar – für die Verwendung des Wallet

Für den Aufbau einer JDBC-Verbindung wird das Objekt `OracleDataSource` genutzt *(siehe Listing 1)*.

Der TNS-Alias setzt sich dabei aus Datenbanknamen und dem Namen einer Regel für die Verwaltung der Datenbankressourcen zusammen. Eine solche Regel definiert ein Laufzeitlimit für Datenbankabfragen, ein IO-Limit und das Verhältnis der möglichen CPU- und IO-Nutzung. Die Regeln sind über die Service-Konsole anpassbar. Vordefinierte Namen sind LOW, MEDIUM, HIGH, TP und TPURGENT *[6]*. Die gültigen Aliasse werden in der Datei „tnsnames.ora" im Wallet-Verzeichnis verwaltet. Für eine Datenbank namens „DBORCL" lautet der Alias für das Profil HIGH beispielsweise `dborcl_high`. Die JDBC-Verbindung lässt sich nun wie gewohnt verwenden *(siehe Listing 2)*.

Der komplette Beispielcode ist auf GitHub zu finden *[7]*. Zum Ausführen des Beispiels braucht man zusätzlich lediglich ein VM-Argument, das auf das Wallet-Verzeichnis zeigt *(siehe Listing 3)*.

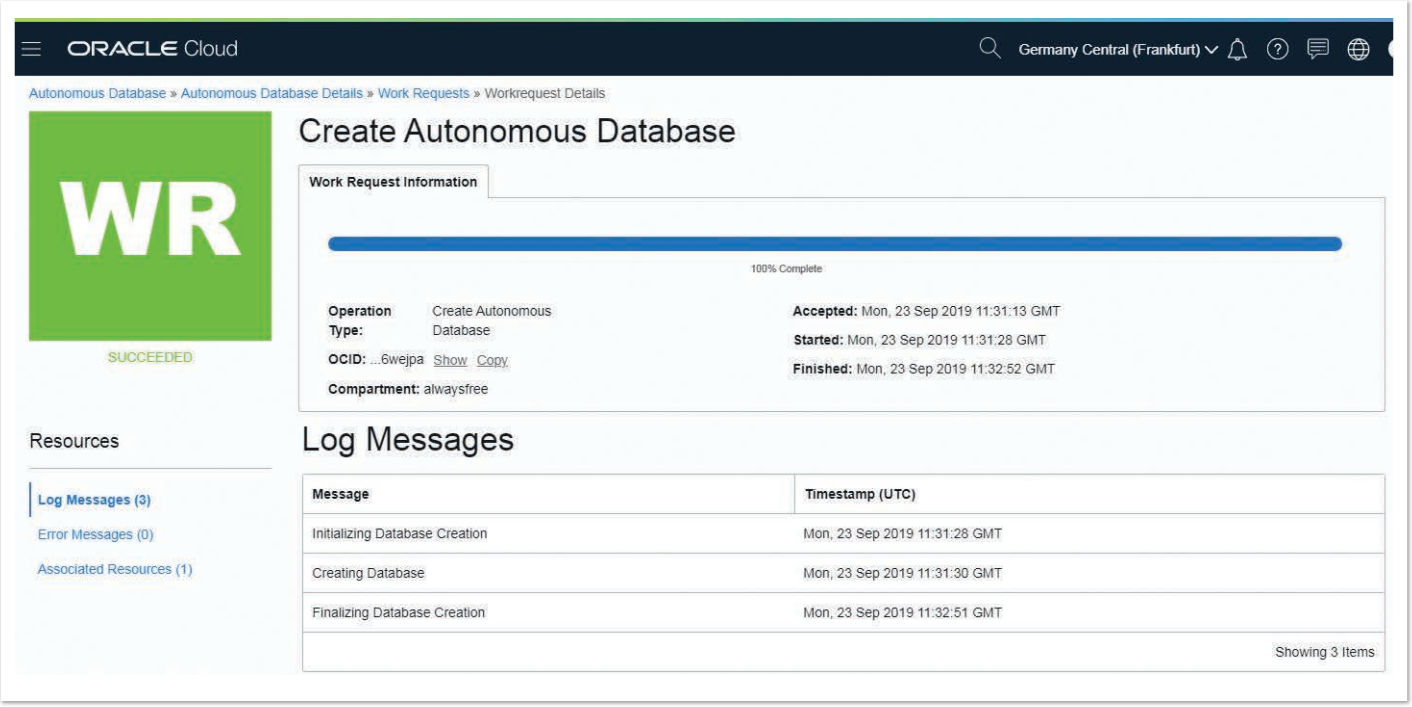


Abbildung 3: Erfolgreiche Provisionierung einer Autonomous Database in unter drei Minuten (Quelle: Marcel Amende)

```
ods = new OracleDataSource();
ods.setURL("jdbc:oracle:thin:/@<TNS_ALIAS>");
ods.setUser("<DB_BENUTZER>");
ods.setPassword("<DB_PASSWORT>");
OracleConnection con = (OracleConnection) ods.getConnection();
```

Listing 1

```
Statement s = connection.createStatement();
ResultSet rs = statement.executeQuery("select name from emp");
```

Listing 2

```
$ java -Doracle.net.wallet_location=<WALLET_VERZEICHNIS> …  autonomous.jdbc.SimpleJDBCTest
```

Listing 3

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL("jdbc:oracle:thin:/@<TNS_ALIAS>");
pds.setUser("<DB_BENUTZER");
pds.setPassword("<DB_PASSWORT>");
pds.setConnectionPoolName("JDBC_UCP_POOL");
pds.setInitialPoolSize(3);
pds.setMinPoolSize(3);
pds.setMaxPoolSize(5);
pds.setTimeoutCheckInterval(5); //in Sekunden
pds.setInactiveConnectionTimeout(20); //in Sekunden
Connection con = pds.getConnection();
```

Listing 4

```
<beans ...>
  <bean id="dataSource" class="oracle.ucp.jdbc.PoolDataSourceImpl">
    <property name="connectionFactoryClassName" value="oracle.jdbc.pool.OracleDataSource" />
    <property name="URL" value="jdbc:oracle:thin:/@<TNS_ALIAS>" />
    <property name="user" value="<DB_Benutzer>" />
    <property name="password" value="<DB_PASSWORT>" />
    <property name="maxPoolSize" value="5" />
    <property name="initialPoolSize" value="1" />
  </bean>
  ...
</beans>
```

Listing 5

```
...
@ImportResource({ "classpath*:AppConfig.xml" })
public class MyApplication {
...
```

Listing 6

Universal Connection Pool

Für eine skalierbare Anwendung sollte man einen JDBC Connection Pool verwenden. Der Oracle-JDBC-Treiber bietet hierfür den „Universal Connection Pool" (UCP), den man über die ucp.jar-Bibliothek einbinden kann. Ebenfalls auf GitHub *[7]* findet sich im Package `autonomous.jdbc.ucp` eine Beispielimplementierung als Singleton *(siehe Listing 4)*.

Microservices mit Spring Boot

Spring Boot ist das derzeit wohl beliebteste und gleichzeitig bewährteste Framework für die Entwicklung von Java-Applikationen in der Cloud. Es ist Open Source *[8]*, wird aber von der Firma Pivotel *[9]* massiv kommerziell unterstützt. Es eignet sich für die schnelle Erstellung von Microservices, ist aber auch mächtig genug für die Entwicklung modularer Monolithen. Natürlich unterstützt es auch JDBC-Verbindungen zur Oracle Autonomous Database und den Universal Connection Pool.

Zusätzlich zu den aufgeführten JDBC-Bibliotheken muss man in einem Spring-Boot-Projekt folgende Spring- (hier exemplarisch in der Version 4.3.4 *[10]*) und Spring-Boot-Bibliotheken (Version 1.4.2 *[11]*) einbinden:

- spring-boot-1.4.2.jar files
- commons-logging-1.2.jar
- spring-aop-4.3.4.RELEASE.jar
- spring-beans-4.3.4.RELEASE.jar
- spring-boot-1.4.2.RELEASE.jar
- spring-context-4.3.4.RELEASE.jar
- spring-core-4.3.4.RELEASE.jar
- spring-expression-4.3.4.RELEASE.jar
- spring-jdbc-4.3.4.RELEASE.jar
- spring-tx-4.3.4.RELEASE.jar

Die Verbindungsparameter der JDBC-Datenquelle werden in der XML-Datei für die Spring-Applikationskonfiguration angegeben *(siehe Listing 5)*. Die Konfigurationsdatei wird wiederum über eine

```
/* File main.js - start */
const oracledb = require('oracledb');

async function run() {
  let pool;

  try {
    pool = await oracledb.createPool({
      user: "<DB_USER>",
      password:  "<DB_PASSWORT>",
      connectString:  "<TNS_ALIAS>"
    });
    const con = await pool.getConnection();
    const res = await con.execute("select sysdate from dual");
    console.log(res.rows[0]);
  } catch (err) {console.error(err);}
}

run();
/* File main.js - end */
```

Listing 7

```
$ node main.js
[ 2019-09-19T13:08:08.000Z ]
```

Listing 8

`@ImportResource`-Annotation der Main-Klasse der Applikation referenziert *(siehe Listing 6)*. Die komplette, lauffähige Spring-Applikation ist ebenfalls auf GitHub zu sehen *[7]*.

JavaScript mit Node.js

Für eine Datenbankapplikation in JavaScript und die Ausführung in Node.js muss man drei Dinge vorbereiten:

1. Setzen der Umgebungsvariable TNS_ADMIN auf das Wallet-Verzeichnis: `$ set TNS_ADMIN=<WALLET_VERZEICHNIS>`
2. Herunterladen des Basic- oder Basic-Light-Pakets des Oracle Instant Client [12]. Nach dem Auspacken beziehungsweise Installieren nimmt man das entstandene Verzeichnis in den Suchpfad auf: `$ set PATH=<INSTANT_CLIENT_VERZEICHNIS>`
3. Installieren des Oracle-Datenbanktreibers mit dem Node-Paketmanager: `$ npm install oracledb`

Ein minimales JavaScript-Programm „main.js" sieht dann unter Verwendung eines Connection Pool wie in *Listing 7* abgebildet aus.
Es handelt sich dabei natürlich um ein eigenständiges Programm, keinen Web-Service. Die Verwendung eines Connection Pool ist daher exemplarisch zu sehen. Bei Ausführung gibt es die aktuelle Systemzeit der Datenbank aus *(siehe Listing 8)*.

RESTful Services und SODA

Natürlich gibt es Alternativen zu Datenbankabfragen über Datenbanktreiber. Oft ist es einfacher, die Datenbak gleich als RESTful Service zu konsumieren. In der Oracle Autonomous Database stehen dafür die „RESTful Data Service" (ORDS) und mit dem „Simple Oracle Document Access" (SODA) eine API-Abfrage im NoSQL-Stil bereit.

```
ORDS.ENABLE_SCHEMA(p_enabled => TRUE,
                   p_schema => '<DB_SCHEMA>',
                   p_url_mapping_type => 'BASE_PATH',
                   p_url_mapping_pattern => '<REL_URL_PFAD>',
                   p_auto_rest_auth => FALSE);
```

Listing 9

```
ORDS.ENABLE_OBJECT(p_enabled => TRUE,
                   p_schema => '<DB_SCHEMA>',
                   p_object => '<DB_TABELLE>',
                   p_object_type => 'TABLE',
                   p_object_alias => '<SERVICE_NAME>',
                   p_auto_rest_auth => FALSE);
```

Listing 10

```
https://<RANDOM>-<DB_NAME>.adb.<DATACENTER_NAME>.oraclecloudapps.com/ords/< REL_URL_PFAD >/<SERVICE_NAME>/
```

Listing 11

Ein Datenbankschema kann man mit einem einfachen PL/SQL-Prozeduraufruf unter einem relativen Pfad für REST Services vorbereiten *(siehe Listing 9)*.

Nun kann eine Tabelle des Schemas als RESTful Service bereitgestellt werden, der CRUD-Operationen in SQL DML und die Daten in JSON-Formate übersetzt *(siehe Listing 10)*. Dabei ist Vorsicht geboten, denn daraus resultiert ein öffentlich aufrufbarer REST Service unter der in *Listing 11* angegebenen URL.

REST-Ressourcen lassen sich durch die Vergabe von Privilegien sehr einfach mit einer OAuth2- oder BASIC-Authentifizierung versehen. Die Operationen zur Erstellung eines REST Service können mit dem SQL Developer auch per Knopfdruck ausgeführt werden.

Zusammenfassung

Für den (Java-)Entwickler ändert sich mit der Oracle Autonomous Database gegenüber der Nutzung der klassischen Datenbank glücklicherweise wenig. Jede Kommunikation mit der Datenbank ist aber zwingend verschlüsselt und signiert. Der organisatorische Wert der autonomen Datenbank kann hingegen immens sein. Bei einer agilen Softwareentwicklung braucht man praktisch auf der Stelle die nötigen Ressourcen, ohne sich auf unabsehbare Kosten und Aufwände einzulassen. Dabei kann die Oracle Autonomous Database einen wichtigen Beitrag leisten. Als „Always Free"-Dienst steht sie in circa drei Minuten kostenfrei für die Umsetzung neuer Ideen bereit. Darüber hinaus zahlt man flexibel nur die tatsächliche Nutzung. Im Erfolgsfall kann man im laufenden Betrieb nahezu beliebig skalieren und hat vom Projektstart bis zum produktiven Einsatz immer ein sicheres, absolut wartungsfreies und hochverfügbares System. Bei höchstem Bedarf an Sicherheit und Datenschutz sogar in einem kompletten privaten Subnetz mit eigenen Schlüsseln. Für den Java-Entwickler sollte die autonome Datenbank daher einen Blick wert sein. Auch die System- und Datenbankadministratoren werden – befreit von Routinearbeiten – anspruchsvollere und interessantere Aufgaben im Entwicklungsprojekt übernehmen können.

Quellen

[1] *https://www.oracle.com/database/autonomous-database.html*
[2] *https://cloud.oracle.com/tryit*
[3] *https://www.oracle.com/cloud/free/*
[4] *https://www.oracle.com/database/technologies/appdev/jdbc-ucp-19c-downloads.html*
[5] *https://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html*
[6] *http://bit.ly/servicenames-doc*
[7] *https://github.com/ora-autonomous/jdbc*
[8] *https://github.com/spring-projects/spring-boot*
[9] *https://spring.io/projects/spring-boot*
[10] *https://repo.spring.io/release/org/springframework/spring/4.3.4.RELEASE/*
[11] *https://jar-download.com/artifacts/org.springframework.boot/spring-boot/1.4.2.RELEASE/source-code*
[12] *https://www.oracle.com/database/technologies/instant-client/downloads.html*
[13] *https://www.oracle.com/tools/downloads/sqldev-v192-downloads.html*

Marcel Amende
Oracle
Marcel.Amende@oracle.com

Geboren als Ingenieur, aufgewachsen bei Oracle, zuhause in der Cloud. Als Solution Engineer repräsentiert Marcel die „Cloud Native"-Entwicklergemeinde von Oracle, die aus den Diensten der Oracle Cloud kreative Kundenlösungen gestaltet.

Kersten Mebus
Oracle
Kersten Mebus@oracle.com

Kersten Mebus ist schon seit seiner Schulzeit von Technologie begeistert. Sein Motto ist: Alles, was digitalisiert werden kann, wird digitalisiert. Als Cloud-Experte und Solution Engineer für Integration/Prozesse, Anwendungsentwicklung und Datenbanken erarbeitet er zusammen mit dem Kunden kundenspezifische Lösungen für die Cloud.

DevOps?! Gerne, aber richtig!

Maximilian Braun, virtual7 GmbH

Viele Organisationen gehen den Weg in Richtung DevOps. Einige von ihnen schon länger und sehr erfolgreich. Von diesen Teams lässt sich lernen, wie eigenständige Teams mit zufriedenen Mitarbeitern schnell Features an ihren spezifischen Markt bringen und dabei eine stabile Produktionsumgebung gewährleisten.

Bevor der DevOps-Frust einsetzt, sollte man sich Gedanken um die Transformation machen, denn DevOps ist weniger Technologie als vielmehr Zusammenarbeit, Kultur, Lean-Produktmanagement und Lean-Management. In Studien und Umfragen wird erforscht, was Organisationen umsetzen, die erfolgreich DevOps einsetzen. In diesem Artikel stelle ich einige Ergebnisse aus meiner Erfahrung aus Ministerien, Behörden und der öffentlichen Hand Deutschlands, der Wirtschaft und mehreren Studien vor.

Was ist DevOps?

Für DevOps gibt es keine allgemeingültige Definition. Das habe ich in den Organisationen, die bereits auf dem Weg sind und die ich begleitet habe, kennengelernt. Dadurch kommt es oft zu Missverständnissen und enttäuschten Erwartungen. Alle Definitionen, die ich kenne, lassen sich für mich jedoch damit zusammenfassen: „DevOps ist eine Reihe von Methoden, die es Organisationen erlauben, Änderungen an Systemen schnell durchzuführen, ohne die Qualität der Systeme zu beeinträchtigen" *[1]*. Konkret heißt dies, dass wir Features mit DevOps schnell und in hoher Qualität an unsere Kunden bringen können. Um das zu ermöglichen, ist es nötig, dass die einzelnen Teams verantwortlich für den kompletten Lifecycle ihres Produktes sind; von der Konzeption über die Entwicklung und Integration bis hin zum Betrieb.

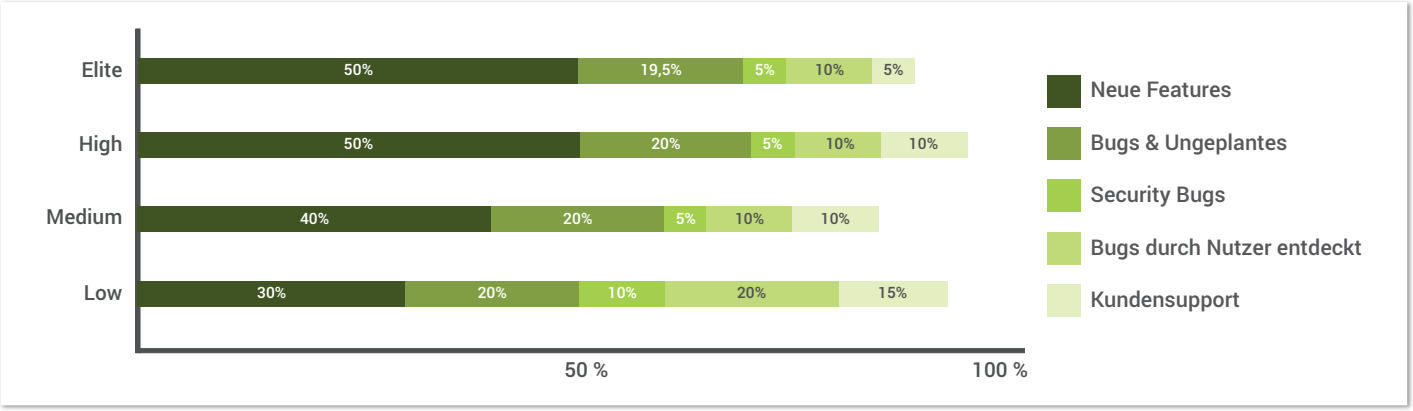


Abbildung 1: Auslastung der Befragten der „State of DevOps"-Studie 2018 [2], gruppiert nach Performance. Visualisierung © virtual7 GmbH

Mehr Features durch DevOps

Ein Aspekt, wieso Organisationen auf DevOps setzen, ist, dass sie mehr Features an den Markt bringen möchten. Die Thesen der „State of DevOps"-Studien unterstützen dieses Vorhaben. Beispielsweise verbringen Organisationen und Teams, die als „Elite" klassifiziert wurden, 50 Prozent ihrer Zeit mit der Entwicklung neuer Features *[2]*, wohingegen „Low-Performer" nur 30 Prozent ihrer Zeit in neue Features investieren. Sie investieren hingegen das Doppelte *[2]* ihrer Zeit in Bugs und Security-Probleme. Außerdem ein Vielfaches in klassische Customer-Support-Tätigkeiten *(siehe Abbildung 1)*.

Der DevOps-Weg

Wenn eine Organisation sich entschieden hat, den DevOps-Weg zu gehen, entscheidet sie sich dafür, all ihre Prozesse und Kriterien zu überdenken. Silos zwischen Betrieb und Entwicklung werden abgerissen. Alte Wege werden von anderen Menschen beschritten und neue Wege tun sich auf. Deswegen ist es wichtig, dass dieser Weg durch regelmäßige Retrospektiven begleitet und neu ausgerichtet wird. Fortschritte sollten messbar sein, damit Experimente evaluiert werden können.

Erfolge messen

Wie für jede Organisation ist es für DevOps-Organisationen wichtig, KPIs (Key Performance Indicators/Kennzahlen) zu erfassen, die neben finanziellen Aspekten den Erfolg der Organisation messen. Integraler Bestandteil einer KPI ist es, die Leistung eines Systems auf den Punkt zu bringen. Der Mehrwert, den DevOps Organisationen bietet, ist meist die bedarfsgerechte Entwicklung und der Betrieb von Software. In manchen Fällen nutzen Organisationen nur betriebsrelevante KPIs, um den eigenen Erfolg zu messen. Die Erfahrung zeigt, dass es hier kein allgemeingültiges Bild auf DevOps-KPIs gibt. Da passiert es oft, dass es Dev- und DevOps-Teams gibt. DevOps löst die Silos „Dev" und „Ops" auf. Als Team sollten wir jedoch Verantwortung für den gesamten Prozess übernehmen und einer Trennung entgegenwirken.

Aus dem Zusammenlegen beider Bereiche lassen sich fünf KPIs ableiten, die zu einer übergeordneten KPI zusammengefasst wird. Der erste Indikator **Delivery Lead Time To Change** beschreibt, wie viel Zeit zwischen Fertigstellen der Änderung (commit) und tatsächlichem Übergang in die Produktion vergeht *(siehe Tabelle 1)*. Die Kennzahl **Deployment Frequency** beinhaltet, wie oft Änderungen in die Produktion gebracht werden *(siehe Tabelle 2)*. Außerdem die **Change Fail Rate**: die Anzahl der fehlgeschlagenen Änderungen, den erfolgreichen Änderungen (zum Beispiel Bugs und zurückgerollte Releases) gegenübergestellt. Unter **Mean Time To Restore** versteht man, wie viel Zeit nach einem Ausfall vergeht, bis die Systeme wieder normal operieren *(siehe Tabelle 1)*. Zuletzt benötigen wir noch die **Availability**, also die Verfügbarkeit im Jahresmittel der Produktionssysteme. Diese fünf KPIs lassen sich unter dem Dach der **Software Delivery Performance** zusammenfassen. Als DevOps-Organisation ist die Software Delivery Performance unsere KPI. Sie fasst alle Themengebiete zusammen, in denen wir Experten sind.

Diese KPIs können verwendet werden, um den DevOps-Weg zu beobachten und zu prüfen. Experimente, Methoden und Technologien zeigen ihre Auswirkungen und sind echte Daten, nicht nur ein Gefühl. Gerade am Anfang lohnt es, sich die Frage zu stellen, wie man innerhalb dieser KPIs steht. Die „State of DevOps"-Studie 2018 *[2]* hat ihre Teilnehmer in mehrere Kategorien aufgeteilt: Elite-, High-, Medium- und Low-Performer. Beim Vergleich von High- und Low-Performern zeigen sich große Unterschiede.

Weniger als eine Stunde
Weniger als ein Tag
Zwischen einem Tag und einer Woche
Zwischen einer Woche und einem Monat
Zwischen einem Monat und sechs Monaten
Mehr als sechs Monate

Tabelle 1: Klassifikation Delivery Lead Time To Change (Wie viel Zeit wird benötigt, um eine Änderung zum Kunden auszuliefern?) und Mean Time To Restore (Wie viel Zeit wird benötigt, um die Systeme in den Normalzustand zurückzuversetzen?) KPIs.

Bei Bedarf (mehrfach am Tag)
Zwischen einmal in der Stunde und einmal am Tag
Zwischen einmal am Tag und einmal in der Woche
Zwischen einmal in der Woche und einmal im Monat
Zwischen einmal im Monat und einmal in sechs Monaten
Weniger als einmal in sechs Monaten

Tabelle 2: Klassifikation Deployment Frequency (Wie oft führe ich Deployments in der Produktion aus?)

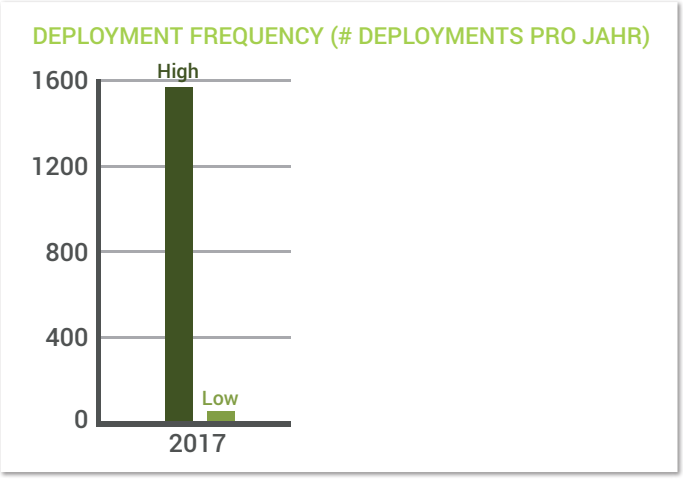


Abbildung 2: Frequenz der Depoyments: High- vs. Low-Performer, „State of DevOps Studie"-2018 [2]. Visualisierung © virtual7 GmbH

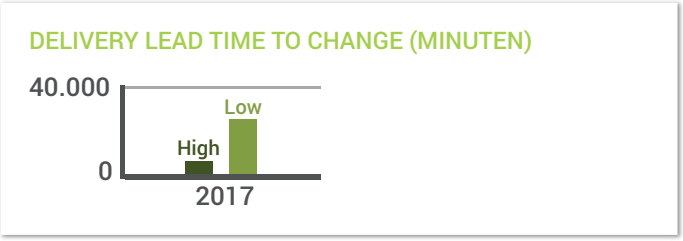


Abbildung 3: Delivery Lead Time To Change: High- vs. Low-Performer, „State of DevOps Studie"-2018 [2]. Visualisierung © virtual7 GmbH

So deployen High-Performer ihre Software beispielsweise 46 Mal *(siehe Abbildung 2)* öfter in die Produktion als Low-Performer. Die benötigte Zeit von Änderungen bis hin zum Übergang in die Produktion ist ganze 440 Mal kleiner *(siehe Abbildung 3)*. Im Falle eines Ausfalls können High-Performer ihre Systeme 170 Mal schneller in den Normalzustand zurückversetzen *(siehe Abbildung 4)*. Außerdem resultieren Änderungen fünf Mal seltener in Rollbacks oder Bugs *(siehe Abbildung 5)*. Das allein sind starke Argumente für den DevOps-Weg.

Capabilities

Gerade bei Themen mit Technologie-Fokus nutzen wir oft Reifegradmodelle, um unseren Fortschritt zu messen. Reifegradmodelle beschreiben einen statischen Pfad, sind unterteilt in mehrere Stufen und charakterisieren einen finalen Zustand. Das passt nicht zum DevOps-Weg. Ähnlich wie eine agile Transition hat es viel mehr mit Menschen als mit Technologie zu tun. Das lässt sich nur schwer in Grade einteilen und wird nie wirklich fertig. Deswegen kann ein flexibleres Modell, dass auf Capabilities (Kompetenzen/Fähigkeiten) setzt, auf die Bedarfe einzelner Organisationen besser eingehen. Basierend auf den „State of DevOps"-Studien der letzten Jahre, wurden 24 Capabilities in fünf Kategorien identifiziert *[3]*, die auf die Software Delivery Performance eingehen: Continuous Delivery, Architektur, Produkt und Prozesse, Lean-Management oder -Monitoring sowie Kultur.

1. Continuous Delivery Capabilities

Ein Teil der 24 Capabilities lässt sich unter Continuous Delivery klassifizieren. DevOps-Teams können viele dieser Methoden ohne Abhängigkeiten einsetzen und so den Grundstock für eine erfolgreiche Transition schaffen.

1.1 Versionskontrolle – überall

Die Nutzung moderner Versionskontrollsysteme bildet das Fundament für den DevOps-Weg. Dennoch sollte nicht nur der Code der Anwendungen versioniert werden, auch die Konfiguration der Anwendung, Skripte für die Automatisierung, System- und Umgebungskonfigurationen sollten versioniert werden.

1.2 Vollautomatisierte Deployment-Prozesse

Deployments, die ohne einen manuellen Schritt durchgeführt werden, beschleunigen das Team beim Rollout überdurchschnittlich. Gleichzeitig erfordert dies verlässliche Tests.

1.3 Regelmäßige Check-ins – Continuous Integration

Als erster Schritt, um Continuous Delivery zu implementieren, gilt Continuous Integration. Wenn wir Continuous Integration einsetzen, dann checken wir regelmäßig unseren Code ein. Dieser Check-in wird automatisiert getestet und Fehler können frühzeitig behoben werden. Der Continuous-Integration-Prozess erstellt die ersten Artefakte, die später „released" und „deployed" werden können.

1.4 Weg mit den langlebigen Branches – Trunk-Based Development

Trunk-Based Development ist eine Methode, Versionsverwaltung zu nutzen. Dabei werden maximal drei Branches genutzt. Branches haben generell eine sehr kurze Laufzeit, beispielsweise von maximal einem Tag, bevor sie in den Master zurückgeführt werden. Diese Methode ermöglicht disziplinierten Teams, durch die Anwendung von Feature-Toggles, eine Code-Freeze-freie Entwicklung. Die „State of DevOps"-Studien zeigen, dass Trunk-Based Development die Methode ist, die hoch-performante DevOps-Teams ausmacht.

1.5 Test-Automatisierung

Gute Automatisierung der Tests zeichnet sich dadurch aus, dass sie verlässlich funktioniert und ohne manuellen Aufwand in unseren Softwareentwicklungsprozess eingebunden ist. Wir müssen uns darauf verlassen können, dass Tests echte Bugs finden und nicht falsch-positive Ergebnisse liefern. Als Entwickler sollten wir für diese Tests verantwortlich sein. Diese Merkmale zeichnen eine gute Test-Automatisierung aus.

1.6 Testdaten-Management

Tests erfordern Daten. Diese müssen regelmäßig und sorgfältig gepflegt werden. Daten, die für die automatisierten Tests benötigt werden, sollten zu jeder Zeit verfügbar und anpassbar sein und nicht durch die Anzahl der Tests oder die Anzahl der Ausführungen limitiert werden. Bei Testdaten gilt der Grundsatz, wie bei Applikationscode: Weniger ist mehr.

1.7 Integration von Security in den Softwareentwicklungsprozess – DevSecOps

Security-bezogene Bugs zählen zu den schwerwiegendsten Problemen. Sie von Anfang an zu vermeiden muss das oberste Ziel sein. Wenn Info-Sec-Experten bereits frühzeitig in das Design, die Entwicklung und das Testing hinzugezogen, Security Reviews abgehalten und geprüfte Security-Bibliotheken verwendet werden, dann können viele Bugs im Vorfeld vermieden werden. Teams lernen von diesen Experten und können die Expertise aus dem Team heraus in die Projekte tragen. Sicherheitsrelevante Features sollten in den automatisierten Test-Suites getestet werden.

1.8 Continuous Delivery implementieren

Continuous Delivery ist eine Methode, mit der Software zu jeder Zeit imstande ist, ausgerollt zu werden, und das zu jedem Zeitpunkt im Entwicklungszyklus. Diesen Zustand zu halten, ist, noch vor neuen Features, von höchster Priorität. Durch Transparenz und schnelles Feedback zu Qualität und Deploybarkeit können Probleme schnell durch das ganze Team behoben werden. Ziel ist es, bei Bedarf die Systeme in der Produktion zu jeder Zeit aktualisieren zu können.

Weitergehendes wird im Buch „Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation" (ISBN 9780321601919) von Jez Humble angesprochen.

2. Architektur-Capabilities

Klassische Architektur fokussiert sich auf Tool- und Technologieauswahl. Erfolgreiche DevOps-Teams wissen, welche Tools und Technologien sie benötigen. Architekten in diesem Umfeld arbeiten mit den Teams eng zusammen, damit sie bessere Ergebnisse erzielen können, zeigen ihnen neue Wege, Tools und Technologien auf.

2.1 Lose gekoppelte Architekturen

Eine lose gekoppelte Architektur ermöglicht es Teams, das Test- und Deploy-Stadium ohne Abstimmungsaufwände mit anderen Teams und Services durchzuführen. Damit erreicht das Team eine Eigenständigkeit, die das Entwickeln und Ausliefern beschleunigen kann.

2.2 Architektur für eigenständige Teams

Architekten in DevOps-Teams können Teams am besten unterstützen, indem sie gemeinsam Fragestellungen bearbeiten. Der Fokus des Architekten sollte darauf liegen, die Ergebnisse des Teams zu optimieren. Dabei können beispielsweise die Minimierung der Bedarfe für Integrationstests und die Unabhängigkeit des Teams bei Test und Deployment Thema sein. Teams wissen, mit welchen Tools und Technologien sie am besten arbeiten können. Eine mögliche Fragestellung kann sein, wie man noch mehr aus diesen Tools herausholen kann.

3. Produkt- und Prozess-Capabilities

Diese Capabilities aus dem Lean-Product-Management zahlen direkt auf die Software Delivery Performance ein, wobei die Software Delivery Performance das Lean-Product-Management positiv beeinflusst. Das heißt, dass das Produktmanagement den DevOps-Weg unterstützen kann. Teams können gleichzeitig das Produktmanagement selbst betreiben oder mit Daten unterstützen.

3.1 Kundenfeedback sammeln und implementieren

Um die Produktentwicklung positiv zu beeinflussen, haben Teams eine Reihe von Möglichkeiten. Neben der Erhebung anonymisierter Daten zu Nutzerverhalten und dem Messen von erfolgreichen und gescheiterten Vorgängen ist das direkte Feedback von Nutzern eine weitere Größe, die in das Design und die Entwicklung des Produktes einfließen kann.

3.2 Experimente fördern

Basierend auf diesen Daten und der Möglichkeit, Spezifikationen direkt anzupassen, kann die Innovation direkt aus den Teams erfolgen. Die Nähe zum Produkt, den Kunden und den Daten erlaubt es, Experimente zu starten und aus ihnen zu lernen. Damit kann das Team direkt neue Werte am Produkt schaffen, ohne „von oben" einen Segen zu erhalten.

3.3 Arbeitsabläufe sichtbar machen

Das Verständnis und die Sichtbarkeit, wie neue Arbeitspakete zum Team gelangen, von Business-Seite bis zum Kunden, hilft Teams, fundierte Entscheidungen im Alltag zu treffen. Die Verwendung von Bibliotheken bis hin dazu, wie Systeme geschnitten werden, ändert sich mit dem Kontext, den das Team hat.

3.4 Arbeiten mit kleinen Paketen

Continuous Delivery ermöglicht es, jederzeit neue Features an den Kunden zu bringen. Im Produktmanagement sollte dies ein Gegenstück haben: Ein großes Projekt wird in kleine Features zerlegt, die in schnellen Zyklen ausgerollt und verbessert werden. Das ermöglicht frühes Feedback durch Methoden wie A/B-Testing.

4. Lean Management und Monitoring Capabilities

Nach den klassischen Ansätzen im Management von Software-Entwicklungsteams und der Adaption agiler Methoden steht nun mehr und mehr Lean im Fokus. Diese Capabilities legen den Fokus auf die Produktivität der Teams.

4.1 Changes ohne aufgeblasene Prozesse

Die Forschung durch DevOps-Studien zeigt *[3]*, dass die Freigabe von Changes durch teamfremde Change-Approval-Boards die Software Delivery Performance negativ beeinflusst und nicht für die Stabilität sorgt, die damit verbunden wird. Leichtgewichtige Prozesse, wie Peer-Reviews, wirken sich hingegen positiv auf die Software Delivery Performance aus.

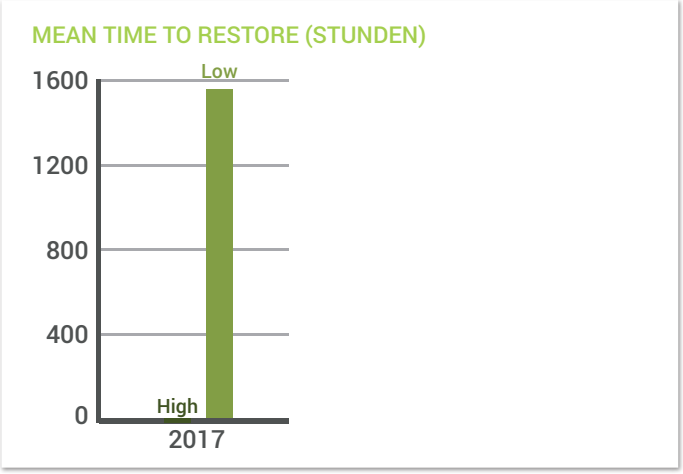


Abbildung 4: Mean Time To Restore: High- vs. Low-Performer, „State of DevOps Studie"-2018 [2]. Visualisierung © virtual7 GmbH

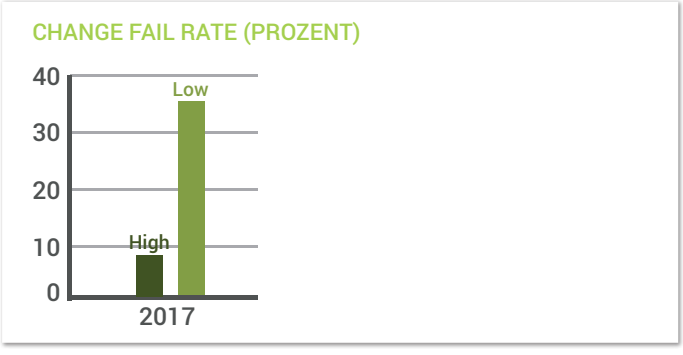


Abbildung 5: Change Fail Rate: High- vs. Low-Performer, „State of DevOps Studie"-2018 [2]. Visualisierung © virtual7 GmbH

4.2 Fundierte Entscheidungen durch Monitoring

Die Daten, die Anwendungen und Infrastruktur produzieren, können dafür genutzt werden, um technische Maßnahmen zu ergreifen. Das Produktmanagement profitiert von diesen Daten und kann datengestützte Entscheidungen treffen.

4.3 Proaktives Monitoring

Erfolgreiche DevOps-Teams überwachen Systeme mithilfe von Schwellwerten und Warnungen bei Änderungsraten. So können diese Teams verbeugende Maßnahmen für Probleme treffen und laufen ihnen nicht hinterher.

4.4 Work-in-Progress Limits

Ein Work-in-Progress Limit, also das bewusste Limitieren von Paketen, die gleichzeitig abgearbeitet werden, wirkt sich positiv auf den Durchsatz und die Prozesse aus. Außerdem werden Randbedingungen und Beziehungen zwischen Arbeitspaketen transparenter.

4.5 Transparenz bei Qualität und Arbeit für das ganze Team

Tools, die Monitoring, Deploybarkeit und Code-Qualität sowie den aktuellen Stand der Work-in-Progress-Pakete auf einer Website oder einem für das ganze Team sichtbaren Monitor zeigen, geben dem Team mehr Transparenz. Diese ist wichtig, um bei Ereignissen wie Ausfällen oder Problemen bei der Deploybarkeit gegenzusteuern.

5. Kultur-Capabilities

Eine Kultur in Technologie-Organisationen erscheint auf den ersten Blick schwer messbar. Ron Westum fasst die meisten Kulturen mit seinem Paper *[4]* zusammen: pathological (machtorientiert), bürokratisch (regelorientiert) und generative (leistungsorientiert).

5.1 Produktive Unternehmenskultur schaffen und fördern

Die „generative"-Kultur unterstützt die Software Delivery Performance durch Vertrauen, ein hohes Maß an Kooperation, starken Informationsfluss, geteilte Risiken, Neugier sowie offene und ausgeprägte Kommunikation zwischen Teams *[5]*.

5.2 Lernen

Kontinuierliches Lernen ist kritisch für die stetige Veränderung des modernen Arbeitslebens. Als Arbeitgeber und Arbeitnehmer sollte das nicht als einmalige Investition verstanden werden, sondern als ein dauerhafter, beidseitiger Prozess.

5.3 Zusammenarbeit von Teams fördern

Produktive Unternehmenskulturen reduzieren das Silodenken zwischen Teams und fördern die Zusammenarbeit. Nicht nur bei Themen wie Entwicklung, Operations- und Information-Security, sondern zwischen dem klassischen Business und der Software-Delivery-Organisation.

5.4 Innovative Führungskultur etablieren

Innovative Führungskultur (Transformational Leadership) bedeutet, dass Führungskräfte durch Inspiration und Motivation Menschen zu Bestleistungen herausfordern. Diese Führungskräfte bringen eine Vision in die Organisation, kommunizieren klar, offen und inspirierend, fordern heraus, unterstützen und erkennen persönliche Leistungen an.

5.5 Tools und Ressourcen bereitstellen

Kritisch für die Zufriedenheit im Berufsalltag ist für viele, dass sie ihre Arbeit gut erledigen. Führungskräfte verstehen es, je nach Situation, das bereitzustellen, was benötigt wird. Sei es der Kontext, die Befähigung, etwas zu erledigen, neue Skills oder einfach nur Hard- oder Software.

Fazit: DevOps ist die Leistung einer kompletten Organisation

Erfolgreiche DevOps-Transformationen werden von der kompletten Organisation gelebt. Sie werden durch eine starke Übernahme von Verantwortung in den Teams gekennzeichnet und von offenen Retrospektiven begleitet. Damit wird es der Organisation ermöglicht, an ihrer Transformation zu wachsen. Wenige KPIs ermöglichen transparente Experimente, die Fehlschläge erlauben und Lernen fördern. Also: Gehen wir es richtig an!

Quellen

[1] *en.wikipedia.org/wiki/DevOps*

[2] Dr. Nicole Forsgren, Jez Humble, Gene Kim (2018): Accelerate: State of DevOps 2018: Strategies for a New Economy, DevOps Research and Assessment LLC, *https://cloudplatformonline.com/rs/248-TPC-286/images/DORA-State%20of%20DevOps.pdf*

[3] Dr. Nicole Forsgren, Jez Humble, Gene Kim (2018): Accelerate: Building and Scaling High Performing Technology Organizations, IT Revolution Press

[4] Ron Westrum (2014): A typology of organisational cultures, *https://qualitysafety.bmj.com/content/13/suppl_2/ii22*

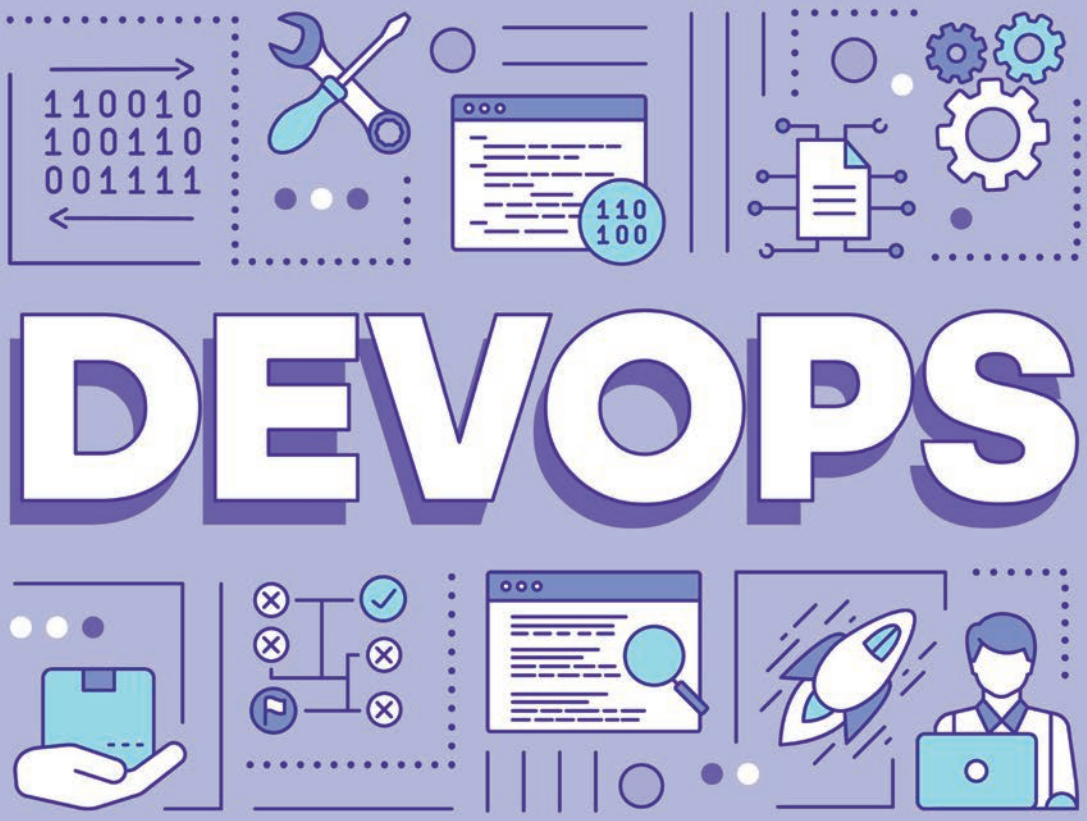
[5] continuousdelivery.com / Culture: *continuousdelivery.com/implementing/culture/*

Maximilian Braun

Virtual7

maximilian.braun@virtual7.de

Maximilian Braun ist Senior Technical Consultant bei virtual7 in Karlsruhe. Er berät internationale Konzerne und Ministerien, Behörden und die öffentliche Hand Deutschlands bei DevOps-Transformationen. Als Software-Entwickler und -Architekt schult er Entwickler und Anwender, gestaltet digitale Produkte und die digitale Zukunft Deutschlands.



Mit Jenkins in Richtung DevOps

Moritz Reinwald, MT AG

Jenkins ist der weltweit am meisten genutzte Open-Source-Automatisierungsserver mit unzähligen Plug-ins für eine Vielzahl von Technologien und Einsatzszenarien rund um Continuous Integration und Continuous Delivery [1]. Dieser Artikel stellt verschiedene Möglichkeiten und Werkzeuge vor, mit denen der Jenkins-Server dazu genutzt werden kann, einen weiteren Schritt in Richtung DevOps zu gehen.

Jenkins Build Jobs

Ist ein Jenkins-Server eingerichtet und aktiv, können Aufgaben in sogenannten Jenkins Build Jobs definiert werden. Diese Jobs können einzeln über die Oberfläche konfiguriert und sowohl manuell als auch automatisiert, zum Beispiel zeitgesteuert oder bei Änderungen in der Versionsverwaltung, gestartet werden. Da die Jobs im Normalfall nach Aufgaben getrennt sind, pro Job also nur eine Aufgabe ausgeführt wird, werden pro Projekt mehrere Jobs benötigt. Über die Zeit kann so aus wenigen Jobs relativ schnell eine große und unübersichtliche Menge an Jobs heranwachsen, die sich nur schwer und mit viel Zeitaufwand pflegen lässt.

Soll nun eine Änderung in mehreren Jobs vorgenommen werden, wie beispielsweise das Versionierungssystem von Subversion auf Git umzustellen, muss jeder Jenkins- Job einzeln abgeändert werden. Da die Jobs von Jenkins intern als XML-Dateien gespeichert werden, ist es durchaus möglich, die Änderungen dort vorzunehmen, allerdings ist dies sehr kompliziert und oftmals fehlerbehaftet, da dort keinerlei Validierung der vorgenommenen Änderung stattfindet. Damit stellt die Änderung der XML-Dateien keinen sinnvollen Weg dar. Für die Bearbeitung von mehreren Jobs gibt es zudem Plug-ins aus der Jenkins-Community, die mehr oder weniger gut funktionieren. Das generelle Ziel sollte hier allerdings nicht sein, Werkzeuge zu finden, die einen bei der aufwendigen Änderungsmethode von Jenkins-Jobs unterstützen, sondern eine Methode zum ausfindig machen, die es ermöglicht, Jenkins-Jobs einfach zu erstellen und später auch zu pflegen.

Ein weiterer Punkt, der gegen eine Vielzahl von Jenkins-Jobs spricht, ist die Dezentralisierung. Verteilt sich ein Prozess auf verschiedene Jobs, können diese zwar verkettet und somit automatisch hintereinander aufgerufen werden, allerdings kann dies auch Risiken mit sich bringen. Ein Nutzer kann eine Verkettung nicht direkt sehen, da er dazu in die Konfiguration der einzelnen Jobs schauen müsste. Führt dieser Nutzer zum falschen Zeitpunkt unwissentlich eine ganze Reihe von Jobs anstelle eines einzelnen Jobs aus, kann es schnell zu Problemen kommen. Zudem ist es schwierig, ein Reporting über mehrere Jobs zu realisieren, um den Fortschritt und die jeweiligen Ergebnisse aller Jobs auf einen Blick sehen zu können. Hier liegen die einzelnen Informationen alle verteilt in jedem einzelnen Jenkins-Job.

Jenkins stammt ursprünglich aus einer Zeit, in der Programme ausschließlich über die Benutzeroberfläche gesteuert wurden. Damals boten viele Programme keinerlei Optionen, erstellte Neuerungen als Code zu speichern, um diesen dann zu versionieren. Auch die Jenkins Build Jobs verlassen sich daher zu sehr auf die Benutzeroberfläche, d.h. sie sind dazu designt, über die Oberfläche erstellt zu werden. Damit Jenkins Build Jobs versioniert werden können, müssen viele verschiedene XML-Dateien in das Versionierungssystem eingespielt werden. Zusätzlich zur Versionierung spielt hier auch die Dokumentation eine Rolle. Da die Jenkins Build Jobs über eine Oberfläche erstellt werden, lassen sich alle Änderungen und Konfigurationen nicht einfach dokumentieren, sondern es müssen alle in der UI vorgenommenen Änderungen detailgetreu aufgeschrieben oder Screenshots angefügt werden. Hier zeigt sich auch das Problem der Reproduzierbarkeit. Wird ein kleines Detail vergessen, kann es vorkommen, dass zwei fast identische Jenkins Build Jobs komplett unterschiedliche Ergebnisse erzielen.

Zusammengefasst haben Jenkins Build Jobs mit umfangreicheren Projekten einige Probleme, sei es die reine Anzahl der einzelnen Jobs, die Unübersichtlichkeit und Dezentralisierung von Informationen, ungewollte Risiken oder die nicht triviale Versionierungsmöglichkeit. Eben diese Hürden gilt es zu überwinden, um weiter in Richtung DevOps zu gehen, und genau dafür gibt es seit Frühjahr 2016 eine offizielle Lösung mit Continuous Delivery Pipelines vom Jenkins-Entwicklerteam rund um Kohsuke Kawaguchi.

DevOps und CI/CD

Vielleicht gehen wir vorher aber nochmal einen Schritt zurück, um das große Ganze zu betrachten und zu verstehen. Wo soll es überhaupt hingehen? Was ist DevOps? Und wie helfen Continuous Delivery Pipelines dabei, mehr in Richtung DevOps zu kommen?

Ganz grob gesagt ist DevOps die Kombination aus Development und Operations. Wird versucht, DevOps genauer zu definieren, könnte es in etwa folgendermaßen klingen: „DevOps bezeichnet eine Reihe von Praktiken zur Automatisierung der Prozesse zwischen Softwareentwicklern und IT-Teams, durch die Software schneller und zuverlässiger entwickelt, getestet, freigegeben *[3]*, betrieben und gewartet werden kann". Eine genaue Definition ist allerdings schwierig, da DevOps kein konkretes Werkzeug beziehungsweise keine konkrete Methode, sondern eher eine Unternehmenskultur ist. Zusammengefasst kann gesagt werden: DevOps vereint alle notwendigen Schritte, sowohl die des Entwicklungszyklus:

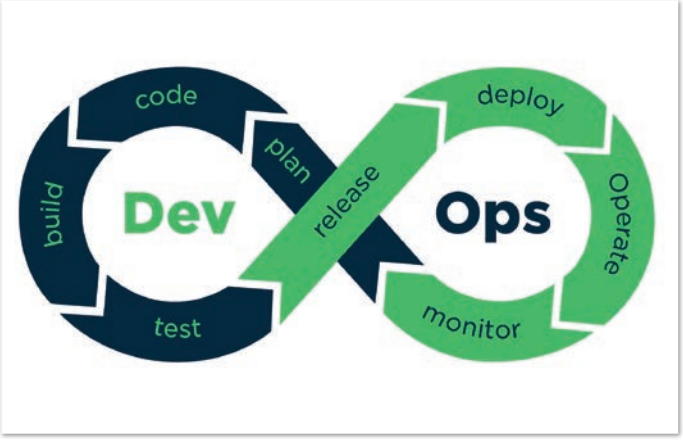


Abbildung 1: Der DevOps-Zyklus (Quelle siehe [2])

- planen
- entwickeln (Code schreiben)
- bauen
- testen

Als auch die des Betriebszyklus:

- fertiges Release erstellen
- in der Kundenumgebung einspielen
- betreiben
- überwachen

Aus diesen beiden, ohne DevOps getrennten, Kreisläufen bildet sich somit ein einzelner Zyklus, der, solange das Produkt lebt, unendlich weiterläuft *(siehe Abbildung 1)*

Continuous Delivery ist, wie auch DevOps, eines der Schlagworte, die überall genannt werden, aber oftmals gar nicht richtig interpretiert werden können. Was ist zum Beispiel der Unterschied zwischen Continuous Integration, Continuous Delivery und Continuous Deployment? Und wie passt DevOps dazu? Die Unterschiede zeigen sich deutlich, wenn die einzelnen Schritte des Software-Lebenszyklus betrachtet werden *(siehe Abbildung 2)*.

Nun ist auch ersichtlich, dass Continuous Delivery kein Synonym von Continuous Deployment ist, auch wenn beide häufig mit CD abgekürzt werden. Ein Release in eine Staging-Area, also Testumgebung, einzuspielen und danach manuell auszuliefern oder ein Release direkt in das Produktivsystem des Kunden einzuspielen, ist ein großer Unterschied.

Jenkins Pipelines

Nachdem nun definiert ist, wohin es gehen soll, müssen jetzt noch die technischen Mittel gefunden werden, um diesen Weg zu gehen. Seit 2016 gibt es vom offiziellen Jenkins-Entwicklerteam das Pipeline-Plug-in *[4]* oder besser die Pipeline Plug-in Suite, da das Pipeline-Feature auf mehrere Plug-ins aufgeteilt ist. Diese Plug-ins erweitern den Jenkins-Server um die Funktionalität, Pipelines auszuführen. Pipelines sind Automatisierungsketten, die im Falle einer Continuous Delivery Pipeline beispielsweise alle Schritte zum Release einer Software abarbeiten. Nach einer einfachen Installation der Plug-ins auf dem Jenkins-Server kann bereits mit den Pipelines gearbeitet werden, in den meisten Fällen sogar ohne einen Neustart.

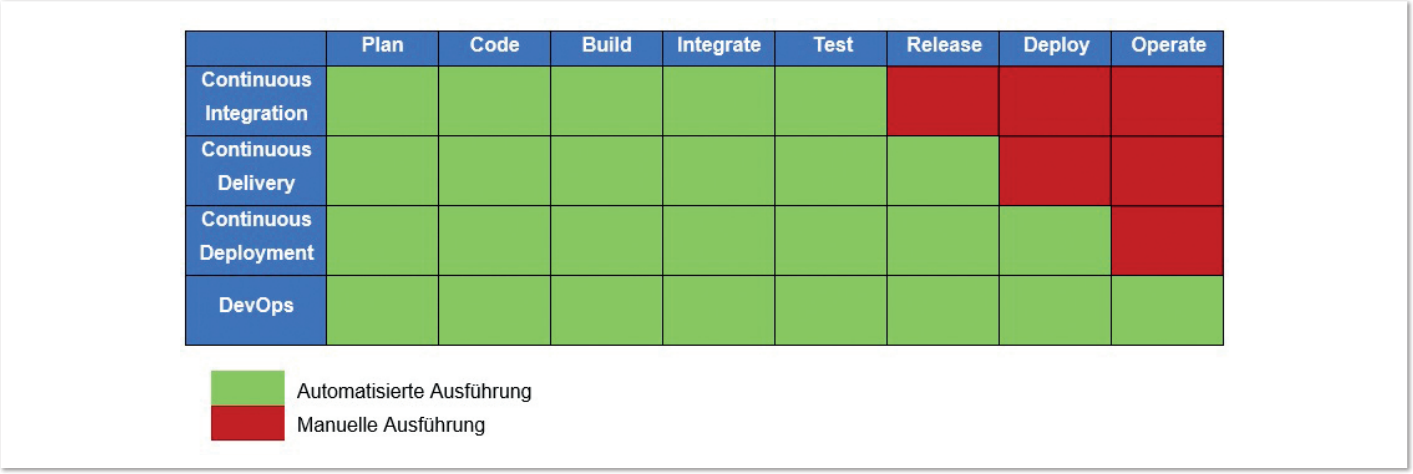


Abbildung 2: Abdeckung des Software-Lebenszyklus durch Continuous Integration, Continuous Delivery, Continuous Deployment und DevOps (Quelle: Moritz Reinwald)

Im Gegensatz zu den Build Jobs wird bei einem Pipeline-Job die gesamte Logik in einem Skript, dem sogenannten Jenkinsfile, hinterlegt. Dieser Jenkinsfile sollte auch in das Versionierungssystem eingepflegt werden. Aktuell gibt es zwei Versionen von Jenkins Pipelines. Die erste, initiale Variante heißt Scripted Pipeline. Hierbei wird der Jenkinsfile, auf dem die Pipeline aufbaut, noch größtenteils in Groovy, einer auf der Java-Plattform basierenden Programmier- und Skriptsprache, geschrieben. Zusätzlich gibt es die zweite und neuere Möglichkeit, die Pipelines deklarativ zu erstellen. Die sogenannten Declarative Pipelines basieren auf einer DSL, Domain-Specific-Language, die auf Groovy aufbaut. Wird eine Pipeline geschrieben, sollte der deklarative Ansatz gewählt werden, da die Scripted Pipeline in die deklarative Variante miteingebunden werden kann und somit obsolet ist.

Da der Jenkins-Server die Pipelines als weiteren Job-Typ definiert, können Pipelines auf die gleiche Weise wie die normalen Build Jobs angelegt werden. Nach einem Klick auf die Schaltfläche „Element anlegen" kann nun als Job-Typ „Pipeline Job" ausgewählt werden *(siehe Abbildung 3)*. Die Oberfläche zur Konfiguration eines Pipeline-Jobs sieht hierbei ähnlich aus wie die der Build Jobs. Hier sollte allerdings nicht der Fehler gemacht werden, dass zu viel in der Oberfläche konfiguriert wird. Diese Konfigurationsoptionen stehen ebenfalls im Jenkinsfile zur Verfügung. Wenn mehrere Pipelines auf einem Jenkinsfile aufbauen, muss hier nicht jeder Pipeline-Job einzeln konfiguriert werden, sondern jeder Job zieht sich die notwendige Konfiguration aus dem Jenkinsfile. Einzig die Verbindung zum bevorzugten Versionierungssystem muss in jedem Pipeline-Job in der Oberfläche gesetzt werden.

Da ein Jenkinsfile genutzt wird, findet sich im Pipeline-Job selbst keinerlei Konfiguration wieder. Theoretisch besteht auch die Möglichkeit, die Konfiguration direkt in der Oberfläche des Pipeline-Jobs zu erstellen und die Pipeline nicht in einen Jenkinsfile zu schreiben. Dies widerspricht allerdings der Idee der Jenkins Pipelines, da diese Pipeline-Jobs dann genau wie normale Build Jobs konfiguriert sind und alle Vorteile verloren gehen. Daher sollte ganz unten in der Konfiguration unbedingt die Option, den Jenkinsfile aus der Versionierung zu laden, gewählt werden *(siehe Abbildung 4)*. Hier stehen alle unterstützten Versionierungssysteme von Jenkins zur Verfügung. Neben Git lassen sich weitere Systeme wie etwa Subversion und BitBucket über Plug-ins installieren.

Jenkinsfile

Der Jenkinsfile einer Declarative Pipeline besteht hauptsächlich aus Sektionen und Direktiven. Der Anfang jeder Pipeline ist der pipeline-Block. Alles, was in der Pipeline passiert, muss innerhalb dieses Blockes stehen. Sind selbst geschriebene Groovy-Funktionen vorhanden, können diese auch außerhalb des Blockes definiert werden. Sinnvoller ist es in diesem Fall aber, eine sogenannte Shared Library *[5]* zu nutzen. Dies ist ein Feature, über das die Standard-DSL-Sprache um eigene Funktionen erweitert werden kann. Die grobe Struktur innerhalb des pipeline-Blockes ist durch sogenannte stages definiert. In jeder stage können verschiedene Befehle innerhalb einer steps-Sektion ausgeführt werden. Zudem kann in jeder stage eine post-Sektion eingefügt werden. In dieser können verschiedene Befehle aufgrund des Ergebnisses der steps-Sektion ausgeführt werden. Diese post-Sektion kann ebenfalls am Ende der gesamten stages-Sektion definiert werden.

Als Erstes innerhalb des pipeline-Blockes muss mithilfe der agent-Sektion ein Exekutor, auf dem die Pipeline oder einzelne Stages aus-

Abbildung 3: Erstellung einer neuen Jenkins Pipeline (Quelle: Moritz Reinwald)

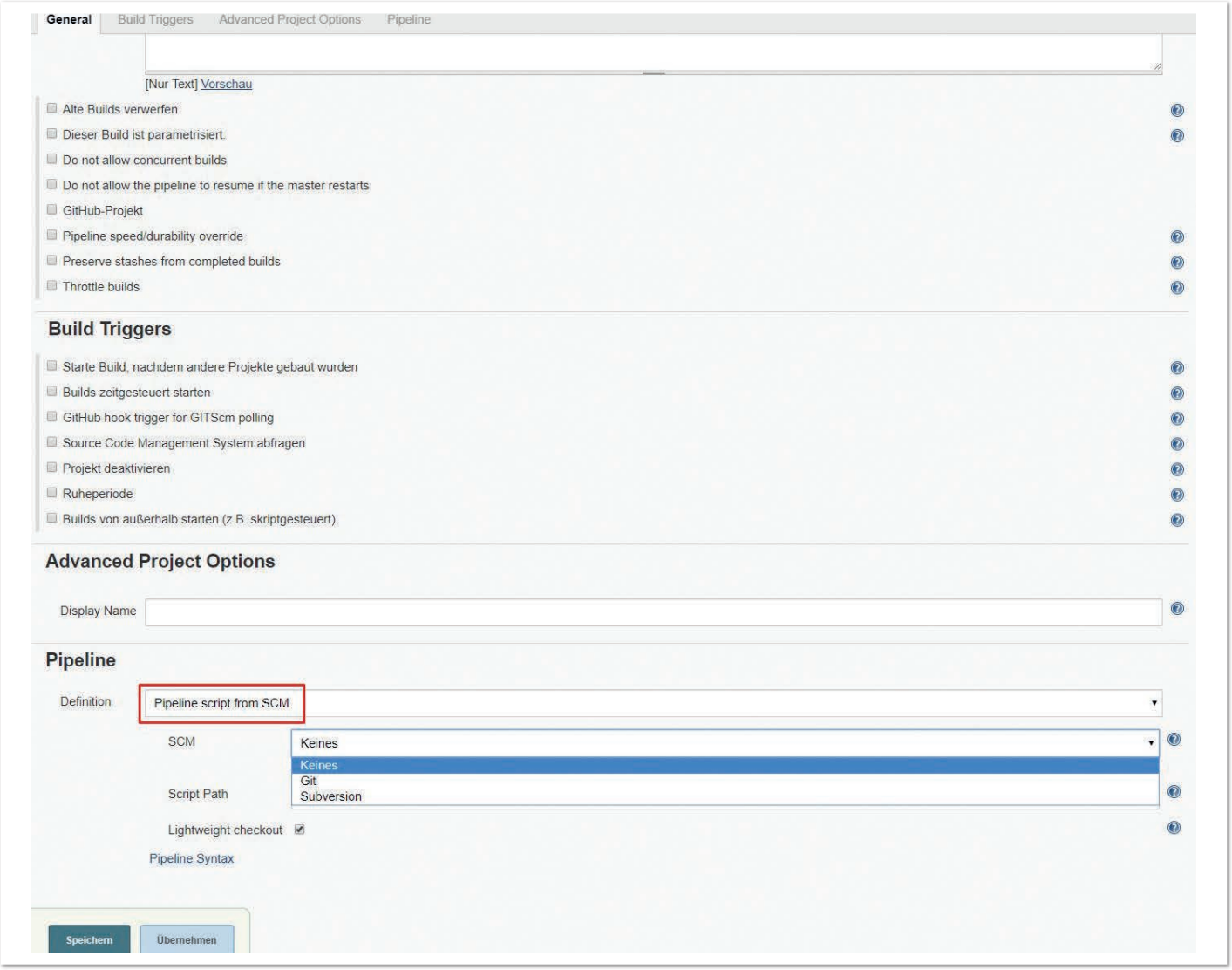


Abbildung 4: Beim Konfigurieren einer Jenkins Pipeline die Option „Pipeline script from SCM" wählen (Quelle: Moritz Reinwald)

Name	Benötigt	Verwendbar in	Beschreibung
pipeline	Ja	(1x)	Der Hauptblock
agent	Ja	pipeline, stage	Exekutor für Pipeline oder jede Stage
stages	Ja	pipeline (1x)	Hauptsektion für einzelne Stufen
stage	Ja mind. 1x	stages	Direktive für eine Stufe (Name muss angegeben werden)
steps	Ja	stage	Sektion für die einzelnen Befehle
post	Nein	pipeline, stage	Aktionen, je nach Ergebnis der Stage/Pipeline ausführen
environment	Nein	pipeline, stage	Definiert Umgebungsvariablen
options	Nein	pipeline (1x), stage (eingeschränkt)	Definiert Build-Optionen für die Pipeline (für eine Stage lassen sich nur bestimmte Optionen aktivieren)
parameters	Nein	pipeline (1x)	Definiert Parameter, die der User mitgeben soll
triggers	Nein	pipeline (1x)	Definiert automatisierte Starts der Pipeline
parallel	Nein	stage	Führt mehrere Stages innerhalb einer Stage parallel aus
script	Nein	steps	Führt Code einer Scripted Pipeline oder Groovy Code aus
when	Nein	stage	Bedingte Ausführung der Stage
catchError	Nein	steps	Fängt einen Fehler ab (ähnlich zu try/catch)

Tabelle 1: Auflistung einiger Sektionen und Direktiven

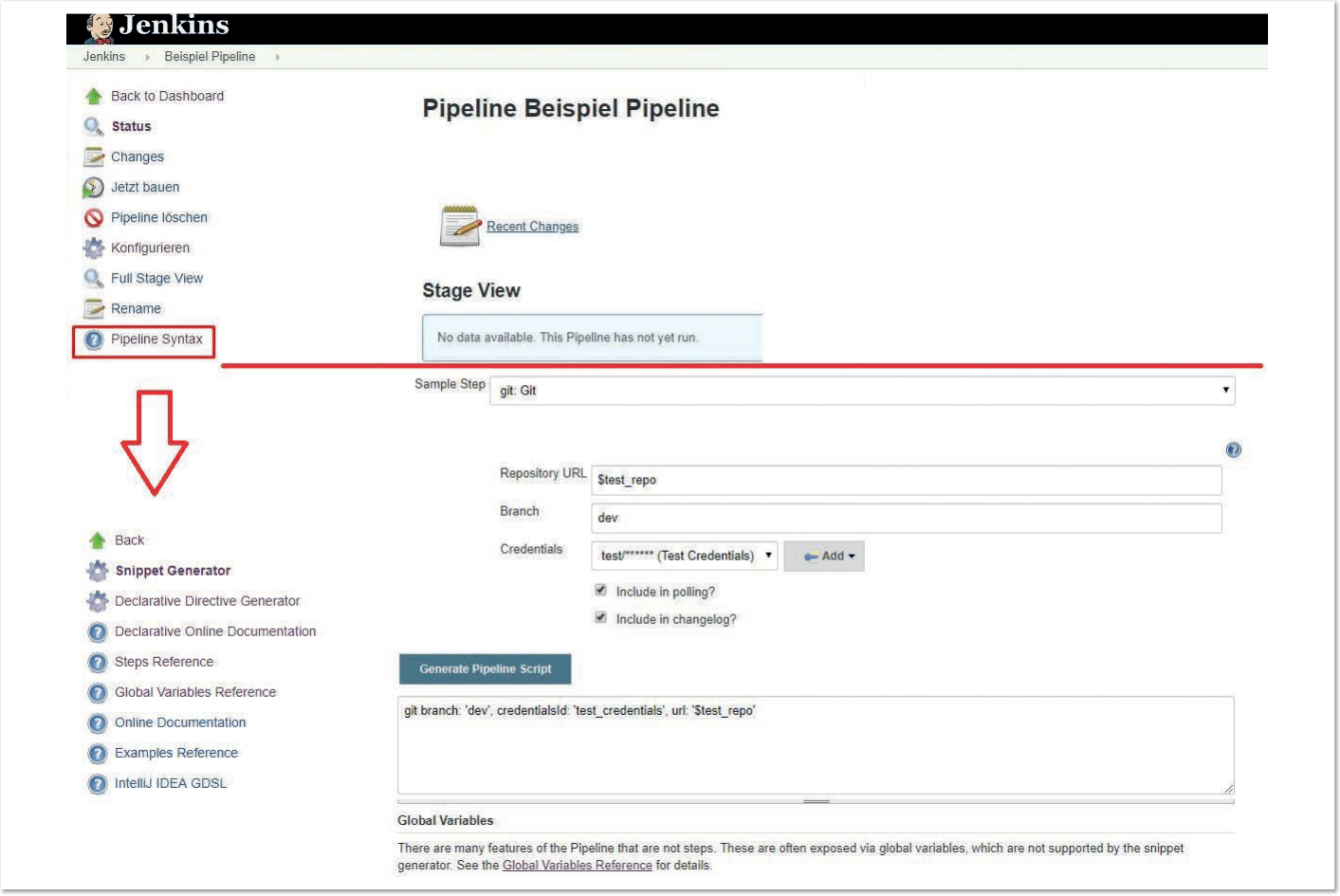


Abbildung 5: In Jenkins integrierter Snippet & Directive Generator (Quelle: Moritz Reinwald)

geführt werden, gewählt werden. Hier bietet Jenkins verschiedene Möglichkeiten. Mit agent any bestimmt Jenkins den Exekutor, mit agent none wird global kein agent festgelegt, dieser muss dann innerhalb der jeweiligen stages definiert werden. Soll ein bestimmter agent gewählt werden, kann dies über ein label geschehen *(siehe Listing 1)*. Weitere Möglichkeiten sind in der Jenkins-Pipeline-Dokumentation beschrieben *[6]*.

Nachdem der Exekutor definiert wurde, können nun durch weitere Direktiven, noch vor den Stages, einige Optionen, Parameter, Trigger und Umgebungsvariablen gesetzt werden. Nachfolgend sind alle häufig genutzten Sektionen und Direktiven aufgelistet, ausführlichere Informationen zu diesen und weiteren Befehlen finden sich ebenfalls in der Dokumentation der Jenkins Pipelines *[6]*.

Um die Pipeline nun mit Funktionalität zu versehen, müssen die gewünschten Befehle in die steps-Sektion geschrieben werden. Hierfür empfiehlt es sich, den durch die Pipeline-Plug-ins in den Jenkins integrierten Pipeline Snippet Generator zu nutzen. Dieser findet sich innerhalb eines Pipeline-Projektes im Menü auf der linken Bildschirmseite. Hier sind alle verfügbaren Befehle für eine Pipeline enthalten und können mithilfe der Oberfläche konfiguriert werden *(siehe Abbildung 5)*. Als Ergebnis liefert der Generator ein Snippet, das nahtlos in die Pipeline integriert werden kann. Für Plug-ins, die Jenkins Pipelines unterstützen, sind die Befehle ebenfalls im Snippet Generator verfügbar. Zusätzlich ist auch ein Direktiven-Generator für Declarative Pipelines und viele Referenzen, beispielsweise für einzelne Steps oder globale Variablen, vorhanden.

```
Pipeline {
    agent {
        label: 'linux'
    }
    …
    stages {
        …
    }
}
```

Listing 1: Einen bestimmten Agenten über ein Label selektieren

Zudem gibt es die Möglichkeit, den Jenkinsfile validieren zu lassen, bevor die Pipeline ausgeführt wird, indem der Jenkins Pipeline Linter genutzt wird. Dieser wird auf dem Jenkins-Server ausgeführt und der Jenkinsfile per HTTP-Post-Request übertragen. Je nach Konfiguration müssen hier noch Vorkehrungen getroffen werden, damit der Jenkins-Server die Anfrage erfolgreich erhält. Mehr dazu befindet sich in der Dokumentation *[7]*.

Besonderheiten und Best Practices

Müssen innerhalb der Pipeline, also dem Jenkinsfile, Passwörter genutzt werden, sollte unbedingt das Jenkins Credentials Plug-in *[8]* genutzt werden, das normalerweise standardmäßig mit dem Jenkins installiert wird. Wie bei den Build Jobs ermöglicht das Plug-in auch in den Pipelines die Nutzung von IDs, um im Jenkins hinterlegte Benutzerdaten zur Laufzeit abzurufen *(siehe Listing 2)*. Somit stehen keine Benutzerdaten im Klartext in Skripten oder im Jenkinsfile.

```
withCredentials([usernamePassword(
credentialsId: 'test_credentials',
passwordVariable: 'test_password',
usernameVariable: 'test_user')]) {
    bat '''cd Projektordner
    connect_db %apex-connection% %test_user% %test_password%
    @scripts\\myScript.sql'''
}
```

Listing 2: Aufruf des Credentials-Plug-ins aus dem Jenkinsfile

Sollen Informationen des aktuellen Builds des Pipeline-Jobs abgerufen werden, kann dies mithilfe der Eigenschaft currentBuild bewerkstelligt werden. Diese enthält Informationen über die ID bzw. Nummer, den Display-Namen, die Beschreibung und das aktuelle Ergebnis, auch zur Laufzeit. Damit ist es möglich, das Ergebnis eines Build-Schrittes oder auch der gesamten Pipeline zu prüfen und manuell das Ergebnis zu setzen, sollte der Jenkins-Server dies einmal nicht automatisch richtig erkennen. Ist das Ergebnis allerdings fehlerhaft (FAILURE), lässt es sich nur noch auf „instabil" (UNSTABLE) und nicht mehr auf „erfolgreich" (SUCCESS) ändern *(siehe Listing 3)*. Bricht ein Benutzer die Pipelineausführung ab, gibt es noch den Status ABORTED. Hier sollte also möglichst vor dem Setzen des Ergebnisses geprüft werden.

Soll ein Pipeline-Job nicht im Standard Jenkins Workspace laufen, kann ein sogenannter Custom Workspace angegeben werden *(siehe Listing 4)*. Hier sollte darauf geachtet werden, dass der Jenkinsfile zuerst aus der Versionierung ausgecheckt und dann erst ausgeführt

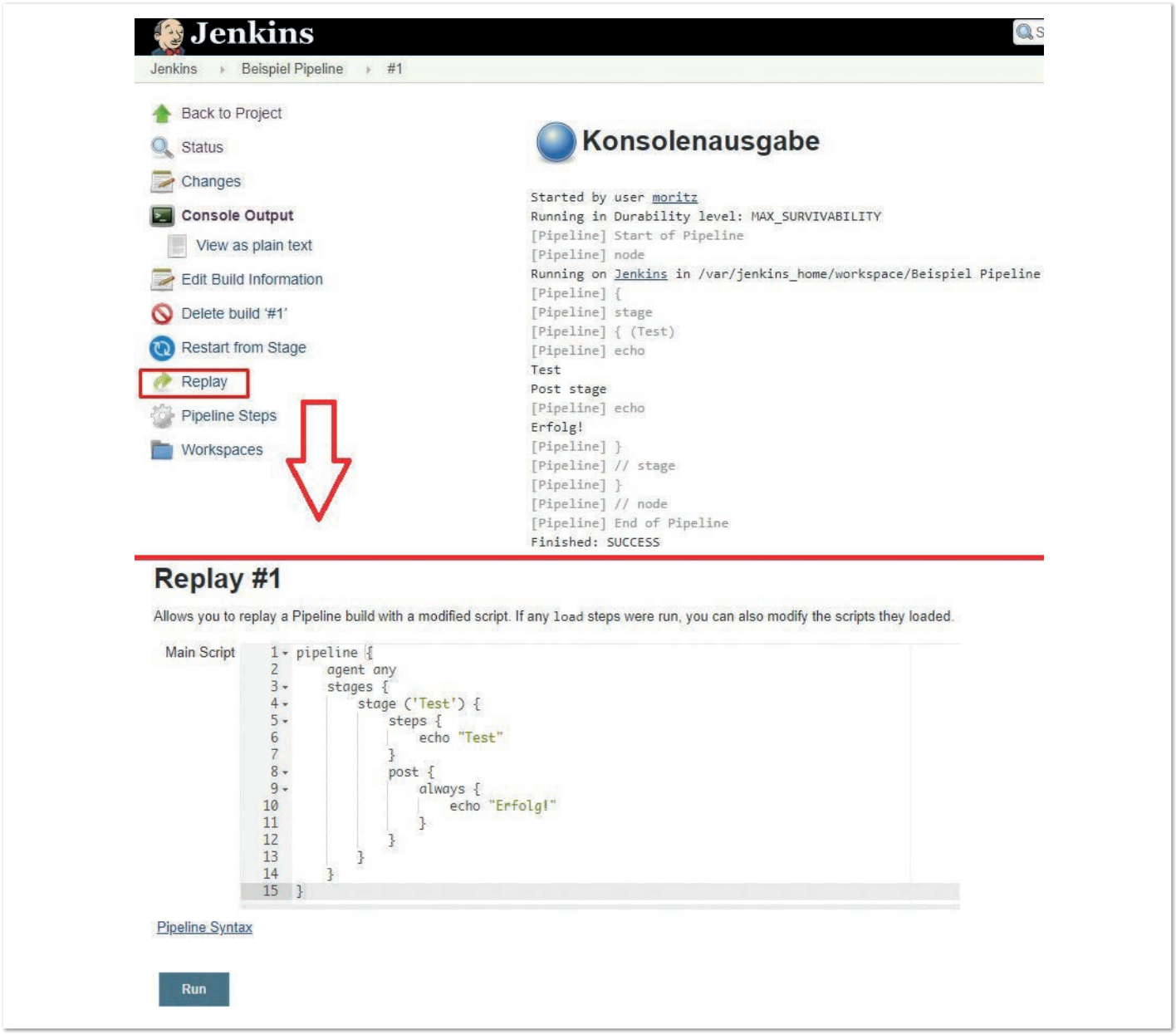


Abbildung 6: Replay-Funktion in der Oberfläche eines Builds (Quelle: Moritz Reinwald)

wird. Da der Custom Workspace allerdings erst im Jenkinsfile definiert wird, wird der Jenkinsfile selbst immer in den Standard Workspace ausgecheckt. Hier sollte wirklich nur der Jenkinsfile und nicht etwa das ganze Repository ausgecheckt werden, um Duplikate zu verhindern und nicht unnötig Speicherplatz zu verschwenden.

Funktioniert eine Pipeline nicht wie gewünscht, gibt es in der Oberfläche des Pipeline-Jobs eine Replay-Funktion. Mit dieser ist es möglich, den Jenkinsfile temporär in der Oberfläche zu ändern, um dann einen erneuten Durchlauf zu starten *(siehe Abbildung 6)*. Der Jenkinsfile im Hintergrund bleibt dabei unberührt. Somit ist es einfacher, einen Fehler innerhalb des Jenkinsfiles zu finden, ohne diesen jedes Mal committen zu müssen.

Oberfläche

Wurde ein Pipeline-Job angelegt und mindestens einmal ausgeführt, erscheint in der Oberfläche eine Übersicht über die Ausführungsdauer sowie das Ergebnis der einzelnen Stages. Zudem lassen sich die Logs der jeweiligen Stages mit einem Klick auf die Stage abrufen. Ist die Pipeline erfolgreich durchgelaufen, erscheinen alle Stages grün. Gab es einen Fehler in einer Stage, sind diese und alle nachfolgenden, nicht ausgeführten Stages dunkelrot eingefärbt. In diesem Fall färben sich auch die zuvor erfolgreich durchgelaufenen Stages hellrot. Oben in der Übersicht stehen die zuvor vergebenen Namen der Stages und eine Anzeige dazu, wie lange die Ausführungsdauer dieser Stage durchschnittlich war. Auf der linken Seite werden die verschiedenen Builds, also Ausführungen, mit der jeweiligen Build-Nummer, dem Start-Datum und der Anzahl an Commits in die Versionierung seit der letzten Ausführung aufgelistet *(siehe Abbildung 7)*.

Neben der Standardoberfläche gibt es per Plug-in *[9]* auch noch die extra komplett neu entwickelte Oberfläche namens Blue Ocean. Sie soll die User Experience von Jenkins deutlich steigern, indem sie folgende Hauptmerkmale mit sich bringt *[10]*:

```
if (result == "UNSTABLE") {
    currentBuild.result = "SUCCESS"
}
```

Listing 3: Abrufen und Ändern des Build-Ergebnisses zur Laufzeit

```
pipeline {
    agent {
        label {
            label ""
            customWorkspace "path/to/workspace"
        }
    }
}
```

Listing 4: Setzen eines Custom Workspace innerhalb des Jenkinsfiles

- Unterstützung von anspruchsvollen Visualisierungen von Continuous Delivery Pipelines
- Intuitives Erstellen von Pipelines in einem visuellen Prozess durch den Pipeline Editor
- Anpassungen an die rollenbasierten Anforderungen jedes Teammitglieds
- Präzision bei der Lokalisierung von Problemen innerhalb einer Pipeline
- Native Integration von Branches und Pull Requests für GitHub und BitBucket

Wird Blue Ocean genutzt, kann auf einen Blick gesehen werden, wo die Pipeline fehlerhaft war und welche Stages eventuell trotzdem danach noch ausgeführt wurden. Zudem sieht die Oberfläche deutlich „frischer" aus und ist von Grund auf für Pipelines entwickelt worden *(siehe Abbildung 8)*.

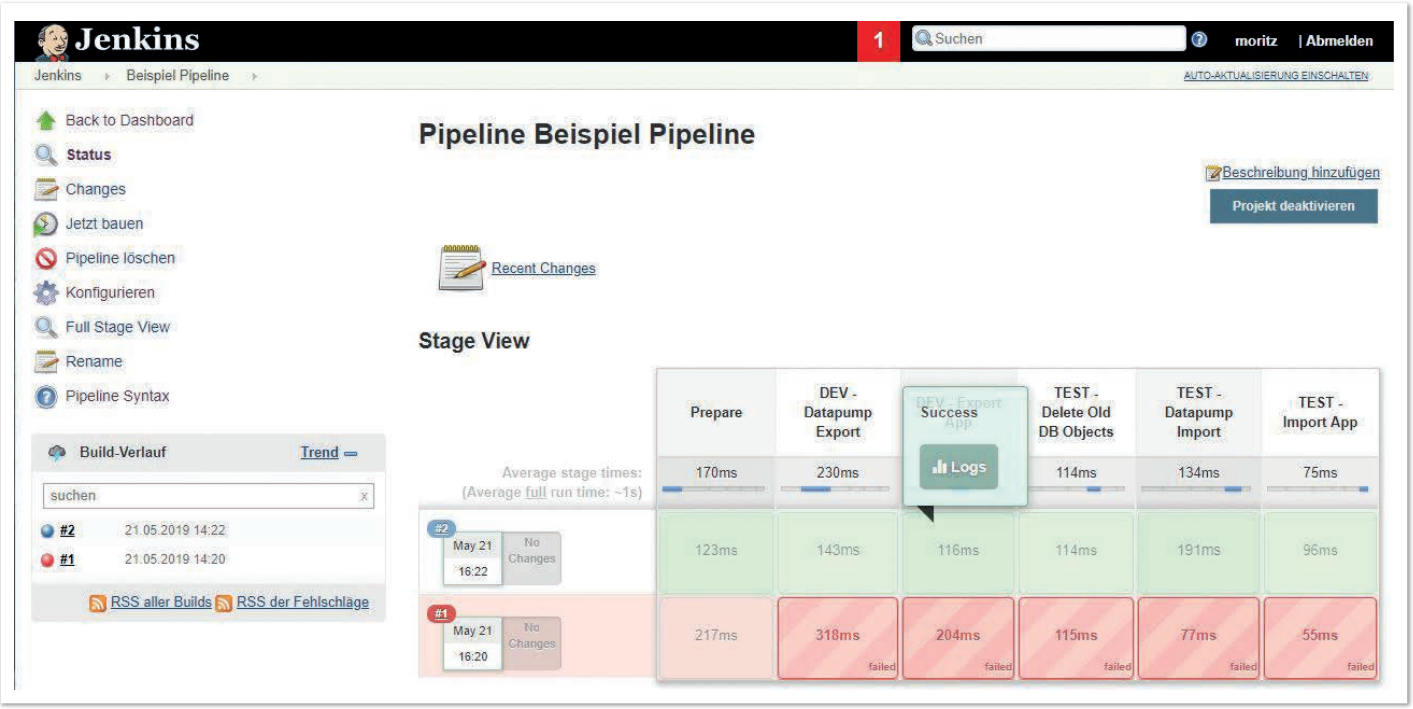


Abbildung 7: Standardoberfläche eines Jenkins-Pipeline-Jobs (Quelle: Moritz Reinwald)

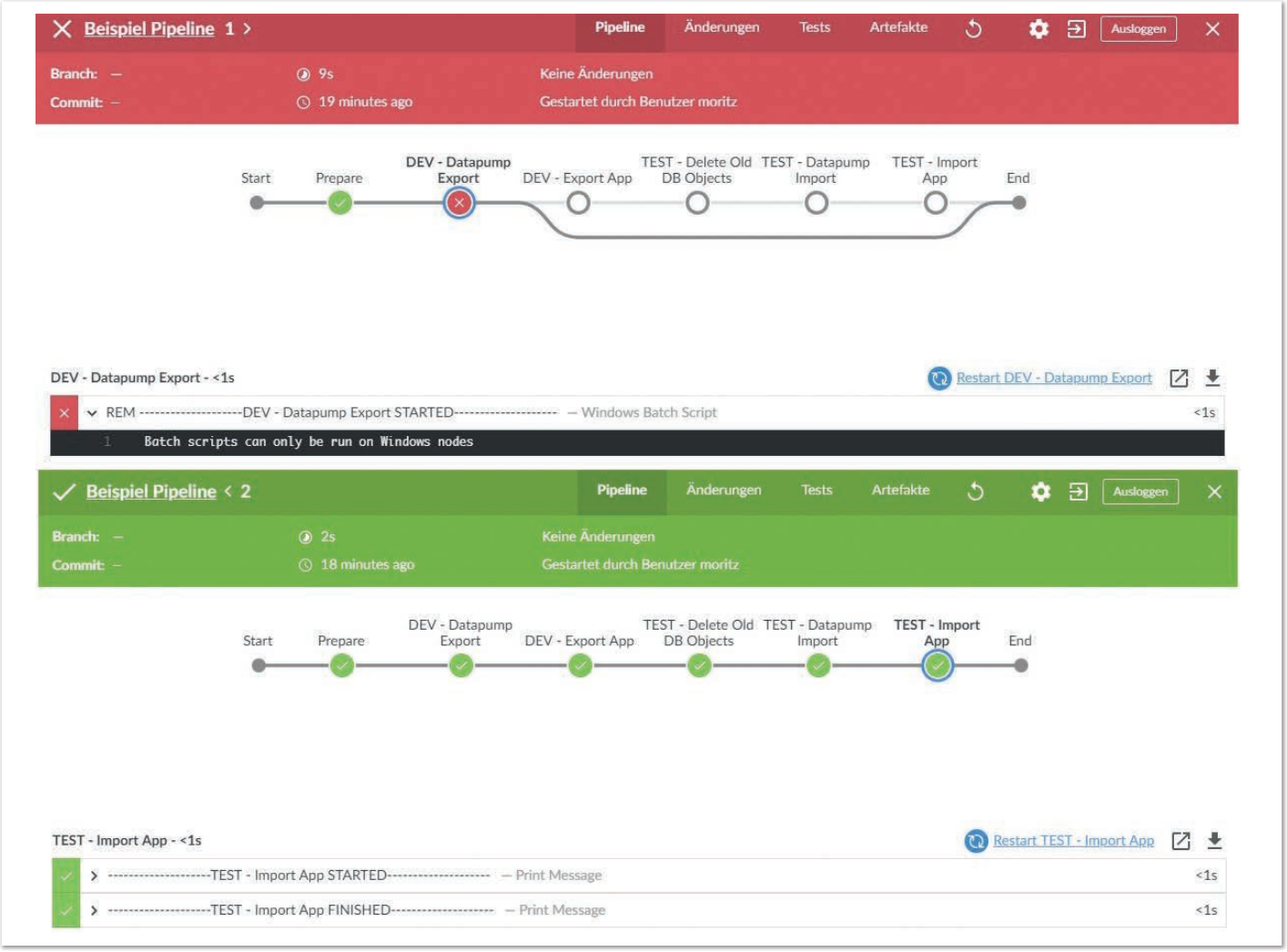


Abbildung 8: Blue-Ocean-Oberfläche für einen Fehlschlag und Erfolg des gleichen Jobs (Quelle: Moritz Reinwald)

Fazit

Jenkins bietet mit den Pipeline Plug-ins und der Blue-Ocean-Oberfläche eine gute Basis für Continuous Delivery Pipelines. Nach einer kurzen Eingewöhnung an den Aufbau des Jenkinsfiles geht die Erstellung von Pipelines, durch den deklarativen Ansatz, einfach von der Hand. Sollte die DSL einmal nicht ausreichen, existiert noch immer die Möglichkeit, eigene Groovy-Funktionen zu schreiben und einzubauen. Mit Blue Ocean bietet Jenkins auch an der Oberfläche viele Funktionen und eine gute Übersicht, somit lassen sich Pipelines gut verwalten und Probleme leichter beheben.

Quellen und weitere Informationen

[1] Jenkins Homepage: *https://jenkins.io/*

[2] DevOps-Zyklus: *https://medium.com/@neonrocket/devops-is-a-culture-not-a-role-be1bed149b0*

[3] Atlassian Definition DevOps: *https://de.atlassian.com/devops*

[4] Jenkins Pipeline Plug-in: *https://wiki.jenkins.io/display/JENKINS/Pipeline+Plugin*

[5] Jenkins Pipeline Shared Libraries: *https://jenkins.io/doc/book/pipeline/shared-libraries/*

[6] Jenkins Pipeline Syntax: *https://jenkins.io/doc/book/pipeline/syntax/*

[7] Jenkins Pipeline Linter: *https://jenkins.io/doc/book/pipeline/development/#linter*

[8] Jenkins Credentials Plug-in: *https://plugins.jenkins.io/credentials*

[9] Jenkins Blue Ocean Plug-in: *https://plugins.jenkins.io/blueocean*

[10] Blue-Ocean-Dokumentation: *https://jenkins.io/doc/book/blueocean/*

Moritz Reinwald
MT AG
moritz.reinwald@mt-ag.com

Moritz Reinwald arbeitet als Berater mit dem Schwerpunkt APEX Development und DevOps bei der MT AG in Ratingen. Neben der Erstellung von Webanwendungen mithilfe von APEX liegen seine Schwerpunkte im Bereich CI/CD mit Jenkins und Testautomatisierung. Zusätzlich beschäftigt er sich mit Technologien rund um das Thema DevOps, insbesondere in Verbindung mit Datenbanken.

"Spaltungen in der kunden-orientierten Software-Industrie sind kontraproduktiv"

Der Name Ed Burns ist dem Großteil der Java-Community durchaus bekannt. Bevor Ed im Juni 2019 zu seinem neuen Arbeitgeber Microsoft wechselte, war er viele Jahre beim Oracle Technical Staff tätig. Ob als Co-Spec Lead von JavaServer Faces, Buchautor oder als Speaker auf zahlreichen Konferenzen: Ein Hauptaugenmerk seiner Arbeit war stets Java. Im Interview mit Christian Luda, DOAG Dienstleistungen GmbH, berichtet Ed, wie sich seine tägliche Arbeit seit seinem Wechsel verändert hat und erinnert sich zurück an die Zeit der Framework-Kriege. Weiterhin berichtet er, was für Vorteile die Microsoft Azure Cloud Java-Usern bietet und was diese in Zukunft von seinem Team erwarten dürfen.

Du bist derzeit Principal Architect für Java Tooling und Experiences des Azure-Teams bei Microsoft. Kannst du uns einen kleinen Einblick in deine derzeitige Arbeit gewähren?

Ed Burns: Selbstverständlich! Gegenwärtig arbeite ich mit meinen alten Kollegen von Oracle daran, WebLogic-Server so einfach wie möglich auf Azure zu bringen. Der beste Weg dafür ist über eine Sammlung von ARM-Vorlagen. ARM steht für „Azure Resource Manager" und ist die zentrale Management-Ebene für Azure. Eine ARM-Vorlage ist eine unhandliche und oftmals große JSON-Datei. Als Mitglied des Developer-Experience-Teams möchte ich dazu beitragen, die Entwicklererfahrung zu vereinfachen. Denn die Erstellung von ARM-Vorlagen ist eine ziemliche Herausforderung. Wir haben Tooling-Unterstützung in Form von Erweiterungen für Visual Studio Code und Visual Studio, was die Erstellung etwas vereinfacht. Es gibt eine kleine Sammlung von Funktionen innerhalb der ARM-Vorlagen namens TLE (Template Language Expressions), über die sich einige Punkte anpassen lassen, aber insgesamt ist das Schreiben von ARM-Vorlagen, als würde man in Bezug auf die von Azure angebotenen Ressourcen sagen: „So sollen meine Azure-Server aussehen." Wir sprechen also über VMs, Netzwerke, Verbindungen zwischen den Lastverteilern, Datenbanken und so weiter.

Ich arbeite derzeit mit Jacob Thomas und seinem Team von Oracle, das wirklich einige hervorragende ARM-Vorlagen erstellt hat. Ich habe sie beraten und ihnen Hinweise gegeben, um eine Vielzahl an WebLogic-Serverkonfigurationen von einfacher VM mit Admin bis zum dynamischen Cluster und mehreren Stufen dazwischen zu ermöglichen. Das alles ist sehr spannend, weil Microsoft im Java-Bereich eine Zeit lang überhaupt nicht vertreten war. Wenn wir mit diesen Schritten fertig sind, werden wir uns darauf konzentrieren, WebSphere und Red Hat EAP auf die gleiche Weise auf Azure zu bringen. Meine Kollegin Theresa Nguyen arbeitet sehr eng mit Red Hat zusammen, um mehrere Services für Azure zu bauen. Sie haben eine Jakarta EE PaaS (Platform as a Service), die im Wesentlichen eine PaaS für WildFly bereitstellt, eine Open-Source-Laufzeitumgebung, die vollständig mit Jakarta EE 8 kompatibel ist. Außerdem arbeitet Nguyens Team an Azure Red Hat OpenShift. OpenShift ist in erster Linie eine Kubernetes-Implementierung; es lassen sich Java und MicroProfile und sogar Jakarta-EE-Workloads darauf verwenden, aber es bietet darüber hinaus noch ein viel breiteres Spektrum. Es passiert also derzeit sehr viel im Java-Bereich bei Microsoft, ich selbst arbeite jedoch im Moment an WebLogic und WebSphere.

Das klingt nach einer Zusammenarbeit von Oracle, Red Hat und Microsoft. Würdest du sagen, dass diese Art der Zusammenarbeit zwischen großen Unternehmen neu ist? Wäre so etwas vor fünf bis zehn Jahren möglich gewesen?

Ed Burns: Ich denke nicht, dass eine solche Zusammenarbeit vor fünf bis zehn Jahren möglich gewesen wäre. Ich glaube, der Grund dafür ist, dass es den Wettbewerb im Cloud-Bereich damals nicht gab. Möglich wird diese Zusammenarbeit heute, weil sich die Unternehmensziele der Cloud-Anbieter und Software-Hersteller angleichen. Im Fall von Oracle und seinen Mitarbeitern, die an Coherence und WebLogic arbeiten, ist das Ziel die Adaptierung und Fortführung ihres Stacks auf anderen Clouds. Das ist die Sicht der Software-Hersteller. Der Cloud-Anbieter wiederum möchte, dass Kunden ihre Workloads auf seine Cloud bringen. In unserem Fall ist das die Azure-Cloud. Wir möchten es den Leuten so einfach wie möglich machen. Alles dreht sich um Erreichbarkeit und deshalb gibt es diese Kooperation. Ausschlaggebend dafür war, dass die Industrie erkannt hat, dass das Thema Cloud die Zukunft ist.

In einem Kurzbericht hast du geschrieben „Der Cloud-Anbieter ist das Wichtigste". Vielleicht kannst du das präzisieren und erzählen, was einen guten Cloud-Anbieter ausmacht?

Ed Burns: In der Vergangenheit gab es viele Plattformkriege und ich erinnere mich gut an diese Zeit, weil ich in der Entwicklung von Java EE 6 und 7 einige von ihnen bestritten habe. Es gab eine große Vielzahl von Java-basierten, serverseitigen Web-Frameworks wie Struts, Wicket und Tapestry sowie JSF (an dem ich gearbeitet und bei der Entwicklung geholfen habe). Jedes dieser Frameworks kämpfte um Aufbau und Häufung von Marktrelevanz. Andererseits zogen diese Auseinandersetzungen auch Interesse auf die jeweiligen Technologien und es ist auch unterhaltsam, mit den Leuten darüber zu debattieren. Darüber hinaus zeigten diese Framework-Kriege, dass das Thema die Beachtung wert war. Unabhängig von der Wahl des Web-Frameworks bestätigte sich, dass man eines braucht, da einige zur Auswahl stehen und also irgendetwas an diesem Architekturansatz dran sein muss. Das Konzept der Framework-Kriege hörte nicht bei Java-Servern und Web-Frameworks auf. Es dehnte sich auf viele andere Dinge aus, beispielsweise Spring Dependency Injection versus CDI (Contexts and Dependency Injection) für Java EE oder Spring MVC versus Java EE MVC. Es gibt noch etliche andere Beispiele. Unser Fokus als Cloud-Anbieter ist, beide Varianten einzubringen.

Microsoft ist ein sehr kundenorientiertes Unternehmen. Betrachten wir also Marktdurchdringung und Marktmacht, dann ist es ein nachweisbarer Fakt, dass Spring Cloud Service eine Vielzahl von Adaptierungen hat. Die Anzahl ist höher als die anderer Microservices wie Helidon, Quarkus oder Micronaut. So großartig und spannend diese auch sein mögen, Spring Cloud hat einen deutlichen Vorsprung. Daher haben wir uns vorrangig auf den Aufbau des „Azure Spring Cloud Service" konzentriert. Dieser Service, eine vollständige Platform as a Service, ergänzt unsere Tätigkeiten im Bereich der Jakarta-EE-ARM-Vorlagen, die eher eine Lift-and-Shift-Lösung sind. Uns ist aufgefallen, dass Unternehmen mit fortgeschrittener Cloud-Integration bereit sind für eine vollständige PaaS, wohingegen die Unternehmen, die erst am Anfang stehen, eine Lift-and-Shift-Lösung bevorzugen, da sie damit vertrauter sind.

Da du die alten Kriege erwähnst – vermisst du diese Zeit? Erfordert das heutige Geschäft eine andere Denkweise?

Ed Burns: Ein bisschen. Es war auch etwas komisch, denn damals musste man in der Lage sein, sich menschlich von seiner Arbeit abgrenzen zu können, denn es gab leider auch Unprofessionalität in diesen Kriegen. Ich erinnere mich sehr gut, dass es zu dieser Zeit zwei unterschiedliche Domänennamen gab, die das Niedermachen von JSF zum Ziel hatten. Das waren Sammlungen von Geschichten, in denen Leute sich über JSF beschwerten. Versteht mich nicht falsch – ich weiß, dass JSF viele Probleme hatte. Allerdings ist es immer noch recht verbreitet und für einige Arten von Anwendungen immer noch sehr gut geeignet für hohe Produktivität. Es war in dem,

was es macht, sehr erfolgreich. Aber man muss das Gute und das Schlechte beleuchten. Wenn man bei diesen Framework-Kämpfen mitmacht, muss man akzeptieren können, dass es nicht jedem gefällt – und das ist in Ordnung, denn so ist es nun einmal.

Heutzutage ist alles etwas reifer und das ist angenehm. In der Software-Industrie haben wir gelernt, dass derartige Spaltungen kontraproduktiv sind. Bei einer kundenorientierten Kultur wie bei Microsoft, wo wir vor allem Kundennutzen liefern wollen, ist das sicherlich der Fall. Daher lassen wir diese Kriege. Beispielsweise startete Azure selbst als Windows Cloud, mittlerweile laufen jedoch mehr als die Hälfte aller CPU-Zyklen und Workloads auf Azure unter Linux. Das ist so, weil wir gemerkt haben, wo die Container sind – also wo Kubernetes ist –, und das ist, wo die Kunden hinwollen. Bei der Cloud geht es vor allem um Standardware – zur Standardware gemachte Hardware und Netzwerke. Alles ist messbar, verrechenbar und prüffähig. Das einzige Betriebssystem, das dafür gut funktioniert, da es kostenlos und Open-Source ist, ist Linux. Alle anderen Betriebssysteme hatten irgendein Lizenzierungsmodell und konnten aus Unternehmenssicht nicht gut in der Cloud skalieren – aus technischer Sicht ja, aber nicht aus Unternehmenssicht. SUN und dann Oracle Solaris hatten viele Innovationen. Einige davon hat Linux bis heute nicht, einige wurden an Linux weitergegeben, wie ZFS, das sehr wichtig für groß angelegten Dateisystem-Netzwerkzugriff vom Betriebssystem ist.

Selbstverständlich gibt es heute in der Cloud unterschiedliche Möglichkeiten bei der Speicherung für das Mapping vom Dateisystem. Man verwendet nicht ZFS zum Skalieren. Jeder Cloud-Anbieter hat seine eigene proprietäre, hoch-performante und höchsteffiziente Methode zur Speicherung. Ich möchte darauf hinaus, dass die proprietären Betriebssysteme nicht mit Linux in der Cloud konkurrieren konnten.

Wie wichtig ist Java im Augenblick für Microsoft?

Ed Burns: Java ist sehr wichtig für Microsoft. Wir nennen es eines der „großen Vier" für die Cloud. Java, Node.js, Python und .NET – das sind unsere Schwerpunkte. Wenn ich „Node.js" sage, dann umfasst das auch TypeScript. Wir investieren auch in die Unterstützung von PHP und Go. Visual Studio Code, das von dem Team stammt, mit dem ich arbeite, bietet Spracherweiterungen für alle diese Technologien. Wir haben verstanden, dass eine Vielzahl an heutiger Unternehmenssoftware bei Banken, Ölfirmen, Pharmaunternehmen und Versicherungen in Java geschrieben wird, häufig in Jakarta EE. Unternehmen versuchen, vom Betrieb eigener Rechenzentren wegzukommen, es muss allerdings wohlüberlegt sein. Viele Unternehmen sind noch nicht bereit, in die Cloud zu gehen, also müssen wir ihnen etwas zur einfachen Migration anbieten oder sogar Teile der Cloud auf ihrer eigenen Website laufen lassen. Wir haben eine Technologie namens „Azure Stack", die auch manchmal Hybrid-Cloud genannt wird. Damit kann die Cloud im eigenen Rechenzentrum betrieben werden. Wir geben ihnen Hardware, die sie in ihrem eigenen Rechenzentrum behalten können, aber die Verwaltung läuft wie alles andere bei Azure. Für Microsoft und Java ist das somit der Punkt, an dem die Kunden sind. Wenn wir also weiterhin Wachstum bei Azure möchten, was der größte Erfolgsfaktor für Microsoft im Hinblick auf Umsatzwachstum war, dann müssen wir genau dorthin schauen.

Kannst du uns beschreiben, was Azure für Java-Benutzer zu bieten hat? Warum sollten sie Azure anstatt anderer Optionen wählen?

Ed Burns: Eine sehr gute Frage! Microsoft hat eine Partnerschaft mit Azul Systems, einem JDK-Anbieter. Wenn Sie Java-basierte Workloads auf Azure laufen lassen, dann läuft das häufig auf dem Azul Zulu JDK, einer vollständig konformen Java-Laufzeitumgebung. Es gibt Hunderte an Services, die man auf Azure laufen lassen kann, alle beliebten Technologien: Redis, Kafka, Hadoop, Cassandra usw. Sie können auf Azure gehen und sagen: „Gib mir eins hiervon und eins davon und hilf mir, sie aneinanderzureihen". Für Services mit einer Java-Anbindung ist das eine Art, um auf Java-basierte Services in der Cloud zuzugreifen. Wir haben auch einige spezifischere Optionen. Es gibt einen Java-App-Service für Linux, der im Wesentlichen ein Tomcat as a Service ist, und dann sind da natürlich noch die bereits erwähnten WildFly- und Spring-Cloud-Services. Wir haben auch einige Azure-Services wie Cosmos DB, eine sehr skalierbare Datenbank-Engine. Wir haben Azure SQL Server, der ebenfalls Java-Zugriff und -Anbindung bietet. Das ist auf der Laufzeitseite. Auf der Tool-Seite haben wir Visual Studio Code, einen sehr erfolgreichen und beliebten Code Editor mit intelligenten Funktionen. Ich bin sehr vorsichtig, es nicht IDE zu nennen, denn eine IDE ist für einen Java-Entwickler sehr spezifisch. Die größte IDE für Java-Entwickler ist natürlich IntelliJ von JetBrains mit einem sehr speziellen Satz an Nutzungsmustern, die sich etwas von dem unterscheiden, was wir bei VS Code für Java umgesetzt haben. Wir klassifizieren VS Code für Java eher als einen intelligenten, leichtgewichtigen Editor als eine Full-Stack IDE. Er bietet allerdings die für die Nutzer beliebtesten Funktionen von IDEs wie Debugging, Syntaxhervorhebung und Refaktorierung. Es macht das, was man braucht, aber es hat nicht diesen schweren, komplexen Rucksack dabei. Beispielsweise gibt es bei VS Code nicht so sehr dieses Projektkonzept. Es ist eher so, dass man auf den Code zeigt und das Tool es dann verstehen wird.

Auf Tool-Seite haben wir einiges an Support für Java in Form von VS Code, das plattformunabhängig und vollständig polyglott ist. Es gibt Erweiterungen in VS Code für Go, aber auch großartige Erweiterungen für Node.js und viele weitere für alle möglichen Technologien. Die Cloud hat wirklich für eine breite Öffnung für mehrsprachige Programmierung gesorgt. Neuere Entwickler, nicht die alten graubärtigen Java-Entwickler wie ich, müssen auf verschiedenen Stacks arbeiten, um ihre Arbeit zu bewerkstelligen. Das ist eine dieser Spannungen auf dem Arbeitsmarkt, da Unternehmen gerne Full-Stack-Entwickler hätten, die sich mit Node, PHP und Python auskennen – was einfach zu viel verlangt ist für eine Person. Man braucht jedoch zumindest ein Tool, das alle diese Themen beherrscht. Vielleicht ist es ja möglich, jemanden zu bekommen, der ausreichend Arbeit in diesen unterschiedlichen Stacks erledigt bekommt, und dann hätte diese Person sicherlich gerne ein Tool wie VS Code.

Oracle hat inzwischen seine Lizenz- und Supportbedingungen für Java geändert. Denkst du, dass das eine gute Entscheidung war?

Ed Burns: Ich war zu dem Zeitpunkt noch bei Oracle und es war für uns als Oracle damals die richtige Entscheidung. Java auf all diesen unterschiedlichen Plattformen zu unterstützen, kostet Ressourcen und Oracle ist eine Firma, die Unternehmen versteht. Unternehmen und ihre Kostenstrukturen sind darauf ausgelegt, Lizenzen für die von ihnen verwendete Software einzupreisen. Für

Zur Person

Ed Burns ist derzeit Principal Architect im Java Tooling und Experiences Team bei Microsoft. In dieser Rolle möchte Ed dazu beitragen, Azure zum besten Ort für Enterprise Java zu machen. Bevor er zu Microsoft kam, war Ed beratendes Mitglied des Technical Staff bei Oracle. Dort war er Co-Spec Lead für JavaServer Faces. Diese Rolle übernahm er von seiner Position bei Sun Microsystems, Inc. Seine beruflichen Interessengebiete sind Web Application Frameworks, AJAX, Komplexitätsreduzierung, testgetriebene Entwicklung, Anforderungserfassung und computergestützte Zusammenarbeit. Ed ist ein erfahrener internationaler Konferenz-Speaker und Buchautor. Weitere Informationen sind in seinem Blog zu finden: *http://ridingthecrest.com/author.html*

Unternehmen, die auf Java spezialisiert sind und von denen Oracle den Support bezahlt haben möchte, war das bereits gang und gäbe. In vielen Fällen hat Oracle nur die Preise vergleichbar, transparent und nachvollziehbar gestaltet. Während sich das abspielte, haben sich einige Leute beschwert, aber Oracle entgegnete ihnen, dass sie OpenJDK verwenden können, wenn sie das möchten. Es war im Endeffekt nur die Bekanntmachung einer gut durchdachten Strategie. Das Support-Thema spielte auch bei der Anforderung an einen anderen Release-Zyklus eine Rolle, mit häufigeren Releases beim OpenJDK und selteneren Releases im Langzeit-Support-Bereich beim Oracle JDK. Der technische Aspekt, der das alles möglich gemacht hat, ist das Modulsystem von Java 9. Es gibt so viele Teile bei dieser Strategie – Business, Technologie und Rechtsbereich –, dass sie wirklich schön anzusehen ist. Es hat wirklich etliche Jahre gedauert, bis wir mit Jigsaw fertig wurden, und nun, da wir es haben, können wir die Themen unabhängig voneinander entwickeln, unterschiedliche Release-Zyklen haben und ein passendes Support-Modell bereitstellen.

Wie hat sich deine tägliche Arbeit geändert, seitdem du bei Microsoft bist? Was können wir von deinem Team für die Zukunft erwarten?

Ed Burns: Der größte Unterschied hier ist die Kundenorientierung. Microsoft entwickelt nichts mehr, das Kunden nicht benutzen. In der Vergangenheit gab es oft Zeiten, in denen Microsoft erst irgendetwas entwickelt und es dann zur Verwendung in die Welt getragen hat. Das stellte sich nicht gerade als die beste Verwendung für die Ressourcen der Anteilseigner heraus. Wenn wir heute etwas entwickeln wollen, dann müssen wir erst einmal nachweisen, dass es eine Nachfrage auf Kundenseite und eine Aussicht auf Erfolg gibt. Wir sind sehr zurückhaltend, was unsere Investments angeht. Es gibt immer noch Raum für Innovationen, aber diese müssen in irgendeiner Art gerechtfertigt sein. Ein Beispiel dafür ist die sorgfältige Recherche, die uns zum Spring Cloud Service geführt hat. Ein weiterer Unterschied im Vergleich zu Oracle ist der weitaus größere Themenpool, an dem die Leute arbeiten. In meiner vorherigen Rolle bei Oracle war es für mich fast ausschließlich Java, andere machten C++ oder .NET.

Was die Leute von uns erwarten können, ist der Azure Spring Cloud Service sowie ARM-Vorlagen für WebLogic und WebSphere. Es wird auch mehr investiert werden, um VS Code für Java zu verbessern und mehr von Entwicklern benötigte Anwendungsfälle abzudecken. Im August 2019 hat Microsoft jClarity übernommen, ein Unternehmen für Java-Performance-Training und Tuning. Aufgrund der weitreichenden Erfahrung beim Erstellen und Optimieren von JDKs wird uns dieses Team auch dabei helfen, die Java-Vision bei Microsoft voranzutreiben.

Außerdem freue ich mich sehr darüber, dass Microsoft sich am OpenJDK-Projekt beteiligt. Dies ist nach der Übernahme von jClarity ein logischer Schritt, denn die Verantwortlichen in diesem Unternehmen waren stark in AdoptOpenJDK involviert. Es ist wirklich toll, dass sie diese Beteiligung bei Microsoft fortsetzen können. Microsoft Azure hat einige der schwersten Java-Workloads auf dem Planeten. Wir nehmen diese Workloads, wenden die Leistungsexpertise von jClarity und die Reichweite von Microsoft an, mischen sie mit der Transparenz von OpenJDK und erhalten ein Erfolgsrezept, um Java noch besser zu machen.

Vielen Dank für das Interview, Ed!

Mitglieder des iJUG

01 Android User Group Düsseldorf
02 BED-Con e.V.
03 Clojure User Group Düsseldorf
04 DOAG e.V.
05 EuregJUG Maas-Rhine
06 JUG Augsburg
07 JUG Berlin-Brandenburg
08 JUG Bremen
09 JUG Bielefeld
10 JUG Bonn
11 JUG Darmstadt
12 JUG Deutschland e.V.
13 JUG Dortmund
14 JUG Düsseldorf rheinjug
15 JUG Erlangen-Nürnberg
16 JUG Freiburg
17 JUG Goldstadt
18 JUG Görlitz
19 JUG Hannover
20 JUG Hessen
21 JUG HH
22 JUG Ingolstadt e.V.
23 JUG Kaiserslautern
24 JUG Karlsruhe
25 JUG Köln
26 Kotlin User Group Düsseldorf
27 JUG Mainz
28 JUG Mannheim
29 JUG München
30 JUG Münster
31 JUG Oberland
32 JUG Ostfalen
33 JUG Paderborn
34 JUG Passau e.V.
35 JUG Saxony
36 JUG Stuttgart e.V.
37 JUG Switzerland
38 JSUG
39 Lightweight JUG München
40 SOUG e.V.
41 SUG Deutschland e.V.
42 JUG Thüringen

iJUG Verbund

www.ijug.eu

Impressum

Java aktuell wird vom Interessenverbund der Java User Groups e.V. (iJUG) (Tempelhofer Weg 64, 12347 Berlin, *www.ijug.eu*) herausgegeben. Es ist das User-Magazin rund um die Programmiersprache Java im Raum Deutschland, Österreich und Schweiz. Es ist unabhängig von Oracle und vertritt weder direkt noch indirekt deren wirtschaftliche Interessen. Vielmehr vertritt es die Interessen der Anwender an den Themen rund um die Java-Produkte, fördert den Wissensaustausch zwischen den Lesern und informiert über neue Produkte und Technologien.

Java aktuell wird verlegt von der DOAG Dienstleistungen GmbH, Tempelhofer Weg 64, 12347 Berlin, Deutschland, gesetzlich vertreten durch den Geschäftsführer Fried Saacke, deren Unternehmensgegenstand Vereinsmanagement, Veranstaltungsorganisation und Publishing ist.

Die DOAG Deutsche ORACLE-Anwendergruppe e.V. hält 100 Prozent der Stammeinlage der DOAG Dienstleistungen GmbH. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. wird gesetzlich durch den Vorstand vertreten; Vorsitzender: Stefan Kinnen. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. informiert kompetent über alle Oracle-Themen, setzt sich für die Interessen der Mitglieder ein und führen einen konstruktiv-kritischen Dialog mit Oracle.

Redaktion:
Sitz: DOAG Dienstleistungen GmbH
ViSdP: Mylène Diacquenod
Redaktionsleitung: Lisa Damerow
Kontakt: redaktion@ijug.eu

Redaktionsbeirat:
Andreas Badelt, Melanie Feldmann, Markus Karg, Manuel Mauky, Bernd Müller, Benjamin Nothdurft, Daniel van Ross, André Sept

Titel, Gestaltung und Satz:
Caroline Sengpiel,
DOAG Dienstleistungen GmbH

Fotonachweis:
Titel: Bild © Serhii Yaremenko | *https://de.123rf.com*
S. 10: Bild © Luca Bertolli | *https://de.123rf.com*
S. 12: Bild © RAJESH RAJENDRAN NAIR | *https://de.123rf.com*
S. 20: Bild © Marc-Steffen Unger
S. 22: Bild © SOPAN HADI | *https://de.123rf.com*
S. 27: Bild © Freepik | *https://de.freepik.com*
S. 28 + 29: Bilder © Schuchrat Kurbanov | DOAG
S. 30: Bild © sashkin7 | *https://de.123rf.com*
S. 35: Bild © aurielaki | *https://de.123rf.com*
S. 37: Bild © laracold | *https://de.123rf.com*
S. 39: Bild © Руслан Коляев | *https://de.123rf.com*
S. 45: Bild © Timur Arbaev | *https://de.123rf.com*
S. 50: Bild © Руслан Нестеренко | *https://de.123rf.com*
S. 58: Bild © Jennifer Reis | *JR Rock Shots*

Anzeigen:
Simone Fischer, DOAG Dienstleistungen GmbH
Kontakt: anzeigen@doag.org

Mediadaten und Preise unter:
www.doag.org/go/mediadaten

Druck:
adame Advertising and Media GmbH,
www.adame.de

Inserentenverzeichnis

Data Analytics 2020

28. & 29. April | in Düsseldorf

DOAG

analytics.doag.org

Programm online

Early Bird
bis 21. Januar 2020

JavaLand 2020

17. - 19. März 2020 in Brühl bei Köln

Ab sofort Ticket & Hotel buchen!

www.javaland.eu

Präsentiert von: DOAG Heise Medien

Community Partner: iJUG Verbund

www.javaland.eu